DOCTORAL SCHOOL
UNIVERSITY OF MILANO-BICOCCA

Department of Informatics, Systems and Communication

PhD program in Computer Science

Cycle XXXV

Three Perspectives on Anomaly Detection in Deep Learning

Francesco CRAIGHERO

Registration number: 854389

Supervisor: Prof. Marco ANTONIOTTI

Co-supervisor: Dr. Alex GRAUDENZI

Tutor: Prof. Francesca ARCELLI

PhD Program Director: Prof. Leonardo MARIANI

ACADEMIC YEAR 2021/22

Abstract

Real-world applications of deep learning are rapidly growing, but so are the concerns on the fairness and safety of such models. Understanding what a deep model knows, while discarding what it is unsure about, represents a key step in reaching a more trustworthy AI. Towards this goal, *anomaly detection* strives to define scoring methods for input examples that distinguish normal, well-represented, instances from the ones that are rarely occurring, corrupted or out-of-distribution. Ranking instances is not only a way to allow users to discard untrustworthy inputs, but provides useful insights about the data itself. Indeed, since modern datasets are ever-growing, methods that perform automated data auditing such as anomaly detectors are of crucial importance.

In this thesis, I will discuss three approaches related to anomaly detection in deep learning. First, I will introduce the Activation Pattern DAG (APD), a Directed Acyclic Graph (DAG) that summarizes the latent space's properties of Deep Neural Networks (DNNs). I will also show that the APD can be used to cluster input instances based on their similarity in the latent space. More in detail, the APD exploits DNNs' activation patterns, a discrete representation of latent features resulting from the application of piecewise linear activations functions. Experiments show that the cluster size can be used to rank input instances by difficulty, allowing one to predict misclassified instances, but also to reduce the size of a dataset by identifying a meaningful subset of representatives. Second, I will present the ENsemble Adversarial Detector (ENAD), a new method for adversarial examples detection in Convolutional Neural Networks (CNNs), based on the idea of integrating score functions from state-of-the-art detectors. Results prove the goodness of its performances against state-of-the-art detectors, with both known and unknown adversarial attacks. Furthermore, ENAD allows for flexibility in the number of detectors in the ensemble, making it possible to incorporate newly introduced methods. Third, I will present a study investigating the behaviour of deep models on imbalanced data, by considering the uncertainty of the model and the complexity of data. I will discuss two case studies, one on predicting the binding affinity between T-cell receptors (TCRs) and epitopes, a recent and important application of deep learning in immunology, and a more standard task of image classification. In order to analyse the deep models, I selected three metrics related to uncertainty estimation and anomaly detection. Moreover, I will discuss the use of data dimensionality estimation to detect imbalanced class complexities.

Overall, these efforts provide three contributions to the broad and important area of anomaly detection in deep learning, which is recently gaining a lot of attention for its connections with generalization and safety.

Three additional projects in computational biology are available in the appendix. They include a review on imputation and denoising methods for single-cell data, a classifier to predict cancer samples from the topological properties of metabolic networks, and a deep learning model to predict relative fluxes in reaction systems.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence.
APD	Activation Pattern DAG.
AUPR	Area Under the Precision-Recall curve.
AUROC	Area Under the Receiver Operating Characteristic.
AU	Aleatoric Uncertainty.
BIM	Basic Iterative Method.
CNN	Convolutional Neural Network.
CW	Carlini-Wagner.
DAG	Directed Acyclic Graph.
DNN	Deep Neural Network.
ENAD	ENsemble Adversarial Detector.
EU	Epistemic Uncertainty.
ExAD	Ensemble approach for Explanation-based Adversarial Detection.
FGSM	Fast Gradient Signed Method.
FNN	Feedforward Neural Network.
FN	False Negative.
FP	False Positive.
GMM	Gaussian Mixture Model.
ID	Intrinsic Dimensionality.
LID	Local Intrinsic Dimensionality.
OCSVM	One-Class Support Vector Machine.
OOD	Out-of-Distribution.
PCA	Principal Component Analysis.
Pr	Precision.
RBF	Radial Basis Function.
RNN	Recurrent Neural Network.
ReLU	Rectified Linear Unit.
Re	Recall.
SVM	Support Vector Machine.
TCR	T-cell receptor.

TITAN	Tcr epITope bimodal Attention Networks.
TN	True Negative.
TP	True Positive.
aK-LPE	averaged K (nearest neighbors) Localized p-value Estimation.
tSNE	t-distributed Stochastic Neighbor Embedding.
twoNN	two Nearest Neighbours.

Chapter 1

Introduction

Deep learning achieved impressive results in many areas, including biosciences [Sap+22], medical imaging [Lit+17; Lei+17], natural language processing [Bro+20], large-scale image recognition [KSH12], autonomous driving [Gri+20; Boj+16], and playing mind games and video games [Sil+16; Vin+19]. Examples of such groundbreaking achievements are protein structure prediction [Jum+21] or beating professional human players in the game of Go [Sil+16], considered the most challenging mind game.

The ever-growing number of applications of deep learning, including safety-critical settings such as medical imaging [Lei+17] and self-driving cars [Boj+16], pushed machine learning research towards a safer and more explainable AI [Mur+19; Amo+16; Moh+23; Die17]. Examples of such results are feature attribution methods, used to explain the model's decisions [STY17; SGK17; STY17; Spr+15] or detectors to identify possible failure points of the model [Yan+22].

This thesis will contribute to the broad field of anomaly detection (also referred to as Out-of-Distribution detection [Yan+22]) in deep learning, with the dual aim of increasing the interpretability and safety of a given model. In contrast to classical anomaly detection [Ruf+21], that usually involves applying a detector directly on the input data, I will present methods that detect anomalies from the properties of a trained Deep Neural Network (DNN), such as its latent features. Accordingly, characterizing anomalies allows us to both interpret the behaviour of the model on outliers [Mur+19] and to define run-time error detection methods [Moh+23].

1.1 Anomaly Detection in Deep Learning

The main objective of anomaly detection in deep learning is to assign scores to input instances in order to distinguish the "normal" from the "anomalous" ones. Normal examples are well-represented in the training set, and the model is more confident when making predictions on them. On the other hand, anomalous examples may fall into different categories such as rare instances from small subpopulations, or inputs deliberately altered by an adversary to fool the model [Sze+14; GSS15].

In real-world applications of deep learning, anomalies can be a result of the so-called *closed-world assumption* [Yan+22]. Indeed, while benchmark data is usually defined in such a way that training and test data are drawn from the same underlying distribution, in real applications we have to deal with Out-of-Distribution (OOD) data [Yan+22]. Moreover, large datasets typically have a long-tailed distribution [ZAR14], therefore the model can be biased towards more frequently occurring instances [Fel20], while performance could degrade for the under-represented examples. These issues will be discussed in section 2.2, as important motivations of anomaly detection in deep learning.

Tackling anomalous examples requires the definition of a ranking that separates normal points from the others. In contrast to standard anomaly detection [Ruf+21], in this thesis we will identify anomalies by considering the properties of trained deep models. As an example, recent approaches consider the learning dynamics [Ton+19] or the variance of model's gradients [ADH22] as informative features to characterize abnormality. Other methods estimate the model's uncertainty [LPB17; GG16] to distinguish when the model is confident from when the prediction is untrustworthy. Indeed, the model's uncertainty might reflect the difficulty of the input data, allowing to discard anomalous inputs [Lei+17]. Other approaches apply standard anomaly detection to DNNs, for example by using nearest neighbours [PM18] or the Mahalanobis distance [Lee+18] in the latent space. An exhaustive discussion of anomaly detection in deep learning is available in section 2.3.

According to the taxonomy of machine learning interpretations defined in [Mur+19], anomaly detection can be classified as a post-hoc analysis method that provides data-level interpretations. In other words, characterizing anomalies provides practitioners useful insights on the relationships (e.g., common vs rare) learned by a fitted model. Indeed, it is of great importance to understand the model's behaviour on outlier inputs [ADH22]. In this regard, in chapter 3 we will introduce a method to perform automated data auditing based on example difficulty and in chapter 5 we will employ anomaly detection to interpret the behaviour of a model on imbalanced datasets.

Anomaly detection is also fundamental for AI safety [Moh+23; Die17; Amo+16], since it can be adopted to improve the robustness to distributional change [Amo+16] by monitoring the model predictions [Moh+23]. In this thesis, we will discuss the important aspect of adversarial attack detection in chapter 4, where we will discuss how to identify images crafted to fool a trained classifier [GSS15].

Additional details on interpretability and safety in deep learning are provided in section 2.4.

1.2 Thesis Goals

In this thesis, we will present three different projects on anomaly detection in deep learning. In particular, all the methods will try to address the following research objectives:

> Given a trained Deep Neural Network (DNN), are there measurable properties that characterize anomalous data? Can such properties be exploited to improve the interpretability and safety of the model?

As I will discuss in section 2.3, many approaches towards these research aims have already been proposed. My contributions to the field either involve the application of anomaly detection in a novel setting or an advancement over existing methods:

1. introduction of an anomaly detector for piecewise linear DNNs chapter 3.
2. improvement of the state-of-the-art in adversarial detection using ensemble approaches in chapter 4.
3. studying DNNs on imbalanced classification tasks with anomaly detection through uncertainty estimation in chapter 5.

The common thread among all the approaches is the use of the latent features of DNNs as relevant knowledge to characterize anomalies. Indeed, we expect anomalous inputs to be outliers in the hidden representation learned by the fitted model. In chapter 3 we introduce an approach that considers a binarized version of the latent features, called activation patterns, proving to be sufficient in characterizing anomalous instances. In chapter 4 each layer of the DNN is considered as a standalone anomaly detection task, and the final outcome is obtained through an aggregation function. Last, in chapter 5 we use the latent features to estimate the uncertainty of the model on a given input.

In the next section, I will describe the thesis structure and the contributions more in detail.

1.3 Contributions

The first contribution to the field presented in the thesis will be the Activation Pattern DAG (APD) [Cra+20a; Cra+20b], discussed in chapter 3. The APD is a Directed Acyclic Graph (DAG) that summarizes the behaviour of a piecewise linear DNN on a dataset. More in detail, input examples will be clustered based on the activations of each hidden layer of the model, that characterize their similarity. Anomalous instances, in this case, will be the input examples that belong to small clusters. Experiments performed on computer vision data will further confirm that misclassified instances belong to small clusters, while larger ones contain similar instances that could be represented by just one prototype, with a small impact on the performances.

A particular type of anomaly, first introduced in computer vision [Sze+14; GSS15], are adversarial examples. These images are crafted by adding imperceptible noise to normal, correctly predicted, instances, in order to fool the model into making a wrong prediction. In the literature, many techniques have been proposed to detect adversarial examples (reviewed in section 2.3.2). ENsemble Adversarial Detector (ENAD) [Cra+21], introduced in chapter 4, is a novel adversarial detector that ensembles existing state-of-the-art anomaly detectors ([Lee+18; Ma+18; Sch+99]) to improve the detection performance. The effectiveness of ENAD is driven by the fact that the crucial layer and detector vary based on the adversarial attack being addressed. As a result, combining these detectors in an ensemble leads to improved generalization capabilities. Detailed experimental results will consider both the known attack scenario, in which the adversarial attack is the same in the train and test set, and the harder transfer attack scenario, in which the generalization to unseen attacks will be evaluated.

In chapter 5, I will discuss a project focused on the application of uncertainty estimation techniques (reviewed in section 2.3.3), and data complexity estimates, in the form of the intrinsic dimensionality (introduced in section 5.2.1), to interpret the behaviour of deep models for imbalanced binary classification tasks. In particular, I will present two case studies: T-cell receptor (TCR) and epitope binding affinity prediction using the TITAN [WBR21] deep model (in section 5.4.1), and a more standard image classification task (in section 5.5). In the first study, we found that instances of one target class were closer together in the latent space than instances of the second class. We referred to this as the "*Epistemic Gradient*" and speculated that it is a result of data imbalance. We also suggested that a binary classifier trained on imbalanced data could act as an anomaly detector by only fitting the well-represented class, and proposed using sensitivity to Out-of-Distribution (OOD) data (defined in section 5.3.3) to detect this behaviour. All in all, this analysis approaches an interpretability problem (understanding deep models on imbalanced data) from an anomaly detection point of view, opening a valuable future research direction given the frequency of these problems in real-world scenarios [FAK19].

To summarize, this thesis will discuss three different perspectives on anomaly detection: example difficulty estimation in chapter 3, adversarial examples detection in chapter 4 and interpreting deep models in imbalanced binary classification tasks in chapter 5. A schematic depiction of the thesis structure is presented in fig. 1.1.

Moreover, in appendix A I reported three additional works in computational biology, to which I contributed during my PhD: a review of imputation and denoising techniques for single-cell data [Pat+20] (appendix A.1), a classification method to predict cancer samples from metabolic networks [Mac+21] (appendix A.2), and a deep model to predict relative fluxes in reaction systems [Pat+21] (appendix A.3).

Throughout this thesis I will use "we", instead of "I", since in this thesis I will discuss works that are either part of an article, and therefore a contribution of multiple authors, or the result of projects involving also other researchers. In particular, the people involved are members of the Data and Computational Biology (DCB) Lab at the University of Milano-Bicocca (Dr. Fabrizio Angaroni, Prof. Marco Antoniotti, Prof. Chiara Damiani, Dr. Alex Graudenzi, Dr. Davide Maspero and Lucrezia Patruno), of the Scientific Computing Group (SCG) at the University

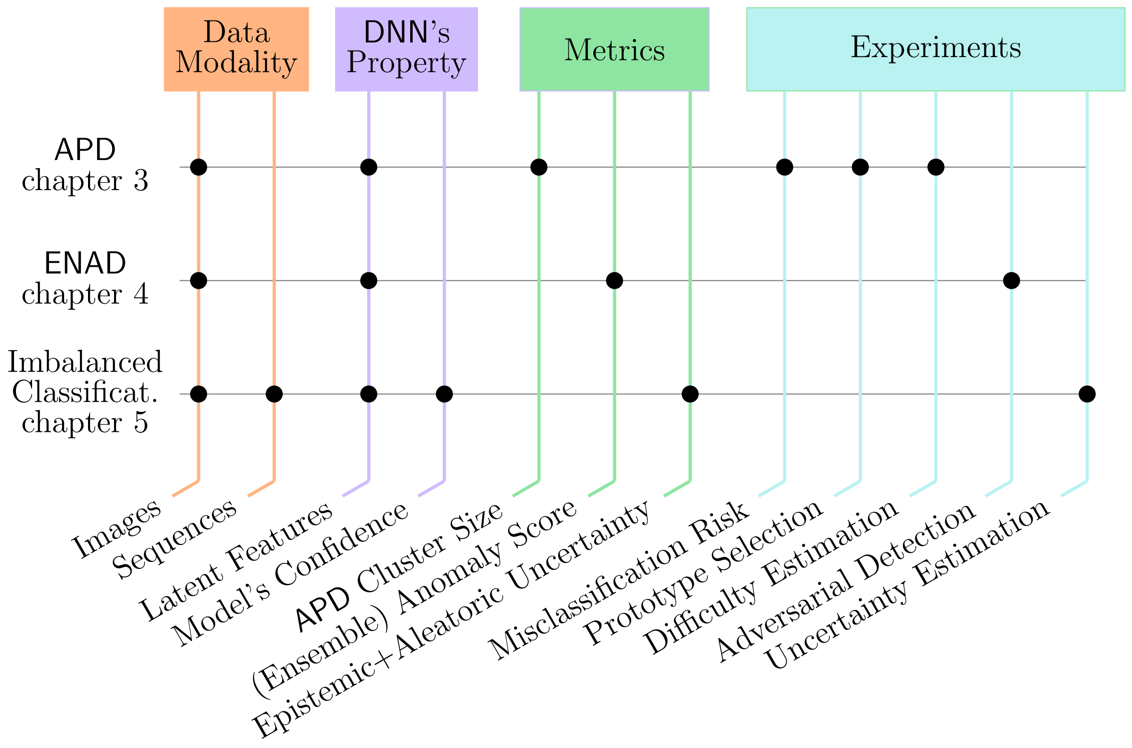


Figure 1.1: **Thesis summary**. In this diagram we provide a description of the three main tasks carried out during the PhD project: the Activation Pattern DAG (APD) (chapter 3), the ENsemble Adversarial Detector (ENAD) (chapter 4) and the project on imbalanced binary classification (chapter 5). In particular, we considered (1) the data modality (images or sequences), (2) the property of the DNN that the detectors uses to output a scoring (latent features or model's confidence), (3) the kind of metric defined, and (4) the experiments performed to validate the metric.

of São Paulo (Prof. Odemir Martinez Bruno and Dr. Jeaneth Machicao) and of the Computational Systems Biology Lab at IBM Research Zürich (Dr. Nicolas Deutschmann and Dr. María Rodríguez Martínez).

Chapter 2

Background: Anomaly Detection in Deep Learning

2.1 Introduction

Anomaly detection in deep learning involves identifying those inputs that differ from the ones considered "normal". On such points, the model can perform badly, therefore their characterization is important in order to discard untrustworthy predictions, in particular for safety-critical settings [Amo+16; Moh+23; Die17].

There are many kinds of anomalous inputs in deep learning, from adversarial examples [GSS15], that are instances perturbed to fool the model, to points drawn from unknown distribution, i.e., Out-of-Distribution (OOD). In this thesis, we are mainly concerned with adversarial examples (introduced in detail in section 4.2.1) and challenging examples, where the difficulty is given by a proxy metric (introduced in section 2.3.1). In chapter 5, we will also consider OOD data to interpret the model's behaviour. For a more detailed characterization of anomalous inputs, refer to [Yan+22].

In section 2.2 and section 2.4, we introduce the motivations that justify the

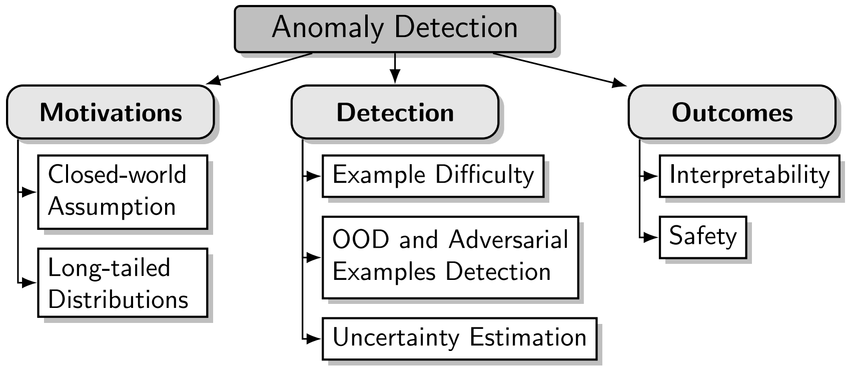


Figure 2.1: **Background summary of anomaly detection in deep learning**.

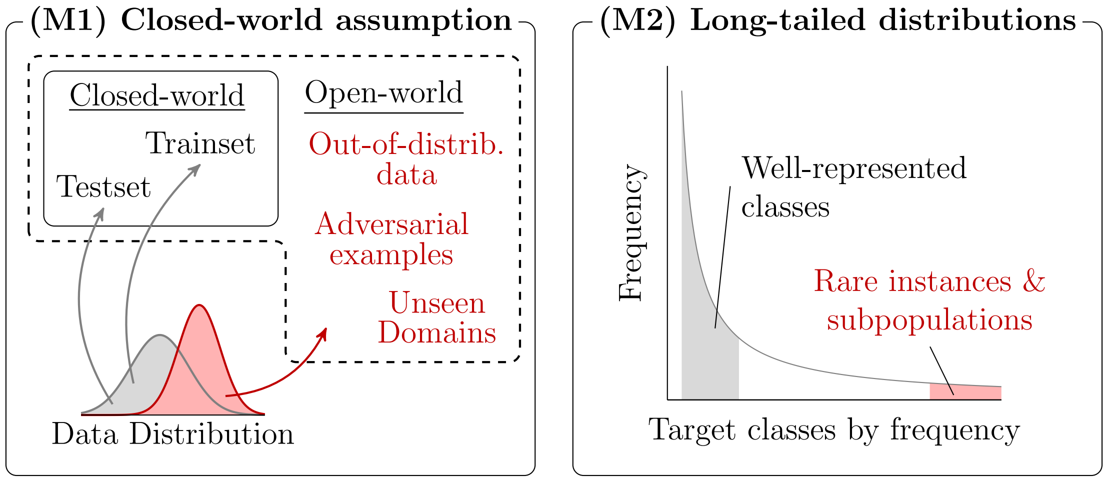


Figure 2.2: **Motivations for anomaly detection in deep learning**. **Closed-world assumption (M1)**, section 2.2.1: when violated, data is no longer drawn from the same distribution of train and test data. Examples are Out-of-Distribution (OOD) data, adversarial examples or instances from unseen domains. **Long-tailed distributions (M2)**, section 2.2.2: real-world data frequently shows a long-tailed distribution, with many examples that are either from small subpopulations or belong to rare events. Dealing with this kind of data is important in order to challenge biases of the model and to improve the trustworthyness of the predictions.

use of anomaly detection in deep learning and its benefits, respectively, while in section 2.3 we introduce three families of anomaly detectors. The chapter structure is reported in the diagram in fig. 2.1.

2.2 Motivations: Why We Have To Deal With Anomalies

Anomaly detection in deep learning is motivated by two main factors (see fig. 2.2). The first, introduced in section 2.2.1, is the fact that in real-world tasks the assumption that the test data is drawn from the same distribution as training data (known as the closed-world assumption) is often violated, making the generalization of the model more challenging. The second factor, described in section 2.2.2, is the presence of long-tailed distributions in real-world datasets, which can contain many under-represented modalities. This can lead to untrustworthy predictions and fairness concerns.

2.2.1 Closed-world Assumption

When evaluating a model based on the so-called *closed-world assumption* [Yan+22], we expect test examples to lie in the same distribution of the training data.

Although, in real-world applications this assumption might no longer hold for many reasons. For example, an attacker might generate adversarial examples that fool the model [GSS15], or the training data could be under-representative of all the possible unseen domains, a common issue in precision medicine such as AI-aided drug discovery [Ji+22b].

There are two main ways to handle OOD data: either by using detectors that detect and reject anomalous inputs (as discussed in section section 2.3.2 and demonstrated in chapter 4 for adversarial examples), or by creating models that are more robust and generalize better to unseen domains [Shu+21].

For a detailed taxonomy of anomalies and detection schemes, refer to [Yan+22].

2.2.2 Long-tailed Distributions

The ever-growing size of datasets used to train and evaluate deep models makes the analysis of all the data points almost impossible. For small data tasks, like precision medicine, we might have a different, but equally challenging, situation in which one does not know which are the anomalous instances. Consequently, the main concerns include the fairness towards under-represented subpopulations [And+21] and untrustworthy predictions, that for safety-critical settings should be detected and discarded.

Vision datasets have been showed to have long-tailed distributions [ZAR14], and are therefore probably susceptible to under-represented modalities. In this regard, Feldman [Fel20] showed that deep models have to memorize long-tailed subpopulations to achieve generalization. Recently, long-tailed distributions have been studied in medical imaging in order to improve the safety of classifier of dermatological conditions [Roy+22].

In order to automate data analysis, reducing the required human contribution and the prior knowledge of the data, one can consider at least two approaches. The first one consists of methods that use the model's properties to define a score for each example based on their difficulty (see section 2.3.1), such as the Activation Pattern DAG (APD) that we introduced in chapter 3. The second approach is estimating the predictive uncertainty (see section 2.3.3), to distinguish simple examples from the untrustworthy ones. The intuition is that by ranking examples, we are able to separate points belonging to the tail from the well-represented ones.

2.3 How to Detect Anomalies

In this section, we will introduce methods to characterize anomalies in deep learning. The first category (section 2.3.1) is about example difficulty estimators, that rank input instances based on their difficulty. Difficult examples could be misclassified instances, rarely occurring examples or corrupted inputs. Second, in section 2.3.2 we will introduce the detectors for OOD and adversarial examples. Last, in section 2.3.3

will describe uncertainty estimation in deep learning, that characterizes anomalous inputs by mainly estimating the predictive uncertainty.

Despite being in three different categories, the detection methods that we will introduce are not completely independent. The taxonomy that we used follows the scope of each work, but example difficulty scores or uncertainty metrics can be applied also for adversarial or OOD detection. Indeed, it is an interesting research question to evaluate how the methods that we will introduce differ, for example from the point of view of aleatoric and epistemic uncertainty (introduced in section 2.3.3).

2.3.1 Example Difficulty Estimation

Example difficulty estimation in deep learning concerns all the methods that define a ranking of input examples based on their expected difficulty or hardness. Uncertainty estimation and anomaly detection techniques (described in sections 2.3.2 and 2.3.3, respectively) can also be used to define rankings of samples, although in this section we consider methods that are designed to estimate a more abstract idea of difficulty. A good ranking considers misclassified, out-of-distribution and rarely occurring points as challenging for the model, while well-represented inputs in the training set should be considered easy examples. The applications of example difficulty estimation methods are many, from detecting rare subpopulations in long-tailed datasets to predicting misclassified examples.

The first category of example difficulty rankings ([Ton+19; ADH22]) considers the learning dynamics to assign a score to each example. For example, in [Ton+19] the ranking is defined by the number of times an example is learned, and then forgotten, during training, where the counts are proportional to the difficulty. On the other hand, in [ADH22] the variance of the gradients through training is used to distinguish easy (low variance) to difficult (high variance) examples.

The second group considers piecewise linear DNNs, by using the (smoothed) local empirical error estimated in each linear region [Ji+22a] or the Jacobian norm of a linear region [Nov+18] as proxies for example difficulty. In addition to the previous categories, in [Jia+21] the authors estimated the contribution of each instance to the model's generalization performance and in [CEP19] the authors compared different scoring methods, such as the adversarial robustness. Lastly, in [BMN21] the authors estimated the layer at which an instance is correctly predicted as an example difficulty proxy.

2.3.2 OOD and Adversarial Examples Detection

The last anomaly detection topic that we will discuss is Out-of-Distribution (OOD) and adversarial examples [GSS15] detection in deep learning. These methods, similarly to example difficulty estimation (describe in section 2.3.1), define a scoring of input instances to distinguish anomalous from normal examples. Although, in contrast to example difficulty, the main purpose of these methods is either detecting if a point is OOD or an adversarial example. For a recent review, refer to [Yan+22]. In the following, mostly taken from [Cra+21], we will refer to anomalies in general, although not all the methods are designed for both OOD and adversarial example detection.

First, let us consider a classifier $\mathcal{N}$ trained on a training set $\mathcal{X}^{train}$ and a test example $\boldsymbol{x_0} \in \mathcal{X}^{test}$, with predicted label $\hat{y}_0$. Detectors can be broadly categorized according to the features they consider: (i) the features of the test example $\boldsymbol{x_0}$, (ii) the hidden features $h_l(\boldsymbol{x_0})$, or (iii) the features of the output of the network $out(\boldsymbol{x_0})$, i.e., the logits $\sigma_{\text{lgt}}(\boldsymbol{x_0})$ or the confidence scores $\sigma_{\text{sm}}(\boldsymbol{x_0})$.

The first family of detectors tries to distinguish normal from anomalous examples by focusing on the input data. For instance, in [HG17b] the coefficients of low-ranked principal components of $\boldsymbol{x_0}$ are used as features for the detector. In [Gro+17], the authors employed the statistical divergence, such as Maximum Mean Discrepancy, between $\mathcal{X}^{train}$ and $\mathcal{X}^{test}$ to detect the presence of anomalous examples in $\mathcal{X}^{test}$. Lastly, in [Var+21] the feature attributions of the input image are considered as features of deep models for anomaly detection.

The second family of detectors aims at exploiting the information of the hidden features $h_l(\boldsymbol{x_0})$. In this group, some detectors rely on the identification of the nearest neighbours of $h_l(\boldsymbol{x_0})$ to detect anomalous examples, by considering either: the Euclidean distance to the neighbours [Car+17], the conformity of the predicted class among the neighbours [PM18; Rag+21], the Local Intrinsic Dimensionality [Ma+18], the impact of the nearest neighbours on the classifier decision [CSG20], or the prediction of a graph neural-network trained on the nearest neighbours graph [Abu+21]. In the same category, additional detectors take into account the conformity of the hidden representation $h_l(\boldsymbol{x_0})$ to the hidden representation of instances with the same label in the training set, i.e., to $\{h_l(\boldsymbol{x_0}) : \hat{y}_0 = y, (\boldsymbol{x}, y) \in X^{train}\}$, by computing either the Mahalanobis distance [Lee+18; KK20] to the class means, or the likelihood of a Gaussian Mixture Model (GMM) [ZH18]. Other detectors take the hidden representation $h_l(\boldsymbol{x_0})$ itself as a discriminating feature, by training either a DNN [Met+17], a Support Vector Machine (SVM) [LIF17], a One-Class Support Vector Machine (OCSVM) [Ma+19] or by using a kernel density estimate [Fei+17]. Additional methods within this family use the hidden representation $h_l(\boldsymbol{x_0})$ as a feature to train a predictive model m_l. The model m_l either seeks to predict the same C classes of the original classifier [Ma+19; Sot+20] or to reconstruct the input data $\boldsymbol{x_0}$ from $h_l(\boldsymbol{x_0})$ [HG17b; HG17a]. The detector then classifies $\boldsymbol{x_0}$ as anomalous either by relying on the confidence of the prediction $\hat{y}_0$ in the former case or on its reconstruction error in the latter.

The third family of detectors employs the output of the network $out(\boldsymbol{x_0})$ to detect adversarial inputs. In this category, some detectors consider the divergence between $out(\boldsymbol{x_0})$ and $out(\phi(\boldsymbol{x_0}))$, where ϕ is a function such as a squeezing function that reduces the features of the input [XEQ18], an autoencoder trained on $\mathcal{X}^{train}$ [MC17], a denoising filter [Lia+21] or a random perturbation [RKH19; HWW19], or an operation of erase and restore of random pixels [ZZ21]. Some others take the confidence score of the predicted class $\hat{y}$, which is expected to be lower when the example is anomalous [HG17b; HG17a; LLS18]. Lastly, in [AD19] a DNN detector was trained directly on the logits, whereas in [Fei+17] Bayesian uncertainty of dropout DNNs was used as a feature for the detector.

Detectors exist that do not fall within any of the above families, which employ, for instance, the layer-wise norm of the gradients [LC20] and the consistency of the softmax scores $\sigma_{\text{sm}}(\boldsymbol{x_0})$ of multiple models [Mon+19].

2.3.3 Uncertainty Estimation

Anomalous data points can be detected by estimating the uncertainty of the model, since we expect abnormal or unexpected examples to be under-represented or not represented at all in the training data manifold. In contrast to example difficulty estimation (introduced in section 2.3.1) and out-of-distribution detection (introduced in section 2.3.1), that mainly define a scoring of input instances, uncertainty is modelled in a probabilistic way and has a stronger theoretical basis (refer to [HW21] for a recent review).

In order to evaluate the uncertainty of a DNN, one can employ variational inference [Blu+15], dropout-based variational inference [GG16] or deep ensembles [LPB17]. For a comparison of methods for uncertainty estimation in deep learning, refer to [Sno+19].

The uncertainty can also be divided in two different components: *aleatoric uncertainty*, a result of randomness and for this reason irreducible, and *epistemic uncertainty*, that is the effect of lack of knowledge, and for this reason reducible [HW21]. On this distinction, in [KG17] the authors investigated both aleatoric and epistemic uncertainty in computer vision tasks.

2.4 Outcomes: How Anomaly Detection Improves Deep Learning

Anomaly detection in deep learning contributes to the important goal of a more interpretable [Mur+19] and safe [Moh+23; Die17; Amo+16] AI, towards the milestone of a Responsible AI [Arr+20]. In the following, we briefly introduce the two challenges and why this thesis contributes to them.

2.4.1 Interpretability

According to Murdoch et al. [Mur+19], interpretable machine learning involves "the extraction of relevant knowledge from a machine-learning model concerning relationships either contained in data or learned by the model".

Interpretability can be obtained by designing models that expose their decision process, as in the case of glassbox models [Nor+19]. On the other hand, when dealing with black-box models such as DNNs, the most adopted approach is post-hoc interpretability. In this case, we try to extract information from an already trained model. Post-hoc interpretations can be local, at prediction-level, or global, at dataset-level. In the former family, we assign feature attribution methods [Spr+15; SGK17; STY17], that assign an importance score to each input feature based on its contribution to a prediction. In the latter category, the interpretations provide an insight of global relationships learned by the model. An example is anomaly detection, that characterizes outlier inputs or trends in the data, allowing practitioners to use this information to improve the model [Lei+17] or to provide example-based explanations [KKK16].

2.4.2 Safety

In addition to interpretability, a crucial requirement for the adoption of AI systems in the real-world is their safety. The objective is to prevent the occurrence of accidents, defined by Amodei et al. as "unintended and harmful behavior that may emerge from machine learning systems when we specify the wrong objective function, are not careful about the learning process, or commit other machine learning-related implementation errors" [Amo+16]. In the following, we briefly list three safety strategies, following the taxonomy introduced in [Moh+23].

A first approach to improve safety is designing models that are interpretable by design, such as glassbox models [Nor+19], or by using post-hoc interpretability techniques such as feature attribution methods [Spr+15; SGK17; STY17]. It is worth noting that interpretability and safety are intertwined goals, as interpretability can be viewed as an essential requirement for achieving safety.

Given a machine learning algorithm, we can also try to increase its robustness with the so-called proactive methods. With regard to deep learning, these techniques include adversarial training [Sze+14; GSS15], defensive distillation [Pap+16] or ensemble approaches [AG17; BBS17; Str+18].

Last, we can perform run-time monitoring of the predictions to detect failure points, named reactive methods. In this category we find anomaly detection, as a last defence layer to prevent accidents. A simple example is prediction with reject options [BW08], in order to prevent the model to make low-confidence predictions.

Chapter 3

Quantifying Example Difficulty from Activation Patterns

Contribution. In this chapter, we report the work introduced in:

[Cra+20a] Francesco Craighero et al. "Investigating the Compositional Structure of Deep Neural Networks". In: *Machine Learning, Optimization, and Data science - 6th International Conference, LOD, Siena, Italy, July 19–23, 2020.*

[Cra+20b] Francesco Craighero et al. "Understanding Deep Learning with Activation Pattern Diagrams". In: *Proceedings of the Italian Workshop on Explainable Artificial Intelligence Co-Located with 19th International Conference of the Italian Association for Artificial Intelligence.* 2020

Summary. The evaluation of deep learning models is becoming more and more difficult due to the ever-growing size of modern datasets and models. Even when data is of moderate size, such as in AI for precision medicine, the data itself can be hard to be interpreted or might be corrupted due to the many sources of technological and experimental noise. These challenges raised the interest towards methods that automate data auditing, so to identify difficult examples or to predict misclassified instances. In this chapter, we introduce the Activation Pattern DAG (APD), a novel graph representation summarizing the features of a piecewise linear DNNs, inducing a ranking of data points based on example difficulty. The APD not only automates data auditing, but is also a useful visualization tool to interpret the behaviour of the model on the input dataset.

Implementation. The experiments performed in [Cra+20a] has been open-sourced on a Github repository[a].

[a] `https://github.com/BIMIB-DISCo/ANNAPD`

3.1 Introduction

An important requirement for the widespread adoption of deep learning models in real-world applications is their trustworthiness. Although, the so-called glassbox machine learning models [Nor+19] allow for an interpretation of the decisions that led to the final prediction, black-box models are, by definition, much harder to be interpreted.

In this chapter we tackle the challenge towards a more trustworthy AI from the perspective of example difficulty (introduced in section 2.3.1). Our aim is defining a score to differentiate simple input examples against more challenging one that are more prone to be misclassified or belong to under-represented modalities. As stated in [ADH22], knowing input difficulty allow us not only to identify and potentially discard atypical examples [BW08], but also to improve interpretability, e.g., through example-based explanations [KKK16]. Ultimately, the ability to extract difficult examples enables a more comprehensive understanding of the model's behaviour.

Example difficulty estimators are particularly helpful with modern large-scale datasets, since they provide an automated interpretation of the input examples without human intervention. Indeed, the long-tailed distribution of common datasets, discussed in section 2.2.2, is an additional reason to employ this kind of scoring metrics. Moreover, the estimation of example difficulty could be of great help also on smaller datasets, as an additional metric on top of the more commonly adopted uncertainty estimators (introduced in section 2.3.3).

Our method, that we will introduce in the following, is specific to piecewise linear DNNs, such as ReLU DNNs [GBB11]. Such models apply a distinct linear function to specific regions of the input space, called activation regions [HR19b]. Each activation region is uniquely identified by an activation pattern, obtained by concatenating the binarized output of each layer. Given that multiple input instances could belong to the same activation region, activation patterns can be shared among multiple examples. Montúfar et al. [Mon+14] showed how shared activation patterns are the result of the redundancies in the data found by the model.

In section 3.3.2, we will introduce a novel graph approach that summarizes the activation patterns given a model and a dataset, called the Activation Pattern DAG (APD). The APD is a Directed Acyclic Graph (DAG) that allows us to better understand how the patterns of each layer are shared among instances. Inspired by decision trees, in section 3.3.3 we will also define a clustering algorithm based on the APD with the aim of finding clusters of instances that share the same pattern in multiple layers and have the same predicted label.

The APD clustering is the main building block of our new example difficulty scoring, that considers the cluster size as a proxy for example difficulty: well-represented examples belong to larger clusters, while challenging points belong to smaller ones. To test our new metric, in section 3.4, we studied if cluster size is able to (i) detect misclassified examples, (ii) correlate with other example difficulty metrics and (iii) compress a dataset by identifying a subset of representative instances.

In addition to example difficulty, in section 3.4.3 we will also propose to employ the APD as a visualization tool, allowing to interpret the learned function given a group of instances. We remark that data visualization is another perspective towards trustworthy AI, in particular for black-box models as DNNs [Bau+17; OMS17]. Indeed, also the ENAD method that we will present in chapter 4 will have a data visualization perspective in section 4.4.3.

Main Contributions The main contributions of this work can be summarized as follows.

- *The Activation Pattern DAG (APD), a novel data structure to summarize activation patterns*: given a dataset and a piecewise linear DNN with ReLU activations, we introduce the Activation Pattern DAG (APD) to both summarize and visualize the activation patterns that occur in the latent space of the model.
- *Clustering the APD for example difficulty estimation*: we defined a clustering algorithm based on the APD that partitions the input data by exploiting the redundancy of the activation patterns. We then used the cluster size as a proxy for example difficulty, where anomalous instances are characterized by small under-represented clusters.
- *Validation of APD cluster size for misclassified example prediction*: we tested the reliability of cluster size as a metric to predict misclassified examples using multiple architectures trained on the MNIST [LCB10] dataset.
- *Using the APD cluster size for dataset compression*: we showed how APD clusters could be used to identify data prototypes: a subset of the training data that can be employed as an alternative to the full dataset, while preserving the model's performances.

3.2 Background

Example difficulty has already been introduced in section 2.3.1. In the following, we will present the main works related to piecewise linear DNNs and activation patterns.

Activation patterns identify the linear regions defined by piecewise linear DNNs, such as ReLU DNNs [GBB11]. Activation patterns can be used as a proxy to estimate function complexity, while the geometry of linear regions can be exploited to study the properties of the learned function. In the following, we will introduce some interesting results on piecewise linear DNNs.

The geometry of piecewise linear DNNs has been investigated from many perspectives, including approximation theory [Bal+19], circuit complexity [Aro+18] and tropical polynomials [CM18].

One of the most studied properties of DNNs is their representational power or expressivity. Early results by Pascanu et al. [PMB14] and Montúfar et al. [Mon+14] studied how deeper networks are more expressive by defining theoretical bounds on the number of activation patterns. Successive results proposed ways to achieve tighter bounds [Rag+17; STR18; SR20; HR19b; HR19a], for example through Mixed-Integer Linear Programming in [STR18; SR20].

On the intersection between example difficulty and activation patterns, the relation between linear regions' properties and the performance of the model has been observed in [Ji+22a; Nov+18]. In particular, in [Nov+18] the authors showed how the linear region's Jacobian norm is predictive of the cross-entropy loss, while in [Ji+22a] the linear region's empirical training error was used as a proxy for example difficulty.

3.3 Methods

In this section, we will formally define the Activation Pattern DAG (APD) and present a novel algorithm to cluster input examples based on activation patterns. In the following definitions, we will employ the notation used in [Mon+14], while we refer to [HR19b] for an extensive formal description of activation patterns and activation regions.

3.3.1 Basic Definitions

Let $\mathcal{N}_\theta(x_0)$ be a Feedforward Neural Network (FNN) with input $x_0 \in \mathbb{R}^{n_0}$ of n_0 dimensions and trainable parameters θ. Each layer h_l, for $l \in 1, \ldots, L$, is represented as a vector of dimension n_l, i.e., $h_l = [h_{l,1}, \ldots, h_{l,n_l}]^T$, where each component $h_{l,i}$ (i.e., a neuron or unit) is the composition of a linear preactivation function $f_{l,i}$ and a nonlinear activation function $g_{l,i}$, i.e. $h_{l,i} = g_{l,i} \circ f_{l,i}$.

Let x_l be the output of the l-th layer for $l = 1, \ldots, L$ and the input of the network for $l = 0$, then, we define $f_{l,i}(x_{l-1}) = W_l x_{l-1} + b_{l,i}$, where both $W_l \in \mathbb{R}^{n_{l-1}}$ and $b_{l,i} \in \mathbb{R}$ belong to the trainable parameters θ. Regarding activation functions, in this thesis we will focus on piecewise linear activation functions. Thus, for the sake of simplicity, we define $g_{l,i}$ as a ReLU activation function [GBB11], i.e., $g_{l,i}(x) = \max\{0, x\}$. When clear from the context, we will omit the second index of $f_{l,i}$ and $g_{l,i}$ to refer to the vector composed by all of them.

Finally, we can represent the FNN $\mathcal{N}_\theta$ as a function $\mathcal{N}_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{out}$ that can be decomposed as

$$\mathcal{N}_\theta(x) = f_{out} \circ h_L \circ \cdots \circ h_1(x), \tag{3.1}$$

where f_{out} is the output layer (e.g., softmax, sigmoid, ...).

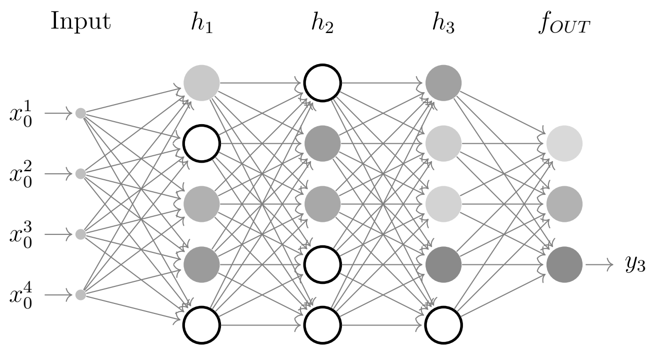


Figure 3.1: **Evaluation of a Feedforward Neural Network with activated units**. The example neural FNN $\mathcal{N}$ is being evaluated on an input instance $x_0 = [x_0^1, x_0^2, x_0^3, x_0^4]$. The hidden units are depicted with varying levels of transparency based on the value of the positive output, and units with output 0 are indicated by a black border. In the final layer, the output label y_3 indicates the output unit with the highest value. In this example, we have $A_1 = [1, 0, 1, 1, 0]$, $A_2 = [0, 1, 1, 0, 0]$, and $A_3 = [1, 1, 1, 1, 0]$.

3.3.2 From Activation Patterns to the APD

Given a FNN $\mathcal{N}_\theta$ and a dataset $\mathcal{D}$, we define the *activation pattern* of layer l given input $x \in \mathcal{D}$ as follows:

Definition 1 (Activation Pattern) *Let $\mathcal{N}_\theta(x_0)$ be the application of a FNN $\mathcal{N}$ with parameters θ on an input $x_0 \in \mathcal{D}$, with $\mathcal{D} \subseteq \mathbb{R}^{n_0}$. Then, by referring to x_{l-1} as the input to layer $l \in \{1, \ldots, L\}$, we can compute the activation pattern $A_l(x_0)$ of layer l on input x_0 as follows:*

$$A_l(x_0) = \{a_i \mid a_i = 1 \textit{ if } h_{l,i}(x_{l-1}) > 0 \textit{ else } a_i = 0,\ \forall i = 1, \ldots, n_l\}. \tag{3.2}$$

Thus, we can represent $A_l(x_0)$ as a vector in $\{0, 1\}^{n_l}$, i.e.:

$$A_l(x_0) = [a_1, a_2, \ldots, a_{n_l}]. \tag{3.3}$$

The above definition can also be easily extended to other binary piecewise activations, e.g. Leaky-ReLUs [MHN+13], or to maxout activations [Goo+13], by using k-ary activation patterns due to the k thresholds. In fig. 3.1 we show a simple example of a FNN $\mathcal{N}(x_0)$ and its activation patterns. In the following, we will represent generic activation patterns as a or a_i, and with $\texttt{layer}(a)$ we will refer to the layer corresponding to that pattern. In addition, we allow us to simplify the notation of A_l and refer to $A_l(\mathcal{X}_0)$ on $\mathcal{X}_0 \subseteq \mathbb{R}^{n_0}$ as $A_l(\mathcal{X}_0) = \bigcup_{x_0 \in \mathcal{X}_0} A_l(x_0)$.

Given an activation pattern $\hat{a}$, or a set of patterns $\mathcal{A}$ belonging to different layers, and a set of instances $\mathcal{X} \subseteq \mathcal{D}$, we call *activation region* the set composed by the instances in $\mathcal{X}$ that generate that activation pattern, or patterns, in their respective layers.

Definition 2 (Activation Region) *The activation region ($\mathcal{AR}$) identified by an activation pattern $\hat{a}$ on an input subset $\mathcal{X} \subseteq \mathcal{D}$ is given by:*

$$\mathcal{AR}(\hat{a}, \mathcal{X}) = \{x \in \mathcal{X} \mid A_l(x) = \hat{a},\ l = \texttt{layer}(\hat{a})\}. \tag{3.4}$$

Given a set of activation patterns $\mathcal{A}$ belonging to different layers, i.e., $\forall a_i, a_j \in \mathcal{A}$ $\texttt{layer}(a_i) \neq \texttt{layer}(a_j)$, we define their activation region as:

$$\mathcal{AR}(\mathcal{A}, \mathcal{X}) = \bigcap_{\hat{a} \in \mathcal{A}} \mathcal{AR}(\hat{a}, \mathcal{X}). \tag{3.5}$$

Given a dataset $\mathcal{D}$ and a FNN $\mathcal{N}_\theta$, we introduce the Activation Pattern DAG (APD) as the Directed Acyclic Graph (DAG) defined by all the activation patterns generated by instances in $\mathcal{D}$ and the way in which they are composed.

Definition 3 (Activation Patterns DAG) *Given a FNN $\mathcal{N}_\theta$ and a dataset $\mathcal{D} \subseteq \mathbb{R}^{n_0}$, the Activation Pattern DAG (APD) is a Directed Acyclic Graph (DAG) $APD_{\mathcal{N}_\theta}(\mathcal{D}) = (V, E)$, where:*

- *V is the set of vertices defined by*

  $$V = \{1, \dots, |\mathcal{A}|\},$$

  where $\mathcal{A} = \bigcup_{l=1}^{L} A_l(\mathcal{D})$ is the set of all possible activation patterns and $|\mathcal{A}|$ is its cardinality. In addition, let $\texttt{patt} : V \to \mathcal{A}$ be a labelling function that associates each vertex to the corresponding activation pattern.

- *E is the set of edges defined by:*

  $$E = \{(v_1, v_2) \in V \times V \mid \texttt{patt}(v_1) \text{ and } \texttt{patt}(v_2) \text{ are consecutive}\}, \tag{3.6}$$

  where two patterns a_i, a_j are called consecutive if

  $$\texttt{layer}(a_i) = l = \texttt{layer}(a_j) - 1$$

  and there exists $x \in \mathcal{D}$ such that $A_l(x) = a_i$ and $A_{l+1}(x) = a_j$.

From definition 3, it follows that the APD has the same depth of the corresponding FNN, and that a node at depth d in the APD corresponds to a pattern of the d-th layer in the FNN. In fig. 3.2 we show a toy-example for the $APD_{\mathcal{N}}$ defined by the FNN $\mathcal{N}$ of fig. 3.1, as generated on five example samples.

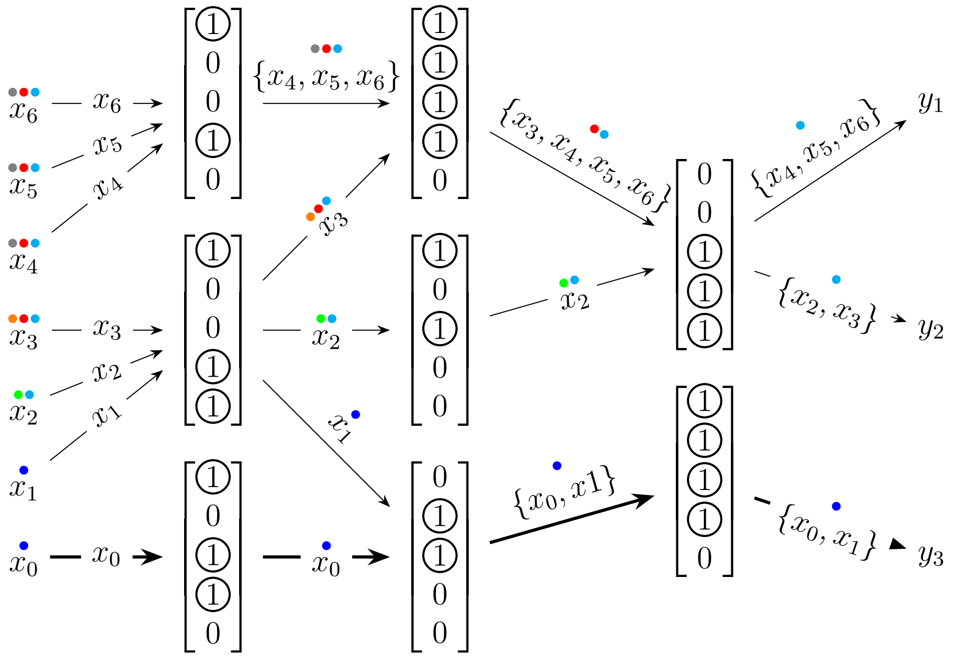


Figure 3.2: **An example APD and the resulting partitioning**. Representation of $APD_{\mathcal{N}}(x_0, \ldots, x_6)$, where the thicker lines indicate the edges generated by the input instance x_0 shown in fig. 3.1. The label predicted by the network is also displayed on the right. The colored bullets above the instances show the splitting history of the clusters, starting from the right and moving towards the left, according to the procedure described algorithm 1.

3.3.3 APD Clustering

Activation patterns identify the linear transformation applied by the model to a given input instance. On the one hand, the model might characterize each instance by its own patterns, on the other hand the model might find redundancies or symmetries in feature space, as shown in [Mon+14], and apply the same linear transformation to a set of examples. For example, in fig. 3.2, the linear transformation defined by pattern $a = [1, 1, 1, 1, 0]$ of the third layer is common to both x_0 and x_1. In the following, we will define a procedure to cluster a dataset based on these symmetries found by the model, expressed by the activation patterns. Moreover, since the symmetries might happen in an intermediate layer, we will consider only the patterns of some layers, instead of the pattern of the whole network.

We are using the meaning of activation patterns to group similar instances together, while also expecting these instances to have the same target label. Activation patterns that represent instances with the same target label will be referred to as "stable", while those that represent instances with different target labels will be referred to as "unstable". In fig. 3.2, for example, the third layer's pattern $a = [0, 0, 1, 1, 1]$ is unstable since its instances belong to both label y_1 and y_2.

We expect well-represented instances to belong to the same (stable) activation pattern, while rare or hard instances would either belong to unstable patterns or be characterized by their own pattern, given that the model is unable to exploit redundancies in the data. In fig. 3.2, instances $\{x_0, x_1\}$ belong to the same target y_3 and are assigned the same pattern in the second and third layer, therefore we expect those instances to be similar and to represent a well-represented property of the dataset.

The APD clustering algorithm is defined in algorithm 1. The first partition of input data is performed by considering only the activation patterns of the last layer. Then, similarly to decision trees, if one of the identified clusters contains instances with distinct labels, we split again by considering which activation pattern they activate in the previous layer. Intuitively, there is a trade-off between the density of the partitioning and the number of stable activation patterns (in the extreme case, there is one stable pattern for each instance). Since we will use the cluster size as a proxy for example difficulty, the information gain measure [Qui86] will be used as a splitting criterion, where a decrease of entropy implies more homogeneous partitions.

In fig. 3.2 coloured bullets mark the splitting history of the six instances, following the clustering order from the output layer to the input layer. For example, the first partition is identified by cyan and blue colour, i.e. $\{\{x_2, x_3, x_4, x_5, x_6\},\{x_0, x_1\}\}$. Cluster $\{x_0, x_1\}$ is not split further, because both instances are classified with y_3 label (stable pattern). Conversely, the other cluster is partitioned twice: the first splitting occurs when considering the second layer, as x_2 has a different activation pattern than the others and is classified with a different class; the same occurs at the first layer, this time between x_3 and the other instances. The final partition is the following $\{\{x_0, x_1\}, \{x_4, x_5, x_6\}, \{x_3\}, \{x_2\}\}$. In section 3.4, we will present some preliminary results on how cluster size of the instances partition can be used to estimate example difficulty.

3.3.4 Forgetting Events

Example difficulty metrics define a score for input examples to characterize their "hardness" (see section 2.3.1). In [Ton+19], the authors proposed to estimate example difficulty by counting the *forgetting events*: the number of times an example is correctly predicted and then misclassified again during training. The intuition is that harder examples close to the decision boundary will have more forgetting events than one that is further away.

Definition 4 (Forgetting event [Ton+19]) *Let x be an instance with label k and $pred_e(x)$ the predicted label of x at epoch e. A learning event at epoch e occurs when $pred_{e-1} \neq k$ and $pred_e = k$. A forgetting event at epoch e occurs when $pred_{e-1} = k$ and $pred_e \neq k$. If an instance has no forgetting event during the learning process, is called unforgettable, otherwise is a forgettable instance.*

Algorithm 1 **APD clustering algorithm**. Procedure that defines a partitioning of dataset $\mathcal{D}$ given an APD G.

```
function SPLIT(APD $G = (V, E)$, dataset $\mathcal{D}$, FNN $\mathcal{N}$)
    $n$.pred() $\leftarrow$ predecessors of node $n \in V$
    L $\leftarrow$ # layers of $\mathcal{N}$
    $out$ $\leftarrow$ dummy ending node
    for $v \in V$ s.t. layer(patt($v$)) == $L$ do
        $E$.add(($v, out$))
    end for
    $\mathcal{P} \leftarrow \{(out, \mathcal{D})\})$                         ▷ Current partition
    $\mathcal{F} \leftarrow \emptyset$                                  ▷ Final partition
    while $\mathcal{P} \neq \emptyset$ do
        (n, $\mathcal{C}$) $\leftarrow$ $\mathcal{P}$.pop()               ▷ Extract (current node, cluster)
        if $n$.pred() == $\emptyset \vee |\mathcal{C}|$ == 1 then   ▷ Check if splittable cluster
            $\mathcal{F}$.add($\mathcal{C}$)
            break
        end if
        $\mathcal{S} \leftarrow \emptyset$
        for $v \in n$.pred() do                          ▷ Split current cluster
            $\mathcal{V} \leftarrow \mathcal{AR}$(patt($v$), $\mathcal{C}$)
            $\mathcal{S}$.add(($v, \mathcal{V}$))
        end for
        $\mathcal{S}' \leftarrow \{\mathcal{V} \mid (v, \mathcal{V}) \in \mathcal{S}\}$
        $ig \leftarrow$ InformationGain($\mathcal{C}, \mathcal{S}'$)
        if $ig > 0$ then                                ▷ Check splitting gain
            $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{S}$
        else
            $\mathcal{F}$.add($\mathcal{C}$)
        end if
    end while
    return $\mathcal{F}$
end function
```

Network	32full	32bottl	16full
FC layers	32, 32, 32, 32, 32	32, 16, 12, 10, 8	16, 16, 16, 16, 16
Optimizer	SGD, $1e^{-3}$		
Epochs	500		
Test Accuracy	98.3%	97.2%	95.8%

Table 3.1: **Architecture tested for the results on the APD as auditing tool**. We tested three different architectures to evaluate the APD on the MNIST [LCB10] dataset: 32full, 32full and 16full. We chosed different widths to evaluate the effect on the dataset partitioning.

3.4 Results

Clustering a dataset using the APD clustering algorithm defined in section 3.3.3 partitions the instances in sets of different sizes. The size of the set will then be used as a proxy for the difficulty of its elements: easy instances should belong to large clusters, while outliers and misclassified instances are expected to end up in smaller clusters.

We will present the following results on the APD and its clustering algorithm:

1. section 3.4.1: experimental setting with the MNIST [LCB10] datasets where we use the APD clustering algorithm to rank test instances.

2. section 3.4.2: analysis on the evolution of the APD during training.

3. section 3.4.3: visualizing the APD as a Sankey diagram for model diagnosis.

3.4.1 APD as an Auditing Tool

We applied the clustering algorithm discussed in section 3.3.3 on the MNIST [LCB10] dataset and tested it on different architectures. We will show that misclassified and difficult instances belong to smaller clusters, confirming that the ranking defined by the APD clustering is a valid example difficulty estimator. The results of figs. 3.3 to 3.5 were obtained on the MNIST [LCB10] dataset set with the architectures listed in table 3.1.

Higher Cluster Size Corresponds to Less Forgetting Events

In fig. 3.3 we reported the clusters size distribution, by architecture type. The majority of the clusters are small (average size $\approx 4, 3, 5$ for 32full, 16full and 32bottl, respectively), while even very large clusters (containing up to 2000 instances) are observed for all architectures.

To validate the goodness of the APD clustering, we estimated example difficulty using another metric: the number of forgetting events [Ton+19] (introduced in section 2.3.1). In fig. 3.3 (right) we display the average number of forgetting events with respect to (log-binned) cluster size. Also, the average number of forgetting events decreases with the cluster size, confirming our hypothesis that challenging instances (more forgetting events) belong to smaller clusters. This trend is also confirmed by looking at the cumulative distributions of forgetting events in fig. 3.4.

Misclassified Examples are Concentrated in Small Clusters

The cluster size is further shown to be important in our results on predicting misclassified instances. In fig. 3.5, we present the distribution of cluster sizes for correctly and wrongly classified instances. Across all architectures, we see that wrongly classified instances are more likely to belong to very small clusters, often singletons, while correctly classified instances are more likely to be in larger clusters with greater variance in size. This trend is also depicted in fig. 3.4, which shows the cumulative distribution of misclassified instances by increasing cluster size. We observe that the majority of misclassified instances are in smaller clusters. These findings suggest correlation between cluster size and the difficulty of the examples.

Using Cluster Representatives for Dataset Reduction

The APD clustering exploits redundancies found by the deep model in the data manifold. Therefore, another test that we can perform is performing dataset compression, by leaving only one example (or prototype) for each cluster. More in detail, we performed the following steps:

1. the MNIST training set was split in 50 000 and 10 000 instances, for training and validation, respectively;
2. the 32full architecture was trained with SGD, a learning rate of 0.01 and early stopping;
3. we performed the APD clustering;
4. two more training sets were defined: one with only one representative for each cluster, and one with the same number of instances, but randomly selected;
5. training was performed again, with the same parameters as (ii).

The results, reported in table 3.2, confirm the validity of our clustering procedure: with randomly selected instances, we decreased the generalization performances by more than 4%, while with selected instances the loss was only of 2%.

Dataset reduction is useful to reduce the computational resources requirements, but also is a proxy for sample complexity: the harder the task to be learned, the less we expect to be able to reduce the dataset size without heavily hurting

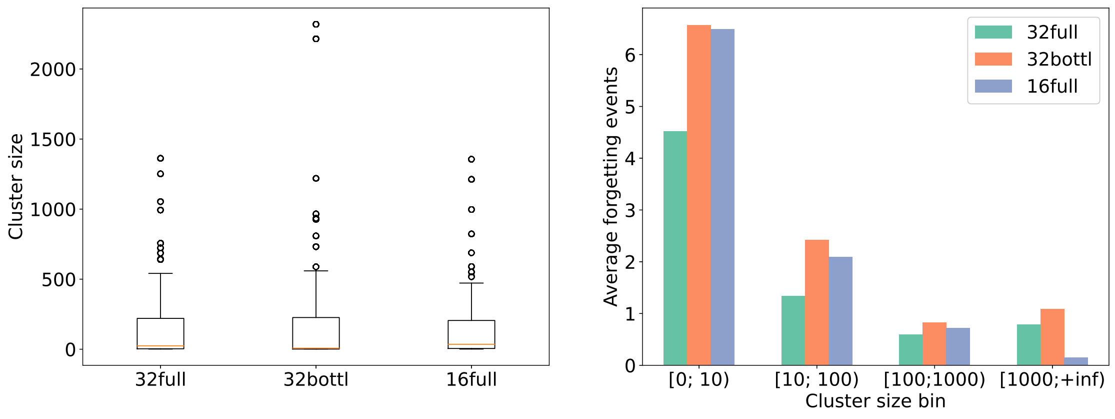


Figure 3.3: **APD cluster size distribution and average number of forgetting events. (Left) Boxplots of cluster size distributions by architecture type.** For all the architectures, the mean is small (average size $\approx 4, 3, 5$ for 32full, 16full and 32bottl, respectively), while there are many clusters with size greater than 500. **(Right) Average number of forgetting events against log-binned cluster size.** As the log-size of the clusters increase, we have a clear decrease in the number of forgetting events. Note that this is an important confirmation of the goodness of the APD clustering as an example difficulty estimation approach.

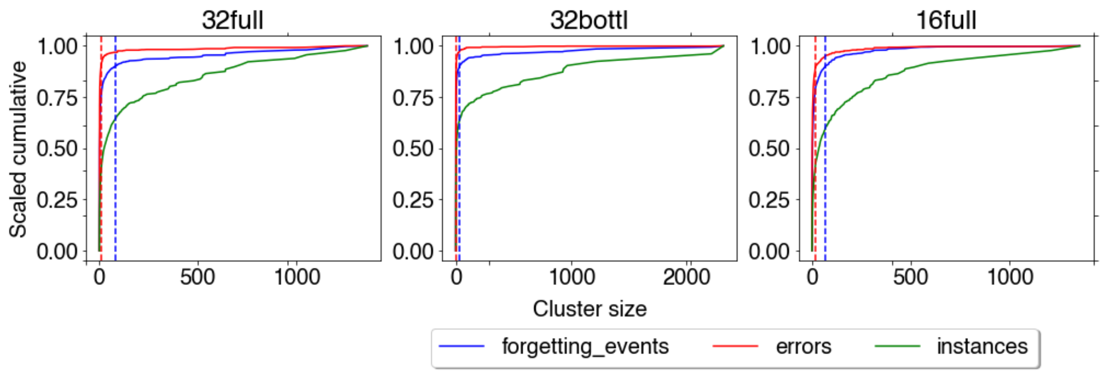


Figure 3.4: **Cumulative distribution of errors and forgetting events, by increasing APD cluster size.** Cumulative distribution of forgetting events (in blue), errors (in red) and number of instances (green), sorted by the respective cluster size and for all the architectures. The vertical dashed lines indicate where 90% of the total is reached.

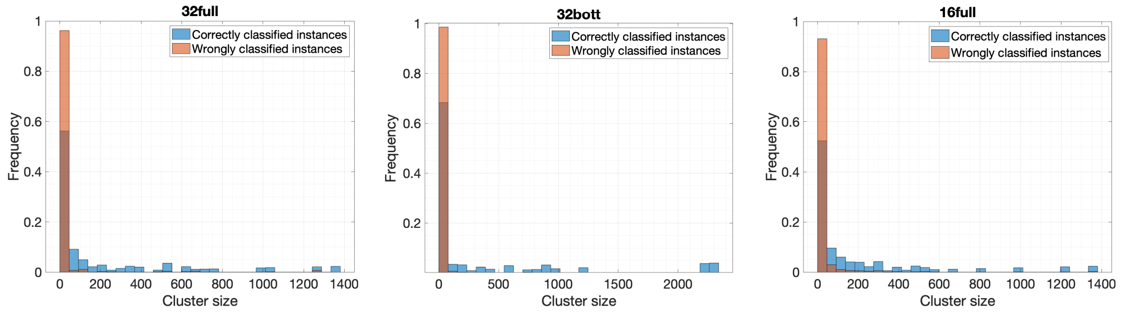


Figure 3.5: **Cluster size distribution for correctly and wrongly classified instances**. For all the architectures, wrongly classified instances (in orange) are concentrated in small clusters.

Trainset selection	Trainset size	Test error
All	50 000 inst.	$96,37 \pm 0,2$
Cluster representatives	8 871 inst.	$94,64 \pm 0,3$
Random	8 871 inst.	$92,26 \pm 0,4$

Table 3.2: **Using the APD for dataset reduction on the MNIST dataset**. The table reports the mean test error and standard deviation of 5 model initializations for 3 possible training sets: (i) the whole MNIST training set, (ii) one representative for each cluster and (iii) a random selection, with the same cardinality of the representative selection.

the generalization performance. With regard to the APD clustering, complex tasks should be characterized by mostly singletons or small clusters. As a future development, it would be interesting to check the sample complexity of different tasks using the APD clustering approach, from the easier MNIST [LCB10] to the harder ImageNet [Den+09].

3.4.2 APD Evolution During Training

In the previous section, we considered the APD after training the model for a fixed amount of epochs. In this section, we will show some results about what happens to activation patterns during training. In particular, the results were obtained with a FNN with $L = 3$ layers with 40 neurons each, trained on the MNIST [LCB10] dataset with SGD and fixed learning rate at 0.001.

In fig. 3.6 (left) we plotted the loss (90 : 10 train/validation split of the 60,000 total instances). In fig. 3.6 (right) we have the evolution of the number of unique activation patterns, i.e. $|A_l(\mathcal{D})|$ for $l \in 1, \dots 3$, where $\mathcal{D}$ is the training set. Interestingly, the number of unique patterns at each epoch decreases with the layer's depth. This justifies starting from the last layer when clustering the APD, given that the raw number of patterns monotonically decreases from the first layer to the last. Moreover, the number of activation patterns of each layer is always far below the theoretical upper bound of 2^{40} possible patterns (refer to [HR19b] for further details) and less than the 54,000 training instances, thus activation patterns are redundant and shared between instances.

In fig. 3.7, we analysed the evolution of the clusters induced by activation patterns during training. We first clustered instances based on the pattern of both the second and last layer, i.e., if x_0, x_1 belong to the same cluster, then $A_2(x_0) = A_2(x_1)$ and $A_3(x_0) = A_3(x_1)$. Note that such clusters correspond to paths from the second to third layer in fig. 3.2. Then, we investigated that the distribution of all (second row) or wrongly classified (first row) instances among the clusters with regard to two measures: purity (proportion of instances of the most frequently predicted class in the cluster) and cluster size. We can observe that there are a number of instances belonging to clusters with high purity and high strength (bottom right of the heatmaps in the second row), that are almost always correct from epoch 150, while wrongly classified instances usually belong to small clusters or clusters with low purity (bottom left of each heatmap).

3.4.3 APD as a Visualization Tool

Until now, we used the APD as a data structure of activation patterns, but here we show its value also as a visualization tool. In Figure 3.8, we plotted APDs for 500 instances of label "1" using a model at epochs 10, 50, 150, and 300. These plots, called Sankey diagrams from the Plotly library [Plo15], show blue rectangles representing the activation patterns of the layers and predicted labels (from left to right), with the height and colour intensity proportional to the number of instances activating or predicting each pattern or label. The edges between the rectangles are coloured according to the proportion of instances that were misclassified, and sized according to the number of instances following that edge. To further clarify the diagram, the thin red edges correspond to the singletons of misclassified instances that we evaluated in the previous results. Since the last layer of the Sankey represents the predicted label, and the correctly classified instances belong to the large blue cluster at the top, the red edges that contain misclassified instances belong to different clusters.

From fig. 3.8 we can observe that activation patterns are shared more in deeper layers, as emerges from fig. 3.6 (large-sized blue rectangles and green edges). In particular, from epoch 150, large clusters appear in the last layer (the penultimate in the diagrams, since the last is the predicted label). Lastly, wrongly classified instances mostly belong to small clusters (the thin lines in red).

Although this represents a small study on using the APD as a visualization tool, we consider the flow diagram an interesting approach to interpret the transformations applied by piecewise linear DNNs.

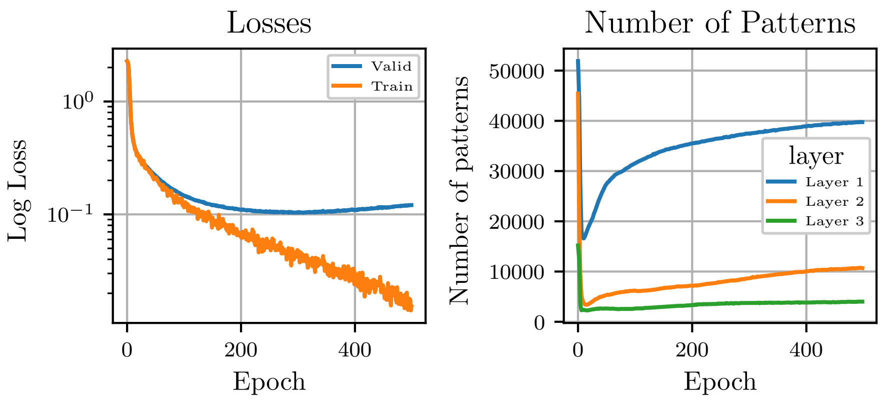


Figure 3.6: **Evolution of activation patterns during training**. **(Left) Train (orange) and validation (blue) loss.** The best validation is achieved at epoch 309. **(Right) Evolution of activation patterns.** Number of unique activation patterns per layer, i.e., $|A_l^*(\mathcal{D})|$, with regard to the network at a given epoch. All the layers steadily increase the number of activation patterns during training. Furthermore, the first layer (blue) has 4 times the number of activation patterns of the second layer, while the third layer (green) doesn't reach more that 4000 activation patterns in all the 500 epochs.

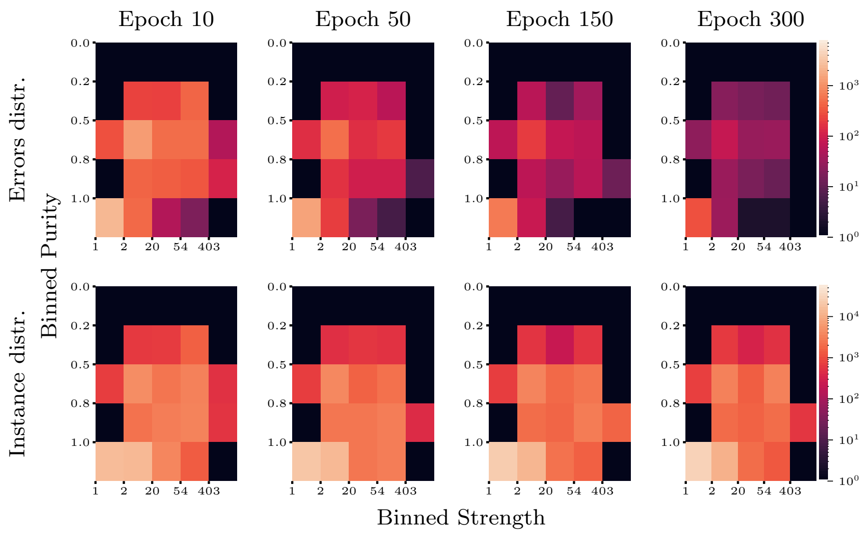
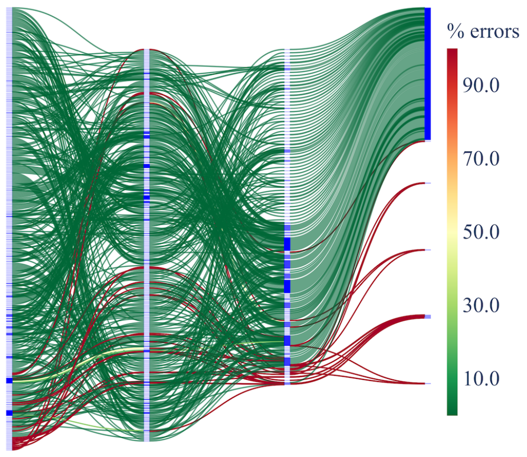
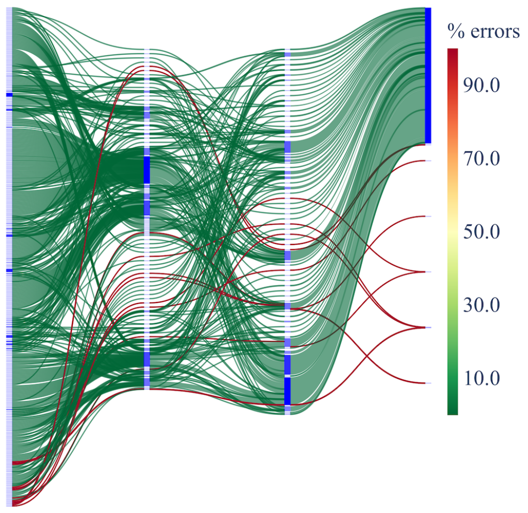


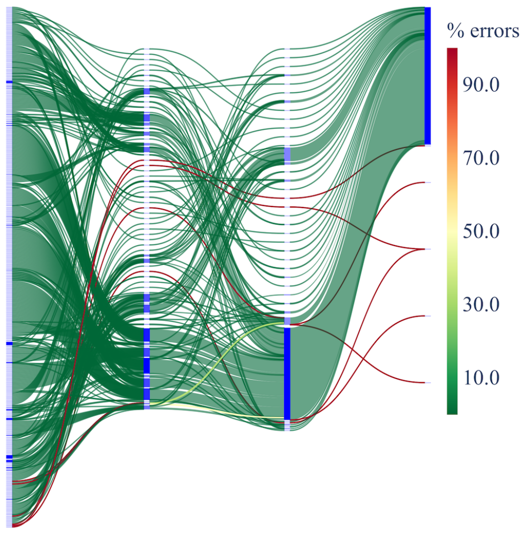
Figure 3.7: **Distribution of cardinalities and misclassified instances of APD clusters at different epochs**. Distribution of wrongly classified instances (first row) and all instances (second row) among clusters defined by activation patterns of second and third layer. Clusters correspond to the edges in the APD between the activation patterns of the second and third layer, respectively. In the x-axis we have the cardinality of the clusters, with log-bins, while in the y-axis we have the purity of the cluster, that is, the proportion of instances belonging to the most frequently predicted label in the cluster.



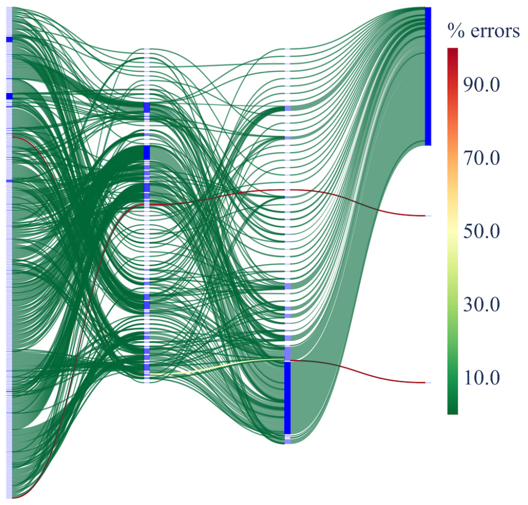
(a) Epoch 10.



(b) Epoch 50.



(c) Epoch 150.



(d) Epoch 300.

Figure 3.8: **APD visualization through Sankey diagrams**. Four APDs for the same 500 instances of label "1" that were learned by the model at different epochs (10, 50, 150, and 300). Each APD is made up of four levels of blue nodes, which represent the activation patterns of the three layers of the network and the predicted labels for each instance. The height and color intensity of the nodes indicate the number of instances they represent. As expected from fig. 3.6, the first layer typically has more patterns than the other layers. In the level for predicted labels, there is a tall node on top that represents the most frequently predicted label, which is "1" in this case. The edges between nodes are sized based on the number of instances they represent and are colored based on the proportion of errors.

3.5 Conclusions

In this chapter, we introduced the Activation Pattern DAG (APD), a novel data structure that summarizes the seen activation patterns given a piecewise linear DNN and a dataset. More in detail, activation patterns are obtained by binarizing the output of ReLU activation functions: a "1" corresponds to a positive output, "0" otherwise. In this way, given an input instance, each layer defines its own activation patterns.

Previous studies observed that overlapping activation patterns between different inputs represent how the model exploits redundancies of the learned function [Mon+14]. We continued on that line, by clustering input instances based on activation patterns, encoded in the APD. The APD allows us to consider not only overlapping patterns in a single layer, but also how input instances overlap in multiple layers. The clustering algorithm that we defined is inspired by decision trees, and has the objective of finding groups of instances with the same patterns in multiple layers and the same predicted label.

After applying the APD clustering algorithm on a given dataset, we claimed that the size of each cluster is a valid scoring to estimate the difficulty of its elements. Accordingly, we propose a connection between activation patterns and example difficulty estimators, introduced in section 2.3.1. The intuition is that large clusters contain well-represented instances that the model assigns to the same activation patterns based on their similarity, while anomalous or misclassified instances should belong to small clusters or singletons. In section 3.4, we validated our hypothesis on the MNIST [LCB10] dataset, considering multiple model architectures. The results confirmed that the cluster size allows distinguishing misclassified from correctly predicted examples. Moreover, cluster size is a good predictor of another example difficulty metric: the number of forgetting events [Ton+19], further confirming our claims.

In addition to example difficulty, in section 3.4.1 we showed how the APD clustering can be used for dataset compression. In particular, by taking only one representative for each cluster, i.e., a prototype, we obtained better generalization performances than a random selection of prototypes. Furthermore, in section 3.4.2 we investigated the evolution of activation patterns during training, validating our design choices for the clustering algorithm.

Given the efficacy of APD clustering in distinguishing well-represented from anomalous instances, it can be employed for automated dataset auditing, in particular for the modern large-scale datasets, but also in safety critical settings, as an alternative to uncertainty estimation, in the context of human-in-the-loop pipelines [Lei+17].

Lastly, in section 3.4.3, we proposed to employ the APD as a visualization tool to diagnose the behaviour of the model on a group of instances. Given the importance of data visualization, in particular for black-box models like DNNs [OMS17; Bau+17], we consider the APD an interesting novel perspective to better understand the behaviour DNNs.

Among the many possible lines of research stemming from the APD, one could study the effect of different hyperparameters on the clustering, such as regularizers [Sri+14; IS15]. Moreover, it would be interesting to extend the APD to other types of models, such as CNNs or Recurrent Neural Network (RNN).

Chapter 4

Adversarial Examples Detection with Ensemble Approaches

Contribution. In this chapter I will discuss the work presented in the following article:

[Cra+21] Francesco Craighero et al. "Unity Is Strength: Improving the Detection of Adversarial Examples with Ensemble Approaches". Preprint (under review).

Summary. In this chapter, we present a framework called ENsemble Adversarial Detector (ENAD) for detecting adversarial examples in Convolutional Neural Networks. ENAD combines the scoring functions of multiple state-of-the-art detectors based on Mahalanobis distance, Local Intrinsic Dimensionality (LID), and One-Class Support Vector Machines (OCSVMs), which analyze the hidden features of Deep Neural Networks (DNNs). Extensive testing on various datasets, models, and adversarial attacks shows that ENAD performs better than competing methods in most cases. ENAD is also highly standardized and reproducible, and its flexible design allows for the easy integration of additional detectors and strategies.

Implementation. The experiments performed in [Cra+21] has been open-sourced on a Github repository[a].

[a] `https://github.com/BIMIB-DISCo/ENAD-experiments`

4.1 Introduction

Recent studies have shown that state-of-the-art DNNs for object recognition tasks are vulnerable to *adversarial examples* [Sze+14; GSS15]. For instance, in the field of computer vision, adversarial examples are perturbed images that are misclassified by a given DNN, even if being almost indistinguishable from the original (and correctly classified) image. Adversarial examples have been investigated in many additional real-world applications and settings, including malware detection [Kol+18] and speech recognition [Qin+19].

Thus, understanding and countering adversarial examples has become a crucial challenge for the widespread adoption of DNNs in safety-critical settings, and resulted in the development of an ever-growing number of *defensive techniques*. Among the possible countermeasures, some aims at increasing the robustness of the DNN model during the training phase, via *adversarial training* [Sze+14; GSS15], *defensive distillation* [Pap+16] or by training more robust models [AG17; BBS17; Str+18] (sometimes referred to as *proactive* methods). Alternative approaches aim at *detecting* adversarial examples in the test phase, by defining specific functions for their detection and filtering-out (*reactive* methods).

In this chapter, we introduce a novel *ensemble approach* for the detection of adversarial examples, named ENsemble Adversarial Detector (ENAD), which integrates scoring functions computed from multiple detectors that process the

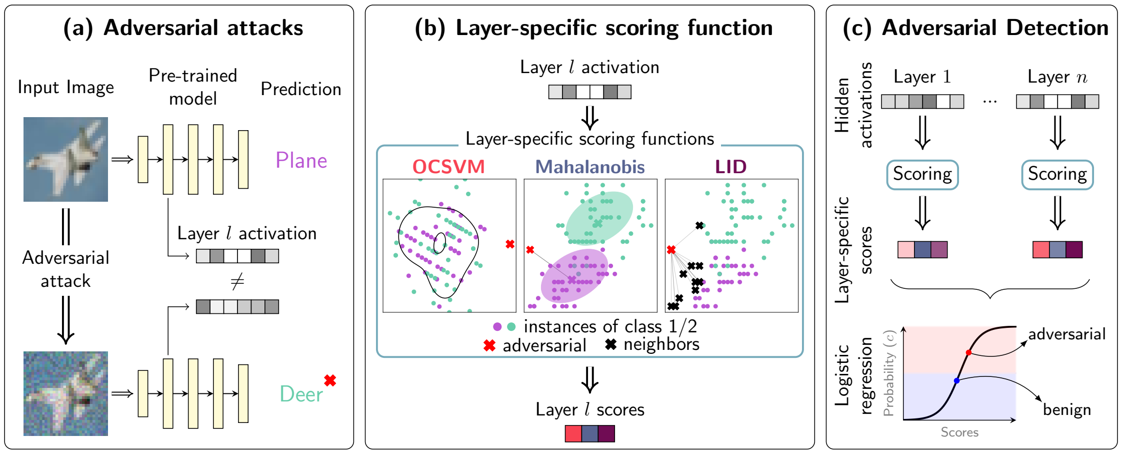


Figure 4.1: **ENAD framework for adversarial detection**. A schematic depiction of the ENAD framework is displayed. **(a)** Given an input image, which can be either benign or adversarial, and a pre-trained deep neural network, the activations of the hidden layers are extracted. **(b)** In order to measure the distance of the image with respect to training examples, layer-specific scores are computed via functions based either on One-Class Support Vector Machine, Mahalanobis distance [Lee+18] or Local Intrinsic Dimensionality [Ma+18]. **(c)** In the current implementation, layer- and detector-specific scores are integrated via logistic regression, so to classify the image as benign or adversarial, with a confidence c.

hidden layer activation of pre-trained DNNs. The underlying rationale is that, given the high-dimensionality of the hidden layers of Convolutional Neural Networks (CNNs) and the difficulty of the adversarial detection problem, different algorithmic strategies might be effective in capturing and exploiting distinct properties of adversarial examples. Accordingly, their combination might allow us to better tackle the classical trade-off between generalization and overfitting, while outperforming single detectors, as already suggested in [AS17], in the distinct context of outlier identification.

To the best of our knowledge, this is the first time that an ensemble approach is applied to the hidden features of DNNs for adversarial detection. However, ensemble-based proactive defences have been previously introduced in [AG17; BBS17; Str+18]. Recently, two ensembling methods for adversarial [Var+21], which focuses on feature attributions rather than the latent features, and out-of-distribution detection [Kau+21] have been proposed. We will compare the former approach, named ExAD, with ENAD in section 4.4.1.

In detail, ENAD includes two state-of-the-art detectors, based on Mahalanobis distance [Lee+18] and Local Intrinsic Dimensionality (LID) [Ma+18], and a newly developed detector based on One-Class Support Vector Machine (OCSVM) [Sch+99]. OCSVMs were previously adopted for adversarial detection in [Ma+19], but we extended the previous method by defining both a pre-processing step and a Bayesian hyperparameter optimization strategy, both of which result fundamental to achieve performances comparable to [Lee+18; Ma+18]. In the current implementation, the output of each detector is then integrated via a logistic regression that returns both the adversarial classification and the overall confidence of the prediction. A schematic depiction of the ENAD framework is provided in fig. 4.1.

For the sake of reproducibility, the performance of ENAD and competing methods was assessed with the extensive setting originally proposed in [Lee+18]. In particular, we performed experiments with two models, namely ResNet [He+16] and DenseNet [Hua+17], trained on CIFAR-10 [Kri09], CIFAR-100 [Kri09] and SVHN [Net+11], and considered four benchmark adversarial attacks, i.e., FGSM [GSS15], BIM [KGB17], DeepFool [MFF16] and CW [CW17b]. In the various tests, ENAD was compared against its constituting standalone detectors (section 4.4.1) and other ensemble strategies (section 4.4.1).

We executed an additional array of experiments aimed at assessing the performance of the distinct detectors when trained on a given attack and tested against others (*transfer attacks*).

Main Contributions The main contributions of this work can be summarized as follows.

- *Improvement of OCSVM performance in adversarial detection*: we introduced an evolved version of the OCSVM strategy for adversarial detection (first described in [Ma+19]), by designing a new pipeline with Bayesian hyperparameter optimization and data preprocessing on top of the default training process. Results highlight performance improvements.

- *Assessment of layer- and attack-specificity of standalone detectors*: thanks to extensive tests on benchmark datasets, models and attacks, we show that the performance of state-of-the-art standalone detectors is layer- and attack-specific, possibly guiding the improvements of such methods. Importantly, we demonstrate that the predictions of different detectors are scarcely overlapping.

- *Introduction of the ENAD ensemble framework for adversarial detection*: we propose a new detector, named ENAD, which integrates layer-specific scoring functions from multiple independent detectors. Its performance is assessed against standalone detectors, different ensemble strategies and other integration schemes (e.g., voting) in a variety of experimental settings. The framework is described by clearly stating all design choices, paving the way to future extensions with different detectors and integration schemes.

- *Performance evaluation in attack transfer settings*: we present a quantitative evaluation of the performance ENAD and competing methods in transfer attack settings. We show the limitations of existing strategies and provide guidelines for future research.

- *Visualization of adversarial examples in low-dimensional space*: we deliver an easy-to-interpret way of visualizing adversarial examples on a low-dimensional projection of the score space, which provides a proxy of their "hardness", and may be particularly useful in real-world applications.

4.2 Background

A survey of OOD and adversarial examples detectors is available in section 2.3.2. In section 4.2.1, we will detail adversarial attacks, that is, methods that generate adversarial examples.

4.2.1 Adversarial Attacks

In the following, we describe the adversarial attacks that were employed in our experiments, namely FGSM [GSS15], BIM [KGB17], DeepFool [MFF16] and CW [CW17a]:

- FGSM [GSS15] defines optimal L_∞ constrained perturbations as:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \cdot \operatorname{sign}\left(\boldsymbol{\nabla}_{\boldsymbol{x}} J(\boldsymbol{x}, t)\right),$$

  such that ϵ is the minimal perturbation in the direction of the gradient with respect to the input image ($\boldsymbol{\nabla}_{\boldsymbol{x}}$) that changes the prediction of the model from the true class y to the target class t.

- BIM [KGB17] extends FGSM by applying it k times with a fixed step size α, while also ensuring that each perturbation remains in the ϵ-neighbourhood of the original image x by using a per-pixel clipping function clip:

$$\begin{aligned}&\tilde{\boldsymbol{x}}_{\boldsymbol{0}} = \boldsymbol{x}\\ &\tilde{\boldsymbol{x}}_{\boldsymbol{n+1}} = \operatorname{clip}_{\boldsymbol{x},\epsilon}\left(\boldsymbol{x_n} + \alpha \cdot \operatorname{sign}\left(\boldsymbol{\nabla}_{\tilde{\boldsymbol{x}}_{\boldsymbol{n}}} J(\tilde{\boldsymbol{x}}_{\boldsymbol{n}}, t)\right)\right)\end{aligned}$$

- DeepFool [MFF16] iteratively finds the optimal L_2 perturbations that are sufficient to change the target class by approximating the original non-linear classifier with a linear one. Thanks to the linearization, in the binary classification setting the optimal perturbation corresponds to the distance to the (approximated) separating hyperplane, while in the multiclass the same idea is extended to a one-vs-all scheme. In practice, at each i-th step the method computes the optimal perturbation p_i of the simplified problem, until $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \sum_i p_i$ is misclassified.

- the CW attack [CW17a], in two variants:

  - the original L_2 norm attack, that we will refer as CW: this variant uses gradient descent to minimize $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 + c \cdot l_{cw}(\tilde{\boldsymbol{x}})$, where the loss l_{cw} is defined as:

$$l_{cw}(\boldsymbol{x}) = \max\left(\max\{\sigma_{\text{lgt}}(\tilde{\boldsymbol{x}})^i : i \neq t\} - \sigma_{\text{lgt}}(\tilde{\boldsymbol{x}})^t, -\kappa\right).$$

    The objective of the optimization is to minimize the L_2 norm of the perturbation and to maximize the difference between the target logit $\sigma_{\text{lgt}}(\tilde{\boldsymbol{x}})^t$ and the one of the next most likely class up to real-valued constant κ, that models the desired confidence of the crafted adversarial.

  - the L_∞ norm variant defined in [ACW18], that we will refer to as CW$_\infty$, since it employs the same l_{cw} loss. In this variant, we optimize for l_{cw}, while clipping the adversarial such that $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^\infty = \epsilon$, with ϵ given as a parameter. Furthermore, we use the constant κ to obtain high-confidence adversarial examples.

4.3 Methods

In this section, we illustrate the ENAD ensemble approach for adversarial detection, as well as the properties of the scoring functions it integrates, respectively based on OCSVM (Detector A–*new*), Mahalanobis (Detector B) and LID (Detector C), which can also be used as standalone detectors. We also describe the background of both adversarial examples generation and detection, the partitioning of the input data and the extraction of the features processed by ENAD and standalone detectors.

4.3.1 Data partitioning

Let $\mathcal{X}^{train}$ be the training set on which the classifier $\mathcal{N}$ was trained, $\mathcal{X}^{test}$ the test set and $\mathcal{L}_{norm} \subseteq \mathcal{X}^{test}$ the set of correctly classified test instances. Following the setup done in [Lee+18; Ma+18], from $\mathcal{L}_{norm}$ we generate (i) a set of noisy examples $\mathcal{L}_{noisy}$ by adding random Gaussian noise, with the additional constraint of being correctly classified, and (ii) a set of adversarial examples $\mathcal{L}_{adv}$ generated via a given attack. We also ensure that $\mathcal{L}_{norm}$, $\mathcal{L}_{noisy}$ and $\mathcal{L}_{adv}$ have the same size. The set $\mathcal{L} = \mathcal{L}_{norm} \cup \mathcal{L}_{noisy} \cup \mathcal{L}_{adv}$ will be our *labelled dataset*, where the label is adv for adversarial examples and $\overline{adv}$ for benign ones. As detailed in the following sections, $\mathcal{L}$ will be split into a training set $\mathcal{L}^{train}$, a validation set $\mathcal{L}^{valid}$ for hyperparameter tuning and a test set $\mathcal{L}^{test}$ for the final evaluation.

4.3.2 Feature Extraction

In our experimental setting, $h_l(\boldsymbol{x})$, with $l \in [1, \ldots, L]$, corresponds to either the first convolutional layer or to the output of the l^{th} dense (residual) block of a DNN (e.g. DenseNet or ResNet). As proposed in [Lee+18], the size of the feature map is reduced via average pooling, so that $h_l(\boldsymbol{x})$ has a number of features equal to the number of channels of the l^{th} layer. Detectors A, B, and C and ENAD are applied to such set of features, as detailed in the following.

4.3.3 Standalone Detectors

Detector A: OCSVM

This newly designed detector is based on a standard anomaly detection technique called One-Class Support Vector Machine (OCSVM) [Sch+99], which belongs to the family of one-class classifiers [Tax01]. One-class classification is a problem in which the classifier aims at learning a good description of the training set and then rejects the inputs that do not resemble the data it was trained on, which represent outliers or anomalies. This kind of classifier is usually adopted when only one class is sufficiently represented within the training set, while the others are undersampled or hard to be characterized, as in the case of adversarial examples, or anomalies in general. OCSVM was first employed for adversarial detection in [Ma+19]. Here, we modified it by defining an input pre-processing step based on PCA-whitening [KLS18], and by employing a Bayesian optimization technique [SLA12] for hyperparameter tuning. The pseudocode is reported in algorithm 2.

Preprocessing OCSVM employs a kernel function (in our case a Gaussian RBF kernel) that computes the Euclidean distance among data points. Hence, it might be sound to standardize all the features of the data points, at the preprocessing stage, to make them equally important. To this end, each hidden layer activation

Algorithm 2 OCSVM detector (see the main text for an explanation of the notation employed)

```
Input: Act. $h_l$ of layer $l$, trainset $\mathcal{X}^{train}$, labelled set $\mathcal{L}$
 1: for each $l$ in $1, \dots, L$ do
 2:     Centering and PCA-whitening of $h_l$: $h_l^*$
 3:     Select best layer-specific parameters $\theta = \{\nu, \gamma\}$
 4:     Fit $\mathsf{OCSVM}_l(\theta)$ on $\{h_l^*(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}^{train}\}$
 5:     $\mathsf{OCSVM}_l(\theta)$ decision function: $\mathrm{O}_l$
 6:     Layer $l$ score of $\boldsymbol{x_0}$: $\mathrm{O}_l(\boldsymbol{x_0})$
 7: end for
 8: Scores vector: $\mathbf{O}(\boldsymbol{x_0}) := [\mathrm{O}_1(\boldsymbol{x_0}), \dots, \mathrm{O}_L(\boldsymbol{x_0})]$
 9: Fit $adv$ posterior on $\mathcal{L}^{train}$: $p(adv \mid \mathbf{O}(\boldsymbol{x_0}))$
10: OCSVM of $\boldsymbol{x_0}$: $\mathsf{OCSVM}(\boldsymbol{x_0}) := p(adv \mid \mathbf{O}(\boldsymbol{x_0}))$
11: return OCSVM
```

$h_l(\boldsymbol{x_0})$ is first centered on the mean activations $\mu_{l,c}$ of the examples of the training set $\mathcal{X}^{train}$ of class c. Then, PCA-whitening $\mathbf{W}_l^{PCA}$ is applied:

$$\begin{aligned} h_l^*(\boldsymbol{x_0}) &= \mathbf{W}_l^{PCA} \cdot (h_l(\boldsymbol{x_0}) - \mu_{l,c}) \\ &= \boldsymbol{\Lambda}_l^{-1/2} \cdot \boldsymbol{U}_l^T \cdot (h_l(\boldsymbol{x_0}) - \mu_{l,c}), \end{aligned}$$

where $\boldsymbol{U}_l^T$ is the eigenmatrix of the covariance $\boldsymbol{\Sigma}_l$ of activations h_l and $\boldsymbol{\Lambda}_l$ is the eigenvalues matrix of the examples of $\mathcal{X}^{train}$. Whitening is a commonly used preprocessing technique for outlier detection, since it enhances the separation of points that deviate in low-variance directions [Agg20]. Moreover, in [KK20] it was conjectured that the effectiveness of the Mahalanobis distance [Lee+18] for out-of-distribution and adversarial detection is due to the strong contribution of low-variance directions. Thus, this preprocessing step allows the one-class classifier to achieve better overall performances[1].

Layer-specific scoring function After preprocessing, OCSVM with a Gaussian RBF kernel is trained on the hidden layer activations h_l of layer l of the training set $\mathcal{X}^{train}$. Once the model has been fitted, for each instance $\boldsymbol{x_0}$ layer-specific scores $\mathbf{O}(\boldsymbol{x_0}) = [\mathrm{O}_1(\boldsymbol{x_0}), \mathrm{O}_2(\boldsymbol{x_0}), \dots, \mathrm{O}_L(\boldsymbol{x_0})]$ are evaluated. More in detail, let $\mathcal{S}_l$ be the set of support vectors, the decision function $\mathrm{O}_l(\boldsymbol{x_0})$ for the l^{th} layer is computed as:

$$\mathrm{O}_l(\boldsymbol{x_0}) = \sum_{sv \in \mathcal{S}_l} \alpha_{sv} k(h_l(\boldsymbol{x_0}), sv) - \rho, \tag{4.1}$$

[1]In a test on the DenseNet, CIFAR-10, CW scenario, the AUROC returned by the OCSVM detector with PCA-whitening preprocessing improves from 82.56 to 90.24, and the AUPR from 78.17 to 82.98, with respect to the same method without preprocessing (see section 4.4 for further details).

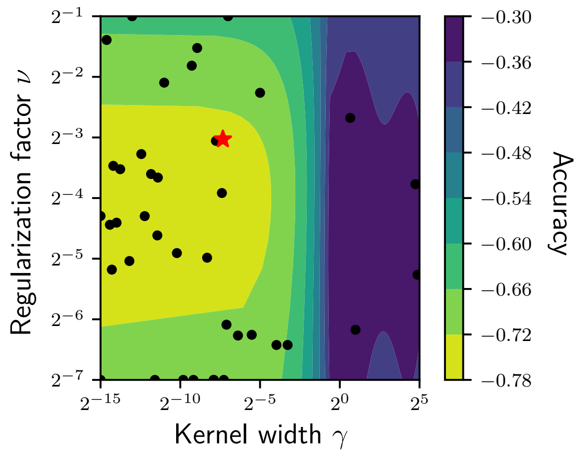


Figure 4.2: **OCSVM hyperparameter optimization**. The influence of different combinations of OCSVM hyperparameters $\{\nu, \gamma\}$ on the validation accuracy is explored via Bayesian optimization [Hea+], in the example scenario of DenseNet model, CIFAR-10 dataset and DeepFool attack. The gradient returns the validation accuracy estimated on $\mathcal{L}^{valid}$. The red star represents the optimal configuration, which is then employed for adversarial detection in both the OCSVM and the ENAD detectors.

where α_{sv} is the coefficient of the support vector sv in the decision function, ρ is the intercept of the decision function and k is a Gaussian RBF kernel with kernel width γ:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\big(-\gamma \|\boldsymbol{x} - \boldsymbol{y}\|^2\big). \tag{4.2}$$

Hyperparameter optimization The layer-specific scoring function takes two parameters as input: the regularization factor $\nu \in (0, 1)$ that represents an upper bound on the fraction of training errors (controlling for overfitting), and the kernel width γ. This hyperparameters must be carefully chosen to achieve good performances. For this purpose, many approaches have been proposed for hyperparameters selection in OCSVM [Ala+20]. In our setting, we used the validation set of labelled examples $\mathcal{L}^{valid}$ to choose the best combination of parameters, based on the validation accuracy. To avoid a full (and infeasible) exploration of the parameters space, we employed Bayesian hyperparameter optimization, via the `scikit-optimize` library [Hea+]. In fig. 4.2, we report the estimated accuracy of the explored solutions in the specific case of the OCSVM detector, in a representative experimental setting.

Adversarial detection Once the scores have been obtained for each layer, they can serve either as input for the standalone Detector A or as partial input of the ensemble detector ENAD. In the former case, in order to aggregate the scores of the separate layers $\mathbf{O}(\boldsymbol{x_0})$, this detector employs a logistic regression to model the posterior probability of adversarial (adv) examples:

$$p(adv \mid \mathbf{O}(\boldsymbol{x_0})) = \Big(1 + \exp\big(\beta_0 + \boldsymbol{\beta}^T \mathbf{O}(\boldsymbol{x_0})\big)\Big)^{-1}, \tag{4.3}$$

The parameters $\{\beta_0, \boldsymbol{\beta}\}$ are fitted with a cross-validated procedure using the labelled training set $\mathcal{L}^{train}$.

Detector B: Mahalanobis

The Mahalanobis detector (Maha) was originally introduced in [Lee+18]. The algorithmic procedure is akin to that of the OCSVM detector, and includes a final layer score aggregation step via logistic regression, but it is based on a different layer-specific scoring function.

Layer-specific scoring function Given a test instance $\boldsymbol{x_0}$, the layer score is computed via a three-step procedure: first, for each instance, the class $\hat{c}$ is selected, such that:

$$\hat{c} = \arg\min_c \mathsf{MahaScore}_l(\boldsymbol{x_0}, c),$$

where $\mathsf{MahaScore}_l(\boldsymbol{x}, c)$ is the Mahalanobis distance for the l^{th} layer between the activations $h_l(\boldsymbol{x})$ and the mean values $\mu_{l,c}$ of the examples in the training set $\mathcal{X}^{train}$:

$$\mathsf{MahaScore}_l(\boldsymbol{x}, c) = (h_l(\boldsymbol{x}) - \mu_{l,c})^T \boldsymbol{\Sigma}_l^{-1} (h_l(\boldsymbol{x}) - \mu_{l,c}), \tag{4.4}$$

where $\boldsymbol{\Sigma}_l$ is the covariance matrix of the examples of $\mathcal{X}^{train}$ in layer l. Then, the instance is preprocessed to obtain a better separation between benign and adversarial examples similar to what is discussed in [LLS18]:

$$\boldsymbol{x_0^*} = \boldsymbol{x_0} - \lambda \,\text{sign}\, \boldsymbol{\nabla}_{\boldsymbol{x_0}} \mathsf{MahaScore}_l(\boldsymbol{x_0}, \hat{c}),$$

where λ is a positive real number, called the perturbation magnitude. The scoring for instance $\boldsymbol{x_0}$ is computed as:

$$\mathbf{M}(\boldsymbol{x_0}) = [\mathrm{M}_1(\boldsymbol{x_0}), \mathrm{M}_2(\boldsymbol{x_0}), \dots, \mathrm{M}_L(\boldsymbol{x_0})],$$

where

$$\mathrm{M}_l(\boldsymbol{x_0}) = -\max_c \; \mathsf{MahaScore}_l(\boldsymbol{x_0^*}, c).$$

Adversarial detection The scores can serve either as input for the standalone Detector B or as partial input for the ensemble detector ENAD. In the latter case, the Mahalanobis detector uses logistic regression to identify adversarial examples, with a procedure similar to that already described for standalone Detector A.

Hyperparameter optimization Differently from Detector A, the hyperparameter selection is performed downstream of the adversarial detection stage. In order to select the best λ (unique for all layers), the method selects the value that achieves the best Area Under the Receiver Operating Characteristic (AUROC) (detailed in section 4.3.5) on $\mathcal{L}^{valid}$ computed on the posterior probability $p(adv \mid \mathbf{M}(\boldsymbol{x_0}))$, which is obtained via the logistic regression fitted on $\mathcal{L}^{train}$.

Detector C: LID

The third detector uses a procedure similar to Detectors A–B, but the layer-specific scoring function is based on the Local Intrinsic Dimensionality (LID) approach [Ma+18].

Layer-specific scoring function Given a test instance $\boldsymbol{x_0}$, the LID layer-specific scoring function L is defined as:

$$\mathrm{L}_l(\boldsymbol{x_0}) = -\left(\frac{1}{k}\sum_{i=1}^{k}\log\frac{r_i(h_l(\boldsymbol{x_0}))}{\max_i r_i(h_l(\boldsymbol{x_0}))}\right)^{-1}, \tag{4.5}$$

where, k is the number of nearest neighbours, r_i is the Euclidean distance to the i-th nearest neighbour in the set of normal examples $\mathcal{L}_{norm}$. The layer-specific scores are:

$$\mathbf{L}(\boldsymbol{x_0}) = [\mathrm{L}_1(\boldsymbol{x_0}), \mathrm{L}_2(\boldsymbol{x_0}), \ldots, \mathrm{L}_L(\boldsymbol{x_0})]$$

Adversarial detection When considered alone, the LID detector employs a logistic regression to identify adversarial examples, similarly to the other detectors (see above).

Hyperparameter optimization Similarly to Detector B, the hyperparameter selection is performed downstream of the adversarial detection stage. k is selected as the value that achieves the best AUROC on $\mathcal{L}^{valid}$ computed on the posterior probability $p(adv \mid \mathbf{L}(\boldsymbol{x_0}))$, which is obtained via the logistic regression fitted on $\mathcal{L}^{train}$. Note that k is unique for all layers.

4.3.4 ENsemble Adversarial Detector (ENAD)

The ENAD approach exploits the effectiveness of Detectors A, B, and C in capturing different properties of data distributions, by explicitly integrating the distinct layer-specific scoring functions in a unique classification framework. More in detail, given a test instance $\boldsymbol{x_0}$, it will be characterized by a set of layer-specific and detector-specific features, computed from the scoring functions defined above, that is: $\mathbf{E}(\boldsymbol{x_0}) = [\mathbf{O}(\boldsymbol{x_0}), \mathbf{M}(\boldsymbol{x_0}), \mathbf{L}(\boldsymbol{x_0})]$. It should be noted that training and hyperparameter optimization is executed for each detector independently.

Algorithm 3 ENAD detector.

```
Input: Act. $h_l$ of layer $l$, trainset $\mathcal{X}^{train}$, labelled set $\mathcal{L}$
1: Select best hyperparameters for OCSVM, Maha, LID
2: for each layer $l$ in $1, \dots, L$ do
3:     Layer $l$ scores of $\boldsymbol{x_0}$: $\mathrm{O}_l(\boldsymbol{x_0}), \mathrm{M}_l(\boldsymbol{x_0}), \mathrm{L}_l(\boldsymbol{x_0})$
4: end for
5: Scores vector: $\mathbf{E}(\boldsymbol{x_0}) \coloneqq [\mathbf{O}(\boldsymbol{x_0}), \mathbf{M}(\boldsymbol{x_0}), \mathbf{L}(\boldsymbol{x_0})]$
6: Fit $adv$ posterior on $\mathcal{L}^{train}$: $p(adv \mid \mathbf{E}(\boldsymbol{x_0}))$
7: ENAD on $\boldsymbol{x_0}$: $\mathsf{ENAD}(\boldsymbol{x_0}) \coloneqq p(adv \mid \mathbf{E}(\boldsymbol{x_0}))$
8: return ENAD
```

Adversarial detection In its current implementation, in order to integrate the scores of the separate layers $\mathbf{E}(\boldsymbol{x_0})$, ENAD employs a simple logistic regression to model the posterior probability of adversarial (adv) examples:

$$p(adv \mid \mathbf{E}(\boldsymbol{x_0})) = \Big(1 + \exp\big(\beta_0 + \boldsymbol{\beta}^T \mathbf{E}(\boldsymbol{x_0})\big)\Big)^{-1}. \tag{4.6}$$

Like Detectors A, B, and C, the logistic is fitted with a cross-validation procedure using the labelled training set $\mathcal{L}^{train}$. Fitting the logistic allows one to have different weights, i.e., the elements of $\boldsymbol{\beta}^T$, for the different layers and detectors, meaning that a given detector might be more effective in isolating an adversarial example when processing its activation on a certain layer of the network. The pseudocode is reported in algorithm 3.

4.3.5 Performance Metrics

Let the positive class be the adversarial examples (adv) and the negative class be the benign examples ($\overline{adv}$). Then, the correctly classified adversarial and benign examples correspond to the True Positive (TP) and True Negative (TN), respectively. Conversely, the wrongly classified adversarial and benign examples are the False Negative (FN) and False Positive (FP), respectively.

To evaluate the detectors' performances, we employed two standard threshold independent metrics, namely AUROC and AUPR [DG06], the Accuracy and the F1-score, defined as follows:

- Area Under the Receiver Operating Characteristic (AUROC): the area under the curve identified by Specificity = $\mathsf{TN}/(\mathsf{TN}+\mathsf{FP})$ and Fall-out = $\mathsf{FP}/(\mathsf{TN}+\mathsf{FP})$.
- Area Under the Precision-Recall curve (AUPR): the area under the curve identified by Precision (Pr) = $\mathsf{TP}/(\mathsf{TP}+\mathsf{FP})$ and Recall (Re) = $\mathsf{TP}/(\mathsf{TP}+\mathsf{FN})$.
- Accuracy = $(\mathsf{TP}+\mathsf{TN})/(\mathsf{TP}+\mathsf{TN}+\mathsf{FP}+\mathsf{FN})$.

- F1-score (F1) = $2 \times (\mathsf{Pr} \times \mathsf{Re})/(\mathsf{Pr} + \mathsf{Re})$.

The AUROC and AUPR were evaluated given the adversarial posterior learned by the logistic function $p(adv \mid X)$, where X is the set of layer-specific scores. The layer-specific scores are computed from $\mathcal{L}^{valid}$ when the AUROC is used for hyperparameters optimization for the Mahalanobis and LID detectors, and from $\mathcal{L}^{test}$ in all the other settings, i.e., the detector's performance evaluation. Moreover, AUROC and AUPR were also used to evaluate the performance of each detector in each layer, by considering the layer-specific scores evaluated on $\mathcal{L}^{test}$ (see, e.g., fig. 4.3b). Note that for the OCSVM and Mahalanobis detectors, lower values correspond to adversarial examples, while the opposite applies to the LID detector. The accuracy was used for the Bayesian hyperparameter selection procedure [SLA12] of the OCSVM detector. Lastly, F1-score, Precision and Recall were used to evaluate ensembling methods in section 4.4.1 and transfer attacks in section 4.4.2.

4.4 Results

Four benchmark adversarial attacks were selected to test the effectiveness of our ENAD approach and competing methods, namely: Fast Gradient Signed Method (FGSM) [GSS15], Basic Iterative Method (BIM) [KGB17], DeepFool [MFF16] and Carlini-Wagner (CW) [CW17b]. In particular, to evaluate the performances in distinct scenarios, we designed two separate arrays of experiments:

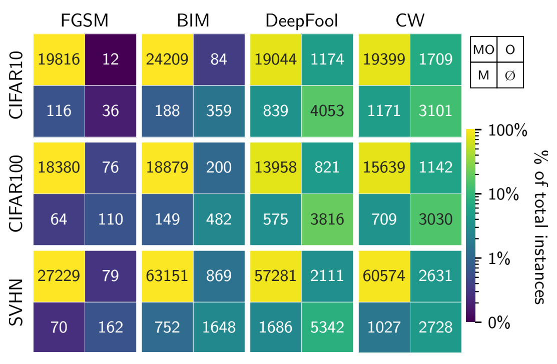
1. *Known attacks* (section 4.4.1): the same adversarial attack is employed both in the training and in the test phase, as proposed in [Lee+18].
2. *Transfer attacks* (also *unknown attacks*, section 4.4.2): a given attack is employed for training and another one for the test phase. In this case, two sub-scenarios were defined:

   a. *cheap training*, i.e., training on the FGSM attack and testing on the other three benchmark attacks.

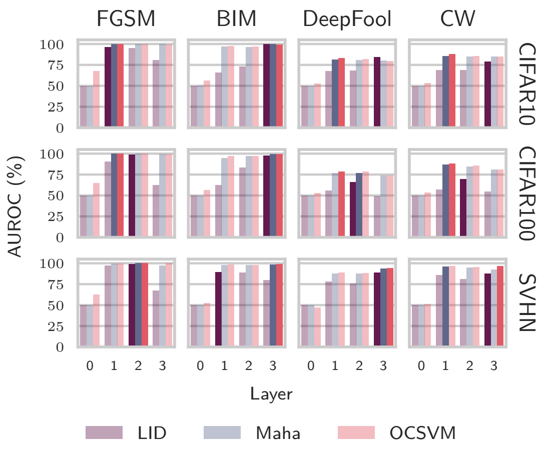
   b. *hard attacks*, i.e., testing against high-confidence adversarial examples with many distortion levels generated using CW_∞.

We compared the performance of ENAD with standalone Detectors A (OCSVM), B (Mahalanobis [Lee+18]), and C (LID [Ma+18]). All four detectors integrate layer-specific anomaly scores via logistic regression and classify any example as adversarial if the posterior probability is > 0.5, benign otherwise (see the Methods for additional details). Moreover, we also evaluated the performance of the ExAD detector [Var+21] and alternative ensembling strategies, i.e., based on voting schemes, in the known attack scenario.

In section 4.4.3 we finally propose a strategy to visualize benign and adversarial examples on a low-dimensional space, based on the similarity of their layer- and detector-specific score profiles, and which may be useful in real-world applications.



(a)



(b)

Figure 4.3: **(a) Comparison of predictions of standalone detectors in the Known Attacks scenario (OCSVM vs. Mahalanobis–DenseNet).** The contingency table shows the number of adversarial examples of the test set $\mathcal{L}^{test}$ correctly identified by: both the OCSVM and the Mahalanobis detectors (MO), either one of the two methods (O or M), none of them ($\emptyset$). The results of the DenseNet model, with respect to the distinct datasets and attacks are shown, whereas the remaining pairwise comparisons are displayed in fig. 4.A.2. **(b) Influence of the hidden layers in adversarial detection in the Known Attacks scenario (DenseNet).** For each configuration of datasets and attacks on the DenseNet model, the AUROC of each layer-specific score for Detectors A–C is returned. For each configuration and detector, the best-performing layer is highlighted with a darker shade (see Methods for further details). The same results for the ResNet model are available in fig. 4.A.1.

The assessment of the computational time of ENAD is discussed in section 4.4.4. Regarding hyperparameter selection, table 4.A.1 contains the hyperparameter configurations and tables 4.A.2 and 4.A.3 contain the best hyperparameters for each setting.

4.4.1 Known Attacks

Standalone Detectors Capture Distinct Properties of Adversarial Examples

In order to assess the ability of standalone Detectors A, B and C to exploit different properties of input instances, we first analysed the methods as standalone, and computed the subsets of adversarial examples identified: (*i*) by all detectors, (*ii*) by a subset of them, (*iii*) by none of them.

In fig. 4.3a, we reported a contingency table in which we compare the OCSVM and Mahalanobis detectors on all the experimental settings with the DenseNet model, while the remaining pairwise comparisons are presented in the fig. 4.A.2. We note that while the class of examples identified by both approaches is, as expected, the most crowded, we observe a substantial number of instances that are identified by either one of the two approaches. This important result appears to be general, as it is confirmed in the other comparisons between standalone detectors.

In addition, in fig. 4.A.3 one can find the layer-specific scores returned by all detectors in a specific setting (ResNet, DeepFool, CIFAR-10). For a significant portion of examples, the ranking ordering among scores is not consistent across detectors, confirming the distinct effectiveness in capturing different data properties in the hidden layers.

To investigate the importance of the layers with respect to the distinct attacks, models and datasets, we also computed the AUROC directly on the layer-specific scores, i.e., the anomaly scores returned by each detector in each layer. In fig. 4.3b, one can find the results for all detectors in all settings, with the DenseNet model (the same results for the ResNet model are available in fig. 4.A.1). For the FGSM attack, the scores computed on the middle layers consistently return the best AUROC in all datasets, while for the BIM attack the last layer is apparently the most important. Notably, with the DeepFool and CW attacks, the most important layers are dataset-specific. This result demonstrates that each attack may be vulnerable in distinct layers of the network.

ENAD Outperforms standalone Detectors

Table 4.1 reports the AUROC and AUPR computed on the fitted adversarial posterior probability for Detectors A, B, and C, respectively, on all 24 experimental settings.

It can be noticed that ENAD exhibits both the best AUROC and the best AUPR in 22 out of 24 settings (including 1 tie with OCSVM), with the greatest improvements emerging in the hardest attacks, i.e. DeepFool and CW. Remarkably, the newly designed OCSVM detector outperforms the other standalone detectors in 12 and 14 settings (out of 24) in terms of AUROC and AUPR, respectively.

Notice that in table 4.A.4, we also evaluated the performance of all pairwise combinations of the three detectors, so to quantitatively investigate the impact of integrating the different algorithmic approaches, proving that distinct ensembles of detectors can be effective in specific experimental settings.

Comparison with voting schemes

ENAD employs a logistic classifier as a meta-learner to aggregate the scoring of every detector in each layer, although other ensembling strategies may be employed, e.g., *voting schemes*.

In this section, we compare ENAD with a voting scheme among the predictions of standalone Detectors A, B, C. In detail, we considered three voting schemes: Or,

		Attack	FGSM		BIM		DeepFool		CW	
Model	*Dataset*	*Detector*	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
DenseNet	CIFAR-10	LID	96.37	98.30	99.52	99.73	75.21	85.22	69.75	80.88
		Maha	**99.80**	**99.96**	99.46	99.75	74.46	82.73	78.43	87.42
		OCSVM	99.58	99.88	99.24	99.69	77.01	84.74	82.98	90.24
		ENAD	99.70	99.89	**99.90**	**99.96**	**83.70**	**89.36**	**85.30**	**91.50**
	CIFAR-100	LID	98.39	99.29	96.31	98.11	55.88	70.12	59.67	72.80
		Maha	99.49	99.87	97.64	99.10	67.79	78.49	74.99	86.86
		OCSVM	99.49	99.84	98.23	99.30	69.17	79.27	78.71	88.95
		ENAD	**99.78**	**99.93**	**98.37**	**99.47**	**73.05**	**83.07**	**83.40**	**92.15**
	SVHN	LID	98.59	99.07	92.13	94.79	85.80	91.83	90.47	94.61
		Maha	99.45	99.85	97.93	99.26	90.00	94.93	90.95	97.16
		OCSVM	99.51	99.86	97.38	99.17	91.40	95.00	96.54	98.50
		ENAD	**99.83**	**99.91**	**98.93**	**99.57**	**93.27**	**96.04**	**98.13**	**99.16**
ResNet	CIFAR-10	LID	99.18	99.67	94.37	96.50	79.40	88.58	73.95	82.29
		Maha	99.87	99.90	99.06	99.58	85.64	91.60	92.28	95.90
		OCSVM	**99.99**	**99.99**	98.95	99.44	83.90	90.83	92.26	95.68
		ENAD	99.99	99.99	**99.58**	**99.78**	**87.77**	**92.89**	**93.35**	**96.46**
	CIFAR-100	LID	97.53	98.78	94.52	96.76	56.10	69.87	65.53	78.51
		Maha	99.48	99.72	93.51	96.92	73.32	85.23	83.00	91.68
		OCSVM	**99.63**	**99.86**	91.70	95.79	71.69	84.17	83.17	91.24
		ENAD	99.63	99.78	**98.22**	**99.26**	**76.58**	**86.34**	**88.26**	**94.08**
	SVHN	LID	94.52	97.84	83.46	90.78	86.60	92.31	79.46	88.16
		Maha	97.90	99.60	92.22	97.16	93.04	95.74	84.95	92.13
		OCSVM	98.06	99.64	95.91	98.12	92.15	95.58	89.19	93.29
		ENAD	**98.33**	**99.69**	**96.80**	**98.59**	**93.70**	**96.18**	**90.66**	**94.62**

Table 4.1: **Comparative assessment of ENAD and competing methods in the Known Attacks scenario**. Performance comparison of the ENAD, LID [Ma+18], Mahalanobis [Lee+18], OCSVM detectors (all the pairwise combinations of the three single detectors are available in table 4.A.4). The Table contains the AUROC and AUPR for all the combinations of selected datasets (CIFAR-10, CIFAR-100 and SVHN), models (DenseNet and ResNet), and attacks (FGSM, BIM). See Methods for further details.

And and Maj (Majority), that is: an example is identified as adversarial if *at least one* detector, *all* detectors, or the majority of the detectors, respectively, classify it as adversarial.

In table 4.2 we report the F1 score comparison in the Known attack setting (see above), while in table 4.A.6 we report Precision and Recall. In almost all settings ENAD proves to be the best-performing strategy. As expected, we note that the Or voting scheme leads to high Recall, but low Precision, while the opposite holds for the And scheme. In this regard, we point out that voting scheme requires the training of three separate logistics, one for each detector, while ENAD requires only one.

Comparison with other ensemble approaches

In this section, we present the comparison of ENAD with another post hoc detector employing an ensembling technique, named Ensemble approach for Explanation-based Adversarial Detection (ExAD) [Var+21]. This method exploits explanation maps [Spr+15; SGK17; STY17] to distinguish normal from adversarial examples. More in detail, ExAD applies multiple types of explanation maps to a given unseen instance, and ensembles multiple DNNs, one for each combination of explanation technique and target class. In comparison to ENAD, only one type of detector is employed (DNNs), the detection task is performed on the feature attribution instead of the latent features and a detector is trained for each target class, in addition to each explanation technique.

In our experiments, we considered only three explanation techniques, given the requirements of the Captum library [Kok+20] and the properties of our pre-trained models: Guided Backpropagation [Spr+15], Input$\times$Gradient [SGK17] and Integrated Gradients [STY17] (using a black image as reference, the Gauss-Legendre quadrature for integral approximation and 100 approximation steps). Furthermore, given the noisy nature of feature attribution techniques [Hoo+19], we employed SmoothGrad-Squared to ensemble 10 attributions obtained from perturbed versions of the original image, given a Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.2)$. Moreover, we tested both CNNs and Autoencoders as detectors, as suggested by the authors.

Autoencoders were trained on normal examples only, and then the reconstruction loss (L_2 loss) is used to distinguish normal (low loss) from adversarial examples (high loss). Each autoencoder was fitted on the explanations of the instances belonging to $\mathcal{X}^{train}$, then the model was regularized using early stopping given the AUROC on the validation set $\mathcal{L}^{valid}$. Lastly, the threshold to separate normal from adversarial was chosen such that the False Positive Rate on $\mathcal{L}^{valid}$ is 5%. As for ENAD, performances were evaluated on the test set $\mathcal{L}^{test}$. $\mathcal{L}^{train}$ was not used. On the other hand, the CNNs were trained on $\mathcal{L}^{train}$, regularized with early stopping using $\mathcal{L}^{valid}$ and tested on $\mathcal{L}^{test}$.

In table 4.3 we report the results for the DenseNet model (the F1-score is reported for all the combinations of dataset and attack). We did not consider CIFAR-100

Model	Dataset	Attack / Ensemble	FGSM	BIM	DeepFool	CW
DenseNet	CIFAR-10	ENAD	**99.13**	**98.90**	**74.83**	**76.89**
		Or	95.15	96.32	70.93	72.71
		Maj	99.12	97.23	67.24	71.98
		And	95.20	96.70	55.75	54.13
	CIFAR-100	ENAD	**99.31**	**96.59**	**65.52**	**78.02**
		Or	97.46	92.93	59.45	69.70
		Maj	98.97	95.30	53.72	63.77
		And	95.30	91.87	34.10	41.31
	SVHN	ENAD	**99.12**	**95.84**	**86.23**	**93.87**
		Or	97.27	91.59	83.40	87.74
		Maj	98.99	94.69	84.00	91.42
		And	95.44	85.27	75.32	83.03
ResNet	CIFAR-10	ENAD	99.72	**97.23**	**77.61**	**85.86**
		Or	97.87	92.64	76.84	80.93
		Maj	**99.78**	96.30	74.93	84.23
		And	98.10	88.32	65.71	64.67
	CIFAR-100	ENAD	98.77	**95.24**	**67.19**	**81.26**
		Or	97.47	86.77	64.45	74.99
		Maj	**98.86**	88.49	61.46	76.03
		And	92.53	82.94	27.80	52.17
	SVHN	ENAD	97.68	**91.44**	**85.51**	**82.98**
		Or	95.25	86.43	83.55	78.93
		Maj	**97.91**	89.41	85.09	79.22
		And	91.42	73.47	76.99	66.59

Table 4.2: **Comparative assessment of ENAD and voting-based strategies in the Known Attacks scenario**. The F1-score returned by ENAD and the ensembling strategies based on voting schemes – i.e., And, Or, Maj(ority) – is shown for the the Known Attacks scenario (see the main text for further details). Precision and Recall results are available in table 4.A.6.

Model	Dataset	Attack / Detector	FGSM	BIM	DeepFool	CW
DenseNet	CIFAR-10	ExAD [CNN]	72.73	37.46	52.69	53.11
		ExAD [AE]	8.55	28.93	14.79	13.73
		ENAD	**99.13**	**98.90**	**74.83**	**76.89**
	SVHN	ExAD [CNN]	50.84	0.02	15.68	27.77
		ExAD [AE]	21.66	27.88	19.40	15.64
		ENAD	**99.12**	**95.84**	**86.23**	**93.87**

Table 4.3: **Comparison between ENAD and ExAD performances**. F1-score for ENAD and ExAD with the two detector types, i.e., the CNN and the Autoencoder (AE), for the DenseNet model, the CIFAR-10 and SVHN datasets and all the attacks (FGSM, BIM, DeepFool, CW). In all the settings, ENAD achieves the best performances. ExAD with the CNN detectors achieves good performances only for the CIFAR-10–FGSM setting, while ExAD with the Autoencoder has always bad performances.

due to the demanding resources required for 300 models (one for each combination of the target class and explanation method). ExAD achieved good performances only with CNN as the detector type and on easy attacks, such as FGSM, and always performs worse than ENAD. Our hypothesis is that a careful hyperparameter search has to be performed on the hyperparameters of both the explanation maps and the detectors to achieve competitive results. However, the resource requirements to perform such search is expected to be huge. For all the above reasons, we did not consider ExAD in further tests.

4.4.2 Transfer attacks

Transfer attacks with cheap training via FGSM

In this section, we discuss the performance of ENAD and Detectors A, B, and C when a transfer attack is performed. In particular, we tested the performance of all the detectors when trained on the cheapest benchmark attack, i.e. FGSM, that only requires the multiplication of the gradient by the perturbation size. Afterwards, we tested the performance on BIM, DeepFool and CW attacks.

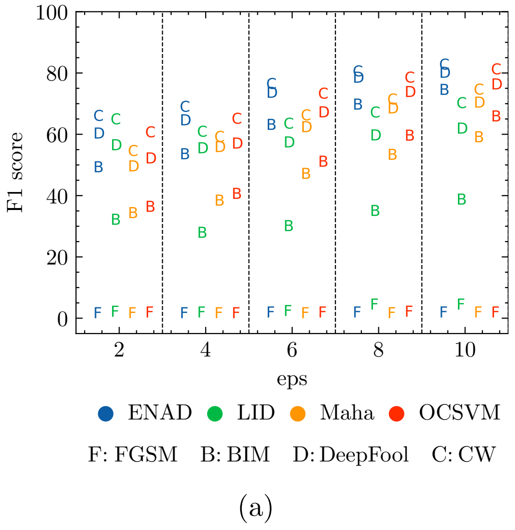
In table 4.4 one can see how, despite the expected worsening of the performances, either ENAD or OCSVM achieve the best AUROC and AUPR in almost all settings, further demonstrating the robustness of our approach.

Training matters with harder transfer attacks

A particular kind of transfer attack is the so-called *adaptive attack*, where an attacker generates adversarial examples exploiting the full knowledge of the detector. Many works address the topic of adaptive attacks [He+17; ACW18; Tra+20; CW17b], in

		Detector	LID		Maha		OCSVM		ENAD	
			AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
Model	*Dataset*	*Attack*								
DenseNet	CIFAR-10	BIM	88.28	92.97	98.00	99.24	98.93	99.59	**99.06**	**99.62**
		DeepFool	57.95	69.94	73.94	**82.88**	73.82	82.81	**74.72**	82.68
		CW	58.25	70.26	76.45	87.23	78.88	88.00	**79.96**	**88.40**
	CIFAR-100	BIM	28.62	31.45	**95.07**	**98.11**	90.94	95.84	57.70	60.27
		DeepFool	55.28	70.30	65.19	76.83	66.23	77.51	**68.90**	**79.21**
		CW	57.77	72.57	66.78	82.06	72.61	84.57	**73.86**	**84.71**
	SVHN	BIM	87.40	91.40	96.32	98.21	**97.26**	**99.14**	96.46	97.54
		DeepFool	73.50	79.50	86.76	91.17	**88.74**	**93.40**	85.59	88.58
		CW	75.48	84.63	88.43	94.33	**91.48**	**97.50**	88.62	92.16
ResNet	CIFAR-10	BIM	89.82	93.34	97.28	98.03	98.85	99.42	**99.02**	**99.47**
		DeepFool	61.62	73.92	75.55	81.34	79.23	**86.38**	**79.63**	85.66
		CW	67.79	78.05	79.01	84.75	**91.04**	**94.69**	90.01	94.55
	CIFAR-100	BIM	43.29	46.06	92.83	95.69	90.26	94.18	**93.70**	**96.68**
		DeepFool	55.23	68.12	68.31	78.24	72.08	**83.45**	**73.26**	83.26
		CW	64.89	76.24	81.72	88.70	80.53	87.56	**85.08**	**91.51**
	SVHN	BIM	71.38	85.03	88.10	95.15	**90.67**	**96.90**	84.54	93.38
		DeepFool	49.97	67.36	53.92	69.29	**59.65**	**75.11**	55.45	68.95
		CW	58.75	76.43	75.45	86.85	**80.03**	**89.70**	68.30	81.72

Table 4.4: **Comparative assessment of ENAD and competing methods in the Transfer Attacks (cheap training) scenario**. Performance comparison of the ENAD, LID [Ma+18], Mahalanobis [Lee+18], and OCSVM detectors when both the hyperparameter optimization and the logistic regression fit are performed on the FGSM attack. The Table contains the AUROC for all the combinations of selected datasets (CIFAR-10, CIFAR-100 and SVHN), models (DenseNet and ResNet), and attacks (BIM, DeepFool and CW). See Methods for further details.



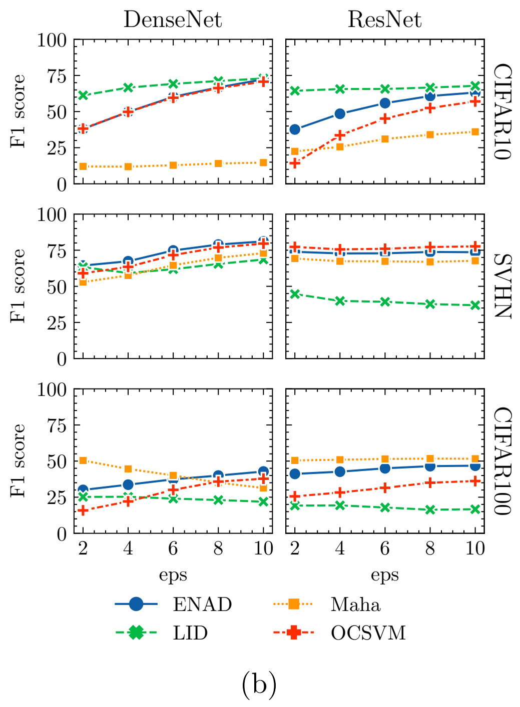


Figure 4.4: **Comparative assessment of ENAD and competing methods in the Transfer Attacks (hard attacks) scenario**. **(a)** The of F1-score returned by ENAD, LID [Ma+18], Mahalanobis [Lee+18], and OCSVM detectors against the CW_∞ transfer attack is shown (DenseNet–SVHN). The dataset employed in this test contains 800 samples of, respectively, adversarial, noisy and clean examples (see the main text for further details). Colors are used to distinguish the detectors, and letters to distinguish the attack on which the detector is trained, e.g., the blue C stands for ENAD trained on CW. The remaining settings are shown in fig. 4.A.4. **(b)** Analysis restricted on the performance of detectors when trained on the CW attack. In both figures, eps stands for the L_∞ attack max perturbation size (in the $[0, 255]$ scale).

some cases by targeting ensembles of detectors [He+17]. In particular, in [ACW18] the authors discuss on how to design an effective attack when a defence method has some gradient masking. The LID detector falls in this category, as its loss function proves to be particularly difficult to differentiate.

Given that ENAD is an ensemble of multiple detectors, making its loss function is *at least* as difficult as that of the LID detector. We here applied the same strategy proposed in [ACW18] to define a *harder* attack. In particular, we considered the L_∞ variant of the CW attack (labelled as CW_∞), which allows one to generate high-confidence adversarial examples with small distortions.

To this end, we generated 800 further adversarial examples with CW_∞ and distortion ϵ, and by scanning $\epsilon \in \{i/255 \mid i \in \{2, 4, 6, 8, 10\}\}$. This setting allowed us to assess the performance of the distinct detectors with respect to the distortion size. Also in this case, we defined three balanced partitions of adversarial, noisy, and clean examples (labels are coherent with the other experiments, see above), and tested ENAD and competing methods when trained on either FGSM, BIM, DeepFool, or CW, and tested against the CW_∞ attack.

In fig. 4.4a we report the F1-score for the DenseNet and SVHN setting. All detectors perform better when trained either on DeepFool or CW, with ENAD showing the overall best performance for all values of ϵ. All detectors appear to be unable to classify CW_∞ adversarial examples when trained on FGSM (cheap training), which in this case is not advisable. All the other settings are reported in fig. 4.A.4.

In fig. 4.4b we report the F1-score for all settings when the detectors are trained on CW, which appears to be best choice as for fig. 4.4a. ENAD is the best performing in the DenseNet and SVHN setting, while being the second best in all remaining settings, showing good overall performance. OCSVM has analogous performances, while the other detectors exhibit unstable performances, confirming the results of section 4.4.2.

4.4.3 Visualizing Adversarial Examples in the Score Space

In the following, we remark our interest in data visualization, as done with the APD in section 3.4.3, by providing a graphical representation of the adversarial examples in a low-dimensional embedding of the score space. More in detail, in order to explore the relationship between the scoring functions and the overall performance of ENAD, it is possible to visualize the test examples on the low-dimensional tSNE space [vH08], using the (logit weighted and Z-scored) layer- and detector-specific scores as starting features (3 detectors $\times$ 4 hidden layers = 12 initial dimensions).

This representation allows one to intuitively assess how similar the score profiles of the test examples are: closer data points are those displaying more similar score profiles, which translates in an analogous distance from the set of correctly classified training instances, with respect to the three scoring functions currently included in ENAD. Importantly, this allows one to evaluate how many and which adversarial examples display score profiles closer than those of benign ones, and vice versa, and visualize them.

As an example, in fig. 4.5 the test set of the SVHN, DenseNet, DeepFool setting is displayed on the tSNE space. The colour gradient returns the confidence c of ENAD, i.e., the probability of the logistic regression: an example is categorized as adversarial if $c > 0.5$, benign otherwise.

While most of the adversarial examples are identified with high confidence (leftmost region of the tSNE plot), a narrow region exists in which adversarial examples overlap with benign ones, hampering their identification and leading to significant rates of both false positives and false negatives. Focusing on the set of false negatives, it is evident that some adversarial examples are scattered in the midst of the set of benign instances (rightmost region of the tSNE plot), rendering their identification extremely difficult.

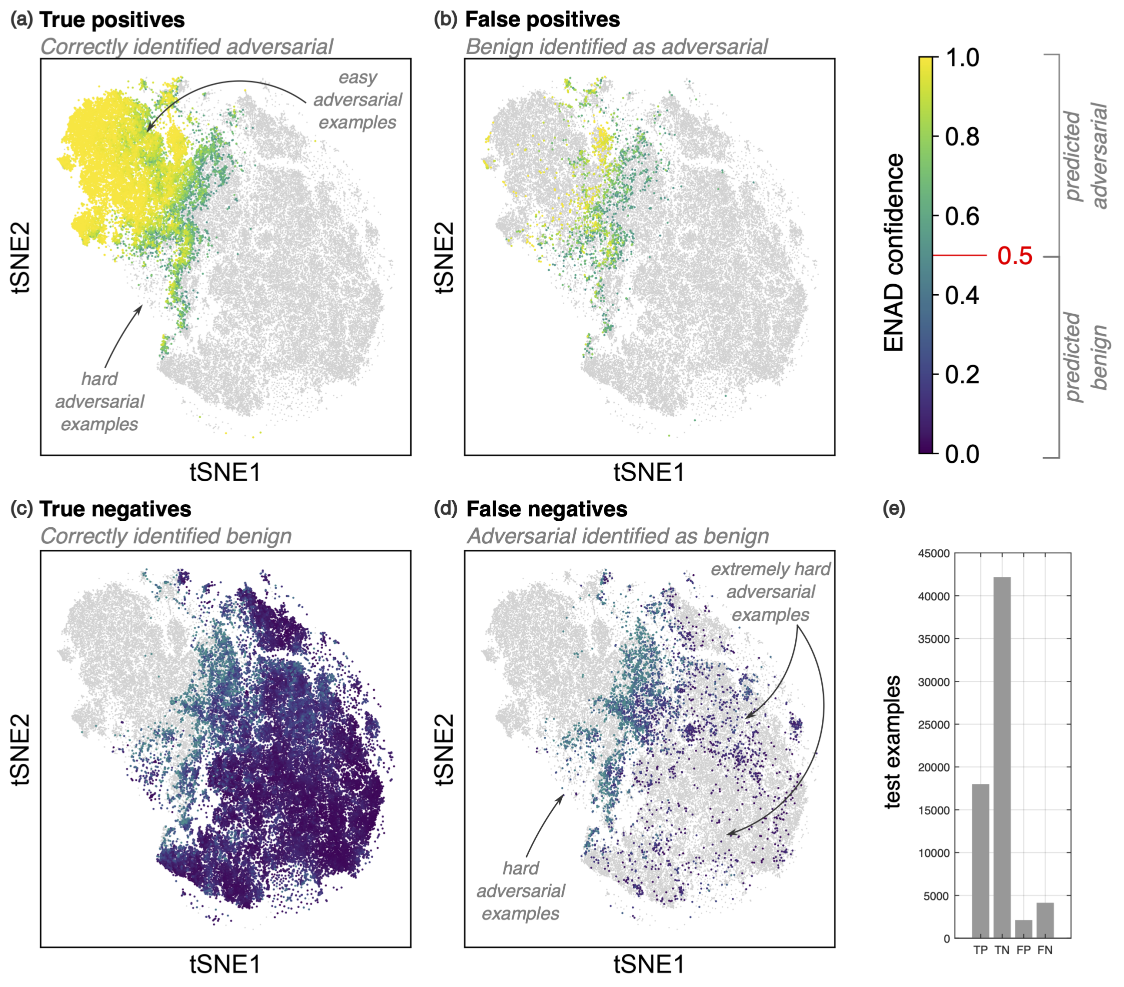


Figure 4.5: **Visualization of adversarial and benign examples in the low-dimensional score space. (DenseNet–SVHN–DeepFool).** Representation of the test data $\mathcal{L}^{test}$ for the configuration DeepFool, DenseNet and SVHN in the tSNE low-dimensional space [vH08]. The layer- and detector-specific scores are weighted with the logit weights, Z-scored, and then used as features for the tSNE computation, via the computation of the k-nearest neighbour graph ($k = 50$) with Pearson correlation as metric (further tSNE parameters: perplexity = 100, early exaggeration = 100, learning rate = 10000). In each quadrant the true positives **(a)**, false positives **(b)**, true negatives **(c)** and false negatives **(d)** are displayed. Each point in the plot represents either an adversarial or a benign example and the color returns the confidence c provided by ENAD, i.e., the probability returned by the logistic classifier. **(e)** The barplots return the absolute number of TPs, TNs, FPs and FNs for this experimental setting.

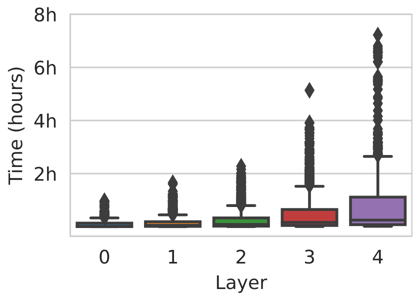


Figure 4.6: **OCSVM fit times by layer**. Fitting times of the OCSVM stand-alone detector, for each reported configuration in table 4.A.2. Attack, model and dataset are aggregated, since the mean fit time depends only on the layer on which the detector was trained, i.e., the deeper the layer, the higher the mean fit time.

4.4.4 Computational time

All tests were executed on a n1-standard-8 Google Cloud Platform instance, with eight quad-core Intel® Xeon® CPU (2.30GHz), 30GB of RAM and a NVIDIA Tesla® K80.

The fitting time of each parameter explored in all configurations for OCSVM is reported in fig. 4.6. For the computation times of the Mahalanobis and LID detectors, please refer to [Lee+18] and [Ma+18].

The computational time of ENAD is approximately the sum of that of the detectors employed to compute the layer-specific scores and, in the current version, it is mostly affected by OCSVM. In particular, its fitting time for each parameter explored in all configurations is reported in fig. 4.6 (note that the time is dominated by outliers that can be easility pruned out). For the computation times of the Mahalanobis and LID detectors, please refer to [Lee+18] and [Ma+18].

OCSVM hyperparameter search time can be improved in many ways: by considering SVM's implementations that support GPU [Wen+18] or by adding a Hyperband scheduler [FKH18], in addition to the Bayesian sampler that we employed in our experiments.

4.5 Conclusions

We introduced the ENAD ensemble approach for adversarial detection, motivated by the observation that distinct detectors are able to isolate non-overlapping subsets of adversarial examples by exploiting different properties of the input data in the internal representation of a DNN. Accordingly, the integration of layer-specific scores extracted from three independent detectors (LID, Mahalanobis and OCSVM) allows ENAD to achieve significantly improved performance on benchmark datasets, models and attacks, with respect to the state-of-the-art, even when a simple integration scheme (i.e., the logistic regression) is adopted.

It is also worth of note that the newly introduced OCSVM detector proved highly effective as a standalone in our tests, indicating that the use of one-class classifiers for this specific task deserves an in-depth exploration. Most important, the theoretical framework of ENAD is designed to be general and as simple as possible, so to show the advantages of adopting ensemble approaches in the "cleanest" scenario. Yet, the framework might be easily extended and improved.

On the one hand, ENAD may accommodate different scoring functions, generated via any arbitrary set of independent algorithmic strategies. In this regard, ongoing efforts aim at integrating detectors processing the hidden layer features with others processing the properties of the *output*, which have already proven their effectiveness in adversarial detection (see, e.g. [HG17a; LLS18; RKH19]). Similarly, one may explore the possibility of exploiting the information on activation paths and/or regions, as suggested in [LIF17; Cra+20a], as well as of refining the score definition by focusing on class-conditioned features.

On the other hand, more effective strategies for the integration of such scoring functions might be devised. As shown the in the results section, adversarial examples generated with a given attack might be more easily identified by a specific detector and by exploiting the properties of a specific layer. In other terms, the attack type is closely related to the detector performance and the layer relevance, and this results in attack-specific optimal weights of the logit currently employed by ENAD and competing methods, possibly limiting its effectiveness as a result. This aspect is even more relevant when facing transfer attacks, for which the logit training is executed on a separate attack, worsening the overall performance. This also suggests that the training phase should be considered with extreme caution when developing a detector for production.

For such reasons, more sophisticated strategies to combine scoring functions might be considered to improve the generality and robustness of our approach, e.g., via weighted averaging or by employing test statistics [Rag+21], as well as via the exploitation of more robust feature selection and classification strategies.

On a side note, we specify that, despite their simplicity, ensembling strategies based on voting schemes might be also considered as an alternative to the logit in safety-critical settings. For example, as presented in the results section, the Or voting scheme may be the method of choice if Recall matters more than Precision, as in several real-world biomedical scenarios (e.g., one might want to minimize the false negatives in diagnostic testing).

To conclude, given the virtually limitless possibility of algorithmic extensions of our framework, the superior performance exhibited in benchmark settings in a purposely simple implementation, and the theoretical and application expected impact, we advocate for a widespread and timely adoption of ensemble approaches in the field of adversarial detection.

4.A Supplementary Material

Detector	Parameter	Configurations
OCSVM	ν	$2^{-7}, 2^{-6}, \ldots, 2^{-1}$
	γ	$2^{-15}, 2^{-14}, \ldots, 2^{5}$
LID	k	$10, 20, \ldots, 90$
Maha	λ	$0.0, 0.01, 0.005, 0.002,$ $0.0014, 0.001, 0.0005$

Table 4.A.1: **Hyperparameters configurations (all settings)**. Hyperparameters space explored in the optimization step for the three detectors OCSVM, LID and Mahalanobis and for all the of models (DenseNet, ResNet), datasets (CIFAR-10, CIFAR-100, SVHN) and attacks (FGSM, BIM, DeepFool, CW).

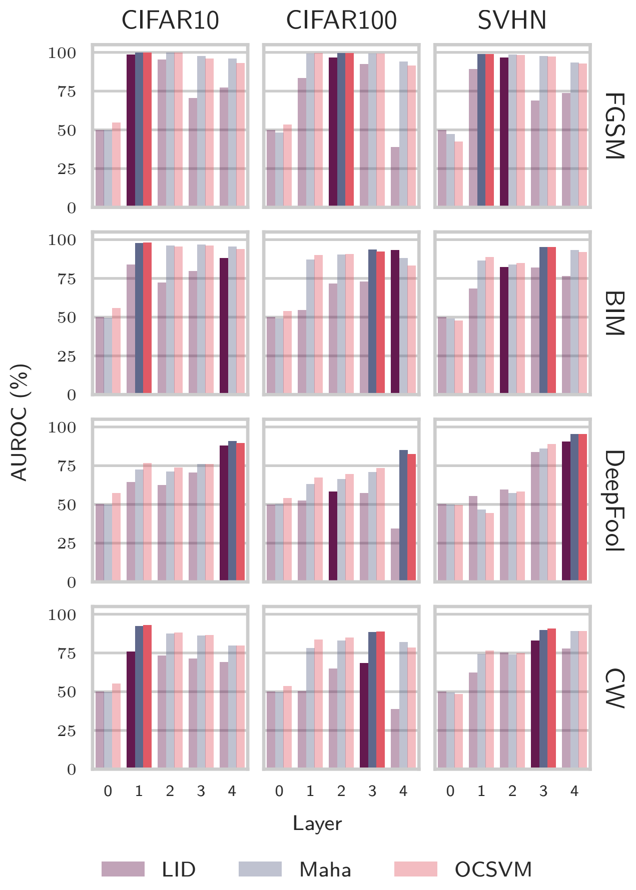


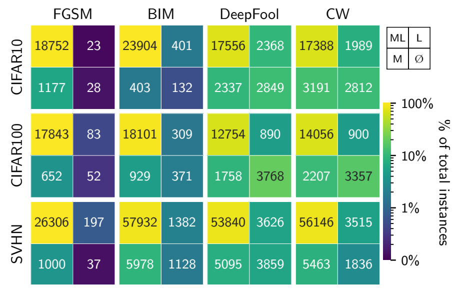
Figure 4.A.1: **Influence of layers in adversarial detection (ResNet model) in the Known Attack scenario**. For each configuration of datasets and attacks on the ResNet model, the AUROC of each layer-specific score for OCSVM, LID and Mahalanobis detectors is returned. For each configuration and detector, the best-performing layer is highlighted with a darker shade.

			Best Value									
		Layer	0		1		2		3		4	
		Parameter	ν	γ	ν	γ	ν	γ	ν	γ	ν	γ
Model	Dataset	Attack										
DenseNet	CIFAR-10	FGSM	1.27e-2	9.43e-2	7.81e-3	2.53e-4	7.81e-3	1.79e-4	2.06e-2	3.05e-5		
		BIM	9.40e-3	9.78e-3	4.15e-2	1.23e-2	1.87e-2	1.45e-2	1.26e-2	6.08e-5		
		DeepFool	1.12e-2	3.05e-5	7.81e-3	1.87e-2	1.29e-1	9.86e-5	1.23e-1	6.21e-3		
		CW	1.13e-2	1.32e-4	4.19e-2	2.86e-2	1.17e-1	1.65e-2	5.69e-2	8.62e-3		
	CIFAR-100	FGSM	8.25e-2	2.81e-4	7.81e-3	3.05e-5	7.81e-3	3.05e-5	2.04e-2	3.05e-5		
		BIM	1.41e-2	1.75e-2	7.81e-3	1.7e-2	6.17e-2	7.93e-4	2.09e-2	4.11e-3		
		DeepFool	1.10e-2	1.2e-4	8.81e-2	7.83e-3	4.67e-2	1.01e-2	1.08e-1	2.51e-4		
		CW	7.81e-3	3.05e-5	1.22e-1	9.69e-3	1.37e-1	3.35e-5	1.10e-1	3.05e-5		
	SVHN	FGSM	7.81e-3	6.1e-2	7.81e-3	2.52e-4	7.81e-3	4.88e-5	7.81e-3	3.05e-5		
		BIM	7.81e-3	2.32e-4	2.17e-2	2.98e-4	4.03e-2	7.63e-4	2.45e-2	3.05e-5		
		DeepFool	7.81e-3	3.05e-5	1.45e-2	1.74e-3	2.84e-2	3.05e-5	4.55e-2	4.16e-4		
		CW	7.81e-3	3.05e-5	1.18e-2	1.35e-2	8.92e-3	1.55e-2	1.43e-2	7.18e-3		
ResNet	CIFAR-10	FGSM	7.81e-3	3.05e-5	7.81e-3	3.05e-5	8.42e-3	7.69e-5	3.06e-2	9.42e-3	7.81e-3	2.25e-3
		BIM	7.81e-3	3.05e-5	2.95e-2	7.7e-3	4.12e-2	3.05e-5	4.09e-2	1.01e-3	1.04e-2	3.63e-3
		DeepFool	7.82e-3	3.16e-5	5.59e-2	3.50e-4	7.15e-2	9.23e-3	7.93e-2	5.22e-3	7.15e-2	4.73e-5
		CW	7.81e-3	3.05e-5	1.01e-1	5.92e-3	9.72e-2	5.53e-4	9.64e-2	9.35e-3	7.81e-3	4.6e-3
	CIFAR-100	FGSM	7.81e-3	9.77e-5	1.93e-2	3.23e-5	2.54e-2	5.58e-4	4.6e-2	1.01e-3	4.29e-2	1.28e-3
		BIM	7.81e-3	3.12e-5	1.2e-1	1.07e-3	1.29e-1	1.10e-4	6.90e-2	2.36e-3	1.18e-1	6.21e-4
		DeepFool	7.81e-3	3.05e-5	1.10e-1	3.28e-3	3.71e-2	9.03e-3	7.24e-2	3.05e-5	1.58e-1	5.43e-5
		CW	7.81e-3	3.05e-5	1.54e-2	4.32e-2	1.44e-1	3.05e-5	1.08e-1	4.46e-5	4.77e-2	2.35e-3
	SVHN	FGSM	7.81e-3	3.05e-5	2.81e-2	1.2e-4	4.36e-2	3.05e-5	5.53e-2	1.02e-3	2.03e-2	1.33e-4
		BIM	7.81e-3	3.05e-5	1.19e-1	3.76e-2	1.4e-1	1.82e-2	6.88e-2	1.18e-3	5.65e-2	1.33e-4
		DeepFool	7.81e-3	3.05e-5	7.81e-3	3.05e-5	1.78e-2	1.12e-2	1.22e-1	1.17e-3	2.4e-2	4.12e-5
		CW	7.81e-3	3.05e-5	7.81e-3	4.63e-2	1.58e-1	1.18e-2	1.09e-2	9.82e-3	4.63e-2	1.62e-4

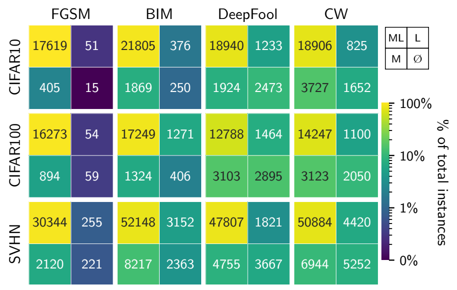
Table 4.A.2: **Best hyperparameters for the OCSVM detector**. Optimal OCSVM hyperparameters (ν, γ) for all the combinations of layer, model (DenseNet, ResNet), dataset (CIFAR-10, CIFAR-100, SVHN) and attack (FGSM, BIM, DeepFool, CW).

			Best Value	
		Parameter	ϵ	k
Model	Dataset	Attack		
DenseNet	CIFAR-10	FGSM	0.001	60
		BIM	0.0	90
		DeepFool	0.0	20
		CW	0.0	20
	CIFAR-100	FGSM	0.0014	90
		BIM	0.0014	90
		DeepFool	0.0	80
		CW	0.0	70
	SVHN	FGSM	0.0005	90
		BIM	0.001	80
		DeepFool	0.0005	20
		CW	0.0	20
ResNet	CIFAR-10	FGSM	0.001	80
		BIM	0.0005	90
		DeepFool	0.001	20
		CW	0.0	50
	CIFAR-100	FGSM	0.0005	90
		BIM	0.001	90
		DeepFool	0.001	80
		CW	0.001	90
	SVHN	FGSM	0.0005	80
		BIM	0.0005	30
		DeepFool	0.001	20
		CW	0.0	20

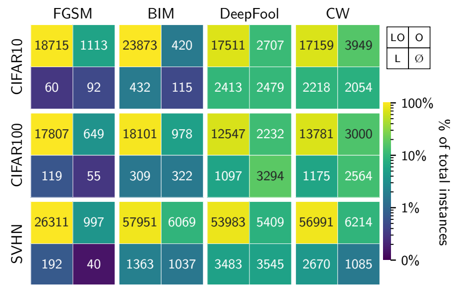
Table 4.A.3: **Best hyperparameters for Mahalanobis and LID detectors**. Optimal Mahalanobis and LID hyperparameters, i.e. perturbation magnitude ϵ and number of neighbors k, respectively, for all the combinations of method, model (DenseNet, ResNet), dataset (CIFAR-10, CIFAR-100, SVHN) and attack (FGSM, BIM, DeepFool, CW).



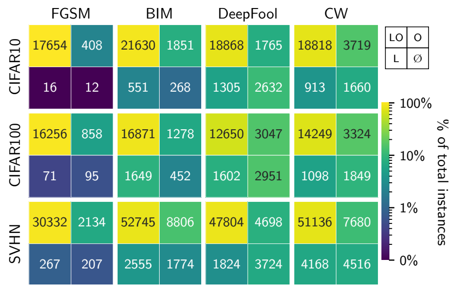
(a): LID (L) vs Maha. (M), DenseNet.



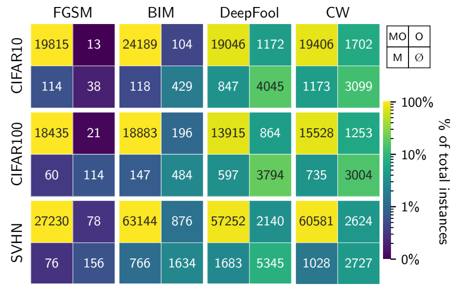
(b): LID (L) vs Maha. (M), ResNet.



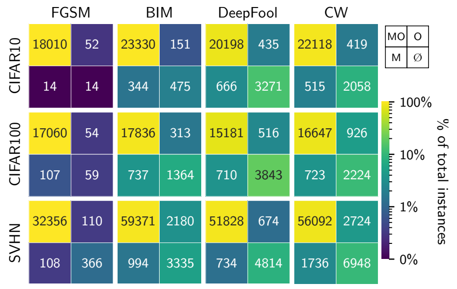
(c): LID (L) vs OCSVM (O), DenseNet.



(d): LID (L) vs OCSVM (O), ResNet.



(e): Maha (M) vs OCSVM (O), DenseNet.



(f): Maha (M) vs OCSVM (O), ResNet.

Figure 4.A.2: **Comparison of predictions of single detectors in the Known Attack scenario**. The contingency table shows the number of adversarial examples correctly identified: by both the detectors (top-left box), by either one of the two methods (diagonal boxes), by none of them (lower-right box), in all the experimental settings described in the main text.

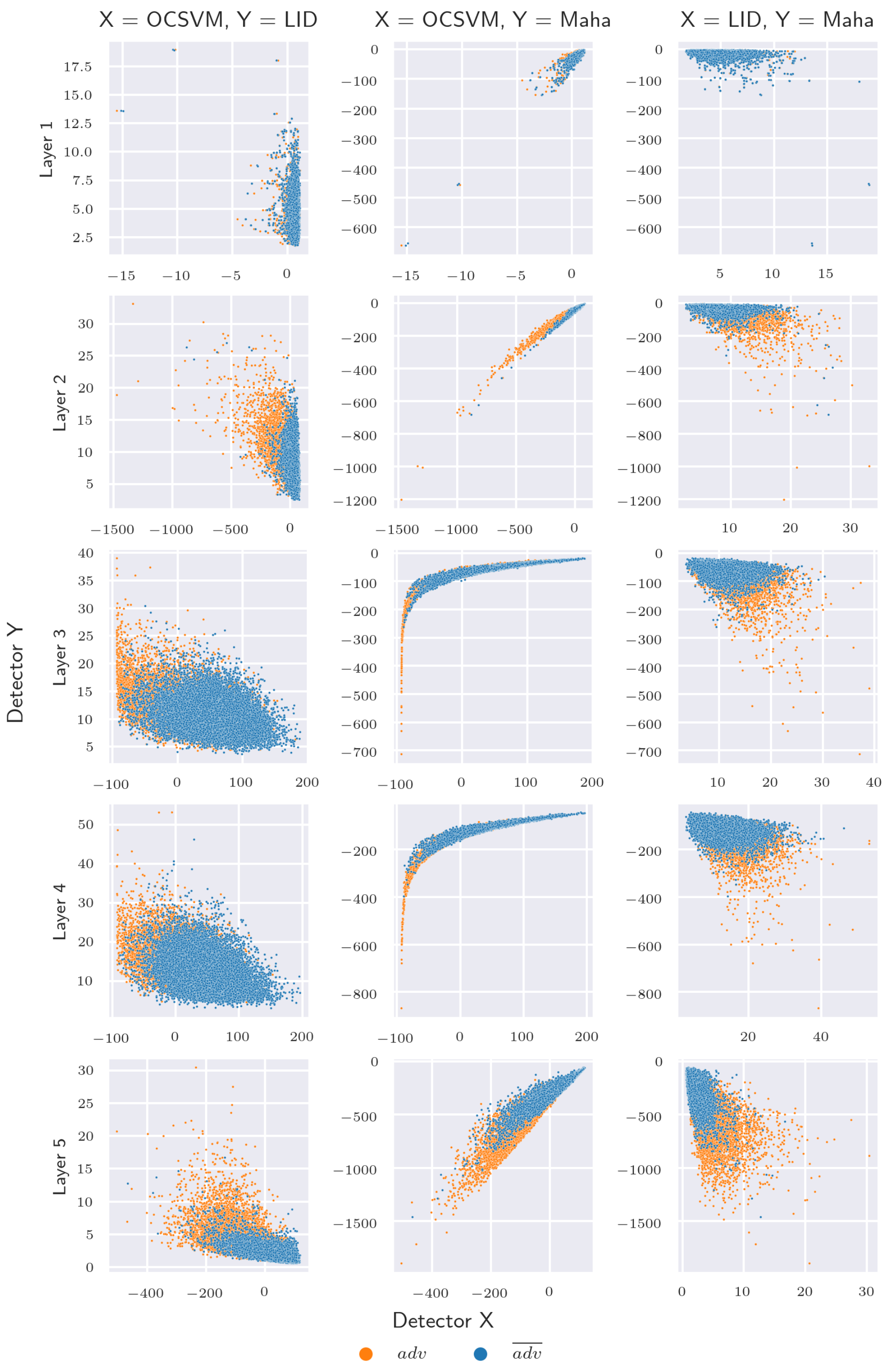


Figure 4.A.3: **Pairwise layer-specific scores comparison in the Known Attack scenario**. Comparison of the layer scores of OCSVM, LID and Mahalanobis detectors, i.e. O_l, L_l and M_l, respectively, for each layer l and for the Resnet, CIFAR-10, DeepFool configuration.

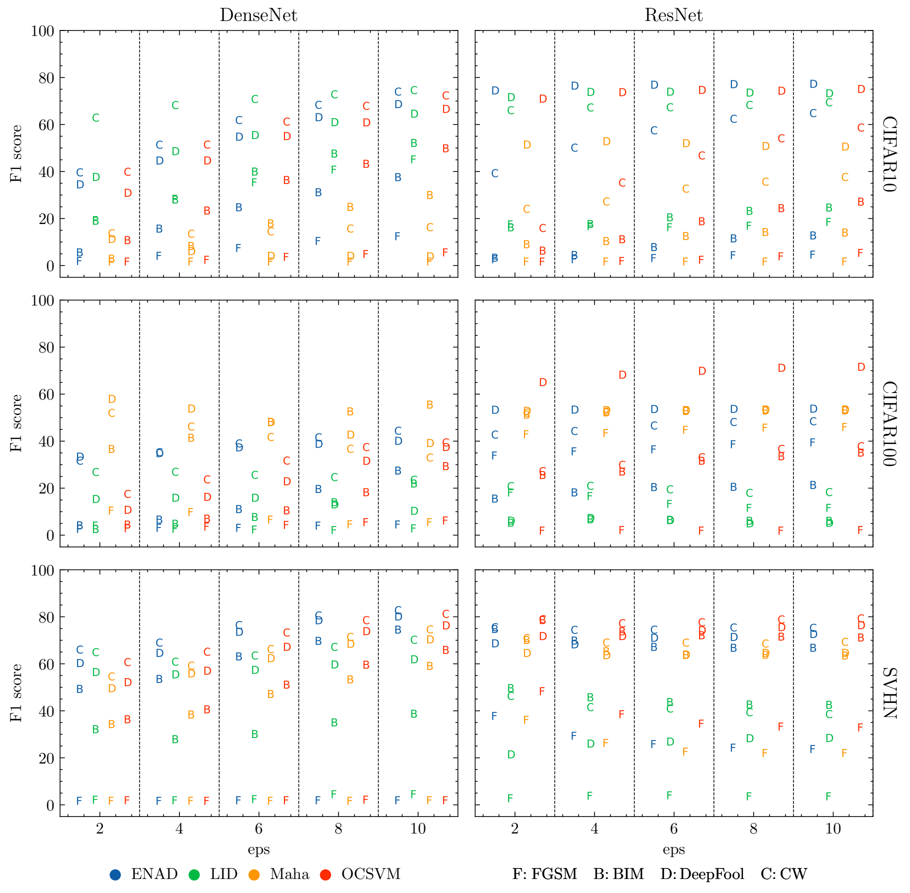


Figure 4.A.4: **Comparative assessment of ENAD and competing methods in the Transfer Attacks (hard attacks) scenario**. The of F1-score returned by ENAD, LID [Ma+18], Mahalanobis [Lee+18], and OCSVM detectors against the CW_{∞} transfer attack is shown (all settings). The dataset employed in this test contains 800 samples of, respectively, adversarial, noisy and clean examples (see the main text for further details). Colours are used to distinguish the detectors, while letters distinguish the attack on which the detector is trained. The eps on the x-axis is the strength of the attack, i.e., the perturbation eps/255. For all the settings FGSM is the worst performing training set, while DeepFool and CW are the best.

		Attack	FGSM		BIM		DeepFool		CW	
			AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
Model	Dataset	Detector								
DenseNet	CIFAR-10	LID	96.37	98.30	99.52	99.73	75.21	85.22	69.75	80.88
		Maha	99.80	**99.96**	99.46	99.75	74.46	82.73	78.43	87.42
		OCSVM	99.58	99.88	99.24	99.69	77.01	84.74	82.98	90.24
		Maha+LID	99.69	99.89	99.83	99.93	81.11	88.49	81.22	89.11
		OCSVM+LID	99.79	99.95	**99.90**	**99.96**	83.17	89.00	**85.30**	**91.50**
		OCSVM+Maha	**99.85**	99.94	99.47	99.78	79.78	86.52	83.17	90.49
		ENAD	99.70	99.89	**99.90**	**99.96**	**83.70**	**89.36**	**85.30**	**91.50**
	CIFAR-100	LID	98.39	99.29	96.31	98.11	55.88	70.12	59.67	72.80
		Maha	99.49	99.87	97.64	99.10	67.79	78.49	74.99	86.86
		OCSVM	99.49	99.84	98.23	99.30	69.17	79.27	78.71	88.95
		Maha+LID	99.69	99.90	97.85	99.11	70.86	81.02	79.06	89.82
		OCSVM+LID	**99.80**	**99.93**	**98.85**	**99.57**	**73.24**	83.06	82.32	91.78
		OCSVM+Maha	99.62	99.89	98.15	99.37	69.71	80.07	81.37	90.20
		ENAD	99.78	**99.93**	98.37	99.47	73.05	**83.07**	**83.40**	**92.15**
	SVHN	LID	98.59	99.07	92.13	94.79	85.80	91.83	90.47	94.61
		Maha	99.45	99.85	97.93	99.26	90.00	94.93	90.95	97.16
		OCSVM	99.51	99.86	97.38	99.17	91.40	95.00	96.54	98.50
		Maha+LID	99.80	**99.93**	98.64	99.50	92.15	95.65	95.29	98.32
		OCSVM+LID	99.81	99.93	98.45	99.44	92.63	95.58	**98.21**	**99.19**
		OCSVM+Maha	99.54	99.87	98.50	99.42	92.38	95.66	96.97	98.65
		ENAD	**99.83**	99.91	**98.93**	**99.57**	**93.27**	**96.04**	98.13	99.16

Table 4.A.4: **Comparative assessment of ENAD with all possible combinations of competing methods in the Known Attack scenario (DenseNet)**. Performance comparison of the ENAD, LID [Ma+18], Mahalanobis [Lee+18], OCSVM detectors, and of the pairwise integration of the three single detectors. The table returns the AUROC and AUPR for the DenseNet model and all the combinations of selected datasets (CIFAR-10, CIFAR-100 and SVHN) and attacks (FGSM, BIM, DeepFool and CW). See Methods for further details.

		Attack	FGSM		BIM		DeepFool		CW	
			AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
Model	Dataset	Detector								
ResNet	CIFAR-10	LID	99.18	99.67	94.37	96.50	79.40	88.58	73.95	82.29
		Maha	99.87	99.90	99.06	99.58	85.64	91.60	92.28	95.90
		OCSVM	**99.99**	**99.99**	98.95	99.44	83.90	90.83	92.26	95.68
		Maha+LID	99.97	99.98	99.45	99.73	86.40	92.26	92.53	96.07
		OCSVM+LID	**99.99**	**99.99**	99.54	99.75	85.36	91.71	92.81	96.08
		OCSVM+Maha	**99.99**	**99.99**	99.34	99.67	**87.95**	92.45	93.24	96.36
		ENAD	**99.99**	**99.99**	**99.58**	**99.78**	87.77	**92.89**	**93.35**	**96.46**
	CIFAR-100	LID	97.53	98.78	94.52	96.76	56.10	69.87	65.53	78.51
		Maha	99.48	99.72	93.51	96.92	73.32	85.23	83.00	91.68
		OCSVM	**99.63**	**99.86**	91.70	95.79	71.69	84.17	83.17	91.24
		Maha+LID	99.58	99.79	97.63	98.94	74.66	85.65	83.56	92.50
		OCSVM+LID	99.21	99.65	**98.24**	99.13	73.64	85.18	84.43	92.74
		OCSVM+Maha	99.63	99.78	94.48	97.63	76.24	85.74	87.16	93.01
		ENAD	**99.63**	99.78	98.22	**99.26**	**76.58**	**86.34**	**88.26**	**94.08**
	SVHN	LID	94.52	97.84	83.46	90.78	86.60	92.31	79.46	88.16
		Maha	97.90	99.60	92.22	97.16	93.04	95.74	84.95	92.13
		OCSVM	98.06	99.64	95.91	98.12	92.15	95.58	89.19	93.29
		Maha+LID	98.05	99.66	93.71	97.74	93.52	96.09	87.43	93.60
		OCSVM+LID	98.12	**99.69**	**96.84**	98.58	93.03	95.96	**90.88**	94.59
		OCSVM+Maha	**98.39**	99.68	95.83	98.14	93.48	96.00	89.19	93.41
		ENAD	98.33	**99.69**	96.80	**98.59**	**93.70**	**96.18**	90.66	**94.62**

Table 4.A.5: **Comparative assessment of ENAD with all possible combinations of competing methods in the Known Attack scenario (ResNet)**. Performance comparison of the ENAD, LID [Ma+18], Mahalanobis [Lee+18], OCSVM detectors, and of the pairwise integration of the three single detectors. The table returns the AUROC and AUPR for the ResNet model and all the combinations of selected datasets (CIFAR-10, CIFAR-100 and SVHN) and attacks (FGSM, BIM, DeepFool and CW). See Methods for further details.

		Att	FGSM		BIM		DeepFool		CW	
			Pr	Re	Pr	Re	Pr	Re	Pr	Re
Md	Ds	Ens								
DN	C10	ENAD	98.68	99.58	98.55	**99.26**	78.91	71.16	76.30	77.48
		Maj	98.88	99.35	97.33	97.14	79.28	58.38	79.12	66.02
		Or	90.82	**99.92**	93.76	99.03	66.48	**76.02**	65.20	**82.17**
		And	**99.75**	91.04	**99.63**	93.94	**91.81**	40.02	**91.55**	38.43
	C100	ENAD	99.04	99.58	95.56	97.64	72.51	**59.77**	77.28	**78.77**
		Maj	98.45	99.48	94.72	95.89	77.11	41.22	79.39	53.29
		Or	95.22	**99.81**	88.33	**98.04**	65.29	54.57	69.90	69.50
		And	**99.41**	91.51	**97.46**	86.89	**84.22**	21.38	**85.87**	27.19
	SVHN	ENAD	99.30	98.94	96.68	95.01	89.51	83.19	92.74	95.01
		Maj	98.53	99.44	95.75	93.65	90.00	78.76	92.20	90.66
		Or	94.86	**99.81**	86.41	**97.43**	78.59	**88.84**	80.06	**97.04**
		And	**99.93**	91.33	**99.00**	74.88	**95.92**	62.01	**98.37**	71.82
RN	C10	ENAD	99.73	99.70	96.41	98.07	85.33	71.17	88.30	83.56
		Maj	99.70	99.85	96.70	95.90	83.22	68.14	89.04	79.92
		Or	95.87	**99.97**	87.72	**98.15**	75.64	**78.08**	76.18	**86.31**
		And	**99.91**	96.35	**99.26**	79.56	**89.02**	52.08	**93.51**	49.43
	C100	ENAD	98.94	98.61	95.12	95.35	74.55	61.16	84.65	78.13
		Maj	99.16	98.56	90.89	86.21	73.76	52.68	85.11	68.70
		Or	95.43	**99.60**	78.72	**96.67**	65.96	**63.01**	69.78	**81.05**
		And	**99.64**	86.37	**97.42**	72.21	**90.38**	16.43	**89.85**	36.75
	SVHN	ENAD	98.09	97.28	91.79	91.09	91.02	80.63	85.03	81.03
		Maj	98.32	97.51	92.65	86.39	90.75	80.09	86.86	72.80
		Or	91.61	**99.19**	79.25	**95.04**	81.36	**85.87**	73.24	**85.58**
		And	**99.15**	84.81	**97.02**	59.12	**95.05**	64.69	**93.46**	51.72

Table 4.A.6: **Precision Pr and recall Re of ENAD and voting-based strategies in the Known Attacks scenario**. This Table complements the result in table 4.2. As expected, the Or voting scheme has the highest recall (an example is adversarial if at least one detector classifies it as adversarial). In contrast, the And voting has the highest precision (all detectors must agree on classifying an output as adversarial). Nevertheless, as reported in table 4.2, ENAD achieves the best overall F1-score and is therefore our method of choice.

Chapter 5

Uncertainty and Complexity in Imbalanced Classification Tasks

Contribution. This chapter contains the work mainly done during my six-month visiting period at the Computational Systems Biology lab at IBM Research Zürich[a], from March to September 2022. During the period, I was supervised by Dr. María Rodríguez Martínez[b] and Dr. Nicolas Deutschmann[c]

Summary. Many real-world applications of deep learning involve solving imbalanced binary tasks, where only one of the two classes is well-represented. In this chapter, we will illustrate two case studies to investigate the behaviour of deep models in imbalanced classification tasks. One will be TCR-epitope binding affinity prediction, a recent and important application of machine learning for immunology, while the other will be a standard image classification task on the CIFAR-100 [Kri09] dataset. In order to analyse the behaviour of a DNN classifier on imbalanced data we will estimate: the epistemic and aleatoric uncertainties, and the sensitivity of each target class to Out-of-Distribution (OOD) data. Moreover, we propose to employ the Intrinsic Dimensionality (ID) of each target class to detect when they have uneven complexities, due to the presence of sub-clasters (small disjuncts) or noisy examples.

Implementation. The work done in this project contributed to "The Uncertainty Quantification 360" (UQ360) open-source toolkit [Gho+21], which is available on Github[d]. Three anomaly detection algorithms (used for Epistemic Uncertainty estimation, as discussed in 2.3.3) have been added to the "Extrinsic UQ Algorithms", as "Latent Space Anomaly Detection Scores". Special thanks goes to Nicolas Deutschmann for having fine-tuned the methods, and for having fulfilled all the requirements for its final distribution.

[a]`https://researcher.watson.ibm.com/researcher/view_group.php?id=8226`
[b]`https://researcher.watson.ibm.com/researcher/view.php?person=zurich-MRM`
[c]`https://research.ibm.com/people/nicolas-deutschmann`
[d]`https://github.com/IBM/UQ360`

5.1 Introduction

In this chapter, we will investigate imbalanced classification tasks from the point of view of uncertainty and complexity. In this context, we are no longer interested in detecting and characterizing unseen anomalies, but rather interpreting the behaviour of the model on the classes of an imbalanced binary classification task.

Anomaly detection is a standard approach when dealing with imbalanced data [Ruf+21], since we can employ one-class methods such as OCSVM [Sch+99] to fit only the well-represented class, while considering all the remaining observations as anomalies. To the best of our knowledge, only few results compared the performance of anomaly detection and binary classification in imbalanced settings, such as [BSJ12].

A dataset can be imbalanced not only due to unequal class proportions, but also uneven complexity, which is another important aspect to consider. In fact, it is not rare in real-world scenarios that one class is well-represented, while the other is characterized by many sub-clusters (small disjuncts) or noisy examples. The problem is even more significant in safety-critical settings such as medical diagnosis [FAK19], where imbalanced data is quite frequent and the performance of the model has a direct impact on the patient and the medical costs. In this case, imbalanced datasets are directly related to the problem of long-tailed distributions we discussed in section 2.2.2, but in this case the tail mostly belongs to one class (the most difficult one).

In the following, we will investigate two case studies of deep models applied to imbalanced data, by considering four metrics to interpret the behaviour of the model and to estimate the complexity of the data. To quantify the uncertainty of the model, and possible biases towards one of the classes, we will first estimate the Aleatoric and Epistemic uncertainties of the model (metrics 1 & 2, section 5.3.1). The first is higher for isolated points, due to lack of information, and will be quantified using an anomaly detector on the latent features of the deep models. The second, on the other hand, will consider the fitted conditional class probabilities, where a high entropy corresponds to the inability of the model to make a decision, such as for points in the overlapping area between two classes. To estimate the complexity of the classes, we will compute their Intrinsic Dimensionality (metric 3, section 5.3.2), that is equal to the minimum number of dimensions required to represent the information in the data. Lastly, we will consider the sensitivity of the easier class to Out-of-Distribution (OOD) data (metric 4, section 5.3.3). The intuition is that if one class is much easier (i.e., well-represented) than the other, the model could behave like an anomaly detector and fit only the easier class, while classifying all the rest (including OODs!) as the harder one.

The first case study (section 5.4) is binding affinity prediction, i.e., the task of predicting whether two molecules bind or not. In particular, we will focus on predicting the affinity of T-cell receptors (TCRs) (a component of the immune system that acts as a detector) and epitopes (the recognized component of a pathogen), that is a significant recent application of machine learning [Mös+19].

Indeed, thanks to advances in high-throughput sequencing techniques [RSS15], we now have sufficient data to train deep models, such as TITAN [WBR21], to predict the binding affinity between TCRs and epitopes. The binding affinity prediction problem resembles an anomaly detection task, since a model could learn the properties that make a TCR and an epitope bind, while all the non-binding couples can be considered anomalies. In our analysis, we discovered a phenomenon we called the *"Epistemic Gradient"*, since the Epistemic Uncertainty (metric 2) is lower for one class (the binding points) and higher for the other. We claimed that the Epistemic Gradient is a consequence of data imbalance, opening a new interesting research line on understanding the learning mechanisms of deep models.

The second case study (section 5.5) will consider a more standard image classification task based on the CIFAR-100 [Kri09] dataset, in which we will evaluate the use of the Intrinsic Dimensionality and the sensitivity to OOD data to investigate imbalanced classification tasks and the behaviour of the classifier. We will first build datasets with an uneven distribution of image labels (modalities), to test whether the number of modalities controls their Intrinsic Dimensionality. The motivation of this experiment is that this should reproduce the phenomena we saw on the TCR-epitope case study. Then, we will evaluate the sensitivity to OOD both with uneven and even distribution of modalities and with different OOD datasets, such as SVHN [Net+11] and LSUN [Yu+15]. The claim is that if the classifier fitted only the easy class, behaving like an anomaly detector (see fig. 5.7 for a toy example), it will predict all the OOD data as the harder class, i.e., "the rest".

Although the results are preliminary and will have to be evaluated in a larger scale experiment, we strongly believe in the importance of this analysis given the relevance of imbalanced classification for many real-world scenarios, such as medical AI [Gao+20; FAK19], and the recent interest of the research community into studying Intrinsic Dimensionality in deep models [Ans+19; Pop+21], OOD detection and related topics [Yan+22] and uncertainty estimation [HW21].

Main Contributions The main contributions of this line can be summarized as follows.

- *Joint analysis of aleatoric and epistemic uncertainty in deep models*: we defined an approach to estimate the epistemic and aleatoric uncertainty in deep models that, given a test example, jointly evaluates both its predictive confidence and its neighbourhood density in the latent space.

- *Discovery of the Epistemic Gradient, a new phenomenon in binary classification of imbalanced datasets*: in a binary classification task, the Epistemic Gradient occurs when the epistemic uncertainty is bimodal and correlated with the target class. We also claim that the gradient is correlated to data imbalance.

- *Controlling class imbalance with modalities*: we introduced a novel approach to control the class complexity by assigning an uneven number of modalities (sub-clusters) to each class.

- *Intrinsic Dimensionality as class complexity estimate*: inspired by [Ans+19], we proposed the intrinsic dimensionality in the latent space, estimated through the twoNN [Fac+17] method, as a proxy for the target class complexity to characterize imbalance in binary classification tasks.

- *Using the sensitivity to OOD to characterize the anomaly detector behaviour of the model*: we propose to use the model's predictions on Out-of-Distribution data to characterize the biases of a model. The hypothesis is that a classifier behaving as anomaly detector fits the easier class, while assigning anomalous data to the harder one.

5.2 Background

5.2.1 Intrinsic Dimensionality and Sample Complexity

What is known as the manifold hypothesis states that "high dimensional data tend to lie in the vicinity of a low dimensional manifold" [FMN16]. As an example, this hypothesis has been supported by results on natural images [Car+08; Pop+21]. In this context, given a dataset with dimensionality d, we call d the *extrinsic dimensionality*, and we define the *Intrinsic Dimensionality* (ID) as the minimal number of variables necessary to represent the information in the dataset.

A number of methods have been proposed to estimate the Intrinsic Dimensionality of a dataset [Bac+21]. For example, in [Fac+17] the Intrinsic Dimensionality is estimated using only two Nearest Neighbours (twoNN), i.e., the distance from the two closest nearest neighbours for each data point. The two Nearest Neighbours (twoNN) algorithm will be described in detail in section 5.3.2.

In [Pop+21; KKL22] experimental results show that only the ID, and not the extrinsic one, is correlated with sample complexity, i.e., the minimal number of training instances required to learn a given task.

With regard to deep learning, the twoNN method has been employed to study the latent representation of DNNs [Ans+19]. While in [Pop+21] the authors investigated the relationship between the Intrinsic Dimensionality of computer vision data and the performance of Convolutional Neural Networks trained on those datasets.

5.2.2 Imbalanced Binary Classification

In [Jap01], the author distinguished *between-class* imbalance, i.e., when the two classes have a different number of samples, from *within-class* imbalance, i.e., when sub-clusters with different sizes exist within a single class. In agreement with the previous result, in [Lóp+13] the authors pointed out that also data intrinsic properties, such as lack of density or sub-clusters, should be considered in imbalanced data classification.

The relationship between Intrinsic Dimensionality and sample complexity in binary tasks have been studied in [Pop+21; KKL22]. Furthermore, in [KKL22] the authors showed that also the entanglement of the class manifolds, i.e., the curvature of the decision boundary, has an effect on sample complexity.

Last, in [BL19] the authors investigated the effect of importance weighting in deep models, with a particular focus on binary tasks. In particular, they showed that the impact of the weighting vanishes during training, also with imbalanced classes. Moreover, they studied how OOD images are classified when varying the ratio of the two classes, with and without importance weighting.

5.3 Metrics

In this chapter, we will define four metrics to analyse and interpret deep models. In particular, we will use the Aleatoric and Epistemic Uncertainty (metrics 1 & 2) as metrics to estimate the confidence of the model in its predictions. Then, we will study the Intrinsic Dimensionality (metric 3) as a proxy to estimate the complexity of a dataset, since it quantifies the minimum dimensions required to represent the information in the data. Lastly, we will consider the sensitivity to OOD data (metric 4) to understand which class is more sensitive to anomalous data.

5.3.1 Metrics 1 & 2: Aleatoric and Epistemic Uncertainty

In a setting with an infinite amount of data, the approximation uncertainty would eventually disappear. The model uncertainty, on the other hand, becomes negligible if the chosen model (the so-called hypothesis space $\mathcal{H}$) can learn the target function f^* ($f^* \in \mathcal{H}$). Consequently, when estimating uncertainty in powerful models such as DNNs, that are well known to be universal approximators [HSW89], approximation uncertainty becomes much more important than model uncertainty.

As we defined in section 2.3.3, the Aleatoric Uncertainty (AU) and Epistemic Uncertainty (EU) are a result of randomness and lack of data, respectively. In fig. 5.1, we provided a simple representation of the two uncertainty types, inspired by [HW21]. Given the absence of global assumptions on f^* by DNNs, the uncertainty is mainly related to the local properties of data: AU is higher when two classes are overlapping (the classifier makes random decisions) and EU is higher when the input example is isolated (due to lack of data).

Let us consider a training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i$ is an input example and y_i is either one of the K target classes. Moreover, given an input $\boldsymbol{x}$, let $p_\theta(y \mid \boldsymbol{x})$ be the probabilistic predictive distribution over the classes estimated by a DNN fitted on $\mathcal{D}$ with parameters θ. The AU can be estimated from the conditional class probabilities $p_\theta(y \mid \boldsymbol{x})$ and the EU can be estimated employing anomaly detection

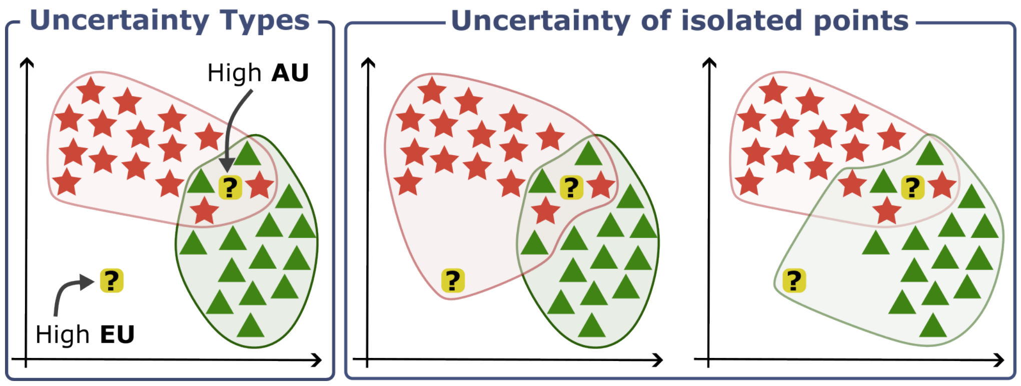


Figure 5.1: **Aleatoric and Epistemic Uncertainty in binary classification**. Reproduction of Figure 12 from [HW21]. On the left, Aleatoric Uncertainty (AU) is higher for a query point (identified by the question mark "?") in between the two classes, while the Epistemic Uncertainty (EU) is higher for the isolated point. For the point with high AU, the model has to "toss a coin" to make a decision, and the same holds also for the isolated point as reported in the remaining two panels.

methods, that typically estimate the density $p(\boldsymbol{x})$. Note that the previous definitions can be easily extended to the regression case where y is a real value.

In the following, we will describe which technique we employed to estimate AU and EU, respectively. Refer to section 2.3.3 and [HW21] for further details on uncertainty estimation in DNNs.

Metric 1: Aleatoric Uncertainty

In order to estimate the Aleatoric Uncertainty (AU), we will use the predictive uncertainty, or conditional class probabilities, of the model $p_\theta(y \mid \boldsymbol{x})$. The predictive uncertainty is higher when a model is unsure about the prediction, such as when a point lies in the overlapping region between two different targets, therefore it is a good proxy for estimating the AU. It should be noted, nevertheless, that the predictive uncertainty is not completely independent of the contribution of EU. For this reason, we will always consider AU and EU jointly in all the experimental results.

The predictive uncertainty can be robustly estimated through *deep ensembles* [LPB17]. More formally, given an ensemble of M models, the average prediction confidence estimated through deep ensembles can be defined as follows:

$$p(y \mid \boldsymbol{x}) = M^{-1} \sum_{m=1}^{M} p_{\theta_m}(y \mid \boldsymbol{x}),$$

where $p_{\theta_m}(y \mid \boldsymbol{x})$ is the predictive confidence of model m with parameters θ_m. Then, the predictive uncertainty can be simply computed considering the entropy, i.e., $H(p(y \mid \boldsymbol{x}))$. Lastly, we approximate the AU as the predictive uncertainty, i.e.,

Algorithm 4 aK-LPE algorithm [QS12].

```
Input: Bootstrapping times $B$, training set $\mathcal{X}^{train}$, query point $\eta$, number of neighbours $K$

1: $n = |\mathcal{X}^{train}|$
2: procedure TRAINING
3:     for each $b$ in $1, \dots, B$ do
4:         Randomly split $\mathcal{X}^{train}$ into two equal sets of size $n/2$: $S_1$ and $S_2$
5:         Define $G_b(x) \coloneqq G(x, S_2, K)$ if $x \in S_1$ else $G(x, S_1, K)$
6:     end for
7:     Define $G^{train}(x) \coloneqq \frac{1}{B} \sum_{i=1}^{B} G_b(x)$
8: end procedure

9: procedure TESTING($\eta$)
10:    for each $b$ in $1, \dots, B$ do
11:        Randomly pick a subset $S$ of $\mathcal{X}^{train}$ with size $n/2$
12:        Define $G_b(\eta) \coloneqq G(\eta, S, K)$
13:    end for
14:    return $\frac{1}{n}|\{G^{test}(\eta) \leq G^{train}(x) \mid x \in \mathcal{X}^{train}\}|$    ▷ Return p-value
15: end procedure
```

given a query point η:

$$\mathsf{AU}(\eta) \approx p(y \mid \eta)$$

The deep ensembles method was chosen because it performs as well as other methods, such as Bayesian DNNs [GG16], while requiring no modifications of the model. Moreover, we considered the version of deep ensembles without adversarial training [GSS15], since its application was not straightforward in the experimental settings and we expected the gain not to be significant.

Metric 2: Epistemic Uncertainty

As already discussed, we will estimate the Epistemic Uncertainty (EU) by means of an anomaly score, that will tell us how much a given point is isolated. More in detail, I will here introduce an anomaly detector, that will then be used on the latent features of the given DNN.

Given that we are estimating the uncertainty, we would like to have a normalized metric in $[0, 1]$. Many anomaly detectors do not satisfy such requirement, up to some post hoc normalization [Kri+11], therefore we chose a detector that provides a p-value estimate. More in detail, we will employ a method based on k-nearest neighbours called averaged K (nearest neighbors) Localized p-value Estimation (aK-LPE) [QS12]. Of the three detectors we employed in ENAD (introduced in chapter 4), only the Mahalanobis distances allows a straightforward p-value estimate [McL99], although we preferred the more flexible approach employing nearest neighbours that was recently shown to have superior for OOD detection [Sun+22]. Indeed, nearest neighbours can be considered a possible effective future addition to the ENAD ensemble.

Let η be a query instance, S a dataset, and K a hyperparameter controlling the nearest neighbours computation, the aK-LPE algorithm defines the G-statistic of η as:

$$G(\eta, S, K) = \frac{1}{K} \sum_{i=K-\lfloor\frac{K-1}{2}\rfloor}^{K+\lfloor\frac{K}{2}\rfloor} D(\eta, S, i),$$

where $D(\eta, S, k)$ the Euclidean distance from x to its k-th nearest neighbour in S.

Algorithm 4 describes the aK-LPE algorithm, that employs a bootstrapping strategy to estimate the G-statistic. In particular, at line 2 the training procedure is defined, computing the bootstrapped G-statistic for all the training points (G^{train}). At line 9 is reported the procedure to compute the p-value for a query point η is reported: first, the bootstrapped G-statistic for η ($G^{test}(\eta)$) is obtained, similarly to the training procedure, then the percentage of training points with G-statistic greater than $G^{test}(\eta)$ is used as an empirical $p-$value. Smaller $p-$values will then characterize points with a higher chance of being anomalous.

Let $\hat{p}(\eta)$ be the empirical $p-$value computed with the aK-LPE algorithm, then we define the epistemic uncertainty of η as:

$$\mathsf{EU}(\eta) \coloneqq 1 - \hat{p}(\eta)$$

To summarize, the empirical uncertainty can be estimated using aK-LPE, given only the hyperparameters K and B, controlling the nearest neighbours and the number of bootstraps, respectively.

The implementation of the aK-LPE algorithm [QS12] can be found in "The Uncertainty Quantification 360" (UQ360) open-source toolkit [Gho+21], which is available on GitHub[1].

5.3.2 Metric 3: Intrinsic Dimensionality Estimation

To estimate the intrinsic dimensionality of a given dataset $\mathcal{X}$, we will employ the TwoNN [Fac+17] approach, described in algorithm 5.

Let $r_{i,1}$ and $r_{i,2}$ be the two closest points to x_i with respect to the Euclidean distance. We define $\mu_i \coloneqq r_{i,2}/r_{i,1}$, for each $x_i \in \mathcal{X}$ (line 3). Moreover, we define the empirical cumulate $F^{emp}(\mu)$ as $F^{emp}(\mu_{\pi(i)}) \coloneqq i/N$ (line 7), where π is the permutation of all μs in ascending order.

[1] `https://github.com/IBM/UQ360`

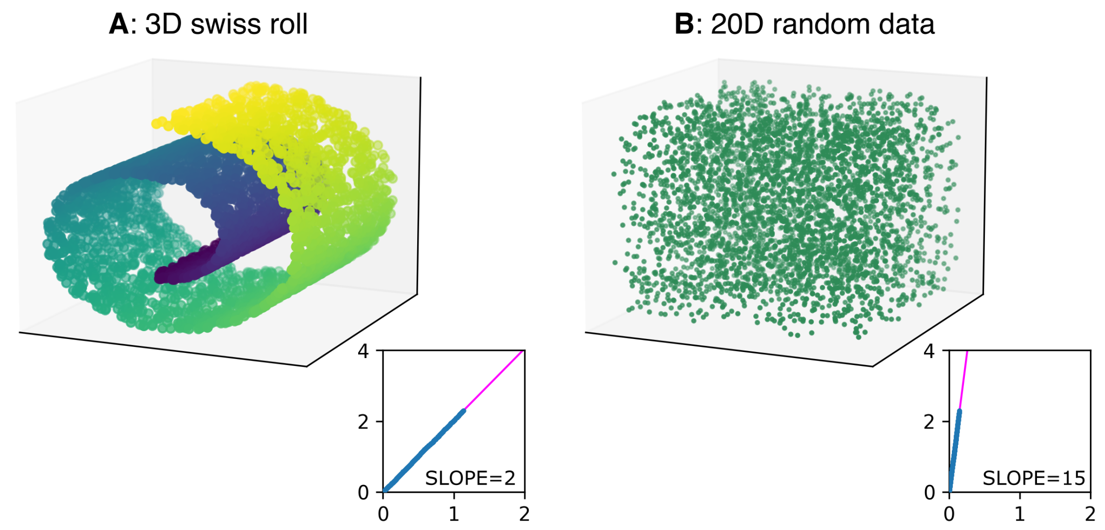


Figure 5.2: **Two examples of estimated intrinsic dimensionality**. In panel **A** there is a three dimensional ($3D$) Swiss roll, while in panel B a three dimensions of a three dimensional slice of randomly generated points from $[0, 1]^{20}$. On the right of each plot, we reported the points (in blue) corresponding to the line defined by the twoNN [Fac+17] method (discard fraction = 10%) using the distance of each point to its two closest neighbors and in pink the fitted straight line, with slope equal to the intrinsic dimensionality. We note that the roll has intrinsic dimensionality 2, due to the number of dimensions of the tangent hyperplane, corresponding to the local dimensionality.

As shown in [Fac+17], given the assumption that the density is constant around x_i (up to the second neighbour), the intrinsic dimensionality d can be approximated as follows:

$$\frac{\log\left(1 - F^{emp}(\mu)\right)}{\log \mu} \approx d$$

In practice, the intrinsic dimensionality d can be estimated by the slope of the straight line passing through L:

$$L = \{(\log \mu_i - \log\left(1 - F^{emp}(\mu_i)\right) \mid i \in \{1, \ldots, |\mathcal{X}|\}\}$$

In order to make the above fit more stable, in [Fac+17] the authors also propose to discard highest values of μ_i (line 6). Importantly, the discard fraction (default = 10%) is the only hyperparameter of this method.

A simple toy example with two datasets and the respective estimated slopes is reported in fig. 5.2.

In the following, we will use for all the experiments the implementation of twoNN from the library `scikit-dimension` [Bac+21].

Algorithm 5 TwoNN algorithm.

```
Input: Dataset $\mathcal{X}$ with size $n$, discard fraction $df$
1: for each $x_i$ in $\mathcal{X}$ do
2:     Compute the distance from $x_i$ to its first and second neighbour, $r_{i,1}$ and $r_{i,2}$, respectively,
3:     Define $\mu_i := \frac{r_{i,2}}{r_{i,1}}$
4: end for
5: Define $\pi$ to be the permutation of all $\mu$s in ascending order.
6: Define $S = \{i \mid \pi(i) \leq n \cdot df\}$                ▷ Discard $df$% of highest $\mu_i$
7: Define the empirical cumulate $F^{emp}(\mu)$ as $F^{emp}(\mu_{\pi(i)}) := \frac{i}{N}$
8: Fit a straight line $l$ through $\{(\log \mu_i, -\log(1 - F^{emp}(\mu_i)) \mid i \in S\}$
9: return slope of $l$                ▷ Return the intrinsic dimensionality
```

5.3.3 Metric 4: Sensitivity to OOD Data

The last metric that we will introduce is the sensitivity to OOD data. Given a binary classification dataset, e.g., cats and dogs from CIFAR-10 [Kri09], and a model trained of $\mathcal{X}$, we want to compute the fraction of OOD points, e.g., instances of the SVHN [Net+11] dataset, that are predicted as either cats or dogs.

Formally, let $\mathcal{X}$ be a labelled dataset with two targets, y_1 and y_2, $p(y \mid \boldsymbol{x})$ be the learned conditional class probabilities by a given classifier. Moreover, let $\mathcal{Y}$ be an OOD dataset. The sensitivity of class y_1 to $\mathcal{Y}$ is defined as the number of samples from $\mathcal{Y}$ classified as y_1, that is:

$$ood(y_1, \mathcal{Y}) = \frac{|\{x_i \mid x_i \in \mathcal{Y} \text{ and } p(y_1 \mid x_i) \geq 0.5\}|}{|\mathcal{Y}|}$$

5.4 Case Study 1: Binding Affinity Prediction

In this section, we will discuss the results of the uncertainty analysis we performed on a deep model for TCR-epitope binding affinity prediction: TITAN [WBR21].

5.4.1 Experimental Setup

TCR-epitope Binding Affinity Prediction

T-cells are a type of lymphocyte and an essential component of the adaptive immune system. The recognition of pathogens is carried out by T-cell receptors, which interact with foreign molecules by binding with peptides (epitopes) presented on their surface by major histocompatibility complex molecules. The effectiveness of the immune system is then expressed by the diversity of its TCR repertoire [LBA15].

Reliably predicting epitope recognition by T-cell receptors would represent a major breakthrough for immunology and cancer therapy [Sah+17]. However, the high diversity of the T-cell receptors (TCRs) repertoire constitutes an important challenge for generating exhaustive datasets.

Recent advances in high-throughput sequencing [RSS15] allowed producing more representative data [Lin+15] and pushed the application of machine learning for TCR–epitope binding affinity prediction [Mös+19]. More in detail, multiple deep learning methods have already been proposed [WBR21; Mor+21; Cai+22], where convolutions and attention layers represent the most frequent techniques.

In practice, we used the same dataset and setup used in [WBR21]. The dataset is a composition of the `VDJ` database [Bag+20] and a `COVID-19` dataset published by the `ImmuneCODE` project [Din+20]. The data is composed by binding pairs only, with 192 unique epitopes and 23,143 unique TCRs. TCRs are then shuffled to generate negative pairs (the chance of getting a binding pair is very low), leading to the final dataset of 46,290 examples. The dataset is then split into the train and test set, with 44,830 and 2360 examples each, such that the test set contains unseen TCRs.

Model

TITAN [WBR21] is a deep model for TCR-epitope binding affinity prediction (an overview is reported in fig. 5.3). That is, given a couple composed by a TCR sequence and an epitope sequence, it will predict whether they bind or not. Starting from the input, TITAN is composed by $1D$ convolutional layers with different kernel sizes, exploiting the local neighbourhood information of each sequence, followed by context attention layers, where one sequence is used as a context for the other in computing the attention scores. Lastly, a classifier head with two dense layers (368 and 184 units each, respectively) leads to the binary prediction.

For additional details on the model and the dataset, we refer to [WBR21] and the implementation[2].

Uncertainty Estimation

Aleatoric uncertainty was estimated using deep ensembles [LPB17], using five models with different initialization seeds. Then, we also applied the aK-LPE algorithm on the last dense layer features to estimate the epistemic uncertainty. For the aK-LPE algorithm, we used $K = 50$ (nearest neighbours hyperparameter) and $B = 5$ (number of bootstraps). All the uncertainty scores are computed for test instances, the training set was used only to fit the aK-LPE algorithm.

5.4.2 The Epistemic Gradient

To visualize Aleatoric and Epistemic Uncertainties, in fig. 5.4 we project the scores into a $2D$ embedding. The embedding of choice is tSNE [vH08], applied on the last layer features of one of the five instantiations of TITAN. From left to right,

[2] `https://github.com/PaccMann/TITAN`

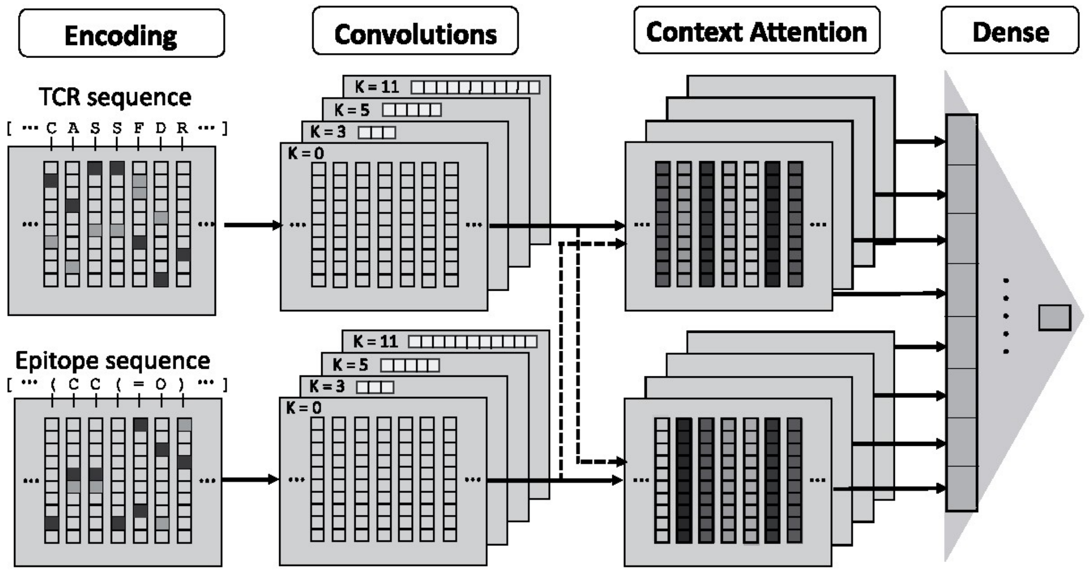


Figure 5.3: **Overview TITAN's architecture**. TITAN's [WBR21] architecture takes as input the encoding of the TCR and epitope's sequences, then $1D$ convolutions with various kernel sizes (K) are applied to both input streams. The convolutions are feeded to context attention layers, that consider one sequence as the context for the other, and viceversa. Lastly, a classifier head of dense layers is applied to predict the binding probability.

the first and second plot are the Epistemic and Aleatoric uncertainties (note that the aleatoric was estimated through an ensemble, but is plotted on an embedding of one of the five instantiations), respectively, and the last one is the target class (either binding or non-binding).

The aleatoric uncertainty follows our expectation, and is higher in the overlapping region between the two targets, and lower in the boundary points. The epistemic uncertainty, on the other hand, is not always higher in the boundary points, as one would expect, but is correlated with the target class: binding points, in fact, have lower epistemic uncertainty than non-binding points.

As we observed in section 2.3.3, is important to compare the two sources of uncertainty jointly, as reported in fig. 5.5. From this visualization, is quite evident that the binding points have higher aleatoric (x-axis, right) and lower epistemic uncertainty (y-axis, lower), while non-binding points have high epistemic uncertainty (y-axis, higher) and a heavier tailed aleatoric uncertainty (x-axis, left) than the binding. From the marginal density, the epistemic uncertainty is bimodal with regard to the target class, from now on we will refer to this phenomenon as the *Epistemic Gradient*.

The Epistemic Gradient suggests an intrinsic imbalance between the binding and non-binding class. Indeed, they are *semantically imbalanced*, given that we

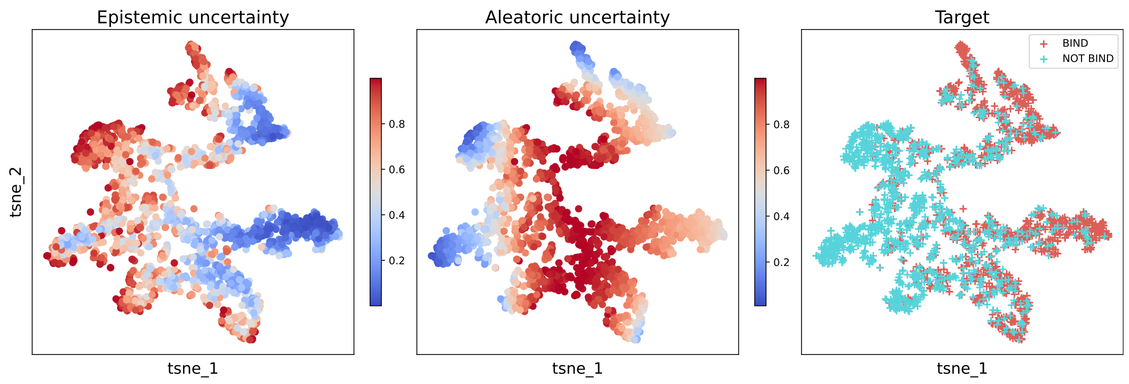


Figure 5.4: **Projection on a tSNE embedding of TITAN's test set aleatoric and epistemic uncertainty**. From left to right, projection of the epistemic and aleatoric uncertainties and the target class on a $2D$ embedding of the last layer's features, obtained using the tSNE algorithm [vH08] (`n_components`$=2$, `perplexity`$=30.0$, `early_exaggeration`$=12.0$).

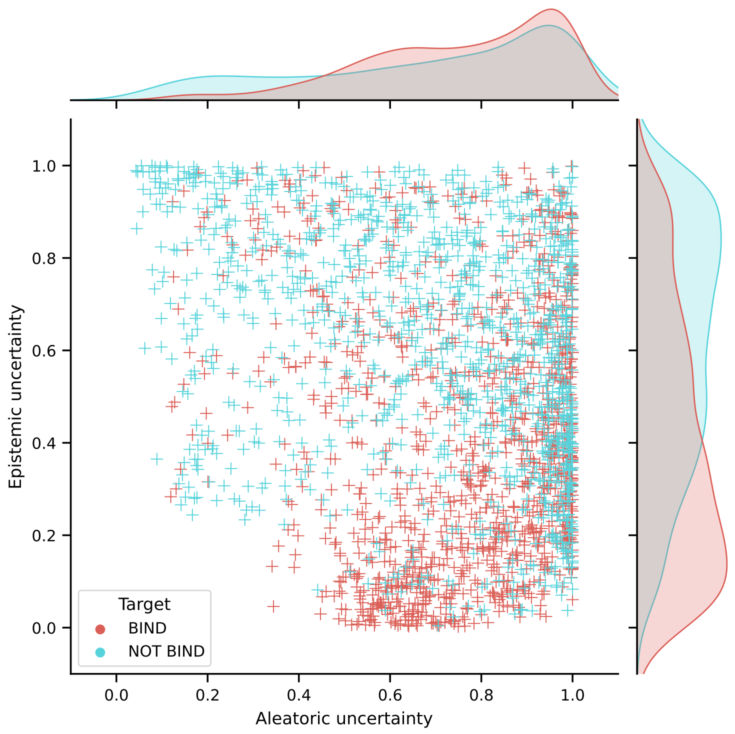


Figure 5.5: **Relation between TITAN's test set aleatoric and epistemic uncertainty**. Joint visualization of aleatoric (x-axis) and epistemic (y-axis) uncertainties. Binding points have higher aleatoric (x-axis, right) and lower epistemic uncertainty (y-axis, lower), while non-binding points have high epistemic uncertainty (y-axis, higher), while the aleatoric one has a heavier tail (x-axis, left) than the binding points. The epistemic uncertainty appears to be bimodal with regard to the target class. We refer to this phenomenon as the *Epistemic Gradient*.

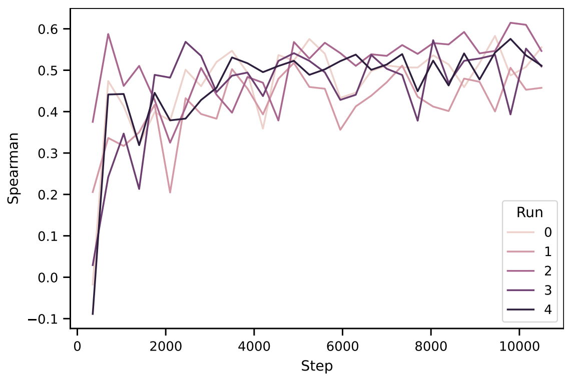


Figure 5.6: **Stability of Epistemic Gradient across multiple seeds**. Evolution during training of the Spearman's rank correlation coefficient between the target class and the epistemic uncertainty for five different instantiations of the TITAN model. All the settins achieve the same value of ≈ 0.5.

would expect the features of a binding pair of sequences to be learnable, while the non-binding sequences could be framed as "anomalies". Consequently, the model appears to behave as an anomaly detector, where the binding points are close to each other, i.e., denser, while the non-binding ones are more sparse.

To further validate the results, it is important to check if the Epistemic Gradient is present in all the models we used to compute the aleatoric uncertainty. More in detail, we computed the Spearman's rank correlation coefficient between the target class and the epistemic uncertainty, to measure if a correlation exists in all settings.

In fig. 5.6 we reported the Spearman correlation evolution during training for all the seeds. Despite some initial fluctuations, a correlation value of around 0.5 is achieved in all settings, confirming the reproducibility of the Epistemic gradient phenomenon.

5.5 Case Study 2: Image Classification

In this case study we will try to reproduce the conditions that led to the Epistemic Gradient, but in a more controllable setting, i.e., with benchmark computer vision data.

As we discussed in the previous case study, we believe that the Epistemic Gradient is a result of the imbalance of the two classes involved in the binary classification (the EASY and the HARD class). Therefore, we anticipated that the model will fit the EASY class while classifying all the other examples, including "anomalies", as the HARD class, essentially behaving like an anomaly detector. Thus, we can formulate the following hypothesis:

Hypothesis 1 *Given an imbalanced binary classification, the model will behave as an anomaly detector by fitting the EASY class.*

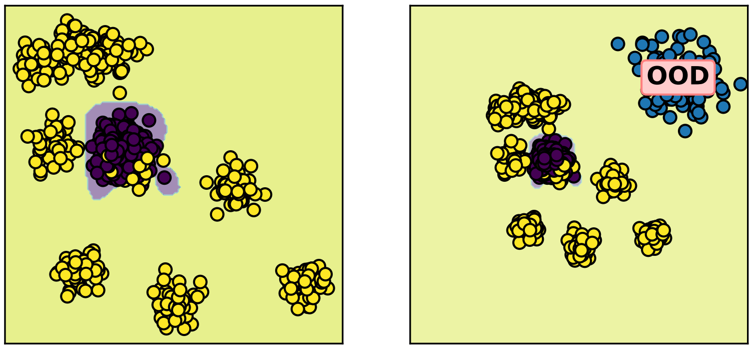


Figure 5.7: **Toy example of imbalanced binary classification and sensitivity to OOD**. **(Left)** A toy binary classification task with a balanced number of instances, but imbalanced number of modalities. The classifier behaves as an anomaly detector (hypothesis 1) by fitting the distribution of the target with one modality only (the EASY class). **(Right)** Given the uneven distribution of modalities of the two target classes and the anomaly detection behaviour of the model (hypothesis 1), we expect OOD to fall in the HARD class (in yellow).

A toy example of this hypothesis depicting two imbalanced classes is reported in fig. 5.7, and will be further discussed in the following paragraphs.

The first step towards investigating our hypothesis is understanding which source of imbalance might be responsible for the observed Epistemic Gradient. Given that the TITAN training set has an even size of target classes, we decided to check if the imbalance was due to the complexity of the two classes. For the TCR–epitope binding problem, for example, we can have a great variety of molecules that do not bind, while conversely only in few conditions they do bind. In the following, we will try to reproduce the variety of the non-binding class by assigning an uneven number of modalities, or clusters, to each of the targets. For example, the HARD class (the non-binding points) could have 15 modalities, while the EASY class (the binding class) only one. We can then formulate a second hypothesis:

Hypothesis 2 *The number of modalities of a class is correlated with its complexity.*

In order to validate hypotheses 1 and 2, in section 5.5.3 we considered a case study on images, employing CIFAR-100 [Kri09]. More in detail, the CIFAR-100 dataset is composed by 20 *superclasses* (flowers, fish, . . .), each one containing 5 classes (e.g., the fish superclass contains the aquarium fish, flatfish, ray, shark, and trout). In section 5.5.2, we will investigate a setting in which the dataset is strongly imbalanced: 15 superclasses vs 1. On the other hand, in section 5.5.3 we will perform additional analysis with only one modality for each target.

Since we believe that the number of clusters or modalities serves as an indicator of class complexity, we also want to assess the impact of uneven modalities on the Intrinsic Dimensionality (metric 3, introduced in section 5.2.1) which was previously used to measure sample complexity in previous studies [Pop+21; KKL22].

Moreover, to test the anomaly detection behaviour (hypothesis 1), we also considered the sensitivity to OOD data (metric 4, defined in section 5.3.3). Referring back to the toy example in fig. 5.7, on the left we have a binary classification task with imbalanced modalities, where the classifier fits the EASY class as an anomaly detector would. On the right, we represented our hypothesis, where OOD data always fall in the HARD class. A similar application of sensitivity to OOD was employed in [BL19] to study the effect of importance weighting in binary classification.

5.5.1 Experimental Setup

Model

For all the experiments, we considered a standard Convolutional Neural Network with two convolutional layers (30 and 15 features maps each, respectively), each followed by a max-pooling layer. After the feature extraction, a classifier head of two fully connected layers (256 units each) was used for the classification. The models were trained using the Adam optimizer, learning rate of 0.001 and early-stopping.

Dataset

In the following settings, we will consider an imbalanced scenario with 10 randomly generated binary classification tasks where one target is composed by 15 superclasses of CIFAR-100 [Kri09] (the HARD class) and the other by only one (the EASY class). In the balanced case, on the other hand, we will assign just one superclass to each target.

To check the sensitivity to OOD, we will first consider the SVHN [Net+11] dataset only. Then, similarly to [LLS18], we will also take into account the LSUN [Yu+15] dataset, by either centre cropping the images or resizing them down to 32×32 pixels. Moreover, we generated images by sampling images from a uniform distribution in $[0, 1]$. Each dataset was downsampled to 10,000 instances.

5.5.2 Controlling Class Complexity with Modalities

In the following, we will present the result of the experiments performed to investigate hypothesis 2. In particular, we will check whether modalities are correlated with the ID.

Dynamics of the Intrinsic Dimensionality during Training

Before considering the Intrinsic Dimensionality (ID) as a proxy to estimate target complexity, we want to study its dynamic during training. In particular, a necessary requirement is that the dimensionality reaches a stable point that we can consider as a complexity estimate. Our analysis complements the work of Ansuini et al. in [Ans+19], even though we focus on small CNNs rather than large-scale CNNs.

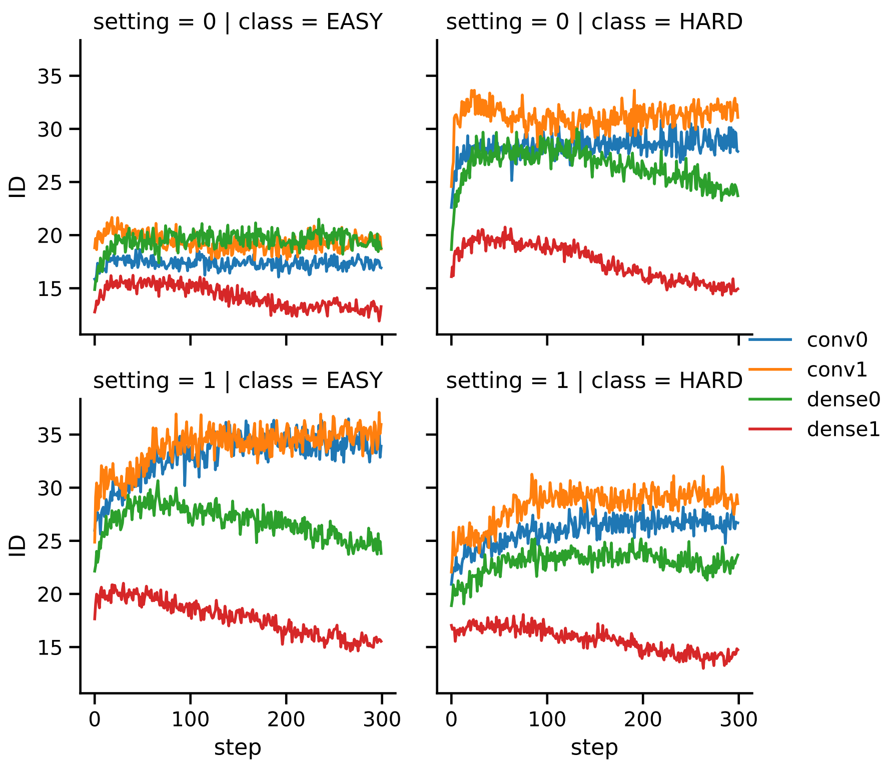


Figure 5.8: **Evolution of the intrinsic dimensionality during training**. Evolution of the intrinsic dimensionality for two different imbalanced settings (rows) of 15 against 1 superclasses, for the HARD and EASY class, respectively (columns). Different colors correspond to different layers of the CNN in which the intrinsic dimensionality was computed, while the columns correspond to the EASY and HARD class, respectively. The first two convolutional layers (`conv0` and `conv1`) have always the highest intrinsic dimensionality, apart from the top-left case, while the last two layers (`dense0` and `dense1`) show a dimensionality compression phenomenon later in training.

In fig. 5.8, we reported the ID in the latest space for two different settings (one for each row), for both the EASY and HARD class (the first and second column, respectively). The ID always reaches a stable point (up to negligible variations), in particular in the first two layers (`conv0` and `conv1`). Moreover, as observed in [Ans+19] the last layers have a smaller ID than the previous ones. However, given that we are considering small CNNs, the highest ID is achieved in the first two layers, rather than in the intermediate ones as in large-scale CNNs [Ans+19].

Another interesting effect, is that the ID of the first two layers shows a monotonic increase in three out of four plots, until reaching a plateau, while the last two layers (in particular the last one) show the opposite behaviour. These preliminary results, in addition to [Ans+19], make the dynamics of the ID in the latent space an interesting line of research to investigate the behaviour of DNNs in the latent space.

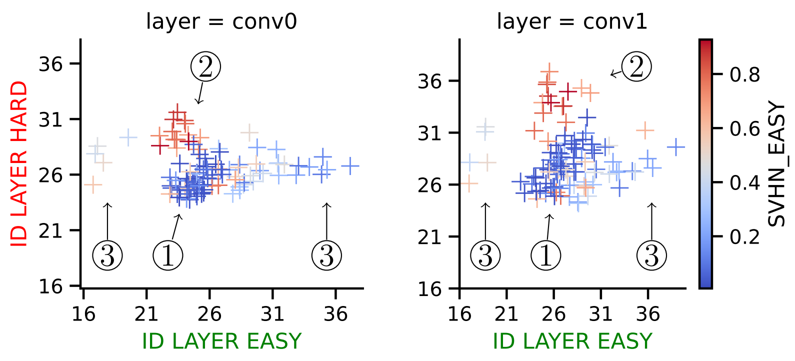


Figure 5.9: **Intrinsic dimensionality and sensitivity to OOD**. Intrinsic dimensionality of the EASY (x-axis) and HARD (y-axis) class of the CNN last layer's latent features. Each point is a different imbalanced dataset with 15 against 1 superclass, for the HARD and EASY class, respectively. The color represents the sensitivity of the EASY class to OOD data, in this case the SVHN [Net+11] dataset. In red the settings where OOD instances are mostly assigned to the EASY class, the opposite for the blue points. We observe a cluster with low sensitivity of the EASY class (①), in contrast to the ones with high sensitivity (②). In ③, SVHN is assigned evenly to the EASY and HARD class.

Lastly, we observe that in setting 0 (first row) the ID of the HARD class is always higher than the EASY class, while in setting 1 (second row) we have the opposite scenario. These results suggest that an uneven distribution of modalities is not sufficient to control the class complexity, refuting hypothesis 2. In this regard, in the next section, we discuss additional results on the relation between modalities and dimensionality.

Intrinsic Dimensionality, Modalities, and Sensitivity to OOD

In fig. 5.9, we show the results of our investigations into hypotheses 1 and 2. First, to validate hypothesis 2, we consider the ID of the EASY (x-axis) and HARD (y-axis) jointly, for both the first and the second layer. Moreover, for each setting, we computed the sensitivity to the SVHN [Net+11] dataset (in this case, an OOD dataset) of the EASY class, to control for hypothesis 1. A blue point means low sensitivity, and therefore the OOD data is assigned to the HARD class, while the opposite holds for the red coloured points.

As we observed in the previous section, we have cases in which the EASY class has a greater ID than the HARD class, refuting hypothesis 2.

Regarding the sensitivity to OOD, we can distinguish three situations:

① The majority of the settings have low sensitivity (in blue).

② Points with high ID of the HARD class have high sensitivity (in red).

③ Points with either the lowest or highest ID of the EASY class, represent settings where OOD data is assigned evenly between the EASY and the HARD classes.

In summary, there appears to be a connection between the ID and sensitivity to OOD data, however, further experimentation is needed to fully understand this phenomenon.

5.5.3 Sensitivity to OOD Data with Even Modalities

Based on the previous section's results, the first conclusion we can draw is that the number of clusters or modalities is either not a reliable measure of class complexity, which would refute hypothesis 2, or the ID is not a suitable metric for our purposes. It is noteworthy that in [KKL22] it has been found that the entanglement between classes should also be taken into account when estimating sample complexity.

Since the number of modalities did not correlate with the dimensionality, we cannot consider these results to be conclusive in regard to hypothesis 1. In the following, we will further investigate the sensitivity to OOD data by using a training set with an even number of modalities and different OOD datasets. Therefore, only one superclass (e.g., "fish" vs "flowers") will be assigned to each of the two targets.

Sensitivity to OOD Data is Dataset Dependant

In fig. 5.11, we studied the sensitivity to four types of OOD: from top-left to bottom-right, SVHN [Net+11], LSUN [Yu+15] cropped and resized to 32×32 and uniform noise OODs. The x and y axis denote the superclass assigned to each of the two targets, while the colour represents the percent of OOD instances that are assigned to the class in the y-axis.

A subset of rows correlates between SVHN [Net+11] and LSUN [Yu+15] (cropped). For example, "flowers" and "fruit_and_vegetables" are blue coloured against all classes in both the OOD datasets. The uniform dataset (bottom-right corner), on the other hand, shows a completely different behaviour than all the other three datasets.

Another interesting observation, is that while the uniform dataset has a sensitivity equal to 0 or 1 in the majority of the cases, the other datasets have a more uniform distribution of values in $[0, 1]$. Since the sensitivity is computed based on the model's predictions, it mainly depends on the position in the input space of the OOD dataset with regard to the decision boundary (as depicted in fig. 5.10). Consequently, while SVHN, LSUN cropped and resized sometimes fall above the boundary, leading to sensitivity between 0 and 1, the uniform dataset is mostly far from the boundary, resulting in exactly 1 or 0 as sensitivity.

Based on these results, we can classify OOD data in two groups: the ones that keep the semantics of the training set (in this case, a natural image), and can fall *above* the decision boundary, and the ones that don't have a specific semantic, like the uniform dataset, and fall away from the boundary. Overall, this analysis is

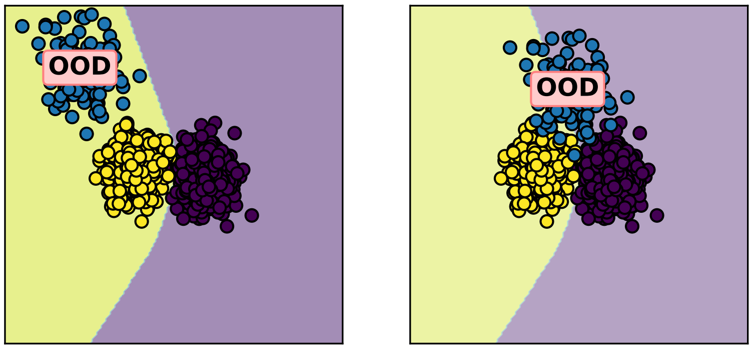


Figure 5.10: **OOD sensitivity and the decision boundary**. Visualization of the different results we get between SVHN and LSUN (cropped and resized) versus the uniform dataset in fig. 5.11. On the left we have the situation of the uniform dataset, where the OOD data is all on one side of the boundary. In this case, the sensitivity of the purple class is 0, while for the yellow class is 1. On the right we have a different situation, in which the OOD data is above the boundary and therefore the sensitivity to OOD data is between 0 and 1 for both classes.

interesting from both the point of view of OOD detection and for understanding the decision boundary of DNNs.

Moreover, it would be interesting to investigate the properties of the "flowers" and "fruit_and_vegetables" classes, that make them have low sensitivity to OOD data.

Lastly, the high variance of the sensitivity to OOD that we observed among different datasets, suggests that we should carefully choose the properties of the OOD dataset to validate the anomaly detection behaviour of the classifier we hypothetized in the previous section (hypothesis 1).

Epistemic Gradient and OOD Sensitivity

In this last analysis, we go back to the Epistemic Gradient we observed in section 5.4.2. In fig. 5.12, we repeated the analysis done for fig. 5.6, for all the settings. More in detail, we computed the epistemic uncertainty using the aK-LPE algorithm in the last layer of the CNN (with the same hyperparameters used in the previous section) and correlated it with the target class. We remark that we consider the class in the y-axis as the EASY class and the class in x-axis as the HARD.

Again, the "flowers" and "fruit_and_vegetables" clearly emerge as having the higher correlation. In the TITAN architecture, we hypothetized this phenomenon as an effect of the model fitting only on the EASY class (hypothesis 1). Furthermore, this result on the epistemic uncertainty, taken together with what we saw in fig. 5.11, make the "flowers" and "fruit_and_vegetables" classes a valuable case study to investigate the Epistemic Gradient phenomenon.

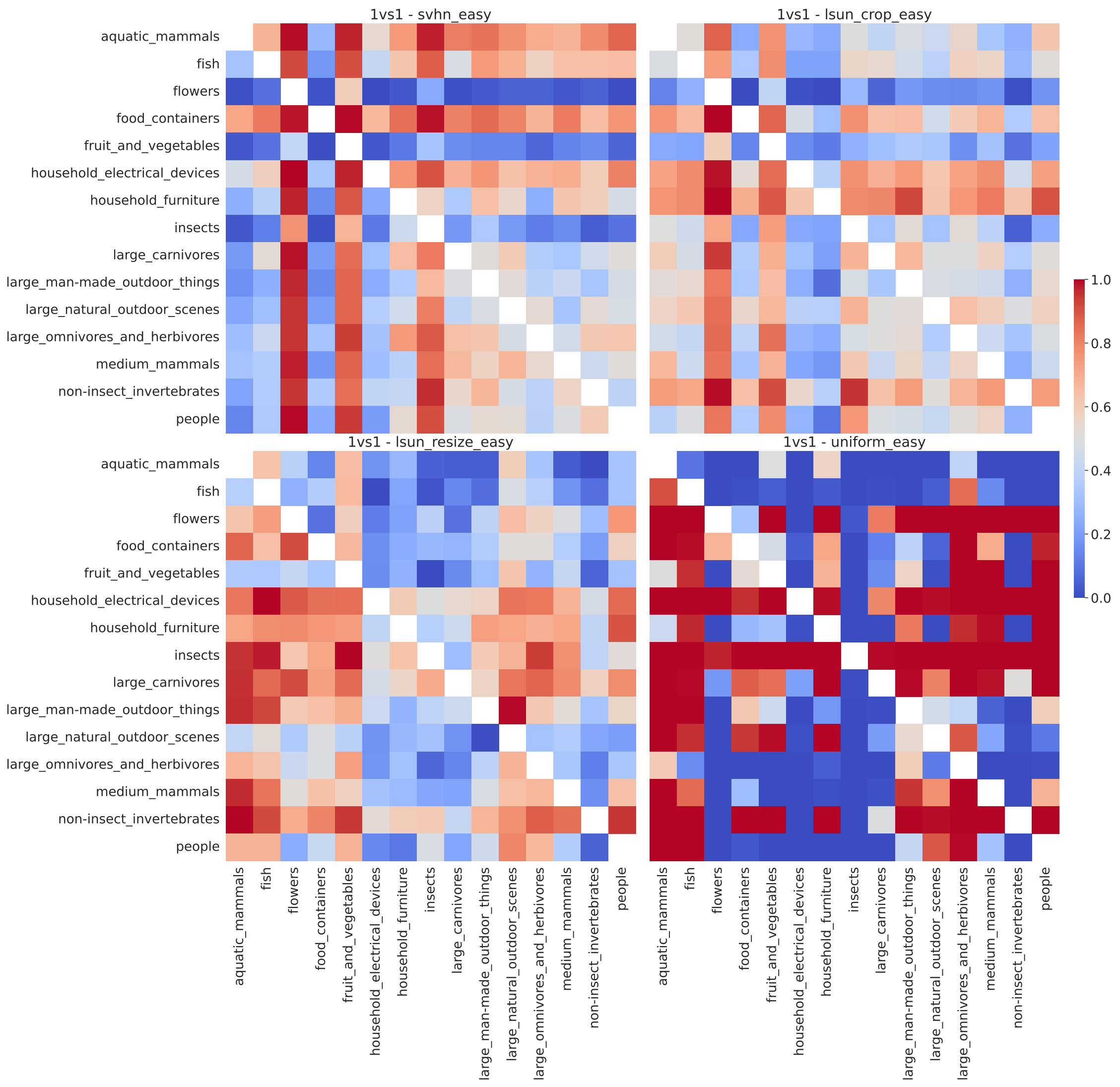


Figure 5.11: **OOD sensitivity in target class dependant**. OOD sensitivity of the class in the y-axis (the EASY class) when changed the other target class, in the x-axis, with regard to the binary classification task. Each heatmap corresponds to a different OOD dataset: from top-left to bottom-right, SVHN [Net+11], LSUN [Yu+15] cropped and resized to 32×32 and uniform noise OODs.

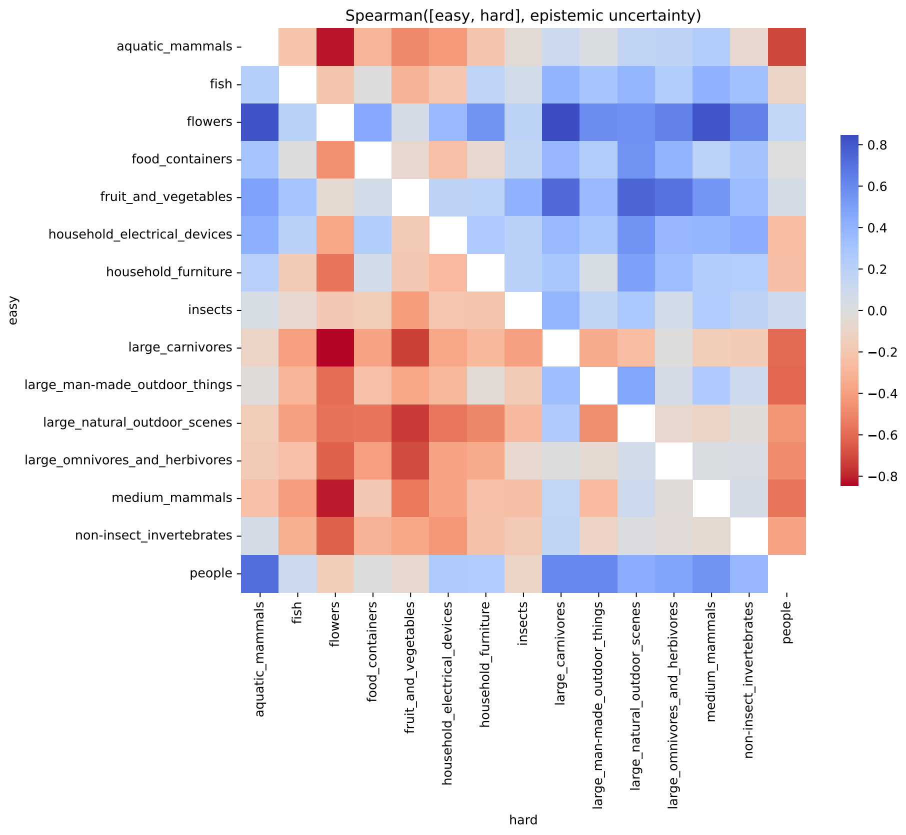


Figure 5.12: **The Epistemic Gradient is correlated with the targets**. Spearman's rank correlation between target class and the epistemic uncertainty, measured with the aK-LPE algorithm as done in section 5.4.2. Each cell corresponds to a binary task using the superclass in the y-axis and in the x-axis. Interestingly, the "flower" and "fruit_and_vegetables" superclasses have the epistemic gradient in most of the cases (high Spearman correlation), while also having low sensitivity of OOD in fig. 5.11.

5.6 Conclusions

In this chapter we discussed the work done during my visiting at the Computational Systems Biology lab at IBM Research Zürich, supervised by Dr. María Rodríguez Martínez and Dr. Nicolas Deutschmann. The project involved investigating imbalanced binary classification tasks from the point of view of uncertainty and complexity.

We presented two different case studies, TCR-epitope binding affinity prediction (section 5.4.1) and image classification (section 5.5), where we tested four different metrics to evaluate the behaviour of the model and the complexity of the classes: the Epistemic and Aleatoric uncertainties (metrics 1 & 2, section 5.3.1), the Intrinsic Dimensionality (ID) (metric 3, section 5.2.1) and the sensitivity to Out-of-Distribution (OOD) data (metric 4, section 5.3.3).

The results on TCR-epitope binding affinity prediction with the TITAN [WBR21] deep model showed that the binding points have a lower epistemic uncertainty than the non-binding points. We called this phenomenon Epistemic Gradient, and claimed that it is a result of the imbalance of the data. In fact, binding affinity prediction resembles an anomaly detection task, where the non-binding points are considered as anomalies.

In order to further investigate the Epistemic Gradient, and also study how we can employ the ID and sensitivity to OOD to interpret the behaviour of deep models in imbalanced tasks, we defined a second case study on image classification. More in detail, we built imbalanced datasets using the superclasses in CIFAR-100 [Kri09] and showed that, in contrast to our expectation, an uneven number of modalities (classes) in each of the two targets does not correlate with the ID. Moreover, we evaluated the sensitivity to OOD data both with an even and uneven number of modalities in each target class. The result showed that the choice of the OOD dataset strongly impacts the final results, since only for a subset of the settings the sensitivity is comparable among different OOD datasets. Interestingly, we observed that for a group of settings, the sensitivity is correlated with the presence of the Epistemic Gradient, motivating additional experimental analyses.

Many research direction can be considered from this preliminary results. First, it would be interesting to further investigate the factors contributing to the ID of data, that we started by taking into account the number of modalities. Additional factors to consider could be the within-class imbalance [Jap01], or the amout and kind of noise in the data. The final objective would be to employ the ID to investigate real-world datasets such as TCR–epitope binding datasets. Second, inspired by the results from [KKL22], we should take into account also the entanglement of the classes when estimating the complexity of data and the causes of the Epistemic Gradient phenomenon.

In conclusion, we think that relating uncertainty, intrinsic dimensionality and sensitivity to OOD is a valuable direction to better understand imbalanced binary classification tasks. In particular, this might help in distinguishing when binary classification and anomaly detection are comparable to when one should be preferred to the other. While for datasets with imbalanced class size this is more trivial (anomaly detection works best when one class has not enough samples to train a classifier), for data that is imbalanced in terms of complexity, to the best of our knowledge, there is still no clear guideline.

Chapter 6

Conclusions

In this thesis, we have discussed three perspectives on anomaly detection in deep learning. The research question was how to exploit properties of Deep Neural Networks (DNNs) to characterize anomalies, and how this result could help us towards a more interpretable and safe AI.

We approached our objective from three different angles, as summarized in fig. 1.1. In particular, we investigated different data modalities (images and sequences), distinct types of anomalies (adversarial, rare or uncertain examples) and various evaluation metrics (misclassified example prediction, adversarial detection, and more). Regarding the similarities between our approaches, all of them used the latent features of the models as relevant knowledge for the detection task.

The main Contributions (C), corresponding to the different chapters, are the following:

C1) The definition of the Activation Pattern DAG (APD) [Cra+20a; Cra+20b], a novel method to perform example difficulty estimation in piecewise linear DNNs (chapter 3).

C2) The design and development of the ENsemble Adversarial Detector (ENAD) [Cra+21], a new approach for adversarial example detection in Convolutional Neural Networks (chapter 4).

C3) Uncertainty estimation and sample complexity analysis on deep models for imbalanced binary classification tasks in two different case studies (chapter 5).

Throughout the thesis, we presented many results contributing to the goal of Responsible AI [Arr+20], discussed in section 2.4:

- **Visualizations** of the input data clustering induced by activation patterns in fig. 3.8 and of the combined anomaly score from multiple layers in fig. 4.5.
- **Understanding trends in data**, by identifying challenging points in chapter 3, adversarial examples in chapter 4 and by analysing the uncertainty distribution among targets in chapter 5.

- **Interpreting the importance of hidden layers**, by investigating the distribution of activation patterns in fig. 3.6 and fig. 3.8, by comparing the detection performances of different layers in fig. 4.3b, and by comparing the Intrinsic Dimensionality (ID) in different layers in fig. 5.8.

Contributions

Before discussing the impact of the contributions to the field of anomaly detection in deep learning, we will describe each of them in detail in the following.

C1) Definition of the Activation Pattern DAG (APD) [Cra+20a; Cra+20b] is a Directed Acyclic Graph (DAG) summarizing the activation patterns of a piecewise linear DNN on a given dataset. Starting from the APD, we designed a new clustering algorithm that partitions examples into clusters of different sizes. By performing experiments on the MNIST [LCB10] dataset and different architectures, we showed that the cluster size is a proxy of example difficulty. In particular, misclassified and challenging instances (mostly) occur in smaller clusters. Moreover, we proved that selecting one representative for each cluster hurts less the generalization performance than choosing examples at random, confirming that our clustering algorithm is able to group similar instances.

C2) The ENsemble Adversarial Detector (ENAD) [Cra+21] is a novel ensembling technique to detect adversarial examples from the latent features of Convolutional Neural Networks (CNNs), based on the intuition that distinct detectors are able to capture different properties of the input data. Our method achieves state-of-the-art performance, after a comprehensive evaluation on 3 benchmark vision datasets (CIFAR-10 [Kri09], CIFAR-100 [Kri09], SVHN [Net+11]), 2 state-of-the-art CNNs (DenseNet [Hua+17] and Resnet [He+16]), 4 adversarial attacks (FGSM [GSS15], BIM [KGB17], DeepFool [MFF16] and CW [CW17b]), and 2 experimental scenarios (testing on a known and unknown adversarial attack). Our method not only achieves state-of-the-art performance, but is also defined to allow an easy extension by adding additional detectors to the ensemble and set a possible foundation for the adoption of ensemble approaches in adversarial detection.

C3) Project investigating the behaviour of deep models on imbalanced binary classification tasks, where the source of imbalance is not the uneven proportion of the classes, but rather their uneven complexity. One of our most important claims is that a binary classifier on imbalanced data will fit the easier class only, while considering the other class as “the rest”, behaving like an anomaly detector. To delve into our hypotheses, we selected four metrics to interpret the model and estimate the input complexity: the Epistemic and Aleatoric uncertainties (impacted by lack of data and randomness, respectively), the

Intrinsic Dimensionality, and the sensitivity to Out-of-Distribution (OOD) data. Moreover, we considered two different case studies: predicting the binding affinity between T-cell receptor (TCR) and epitopes and image classification. Interestingly, we observed that the deep model we used for the first study (TITAN [WBR21]) had the epistemic uncertainty correlated with the target class, a phenomenon we named "*Epistemic Gradient*". In the second case study we tried to investigate the sources of data imbalance, like the number of modalities in each class, and how a binary classifier trained on such data behaves with regards to OOD data. Preliminary results opened possible research lines to understand the causes of the Epistemic Gradient.

We want to emphasize that all code used to obtain these results was developed following the principles of open science, and is publicly accessible through various repositories referenced in the text. Reproducibility of research and results is a crucial aspect in artificial intelligence and computational sciences, and it is important to prioritize it.

Impact

In this thesis, various techniques were presented to identify abnormal inputs for a trained Deep Neural Network (DNN). These techniques can be utilized in various ways such as: identifying misclassified inputs (**C1**), detecting adversarial examples (**C2**) and recognizing when the model is operating as an anomaly detector (**C3**). In the following, the potential real-world applications and improvements to current state-of-the-art techniques that result from our contributions will be discussed.

The Activation Pattern DAG (**C1**) is a new, valuable tool in efforts to "open the black-box" of DNNs [Pet+21; OMS17], as we leveraged the properties of piecewise linear networks to provide both an interpretation and a visual representation of the learned function. Unlike previous works that used activation patterns to assess the expressivity of the model [HR19b; STR18], the APD contributes to the research line exploiting activation patterns to evaluate the model's performance [Nov+18; Ji+22a]. Use-case scenarios of the APD are: automating data auditing [ADH22] by ranking input examples by difficulty, to develop human-in-the-loop pipelines [Lei+17] or to re-train the model using curriculum learning [Ben+09], and data visualization, to understand and interpret how a deep model process the input data (see section 3.4.3).

The research on the ENsemble Adversarial Detector (ENAD) (**C2**) advanced the field by demonstrating the ability of ensembles to enhance adversarial detection performance in Convolutional Neural Networks (CNNs) compared to state-of-the-art methods. Our evaluation of standalone detectors can serve as a guide for future enhancements to the ENAD, which, due to its versatility, enables the seamless integration of additional detectors. ENAD can be applied to pre-trained CNNs to enhance their robustness and safety, making it suitable for safety-critical computer vision applications such as medical imaging [Ma+21; KHS22].

In chapter 5, we presented an analysis of deep models for imbalanced binary classification scenarios. Our preliminary analysis can be considered a first step towards an interpretability toolkit focused on imbalanced classification problems. The first use-case is to detect uneven class complexities through intrinsic dimensionality estimators [Fac+17], recently employed to study the latent representation in deep models [Ans+19]. Second, we showed how to employ uncertainty estimation and Out-of-Distribution data to interpret the behaviour of deep models on imbalanced data. Given the recent interest towards Out-of-Distribution data detection and generalization [Yan+22], even in topic-specific scenarios [Ji+22b], as we observed many times throughout this thesis, we consider our analysis timely and valuable to the field. Furthermore, we remark the importance of studying imbalanced classification tasks, given their frequency in real-world applications such as medical diagnosis [FAK19].

Limitations

In this thesis, we mostly focused on image datasets, except the binding affinity prediction task in chapter 5. Moreover, all the image datasets we considered are benchmark data and not real-world datasets. Hence, applying our methods in practical use-case scenarios should take into consideration additional tests and adaptations.

Going through the specific limitations of each contribution, the APD (**C1**) was tested only on vanilla small-scale Feedforward Neural Network, therefore additional experiments should take into account more complex datasets, standard regularization strategies (e.g., Dropout [Sri+14] and Batch Normalization [IS15]) and larger models.

ENAD (**C2**) ensembles multiple detectors to improve the adversarial detection performance. The main downside is that fitting the overall pipeline requires training each detector, scaling at worst as the sum of each training time (results as shown in section 4.4.4). For this reason, a trade-off between resources and performance should be considered. Moreover, as noted in the transfer attack experiments (section 4.4.2), fitting the hyperparameters and the logistic aggregator function might be challenging due to the problem of selecting representative anomalies. In our experiments, we found that choosing harder attacks improve the generalization performance of the detector, although further experiments should confirm our results in a broader repertoire of adversarial attacks and datasets.

Lastly, our analysis on imbalanced dataset in chapter 5 (**C3**) should be considered a preliminary investigation of the problem, given the novelty of the research and the number of open questions that still have to be investigated. In particular, we are still trying to understand the reasons behind the Epistemic Gradient phenomenon (see section 5.4.2) and the factors correlated with the intrinsic dimensionality of a class in the latent space of a DNN.

Future Work

Example difficulty estimation (**C1**), anomaly detection (**C2**) and uncertainty estimation (**C3**) are three approaches to quantify the properties of anomalous instances in deep learning. In this thesis, we used them for different tasks, like misclassification prediction or adversarial detection, although we did not investigate the correlation between the different scoring methods. A challenging and important future work could focus on how scoring methods differ with regard to their assumptions and properties. For example, an interesting first analysis could classify each approach based on the type of uncertainty that predominantly impacts it, either the Epistemic or the Aleatoric Uncertainty (due to lack of data and randomness, respectively). While for the APD ranking further experiments have to evaluate the correlation with the Uncertainty type, we claim that ENAD is mostly affected by the Epistemic Uncertainty, since it measures how much an instance is isolated in the latent space.

All the three methods we proposed are data agnostic, since they either work with the latent features (all of them) or by considering the softmax output of the network (**C3**). As we already addressed in the limitations, future analysis could consider the validity of our approaches with different data modalities, such as sequences or tabular data.

Examining the future research lines for each contribution in detail, we tested the APD (**C1**) only in one small-scale scenario, therefore a large batch of experiments should be considered. Moreover, inspired by recent results [Ji+22a], we could revisit the clustering algorithm to consider smoothed clusters, by employing a kernel and a proper distance for activation patterns, such as the Hamming distance.

Regarding ENAD, future work should consider novel aggregation schemes first. We already tested voting schemes in section 4.4.1, but alternative approaches could consider other meta-learners beside the logistic regression, such as OCSVM [Sch+99] or the aK-LPE algorithm [QS12], as done in [Rag+21]. Furthermore, it would be interesting to formalize why the best performing layer and detector, as we studied in section 4.4.1, change depending on the chosen adversarial attack.

Finally, our preliminary analysis of imbalanced datasets (**C3**) can be developed in many ways. Key areas for further studies are the Epistemic Gradient phenomenon (section 5.4.2) and its relation with the sensitivity to OOD (section 5.5.3), and using the intrinsic dimensionality to quantify the target class complexity (section 5.2.1). One line of research could also consider the classes' entanglement to evaluate their complexity, as suggested in [KKL22]. Another work could compare the uncertainty of a binary classifier and a model trained with a one-class loss function, such as Deep-SVDD [Ruf+18], to control if an imbalanced dataset makes the two models behave similarly, with regard to our analysis, like we claimed in hypothesis 1.

Appendix A

Interdisciplinary Publications in Computational Biology

In addition to my efforts in developing new theoretical approaches and methods for anomaly detection, I also worked on the implementation and application of deep and machine learning methods to real-world problems. In particular, I have collaborated to the following three projects in computational biology:

- Denoising and Imputation of Single-Cell Transcriptomic Data (appendix A.1).
- Classifying Cancer Samples from Metabolic Networks (appendix A.2).
- Deep Learning for Predicting Relative Fluxes in Reaction Systems (appendix A.3).

A.1 Denoising and Imputation of Single-Cell Transcriptomic Data

Contribution. In this chapter I will discuss the work done for the following articles:

[Pat+20] L. Patruno, D. Maspero, F. Craighero, F. Angaroni, M. Antoniotti, A. Graudenzi. "A Review of Computational Strategies for Denoising and Imputation of Single-Cell Transcriptomic Data". In: *Briefings in Bioinformatics* 22.4 (Oct. 2020)

Summary. Single-cell RNA sequencing (scRNA-seq) data is now widely adopted [Vie+19] given its high single-resolution, providing insights into cell population heterogeneity that were not previously achievable with traditional bulk sequencing [KBQ11]. However, single-cell sequencing protocols are still characterized by experimental limitations such as capture efficency [Haq+17] or the batch effect [GWW17] that may lead to noisy measurements. Consequently, a zero in the data could be both a non-expressed gene or a missing read, while non-zero values could be different from the true transcript abundance. To solve these issues, a number of imputation methods that recover missing values, and denoising methods that correct wrong measurements have been recently proposed. In order to investigate the strengths and weaknesses of these methods, and provide guidelines for their use, we performed a comprehensive evaluation of 19 imputation and denoising methods. Our contributions can be summarized as follows:

- *Extensive benchmarking on both synthetic and real datasets.* We compared each method using synthetic data generated using the tool SymSim [ZXY19] and four real-world datasets.
- *Comprehensive evaluation of the methods under different objectives.* We characterized the quality of each method from different perspectives, such as their ability to recover the true expression value or their reliability in respecting cell-similarity.
- *Selection of the best performing methods among* 19 *tested.* We reduced the 19 methods to a selection of 4 that, in our experiments, performed best in the majority of the tests.
- *Operative guidelines for practitioners.* We provided guidelines for practitioners based on the different qualities of each method, including the scalability and the usability of the code.

Implementation. The experiments performed in the paper has been opensourced on a Github repository[a].

[a] `https://github.com/BIMIB-DISCo/review-scRNA-seq-DENOISING`

Briefings in Bioinformatics, 22(4), 2021, 1–19

https://doi.org/10.1093/bib/bbaa222
Method Review

A review of computational strategies for denoising and imputation of single-cell transcriptomic data

Lucrezia Patruno, Davide Maspero, Francesco Craighero, Fabrizio Angaroni, Marco Antoniotti and Alex Graudenzi

Corresponding authors: Marco Antoniotti, Department of Informatics, Systems and Communication, University of Milan-Bicocca, Milan, Italy. Tel: +39 0264487901; E-mail: marco.antoniotti@unimib.it; Alex Graudenzi, Institute of Molecular Bioimaging and Physiology, Consiglio Nazionale delle Ricerche (IBFM-CNR), Segrate, Milan, Italy. Tel: +39 0221717551; E-mail: alex.graudenzi@ibfm.cnr.it
Lucrezia Patruno and Davide Maspero are equal contributors. Marco Antoniotti and Alex Graudenzi are co-senior authors.

Abstract

Motivation. The advancements of single-cell sequencing methods have paved the way for the characterization of cellular states at unprecedented resolution, revolutionizing the investigation on complex biological systems. Yet, single-cell sequencing experiments are hindered by several technical issues, which cause output data to be noisy, impacting the reliability of downstream analyses. Therefore, a growing number of data science methods has been proposed to recover lost or corrupted information from single-cell sequencing data. To date, however, no quantitative benchmarks have been proposed to evaluate such methods. **Results.** We present a comprehensive analysis of the state-of-the-art computational approaches for denoising and imputation of single-cell transcriptomic data, comparing their performance in different experimental scenarios. In detail, we compared 19 denoising and imputation methods, on both simulated and real-world datasets, with respect to several performance metrics related to imputation of dropout events, recovery of true expression profiles, characterization of cell similarity, identification of differentially expressed genes and computation time. The effectiveness and scalability of all methods were assessed with regard to distinct sequencing protocols, sample size and different levels of biological variability and technical noise. As a result, we identify a subset of versatile approaches exhibiting solid performances on most tests and show that certain algorithmic families prove effective on specific tasks but inefficient on others. Finally, most methods appear to benefit from the introduction of appropriate assumptions on noise distribution of biological processes.

Key words: denoising; imputation; single-cell RNA-sequencing; machine learning

Lucrezia Patruno is a PhD student in computer science at the Department of Informatics, Systems and Communication of the University of Milan-Bicocca. Her studies focus on data analysis and machine learning methods for the study of complex biological phenomena.
Davide Maspero is a PhD student in computer science at the Department of Informatics, Systems and Communication of the University of Milan-Bicocca. His studies focus on data integration methods for complex biological systems.
Francesco Craighero is a PhD student at the Department of Informatics, Systems and Communication of the University of Milan-Bicocca. His research is devoted to deep learning and explainable AI, with the aim of explaining deep networks inner sparse representations.
Fabrizio Angaroni is a postdoc researcher at the Department of Informatics, Systems and Communication of the University of Milan-Bicocca. His research is focused on mathematical methods for modeling and data analysis of complex biological system.
Marco Antoniotti is an associate professor at the Department of Informatics, Systems and Communication of the University of Milan-Bicocca. His main research topics are bioinformatics, computational systems biology, simulation, verification and cancer data analysis.
Alex Graudenzi is a research fellow at the IBFM-CNR. His research integrates (bio)informatics, complex systems, statistics and systems biology to deliver computational methods for the investigation of complex biological phenomena.
Submitted: 26 May 2020; **Received (in revised form):** 7 August 2020

Introduction

In recent years, an increasing number of studies has involved data generated from single-cell RNA sequencing (scRNA-seq) experiments [1, 2], which quantify gene expression levels at single-cell resolution, thus providing insights into cell population heterogeneity [3]. scRNA-seq methods can be used to perform accurate transcriptome quantification with a relatively small number of sequencing reads, isolating a typically large number of single cells. In optimal conditions, scRNA-seq data can recapitulate the results of standard sequencing experiments from bulk samples, yet with a much higher resolution [4].

This is a great advantage, as many works report that even cells in a homogeneous population may have heterogeneous expression profiles [5–8]. For instance, scRNA-seq data can be used to characterize rare cell subpopulations that had been hidden in the output of bulk RNA sequencing experiments [9], as well as in the analysis of cancer evolution, where they can be exploited to study the heterogeneity of tumor cell subpopulations [10] and the processes that lead to drug resistance or metastasis [11]. The wide use of scRNA-seq technologies has also allowed the creation of cell atlases for simple organisms such as, for example, the *Caenorhabditis elegans* [12]; most importantly, there is an ongoing effort to create such map for the human organism, i.e. the Human Cell Atlas [13]. However, the analysis of single-cell sequencing data is affected by the complex combination of biological variation and technical noise, which typically result in sparse and noisy single-cell expression profiles.

On the one hand, stochasticity of gene expression is inherent in most biological systems, with respect to both the biochemical processes related to gene regulation and the fluctuations of other cellular components and phenomena [14]. For this reason, even cells of the same type within the same tissue may display different gene expression distributions, complicating the identification and characterization of cellular states and transitions [15].

On the other hand, currently available sequencing technologies are still hindered by various technical issues [2, 16, 17]. In particular, the most common approaches for scRNA-seq are based on either droplet platforms (e.g. Drop-seq [18], InDrop [19] and Chromium 10x [20]) or plate-based platforms (e.g. Smart-Seq2 [21], MATQseq [22], MARS-seq [23], CEL-seq [24] and SPLIT-seq [25]), while some further approaches rely on microfluidics (e.g. C1 SMARTer [26]) or nanowell arrays (e.g. SEQ-well [27]). Typically, droplet platforms allow to isolate a large number of single cells (from a few to many thousands), by sequencing the 3′-end and by employing unique molecular identifiers (UMIs) [28], which allow the tagging of each transcript before amplification, thus distinguishing original transcripts from amplification duplicates [29]. Conversely, plate-based platforms usually employ full-length sequencing protocols and, accordingly, allow to sequence a much lower number of single cells (∼hundreds), yet with a considerably higher coverage. Overall, all sequencing protocols are affected by a number of technological and experimental issues, which typically result in noisy measurements.

- Capture efficiency: due to (*i*) the low quantity of RNA in a given single cell, and (*ii*) the stochastic nature of gene expression patterns at the single-cell level, certain gene can display null expression level, since none of its transcripts may be captured, thus resulting in zero expression levels. These are the so-called dropout events [30] and might be particularly relevant for scarcely expressed genes. This issue causes both noise and a high sparsity in the data [9].
- Amplification bias: the amplification phase may be subject to potential PCR biases in the quantification of the abundance of each gene, such as preferential amplification of certain templates. UMI-based approaches are able to mitigate this issue, yet in any case, amplification biases can be a potential source of noise in the data.
- Sequencing depth: the number of sequenced reads per cell varies between different experimental settings and platforms, and this can result in noisy and sparse outputs, especially when the depth is relatively low [29].
- Batch effects: technical sources of systematic variation may add a confounding factor in downstream analysis. Batch effects can be generated by analyzing samples with different technologies, in different laboratories or in different runs [31, 32]. When multiple experiments are considered, it is appropriate to remove such bias. In recent years, many methods were proposed to reach this goal. However, the comparison of the performance of methods for batch removal requires an in-depth investigation that is beyond the scope of this work (see [33] for a recent review).

As a consequence, it is safe to suppose that (i) nonzero expression values may not coincide with the true transcript abundance in the cell and (ii) zero values observed in the gene expression profiles may be either due to truly non-expressed genes—in this case, we refer to structural zeros, as proposed in [34]—or to technical limitations of the sequencing technology, i.e. dropout events.

For this reason, many computational approaches have been developed to retrieve lost and corrupted information from scRNA-seq data, with the goal of returning an estimation of the correct expression levels in each single cell. Such methods are typically grouped in two major categories: (i) imputation methods, with the general goal of recovering the missing values in the data and (ii) denoising methods, aimed at adjusting the data by removing biological and technical noise. Very often, the two categories are mentioned indistinctly (see e.g. [35]), even though they comprise substantially different computational tasks.

To better distinguish the two categories, here, we propose a rigorous categorization of imputation and denoising methods for scRNA-seq data, in order to reduce the possible ambiguity in the definition of the underlying computational tasks (an analogous distinction was recently proposed in [36]).

- Imputation methods for scRNA-seq data include two major steps. The first step is aimed at distinguishing structural zeros (associated to non-expressing genes) from dropout events (i.e. genes whose transcripts were not captured during the sequencing process due to technical issues). Accordingly, in the second step, such methods strive to impute the values of dropout entries only. Nonzero entries and structural zeros are left unchanged.
- Denoising methods for scRNA-seq data ideally include both an imputation step (see above) and an additional computational step, which is aimed at modifying the entries which include falsely increased or decreased gene expression levels due to, e.g. biological variation or technical noise. According to this definition, all denoising methods are also imputation methods while the opposite is typically not true (a rigorous definition of the two categories is provided in section 1 of the Supplementary Material).

Methods in both categories rely on different assumptions and employ different algorithmic strategies to perform their tasks.

Thus, as reported in [37], a comprehensive comparison of all available approaches might be useful and timely to clarify which methods are more suitable for different circumstances and distinct data types. In particular, in [37], the different approaches are grouped in the following typologies.

- Data smoothing: the methods in this category aggregate the expression profiles of similar cells in order to perform denoising and imputation. In this category, we find DrImpute [38], DEWÄKSS [39], scHinter [40], *k*NN-smoothing [41], LSImpute [42], MAGIC [43], netSmooth [44], PRIME [45] and RESCUE [46]. Finally, other methods that use data smoothing to impute missing values are G2S3 [47] and scTSSR [48]. However, the former aggregates the information across similar genes to perform imputation, while the latter considers both similar cells and similar genes.
- External knowledge integrators: these methods exploit external knowledge to impute or denoise gene expression profiles. In this category, we find ADImpute [49], netSmooth [44], netNMF-sc [50], SAVER-X [51], SCRABBLE [52], scNPF [53] TRANSLATE [54] and URSM [55].
- Machine learning (ML): these methods employ ML techniques to correct for technical noise. We can find very recent methods that employ Artificial Neural Networks (ANNs) to infer the denoised or imputed version of the dataset, which are AutoImpute [56], DeepImpute [57], DCA [58], EnImpute [59], GraphSCI [60], LATE [54], scIGANs [61], SAUCIE [62], scScope [63], scVI [64] and SISUA [65]. Next, we have methods that use regression to correct for noise in the dataset, which are 2DImpute [66] and RIA [67].
- Matrix theory: these methods decompose the observed gene expression matrix in a low-dimensional space to remove noise. In this category, we find ALRA [68], ENHANCE [69], scRMD [70], CMF-Impute [71], deepMc [72], McImpute [73], PBLR [74], WEDGE [75], ZIFA [76] and Randomly [77].
- Model-based: these methods make assumption on the statistical model of the distribution of technical and biological variability and noise and perform denoising and imputation by estimating the parameters of the distributions. In this category, we find bayNorm [78], BISCUIT [79], BUSseq [80], CIDR [81], MISC [82], SAVER [83], scImpute [84], scRecover [85], SCRIBE [86], SIMPLEs [87] and VIPER [88].

We here present a comparative assessment of denoising and imputation methods for scRNA-seq data, with the goal of providing a general overview of their features, strengths and limitations, in order to understand in which data analysis task they are most computationally and statistically efficient. In particular, we selected a subset of 19 different methods out of the list mentioned above, by including some of the most widely used approaches and which fall in the following categories.

- Data smoothing methods: DrImpute [38], *k*NN-smoothing [41] and MAGIC [43].
- ML methods: AutoImpute [56], DCA [58], DeepImpute [57], SAUCIE [62], SAVER-X [51], SCScope[63] and scVI [64].
- Matrix factorization/theory methods: ALRA [68], ENHANCE [69], McImpute [73], Randomly [77] and scRMD [70].
- Model-based methods: bayNorm [78], SAVER [83], scImpute [84] and VIPER [88].

The comparative assessment was carried out both on simulated data, generated via the widely used tool SymSim [89], and four real-world scRNA-seq datasets from [90–93]. All computational methods were tested with respect to a number of metrics, in order to assess the effectiveness in imputing dropout events, recovering the true expression profiles, characterizing the similarity among cells and improving the identification of differentially expressed genes (DEGs), in addition to quantify their scalability. In the Results section, we present the results of the extensive comparative assessment, also by releasing a summary for a quick evaluation of the distinct techniques in different scenarios and experimental settings.

We note that previous works reviewing imputation methods have been proposed. In particular, in [94], the authors focus on understanding whether six different imputation strategies introduce false positives in the results of differential expression analysis. In [95], eight different methods are analyzed to understand whether they improve the result of clustering and differential expression analysis. Both works, however, do not include in the analysis the most recent methods and assess the performance of a relatively limited number of computational strategies. In addition, both works mainly focus on the imputation task, without assessing how denoising techniques may recover corrupted information. Finally, a recent preprint on a similar subject [35] exploits real-world data to assess the performance of imputation methods on downstream analyses. While this work includes a more extensive assessment of recent methods, it does not employ simulated data, which are necessary to evaluate a number of ground truth (GT)-based performance metrics. Further comments in this respect are provided in the Discussion section.

In the Methods section, we provide a brief description of each denoising and imputation method included in the study, discuss the performance assessment describing both the synthetic data generation and the real-world datasets and present the different metrics used in the analysis. In the Results section, we present the results of the comparative assessment on both simulated and real data, also by releasing a summary for a quick evaluation of the distinct techniques in different scenarios and experimental settings. Finally, in the Discussion section, we draw conclusions about the comparison and discuss possible future developments.

Methods

In this section, we describe in detail the 19 methods included in the comparative assessment; we discuss the synthetic data generation and present the 4 real-world scRNA-seq datasets from [90–93] employed in the analysis, as well as the performance metrics.

Description of denoising and imputation methods

The 19 methods that have been analyzed and tested can be partitioned into the following four families, according to their assumptions and modeling techniques: smoothing, model-based, matrix factorization/theory and ML. In the following sections, we provide a brief description of each method. For additional details, we refer the reader to the original papers.

Data smoothing methods

The first category includes methods that aggregate the expression profiles of similar cells, e.g. by averaging the expression values, in order to impute (DrImpute) or denoise (MAGIC and *k*NN-smoothing) their expression values.

DrImpute [38] imputes dropout events with the following three steps: first, it computes a distance matrix between cells, then it runs the *k*-means algorithm and, lastly, it defines the expected value of a dropout event as the average value of that gene over the cells belonging to the same cluster. To make the estimations more robust, the similarity matrix is computed with both Pearson and Spearman correlations and a range of number of clusters is tested. The averaged estimation over all combinations is taken as the final imputation value, reducing the risk of over-imputation.

***k*NN-smoothing** [41] improves the signal-to-noise ratio of single-cell expression profiles with a two-phase algorithm: first, the k-nearest neighbors (kNNs) of each cell are identified, then the gene expression profile of each cell is smoothed by considering its neighbor profiles. The initial step of the algorithm is performed by normalizing the expression profiles and stabilizing their variance. Then, to overcome the problem of finding the best assignment for k, smoothing is applied in a progressive fashion, by starting from $k = 1$ and increasing k step-by-step until the desired level of smoothness is reached.

MAGIC [43] extracts the true similarity between cells by amplifying biological trends, while simultaneously filtering out spurious correspondences due to noise in the data. First, to overcome the problem of data sparsity, a nearest neighbor graph based on cell–cell expression distance is built. Then, an affinity matrix is defined by applying a Gaussian kernel on the principal components of the graph. Lastly, a diffusion process [96] is applied on the similarity matrix to obtain a smoothed, more faithful affinity matrix. The final imputation involves computing the new expression of each gene as a linear combination of the same expression in similar cells, weighted by the similarity strength obtained in the previous steps.

ML methods

This group includes methods that apply ANNs to solve the denoising problem (further details on ANN types are reported in section 2 of the Supplementary Material). As reported in [37], an increasing number of methods fall in this category (see above). In particular, we selected DeepImpute, DCA, SAVER-X, SAUCIE, scScope, AutoImpute and scVI.

AutoImpute [56] employs a sparse autoencoder, to learn the distribution of the input gene expression matrix and perform imputation. With regard to the implemented loss function, this method takes advantage of standard reconstruction errors such as (root) mean squared error, applied only on the nonzero expressed genes. After training the autoencoder (AE), the reconstructed matrix is taken as the imputed output.

DCA [58] employs AEs to perform denoising. Instead of the classical AE decoder output, it defines a parametric decoder that models each gene count as a negative binomial (NB) or a zero-inflated negative binomial (ZINB) distribution; consequently, the reconstruction error is defined as a likelihood. The predicted distribution is then used to generate the denoised output.

DeepImpute [57] employs a deep feedforward network (DFN) to perform imputation. After the initial preprocessing, where only relevant genes are kept, N random groups of genes G_i are defined. Then, for each gene in each G_i, a set I_i with the top five Pearson correlated genes not in G_i is built. Lastly, each I_i will be an input for a different DFN, trained to output G_i. The output of each DFN is then used for imputing dropout events.

SAUCIE [62] is an AE-based denoising method that also supports batch correction and enhanced clustering and visualization capabilites. More in detail, the AE embedding layer is used for both low-dimensional visualization and batch correction, by minimizing the difference between the probability distribution of layer's activations belonging to different batches. Moreover, the activations of the decoding part are binarized to define an encoding of each cell, which is then used for clustering. Lastly, denoising is performed by minimizing the reconstruction error, i.e. the mean squared error, that deals both with noise and dropout events.

SAVER-X [51] is an extension of SAVER [83] that pairs the Bayesian model with an AE. A NB distribution is used to model technical and biological noise, while the AE is used to estimate the portion of gene expression that is predictable by the other genes. Lastly, Bayesian shrinkage is used to compute a weighted average of the predicted expression values and the observed data, to get the final denoised value. Additionally, SAVER-X allows transfer learning [97] across species, thanks to the flexibility of AEs, allowing to extract information from data belonging to different species and experimental conditions.

scScope [63] exploits a deep learning approach for imputation, combining an AE with a recurrent layer. The architecture of the neural network is composed by a first layer that performs batch correction. Successively, the encoding and decoding layers of the AE perform compression and reconstruction, respectively, of the batch corrected input. Lastly, the imputation layer corrects the missing values and sends back the imputed output to the encoding-decoding layers, to re-learn a compressed representation. The loss function is defined as a standard reconstruction error, on the nonzero entries.

scVI [64] employs a variational AE to specify a ZINB distribution, which models the true gene expression. More in detail, the neural network takes as input each batch-annotated cell expression and successively learns a variational distribution accounting for, separately, the cell-specific scaling factor and the remaining gene variation; furthermore, the defined latent space allows to perform both clustering and visualization. Lastly, the ZINB distribution is specified based on the learned latent representation and the cell scaling factor.

Matrix factorization and matrix theory methods

The third category comprises four methods that denoise (ENHANCE) or impute (ALRA, McImpute and scRMD) the observed gene expression data by solving a matrix factorization problem [98]. For the sake of simplicity, we added to this category also a method that performs imputation by exploiting random matrix theory (RMT): Randomly.

ALRA [68] performs imputation by low-rank matrix completion [99] of the observed gene expression matrix. The algorithm is composed by two phases: firstly, a low-rank approximation with Singular Value Decomposition [100] is computed. Then, to distinguish dropouts from true zeros, the authors observed that biological zeros in the computed low-rank matrix are assigned to small values around 0, due to the approximation error. Consequently, by taking the magnitude of the smallest negative value of each gene as an approximation of the error, it is possible to define a gene-wise threshold to distinguish dropouts and extract the imputed values.

ENHANCE [69] is a method that combines PCA and cell aggregation using kNNs to denoise the observed count matrix. The algorithm can be divided into two main steps. The first one accounts for reducing the bias toward highly expressed genes, by aggregating the expression of similar cells based on the distance between their principal component scores. The second phase projects the aggregate matrix on the first k principal components, where k is selected to represent only true biological differences. Lastly, the selected components are used to derive the final denoised matrix.

McImpute [73] is a low-rank matrix completion approach to impute missing values in a gene expression matrix. This method aims at finding a lower-dimensional decomposition of the input matrix. They formulated a low-dimensional nonnegative matrix factorization problem as an optimization problem, solved using the majorization-maximization technique [101]. To ensure the convexity of the problem, McImpute solves a relaxed version of the original objective: nuclear norm minimization. Lastly, the resulting decomposition is used to impute missing values.

Randomly [77] is a recent denoising method that extracts the true biological signal from the gene expression data by analyzing the eigenvector statistics predicted by RMT [102]. The algorithm is composed by three steps. In the pre-processing step, expression counts are normalized and genes contributing to a sparsity-induced nonbiological signal are removed; then, the random matrix accounting for the noise is estimated. Lastly, the eigenvalues carrying the true biological signal are extracted following RMT, providing a low-rank representation of the input data; additionally, the genes that are mostly responsible for the signal directions can be separated from the less relevant ones.

ScRMD [70] is a method that approaches the imputation task by means of a robust matrix decomposition (RMD) approach [103]. The authors assumed that we can decompose each gene expression in the following components: the mean expression of cells belonging to the same cluster, the specific cell variability, the measurement error and the dropouts events. The method defines each component as a matrix decomposition problem, solved with an alternating direction method of multiplier, by also applying a regularizer to account for the low-rank of the biological signal and the sparseness of the observed counts.

Model-based methods

This category is composed by methods that model the observed expression value of each gene in each cell as a random variable and perform imputation (scImpute and VIPER) and denoising (bayNorm and SAVER) by estimating the parameters of their distributions.

bayNorm [78] employs a Bayesian approach to perform denoising. The posterior distribution of the original counts is composed by (i) the likelihood of obtaining the observed transcripts, modeled as a Binomial distribution, and (ii) a prior on each gene expression value. In order to model biological variability, bayNorm employs a prior on the underlying true gene expression levels, by modeling them as variables following an NB distribution. Parameters can then be estimated locally or globally, depending on one's interest in amplifying or not, respectively, the intergroup differences between cells.

SAVER [83] estimates the true gene expression levels by modeling observed counts as a NB distribution. More in detail, the technical noise in the gene expression signal is approximated by the Poisson distribution, while the gamma prior accounts for the uncertainty in the true expression. The final recovered expression is a weighted average of the normalized observed counts and the predicted true counts.

scImpute [84] is a method that performs imputation, in a three-step algorithm. Initially, it identifies subpopulations of cells by first applying PCA and, successively, spectral clustering [104] on the remaining dimensions. To infer which genes are affected by dropout, it models genes in each subpopulation with a gamma-normal mixture model. Lastly, only highly probable dropout events are considered, to reduce over-imputation, and the final imputation value is computed as a linear combination of the expression of the other cells in the same subpopulation, weighted by the pairwise similarity.

VIPER [88] is an imputation method composed of four phases. The first step performs a pre-selection of candidate similar cells, to reduce overfitting. Then, a least-squares method is used to choose a local neighborhood for each cell. To prevent imputing missing values, VIPER estimates the dropout probability and the expected expression for each zero-valued neighbor. Furthermore, to adjust dropout events, the gene expressions in each neighborhood are assumed to follow a zero-inflated Poisson mixed model, estimated using expectation maximization. Lastly, imputation is performed by computing the weighted sum of the expression of each neighbor, by also taking into account the computed dropout adjustments.

Performance assessment

In the original articles, the imputation and denoising methods introduced above are often compared with competing approaches. However, such comparisons typically involve a limited number of denoising methods and a small number of selected experimental settings. In order to provide a comprehensive evaluation of performances, in this work, we tested all methods on a large number of both simulated and real-world datasets, with respect to several metrics.

In particular, we generated an extensive array of simulated data, for which the GT is available and which allow to quantify the ability of each method to actually recover the lost information (see Supplementary Material section 3 for details about the generation of such data). Moreover, we tested all methods on four real-world scRNA-seq datasets generated via distinct experimental protocols and settings.

Simulations

We employed the tool SymSim [89] to generate a large number of synthetic scRNA-seq datasets (for a total of 90 distinct synthetic datasets). SymSim takes as input the number of single cells, the number of genes, the number of cell subpopulations (characterized by distinct gene expression patterns) and a number of parameters that tune the amount of biological variability and technical noise.

The tool returns as output (i) a GT expression matrix, which includes biological variability but no noise; (ii) a theoretical expression profile (TEP) for each cell subpopulation, which is obtained by removing the biological variability from the GT; and (iii) a noisy (and sparse) expression matrix (NEM), which is finally derived by simulating the steps of a sequencing experiment.

In this work, we generated datasets simulating two main experimental scenarios and, in particular,

(i) non-UMI full-length datasets (i.e. high-coverage, high-amplification bias), including 100 single cells and modeling a typical plate-based full-length sequencing experiment (e.g. Smart-Seq2). Thirty datasets were generated with distinct parameter settings;
(ii) UMI datasets (i.e. low-coverage, low-amplification bias), including 3000 single cells (30 datasets) and 10000 single cells (30 datasets) and modeling a typical droplet sequencing experiment (e.g. Chromium 10x).

The different datasets in each scenario are characterized by distinct parameter settings, in terms of number of cell subpopulations ($\{3, 5\}$), noise level (5 levels), number of selected (most variable) genes ($\{500, 2000, 10000\}$) (Table 1). A detailed

Table 1. Summary of the simulated datasets. We simulated a total of 90 datasets, with the following combinations of parameters: 3 values of sample size (number of single cells) ×2 different numbers of subpopulations ×5 levels of noise ×3 numbers of selected most variable genes

Protocol	UMI	Non-UMI full-length
No datasets	60	30
No cells	{3000; 10000}	100
GT sub-populations	{3; 5}	{3; 5}
Capture efficiency	Low	High
Amplification Bias	No	Yes
Coverage	Low	High
Noise level[a]	{1; 2; 3; 4; 5}	{1; 2; 3; 4; 5}
No genes	{500; 2000; 10000}	{500; 2000; 10000}

[a] The levels of noise present in the simulated datasets are defined in section 3 of the Supplementary Material, which we refer for further details on synthetic data generation.

description of synthetic data generation can be found in the Supplementary Material section 3.

Real-world datasets

All methods were tested also on four distinct real-world scRNA-seq datasets, generated with distinct protocols and experimental specifications. In detail, we have the following.

- **RW-D** #1 (PBMCs – 10x) [90]: this widely employed scRNA-seq dataset is generated via 10x Genomics platform [20] and includes 68579 peripheral blood mononuclear cells (PBMCs), which are annotated with 11 cell types of the immune system, via correlation with benchmark gene expression profiles. This dataset was used in our analysis to assess the performance of imputation and denoising methods in characterizing cell similarities (for further details on the dataset, please refer to [90]; instructions for download are provided in the Supplementary Material).
- **RW-D** #2 (lung cell lines – 10x) [91]: this scRNA-seq dataset is generated via the 10x Genomics platform and includes 3918 cells from 5 distinct cell lines, which were assigned to its corresponding identity by exploiting known genetic differences (i.e. SNPs) between cell lines [91]; this allows not to rely on gene expression profiles for cell labeling. We employed this dataset to assess the robustness of the characterization of cell similarity.
- **RW-D** #3 (pancreatic islets – Smart-Seq2) [92]: this scRNA-seq dataset is generated via the full-length Smart-Seq2 protocol and includes 3514 cells from human pancreatic islets of four diabetic patients and five healthy samples. We employed this dataset to assess the performance of imputation and denoising methods with respect to cell similarity characterization when processing data from non-UMI full-length protocols.
- **RW-D** #4 (melanoma cell lines – 10x, Fluidigm/Smart-Seq, bulk) [93]: this dataset includes three different measurements from the same biological samples, namely (*i*) bulk RNA-seq experiments, (*ii*) 10x Genomics scRNA-seq experiments with 737280 barcodes, (*iii*) Fluidigm/Smart-Seq scRNA-seq experiments with approximately 100 single cells. Since no cell type labels are provided in this dataset, we here used the data to compare the performance of imputation and denoising methods with respect to the correct identification of DEGs, by setting the results obtained on bulk data as baseline.

All real-world datasets were preprocessed to consider only high-quality single cells, and downsampled, to ensure a uniform assessment scheme for all methods. In Table 2, one can find the main features of all datasets employed in the analyses (see Supplementary Material section 4 for further details on preprocessing and downsampling).

Performance metrics

To evaluate the performance of the 19 selected methods, we employed a number of metrics, which were assessed with respect to either simulated or real-world data, according to the specific cases. All metrics are further detailed in section 5 of the Supplementary Material.

Imputation of dropout events (simulations) The effectiveness of the methods in identifying and correcting dropouts events can be evaluated by employing the GT expression matrix obtained from simulations (see Supplementary Material section 5 for additional details). In order to quantify the correct imputation of the dropout entries present in the GT, we employed three distinct metrics.

In particular, we computed (*i* and *ii*) precision and recall on dropout entries only (i.e. entries that are > 0 in the GT and are $= 0$ in the NEM), (*iii*) the Spearman correlation delta between the imputed/denoised expression matrix (for the sake of readability, we will refer to as denoised expression matrix, from now on) and the GT with respect to all the zero entries in the NEM, which allows to evaluate how imputed entries are correlated with GT values (this metric is shown in the Supplementary Material section 5).

Notice that the false discovery rate (FDR) can be easily determined from precision (FDR $= 1-$precision) and, in this case, allows to evaluate the effectiveness of the methods in not imputing structural zeros (i.e. entries that are 0 both in the GT and in the NEM).

Recovery of true gene expression profiles (simulations) To estimate the ability of each method in recovering the true single-cell gene expression profiles, we relied on both the GT and the NEM obtained from simulations.

In particular, we computed the difference between the Spearman correlation coefficient ρ computed after imputation or denoising (i.e. ρ between denoised expression matrix and GT) and that computed before imputation or denoising (i.e. ρ between NEM and GT). This measure is denoted as delta correlation in the following, $\Delta\rho$.

Characterization of cell similarity (simulations and real-world data) In order to evaluate the effectiveness of each method in capturing the similarity among cells, we computed the average silhouette coefficient (or width) [105] by grouping single cells according to the GT labels, i.e. cell subpopulations labels for both simulated data, and cell type/line labels for real data. Higher values of the average silhouette coefficient indicate that cells are grouped consistently with GT labels. Therefore, we here measured the difference between the average silhouette coefficient obtained from denoised data and that computed from the NEM (i.e. silhouette delta). Further detail about the evaluation of such metric is given in the Supplementary Material section 5.

We finally remark that, with regard to simulations, we here employed the TEP of all cell subpopulations as performance benchmark.

Table 2. Features of real-world datasets. Main features of the four real-world datasets used in the assessment of imputation and denoising methods: **RW-D**#1 [90], **RW-D**#2 [91], **RW-D**#3 [92] and **RW-D**#4 [93]

Dataset			Number of cells		
RW-D	Name	Protocol	Original	Employed	Task
#1	PBMC [90]	UMI	68579	3000	Cell sim.
		UMI	68579	10000	Cell sim.
#2	Lung cell lines [91]	UMI	3918	3918	Cell sim.
#3	Pancreatic islets [92]	Non-UMI	352	245	Cell sim.
		Non-UMI	383	243	Cell sim.
		Non-UMI	383	197	Cell sim.
		Non-UMI	383	224	Cell sim.
		Non-UMI	383	196	Cell sim.
		Non-UMI	383	263	Cell sim.
		Non-UMI	383	93	Cell sim.
		Non-UMI	384	275	Cell sim.
		Non-UMI	384	293	Cell sim.
#4	Sake (Parent.) [93]	UMI	737280	3178	DEGs
		Non-UMI	113	113	DEGs
	Sake (Resist.) [93]	UMI	737280	3085	DEGs
		Non-UMI	84	84	DEGs

Identification of DEGs (real-world data)
To assess the improvement on the identification of DEGs due to the application of imputation/denoising methods, we employed real-world dataset **RW-D**#4 which includes two independent cell populations, namely parental and resistant, for which single-cell 10x, single-cell Fluidigm/Smart-Seq and bulk sequencing experiments were executed.

We proceeded as follows: for each single-cell dataset (10x and Fluidigm/Smart-Seq), we performed a standard Wilcoxon test to select the DEGs ($p < 0.05$) between parental and resistant populations, with respect to both the NEM and the denoised expression matrix, and which results in two distinct lists of DEGs.

The expression profiles of the DEGs are then used to calculate the Spearman correlation coefficient between each single cell and the corresponding bulk profile. The distribution of the difference of the Spearman correlation coefficient as computed on denoised data and that on the NEM is used to evaluate the performance for this task.

Computation time (simulations) We finally analyzed the computational time of each tested method to impute or denoise datasets with distinct numbers of observations (i.e. single cells) and of variables (i.e. genes), with respect to a selected number of simulated datasets. All computations were performed on a HP$^{®}$ Z8 G4 Workstation equipped with two Intel$^{®}$ Xeon$^{®}$ Gold 6240 processors at 2.60 GHz, 1 TB DDR4 RAM at 2933 MHz and Linux Mint 19.2 Tina.

We note that, in the original papers, the authors do not declare any theoretical worst-case performance in terms of $O(\cdot)$ notation; although for many of them, it would be derivable from literature. We therefore present an empirical study of the relative performances of the methods.

Parameter settings of computational methods

Most methods were run on both simulated and real-world datasets using default settings and following guidelines provided from the authors, if any. For additional details on parameter settings of all methods, please refer to section 6 of the Supplementary Material and to Supplementary Table 4.

Note that we report the results SAVER-X without pre-training, as its performance seems to be only slightly affected by pre-training on real-world datasets, as shown in Supplementary Figure 9. Besides, for analyses involving synthetic datasets, we did not run AutoImpute, McImpute, scImpute and VIPER on datasets with 10 000 cells and 10 000 genes, and we did not execute VIPER on **RW-D**#1 (downsampled to 10 000 cells and 10 000 genes), due to the high computational time required by such methods. Furthermore, for 10 out of 30 non-UMI full-length simulated datasets, SAUCIE collapsed all cells into one unique profile. Thus, such datasets were not included in the analysis. Finally, please note that for Fluidigm/Smart-Seq datasets in **RW-D**#4, the computation of bayNorm and ENHANCE raised errors and, therefore, their results are not reported.

Results

We start by providing a qualitative example of the effect of the tested imputation and denoising methods: Figures 1 and 2 show the tSNE low-dimensional representation [106] of a synthetic dataset (3000 cells, 5 subpopulations and 2000 genes) and of one real dataset (**RW-D**#1, downsampled to 3000 cells and 2000 genes; see the Methods section for further details). For the synthetic dataset, we show the GT expression matrix, the NEM and the denoised datasets returned by each method, whereas for **RW-D**#1 we show its original expression matrix and the corresponding denoised versions.

From this qualitative analysis, one can appreciate the substantial different data transformations which are determined by the distinct methods.

While it is difficult to draw conclusion from single experiments, certain methods apparently tend to reduce the variability of gene expression profiles, resulting in more compact representations on the tSNE space (e.g. *k*NN-smoothing, SAUCIE, MAGIC), some others appear to enhance the inter-cluster distance (scImpute, SAVER and ENHANCE), while most methods seem to preserve the original disposition in the transcriptomic space, with some exceptions (note that in this and subsequent analyses, AutoImpute seems not to have reached convergence, with default parameters).

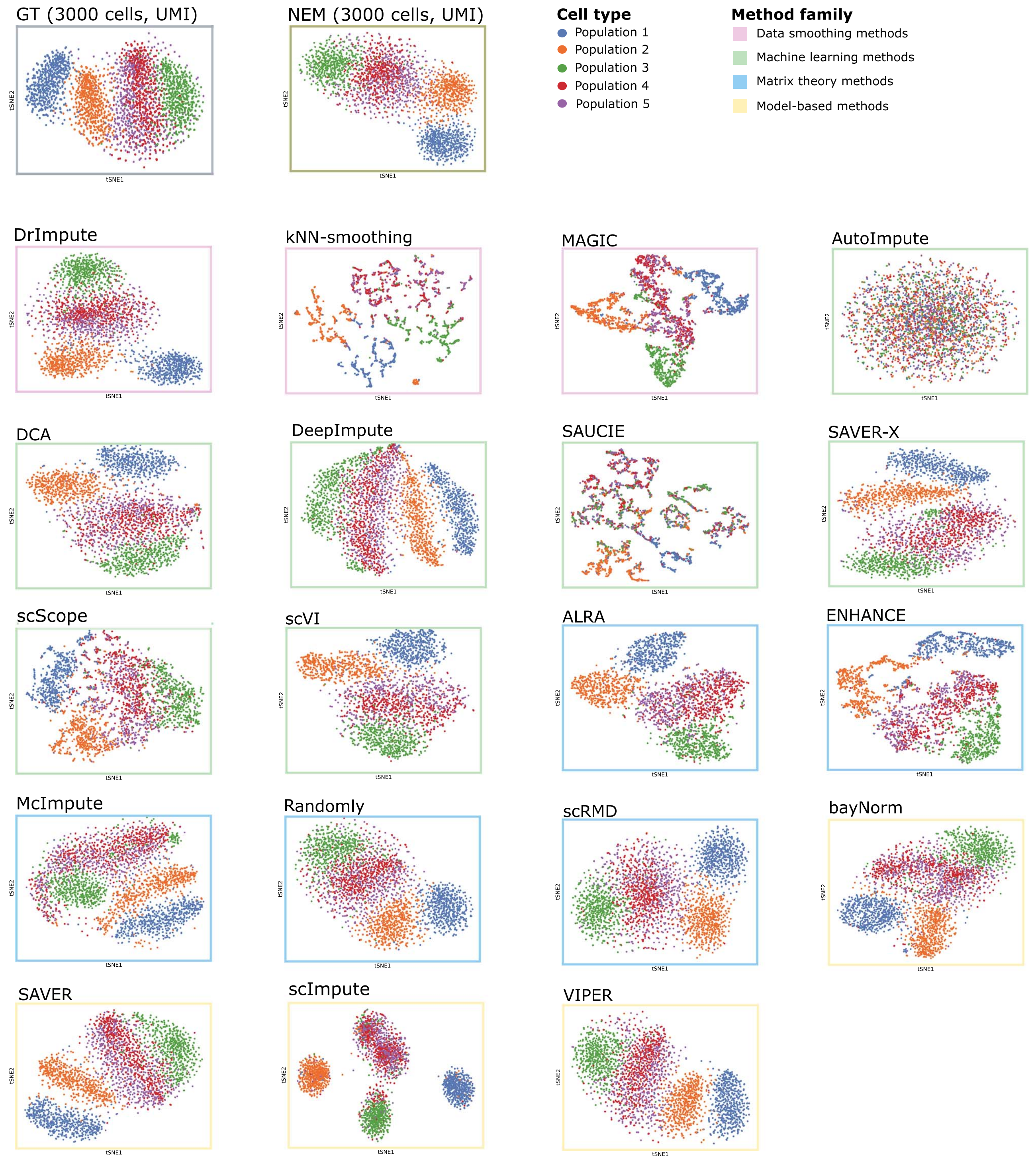


Figure 1. Effect of 19 imputation and denoising methods on a selected simulated scenario via tSNE low-dimensional representation. tSNE low-dimensional representation [106] of the gene expression profile of 3000 single cells of a selected synthetic UMI dataset with 5 subpopulations and 2000 genes. For this dataset, we present the tSNE plot of the GT expression matrix generated via SymSim and the NEM obtained after simulating the sequencing experiment. The remaining tSNE plots represent the gene expression of the cells after the application of all tested denoising and imputation methods to the NEM.

The visualization of three further synthetic datasets and of real-world datasets **RW-D**#2 and **RW-D**#3 are shown in Supplementary Figures 1–5. The results of the quantitative assessment with respect to the metrics described in the Methods section are presented in the following.

Imputation of dropout events (simulations)

We first assessed the performance of all methods in imputing dropout events (i.e. entries = 0 in the NEM but > 0 in the GT expression matrix), leaving structural zeros unchanged (i.e. entries = 0 both in the NEM and the GT). The parameters of all

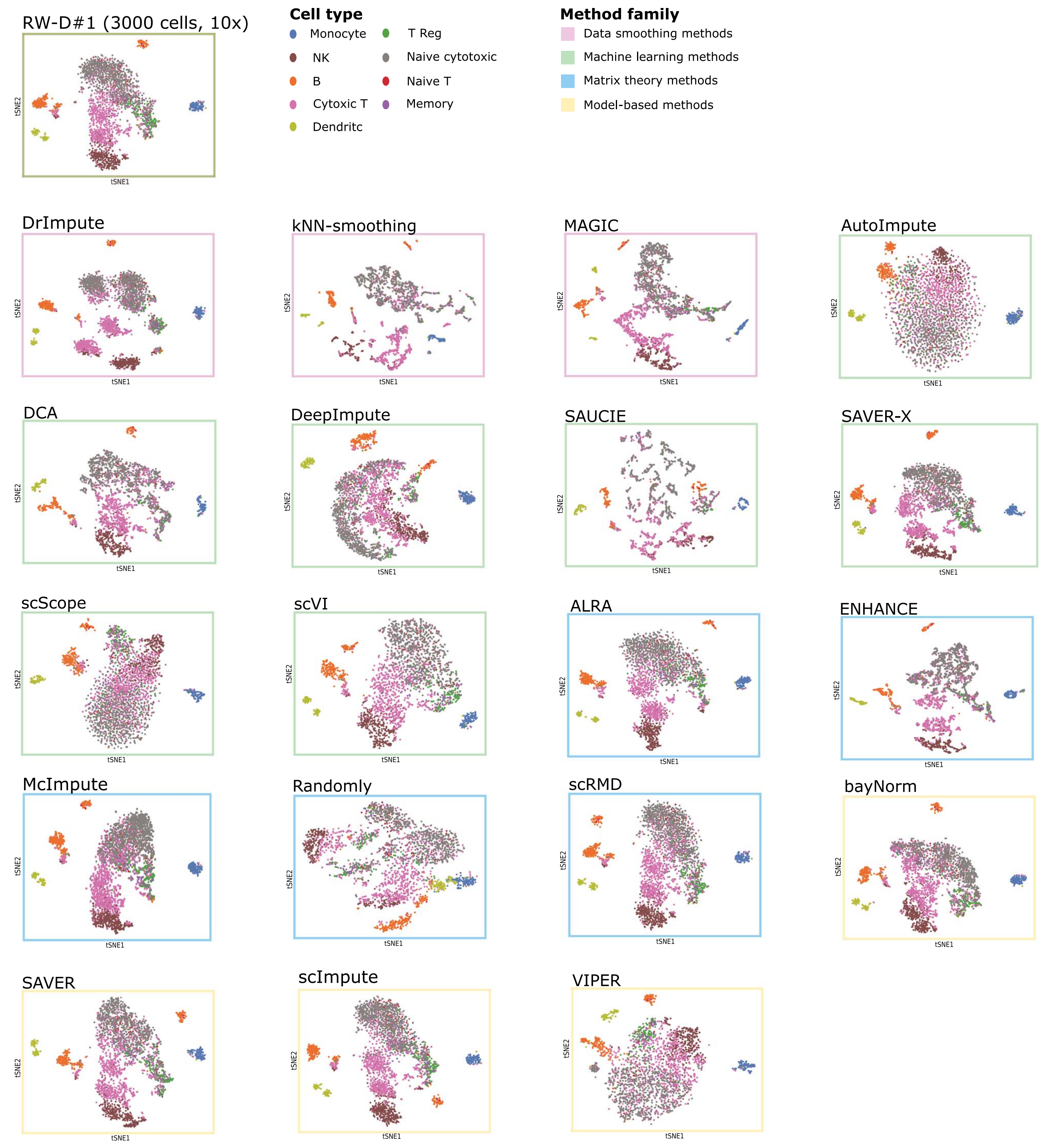


Figure 2. Effect of 19 imputation and denoising methods on real-world dataset RW-D #1 via tSNE low-dimensional representation. tSNE low-dimensional representation [106] of the gene expression profile of 3000 selected cells from RW-D #1 (PBMCs – 10x) [90] as computed on the 2000 most variable genes. For this dataset, we present the tSNE projection of the original dataset, which includes nine cell types and the tSNE plots of the single-cell expression profiles after the application of all methods under analysis.

simulations are recapitulated in Table 1 and in Supplementary Tables 1 and 2. Please refer to the Methods section and to Supplementary Material sections 3 and 5 for details on synthetic data generation and performance metrics. Note that Randomly was not included in this test, since it provides an already scaled expression matrix as output.

In Figure 3, one can find, for each method, the median precision and recall on correctly imputed dropouts (in this case, a true positive is an entry > 0 both in the GT and in the denoised expression matrix but $= 0$ in the NEM), grouped according to the number of (most variable) selected genes ({500, 2000, 10 000}) and the number of single cells (100 for non-UMI full-length

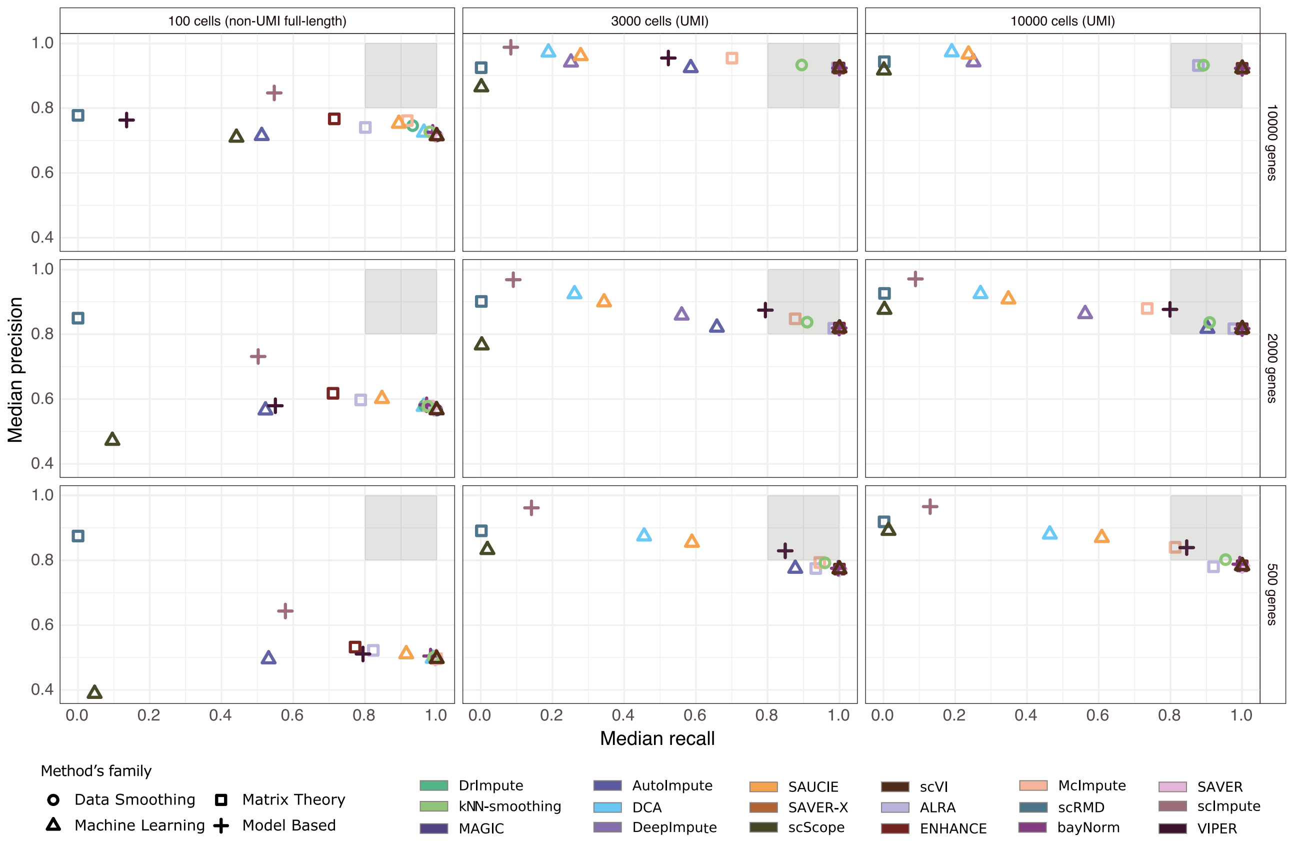


Figure 3. **Performance assessment on imputation of dropout events (simulations).** Assessment of imputation of dropouts, as evaluated on non-UMI full-length simulated datasets (100 single cells) and UMI simulated datasets ({3000, 10000} single cells), with {500, 2000, 10000} genes. In each panel, we display a scatter-plot returning, for each imputation and denoising method, the median precision (y-axis) and recall (x-axis) as computed on correctly imputed dropouts (computed on 10 simulations per setting). In this case, a true positive, is an entry that is > 0 in the denoised expression matrix and in the GT but is $= 0$ in the NEM (see the Methods section for further details and Supplementary Table 3 for the confusion matrix). The squared shade indicates methods with precision and recall > 0.80. In Supplementary Figure 6, the distribution of precision and recall is displayed.

and {3000, 10000} for UMI datasets). In order to identify the methods showing high precision (i.e. how many imputed entries are dropouts) and high recall (i.e. how many dropouts are imputed) scatter-plot areas corresponding to high values for both measures (> 0.80) were highlighted (in Supplementary Figure 6 the distributions of precision and recall on settings are displayed).

As a first result, most methods struggle when dealing with non-UMI full-length datasets (with 100 cells), as proven by the relatively lower value of average precision. This aspect is likely due to the low number of observations (single cells) as compared with the number of variables (genes) and consistently affects the performance of all methods on most tasks (see below).

Conversely, we observe a subset of methods that achieve extremely positive performances (both precision and recall > 0.80) for UMI datasets with 3000 and 10 000 cells. In detail, VIPER provides the best performance with datasets with 500 genes, while for datasets with 2000 and 10 000 genes, ALRA, bayNorm, DrImpute, ENHANCE, *k*NN-smoothing, MAGIC, SAVER, SAVER-X and scVI consistently provide optimal and analogous performances. In particular, such methods show values of recall very close to 1 in all experimental settings (with the exception of *k*NN-smoothing). While this effect might be due to over-imputation, such methods also display significantly high precision in most settings. Notice also that higher values of precision implicate a lower fraction of wrongly imputed structural zeros (entries $= 0$ both in the GT and the NEM), as measured by the false discovery rate (FDR $= 1-$ precision).

Finally, we note that scRMD and scImpute display the highest values of precision in most settings, which, however, are most likely due to the conservative nature of the approaches, which tend to limit the number of imputed values. This observation is strengthened by considering the low values of recall for both methods: indeed, as recall corresponds to the fraction of imputed dropouts, a value close to 0 indicates that the method did not impute most of the events.

To further extend the analysis on imputation of dropouts, in Supplementary Material section 7 (Supplementary Figure 7), we return the analysis of the Spearman correlation coefficient computed considering zero entries of the NEM and which allows to quantify the correlation between imputed entries and the corresponding GT expression values. On the one hand, bayNorm, DrImpute, ENHANCE, MAGIC, SAVER and SAVER-X provide the most accurate and robust results in most scenarios, proving effective in correctly recovering the true expression values of imputed entries. On the other hand, ALRA, *k*NN-smoothing and scVI and VIPER, which exhibit good values of precision and recall on imputed dropouts (see above), display a relatively lower performance in terms of correlation of the imputed entries with respect to the GT expression values.

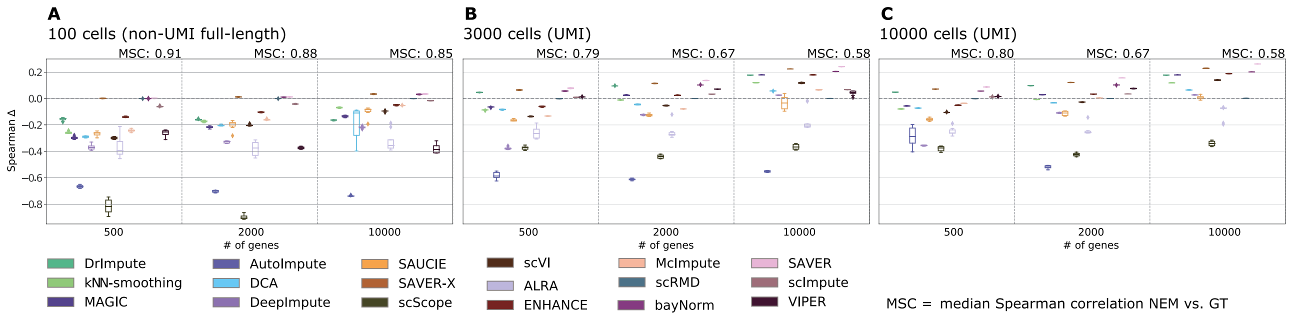


Figure 4. Performance assessment on recovery of true gene expression profiles (simulations). Assessment of recovery of true expression profiles, as evaluated on non-UMI full-length simulated datasets (100 single cells, panel A) and UMI simulated datasets ({3000, 10000} single cells, panels B and C), with {500, 2000, 10000} genes. The boxplots return the distribution of correlation delta, $\Delta\rho$, i.e. the difference between the Spearman correlation coefficient computed between the denoised expression matrix and the GT and that computed between the NEM and the GT, for all methods in each experimental setting. The baseline median Spearman correlation coefficient (MSC) between the NEM and GT is reported on top of the panels, for each setting, while in Supplementary Figure 8, the relative distributions are returned.

Recovery of true gene expression profiles (simulations)

We next tested the capability of each method in recovering the GT gene expression profiles, by using simulated data. In Figure 4, one can find the difference of the Spearman correlation coefficient as computed between the GT and the denoised expression matrix after the application of all 19 methods and that computed between the GT and the NEM. Such difference is denoted as correlation delta, $\Delta\rho$, from now on (see the Methods section and Supplementary Material section 5 for further details).

In particular, the results are displayed according to the number of genes, {500, 2000, 10000} and number of cells, 100 for non-UMI full-length and {3000, 10 000} for UMI experiments, as this allows to analyze the performance under different experimental settings. Note that, as for the analysis on imputed entries, we here do not include the output of Randomly, which provides a scaled output matrix.

As expected, sample size and protocol-type highly influence the capability of any method to recover corrupted information, as the performance of all methods generally improves with datasets with a larger number of single cells and generated via UMI-based protocols. More specifically, most methods appear to struggle when processing non-UMI full-length datasets characterized by a low number of cells (i.e. = 100), delivering unreliable and often erroneous denoised expression profiles, as proven by the negative Spearman correlation delta observed in most cases (up to −0.45 for some methods).

Conversely, correlation deltas progressively improve with UMI datasets including larger numbers of cells and/or genes, and, in particular, all methods with the exception of ALRA and scScope, achieve a positive median delta with datasets with 10 000 genes and 10 000 cells.

Examining the methods in greater detail, we observe that bayNorm, SAVER and SAVER-X are the methods with the best overall performance, as they always provide a positive correlation delta and achieve the best results with both non-UMI full-length and UMI datasets. Furthermore, we note that such approaches show an extremely low variance, suggesting that the results are robust. Among the other approaches, we note that DrImpute displays a high correlation delta with UMI datasets, whereas both ENHANCE and MAGIC exhibit remarkable performances with datasets with more than 3000 cells and more than 2000 genes.

All in all, the results of this and the previous analyses suggest that bayNorm, SAVER and SAVER-X might be an adequate choice for both imputing dropouts and recovering corrupted information, as they show the most accurate and stable performances with both UMI and non-UMI full-length datasets, whereas DrImpute, ENHANCE and MAGIC are similarly effective when processing UMI datasets.

Characterization of cell similarity (simulations and real-world data)

When analyzing scRNA-seq data, one might be interested in characterizing the possible heterogeneous populations included in the dataset, typically performing unsupervised clustering. For example, the Scanpy [107] and Seurat [108] packages for single-cell analyses incorporate the Louvain and Leiden algorithms for community detection [109], which identify clusters based on a nearest neighbors graph constructed from the profiles of each single cell. Therefore, it is clear that improving the identification of cell similarities might result in better clustering performances. To this end, we assessed the effectiveness of all tested methods in enhancing cell similarity with respect to both simulated and real data.

In Figure 5, we show the difference between the average silhouette coefficient computed on denoised expression matrix and that obtained from the NEM, by grouping single cells according to the GT labels. Higher values of the average Silhouette coefficient indicate that cells are close to other cells of the same subpopulation and separated from those belonging to other subpopulations. In particular, GT labels are provided by cell subpopulation labels for simulated data and by cell type/line labels for real-world datasets (see the Methods section and the Supplementary Material for further details). We remark that the silhouette coefficient allows one not to rely on arbitrarily chosen clustering approaches, to evaluate the correct grouping of single cells. In fact, currently available clustering methods for scRNA-seq data are characterized by different properties, goals and specifications and produce results that are extremely sensitive to parameter choices and variations, and which might, in turn, undermine the comparison of denoising and imputation methods on this specific task.

Results are shown for simulated datasets with {500, 2000, 10000} genes and 100 (non-UMI full-length) or {3000, 10000} single cells (UMI), as well as for real-world datasets **RW-D**#1, **RW-D**#2 and **RW-D**#3. Note that we employed the TEP of all cell subpopulations as benchmark for the assessment on simulated datasets: in particular, the silhouette coefficient delta between the TEP and the NEM represents the largest theoretical improvement in each setting.

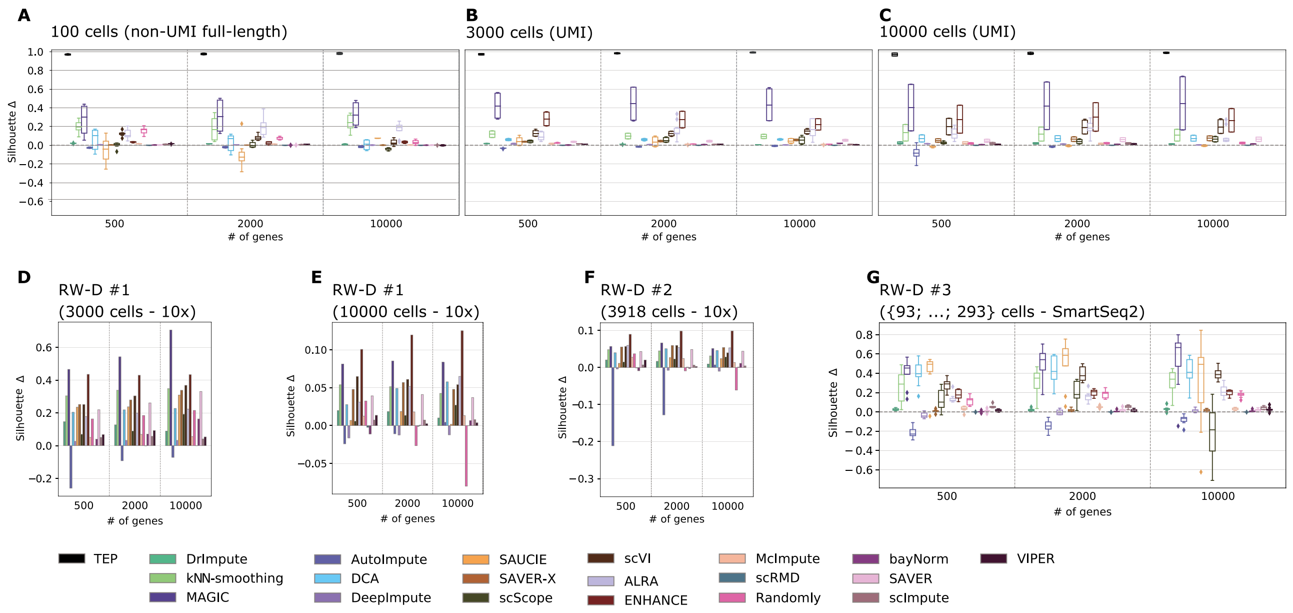


Figure 5. Performance assessment on cell similarity characterization (simulations and real-world data). Assessment of enhancement of cell similarity characterization after denoising, as evaluated on (i) simulated datasets (non-UMI full-length with 100 single cells and UMI-simulated datasets with {3000, 10000} single cells, panels A–C) and (ii) real-world datasets RW-D#1 (downsampled to {3000, 10000} cells, 10x platform, panels D and E), RW-D#2 (3918 cells, 10x platform, panel F) and RW-D#3 ({93, 196, 197, 224, 243, 245, 263, 275, 293} cells, Smart-Seq2, panel G). The boxplots (respectively, barplots) in all panels, depict the distribution (respectively, values) of the Silhouette delta, i.e. the difference between the average silhouette coefficient computed on the denoised expression matrix and that computed on the NEM, for all methods. The difference between the average silhouette coefficient evaluated on the TEP and that computed on the NEM is also shown for all simulated datasets.

Overall, most methods cause an increase of the average silhouette coefficient in most settings, suggesting that imputation and denoising approaches are indeed effective in enhancing the similarity of the expression profiles of cells belonging to the same sub-populations.

This effect is significantly intensified with datasets with larger sample size and generated (or simulated) with UMI protocols, as proven by the overall increase in delta magnitude. In particular, MAGIC and ENHANCE appear to produce the best results, with respect to both simulated and real-world datasets, yet with noteworthy variance in some scenarios, and with the latter method improving its performance with UMI datasets. We further notice that ALRA, *k*NN-smoothing and scVI deliver notable performances in most scenarios, closely followed by DCA. Surprisingly, SAUCIE exhibits a negative delta with simulated non-UMI full-length datasets but produces good results with real-world Smart-Seq2 dataset RW-D#3.

We recall that, among the best performing methods for the imputation and expression recovery tasks (see above), in addition to the aforementioned MAGIC and ENHANCE, SAVER-X and SAVER consistently produce improvements of the average silhouette delta in most simulated and real-world scenarios, whereas bayNorm and DrImpute appear to be less effective with respect to this specific task.

We finally specify that the results on simulated and real-world datasets are mostly coherent across experimental scenarios, further proving the suitability of simulations in assessing the performance of imputation and denoising methods.

Identification of DEGs (real-world data)

In order to quantify the effect of denoising and imputation methods on the identification of DEGs, we leveraged on bulk RNA-sequencing data included in real-world dataset RW-D#4 [93]. In detail, we first computed the DEGs between the parental and resistant samples included in the dataset, with respect to both the original expression matrix and the denoised matrix (via Wilcoxon test, $P < 0.05$), and which resulted in two distinct lists of DEGs. The analysis was repeated for both the Fluidigm/Smart-Seq dataset (84 and 113 single cells for resistant and parental cell lines, respectively) and the 10x datasets (3085 and 3178; see the Methods section and the Supplementary Material section 4 for further details).

In Figure 6, we display the difference of the Spearman correlation coefficient between the expression profile of the DEGs obtained from the denoised expression matrix and the bulk expression profile (computed for each single cell), and the one computed on the profiles of DEGs determined from the original expression matrix.

Noteworthy, most approaches produce an increase of the correlation with respect to the bulk expression profile. In particular, *k*NN-smoothing, MAGIC and SAUCIE deliver a median Spearman delta > 0.10 for both the Fluidigm/Smart-Seq and the 10x datasets, while bayNorm, ENHANCE, SAVER, SAVER-X and scVI show a median Spearman delta > 0.10 for the latter protocol only.

Overall, this result indicates that, in many cases, imputation and denoising methods might be effective in improving downstream analyses, such as the identification of DEGs.

Computation time (simulations)

Figure 7 reports the results of the computational time assessment on three simulated datasets: (i) non-UMI full-length (100 cells) (ii) UMI (3000 cells), and (iii) UMI (10 000 cells), with respect to {500, 2000, 5000} genes, plotted in logarithmic scale.

We can observe that all methods suffer an approximately exponential increase of computational time with respect to the number of cells and the number of genes, with extremely significant difference in magnitude. Overall, the most scalable

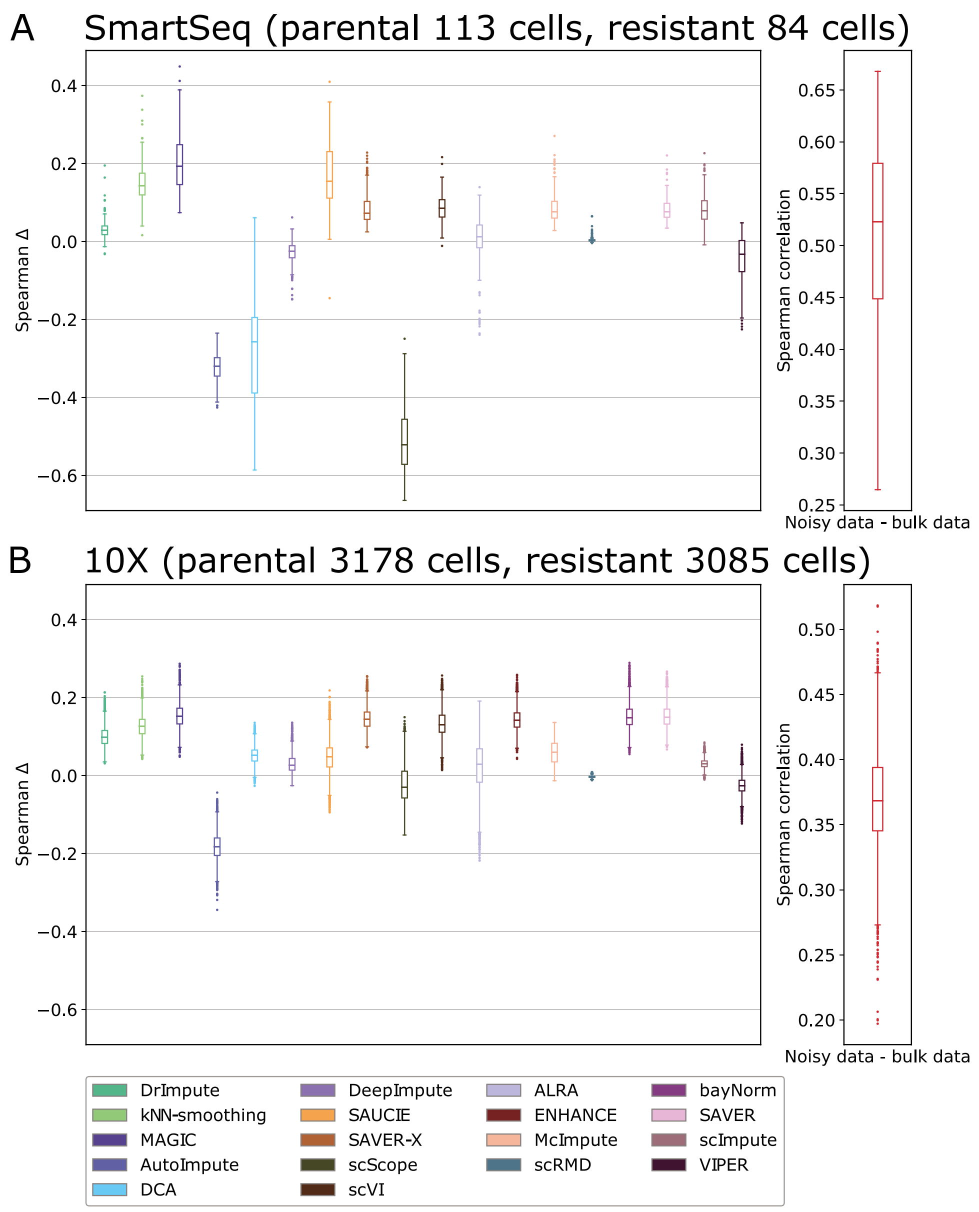


Figure 6. Performance assessment on identification of DEGs (real-world data). Assessment of identification of DEGs, as computed on **RW-D#4** [93]. DEGs between parental and resistant cell lines of RW-D#4 are identified via Wilcoxon test ($P < 0.05$), both starting from the original scRNA-seq dataset and from the corresponding denoised matrices, for both Fluidigm/Smart-Seq and 10x datasets (panel A and B). The Spearman correlation coefficient between the expression profile of all single cells and the corresponding bulk expression profile is computed with respect to all the DEGs included in the distinct lists. The distribution (on all single cells) of the difference between the Spearman correlation coefficient computed with original data matrix and that computed with the denoised version is then shown as boxplots for both 10x and Fluidigm/Smart-Seq datasets. In the rightmost panels, the baseline distribution of the Spearman correlation coefficient between the NEM and bulk data (with respect to the corresponding list of DEGs) is shown, for both scenarios.

algorithms appear to be ALRA, *k*NN-smoothing and scRMD while, in general, matrix theory appears to be the most computationally efficient category.

Summary of the performance assessment on denoising and imputation methods

In Figure 8, we present a recapitulation of the performance assessment. The schema includes seven panels, structured as follows:

- imputation of dropout events,
- recovery of gene expression profiles,
- characterization of cell similarity,
- identification of DEGs,
- computation time,
- task,
- release code quality.

In particular, we selected a subset of simulated datasets, characterized by selected parameter settings in terms of single-cell number ($\{100, 3000, 10000\}$), sequencing protocol $\{$non-UMI full-length, UMI$\}$ and number of genes (2000 for all settings)—and all four real-world datasets (see the Methods section), which we employed to compute a schematic ranking of all methods with respect to the distinct tasks.

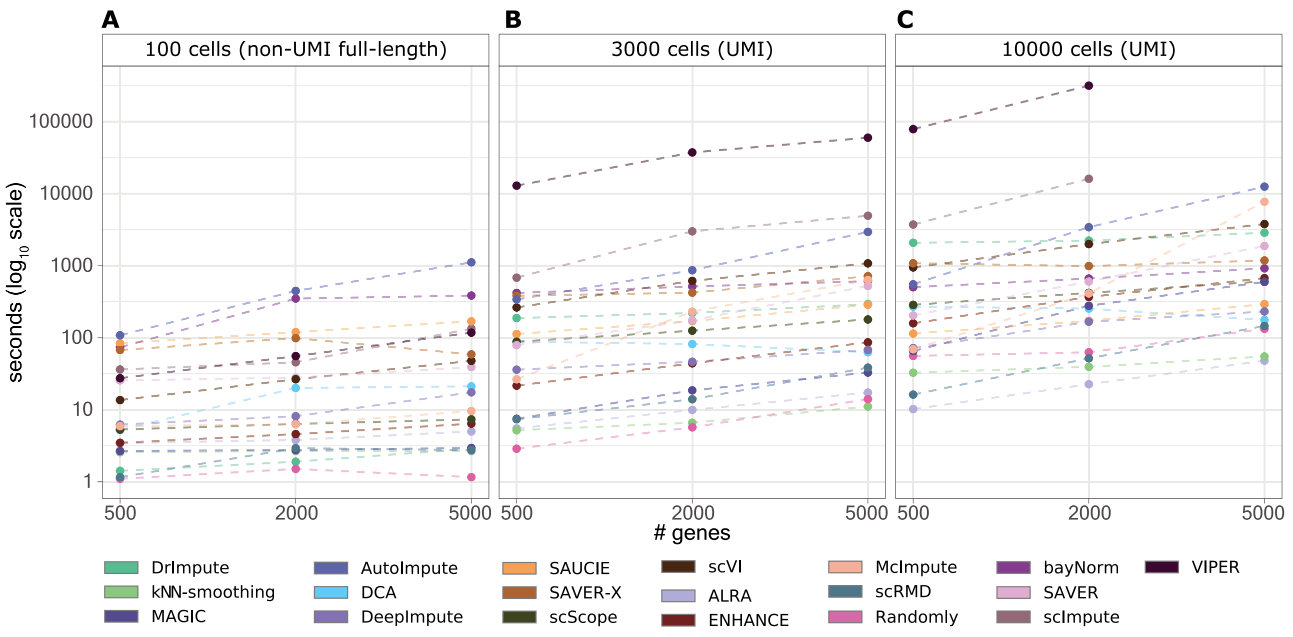


Figure 7. **Computational time assessment.** Running time of each method in denoising/imputing datasets with increasing number of cells and genes. In **(A)** results with 100 cells, in **(B)** results with 3000 cells and in **(C)** results with 10 000 cells. Values are plotted in logarithmic scale.

More in detail, for each selected parameter setting of the simulated dataset and for each real-world dataset, we ordered all 19 methods with respect to the average values of the following metrics:

(i) average Spearman correlation delta for zero entries of the NEM (for imputation of dropout events),
(ii) average Spearman correlation delta on the whole expression matrix (for recovery of gene expression profiles),
(iii) average silhouette delta (for characterization of cell similarity),
(iv) average Spearman correlation delta (for identification of DEGs),
(v) computation time.

The ranking is visually represented with dots with respect to each experimental setting, where the largest dot corresponds to the best performing method (green) and the smallest dot to the worst performing method (red).

The task panel indicates whether each method performs either denoising or imputation (see the Introduction section and Supplementary Material section 1 for a rigorous classification of the two tasks). Finally, the last panel reports a summary of selected quality code metrics, which were used to evaluate the different tools. In particular, usability and documentation range from 1, i.e. the worst result, to 4, corresponding to the best score. Usability is calculated by considering a set of characteristics that contribute in worsening the overall usability of the tool: (*i*) either input preprocessing, preliminary operation, e.g. clustering, or output post-processing, e.g. re-normalization, are required to the user; (*ii*) at least one parameter depends on the input, i.e. a grid-search is required; (*iii*) parameters meaning is not intuitive, e.g. it has no biological meaning; (*iv*) the tool is not available on a package distribution platform, e.g. `Bioconductor` or `pip/conda`. If a tool has none of the previously introduced features is assigned to the maximum score of 4; otherwise, the scoring is reduced to a minimum of 1. Documentation score is assigned as follows: 1 indicates that the authors did provide neither a documentation nor a detailed tutorial, 2 indicates that the authors provided a tutorial but did not write a detailed explanation for the parameters, 3 indicates that a detailed tutorial is available and 4 indicates that the authors provided both a detailed tutorial and a full explanation of all parameters. Finally, we indicate both whether the program is maintained, i.e. updated in the past 2 years, and the programming language on which the tool was implemented.

Discussion

We presented a review of the current state-of-the art of computational approaches for denoising and imputation of scRNA-seq data. Extensive tests on both real and synthetic datasets allowed to evaluate the performances and the robustness of each method under different experimental scenarios.

In light of the presented results, distinct methods appear to be more suitable for different tasks. In particular, ENHANCE, MAGIC, SAVER, and SAVER-X provide the best overall compromise and show robust performances with respect to all considered tasks. In addition to such methods, bayNorm and DrImpute are especially effective in recovering the true expression profiles and imputing dropout entries, while *k*NN-smoothing and scVI in improving the characterization of cell similarity and the grouping of single cells in coherent subpopulations, as well as the identification of DEGs.

We also note that, as expected, most methods appear to struggle with non-UMI full-length datasets, likely due to the low number of observations (cells) as compared with the high number of variables (genes). Furthermore, as already mentioned and as reported in [110], denoised expression values returned by any method should be considered with caution, due to the presence of possible artifacts, as proven by the low correlation with GT expression profiles from simulations recorded in many cases and, particularly, with non-UMI full-length datasets.

By focusing on machine learning frameworks, we notice that methods that employ assumptions on biological variability and technical noise (i.e. DCA, SAVER-X, scVI) typically exhibit

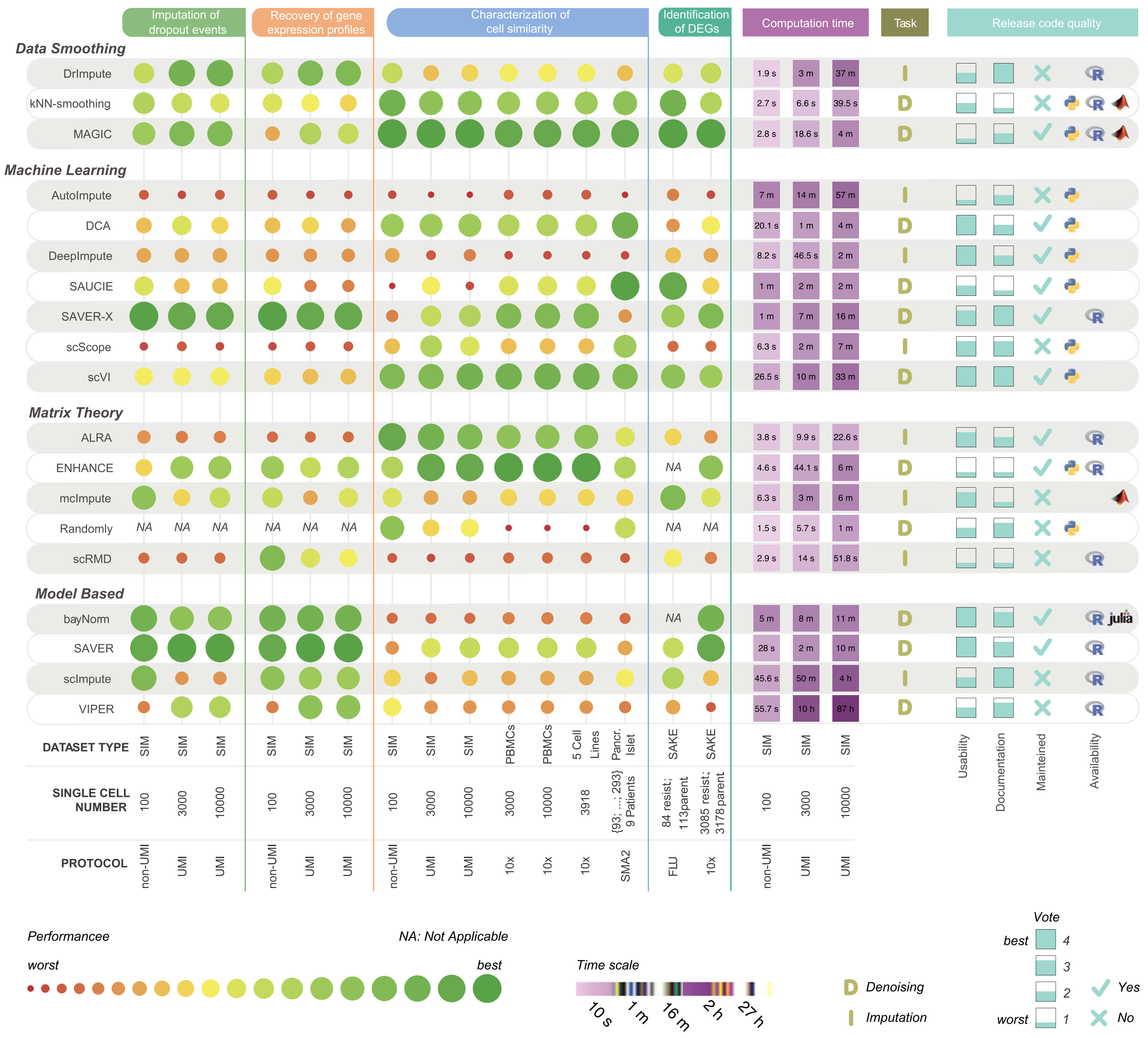


Figure 8. **Summary of the performance assessment on denoising and imputation methods**. The five leftmost panels report a schematic ranking of all 19 tested denoising and imputation methods, as computed on a selected panel of synthetic and real-world datasets, in terms of average Spearman correlation delta for zero entries of the NEM (for imputation of dropout events), average Spearman correlation delta on the whole expression matrix (for recovery of gene expression profiles), average silhouette delta (characterization of cell similarity), average Spearman correlation delta (identification of DEGs) and computation time. The size of each round marker is proportional to the ranking, with the largest (green) dots corresponding to the best performing tool and the smallest (red) dots to worst performing tool, with respect to the considering metric. The task panel indicates whether the method can perform either denoising or imputation tasks. Finally, the rightmost panel reports a summary of a quality code metrics that were used to evaluate the different tools in terms of usability, documentation, maintenance and availability (please refer to the Methods section (Summary of the performance assessment on denoising and imputation Methods) for further details).

better performances, hinting at the importance of including prior knowledge to inform the learning algorithms. Model-based methods present a typically good performance in both imputation and expression recovery, yet at a usually high computational cost, and generally showing suboptimal performance in cell similarity enhancement. Matrix theory-based techniques show good performance in terms of characterization of cell similarity, in addition to noteworthy scalability, even with large datasets. Finally, data smoothing approaches present typically good performances, yet with significant differences according to the specific task.

All in all, the performance of all methods appear to be highly dependent on the specific features of the dataset, as very distinct results are observed for the same method in different experimental scenarios, as recapped in Figure 8. This summary should guide potential users in selecting an optimal method according to the research needs and the available data types.

We further note that a review on a similar subject can be found as a preprint in [35]. Despite achieving similar conclusions on several methods included in our review, such work does not include comparisons on simulated data, which allow to evaluate a number of metrics with respect to the GT. For instance, certain methods that were identified as highly performing in [35], appear to struggle in dealing with true expression profiles recovery, an effect that can be evaluated only via simulations. The virtually unlimited number of in silico scenarios that can be generated via methods such SymSim [89] suggests that simulations should be increasingly used to quantitatively assess the performance of data science methods and especially to test the robustness of their results.

Possible limitations of our assessment might be related to the application of most methods with default parameters, while one can expect improvements when fine tuning the parameters. In this respect, setting guidelines provided by the authors were followed when present and appear to be extremely beneficial to increase the overall usability and performance of the methods.

We also recall that for some methods, such as those based on AEs, it would be possible to use the latent variable space to perform single-cell clustering, while in our analysis we chose to use the denoised expression profiles, to provide a fair comparison for all methods.

We finally remark that scalable methods for denoising of single-cell transcriptomic data might pave the way for refined downstream analyses, for instance, by improving the reliability and accuracy of variant calling pipelines from scRNA-seq data to provide an accurate mapping of genotype and phenotype of single cells [111, 112], as well as by allowing a better estimation of metabolic fluxes from scRNA-seq data in the investigation of cancer metabolism [113, 114].

Key Points

- Extensive tests on synthetic and real datasets provide a quantitative assessment of the performance of denoising and imputation methods in distinct scenarios.
- Some methods are effective in improving the characterization of cell similarity, some others in recovering the true gene expression profiles and imputing dropouts.
- Appropriate assumptions on the noise model are beneficial to recover lost information.
- Overall, ENHANCE, MAGIC, SAVER and SAVER-X constitute a good compromise on all tasks.
- Corrected expression values returned by any method should be considered with caution in downstream analyses.

Supplementary Data

Supplementary data are available online at https://academic.oup.com/bib.

Data availability

The source code used to replicate all our analyses, including synthetic and real datasets, is available at this link: https://github.com/BIMIB-DISCo/review-scRNA-seq-DENOISING.

Acknowledgments

We thank Chiara Damiani, Daniele Ramazzotti and Giulio Caravagna for helpful discussions.

Funding

This work was supported by the Cancer Research UK and Associazione Italiana per la Ricerca sul Cancro (CRUK/AIRC) "Accelerator Award" (award #22790) 'Single-cell Cancer Evolution in the Clinic'. Partial support was also provided by the Italian node of the Elixir network (https://elixir-europe.org/about-us/who-we-are/nodes/italy) and the SysBioNet project, a Ministero dell'Istruzione, dell'Universitá e della Ricerca initiative for the Italian Roadmap of European Strategy Forum on Research Infrastructures.

References

1. Dalerba P, Kalisky T, Sahoo D, *et al*. Single-cell dissection of transcriptional heterogeneity in human colon tumors. *Nat Biotechnol* 2011;**29**(12):1120.
2. Vieth B, Parekh S, Ziegenhain C, *et al*. A systematic evaluation of single cell RNA-seq analysis pipelines. *Nat Commun* 2019;**10**(1):1–11.
3. Angela RW, Norma F, Neff TK, *et al*. Quantitative assessment of single-cell RNA-sequencing methods. *Nat Methods* 2014;**11**(1):41.
4. Kalisky T, Blainey P, Quake SR. Genomic analysis at the single-cell level. *Annu Rev Genet* 2011;**45**:431–45.
5. Huang S. Non-genetic heterogeneity of cells in development: more than just noise. *Development* 2009;**136**(23): 3853–62.
6. Li L, Clevers H. Coexistence of quiescent and active adult stem cells in mammals. *Science* 2010;**327**(5965):542–5.
7. Shalek AK, Satija R, Shuga J, *et al*. Single-cell RNA-seq reveals dynamic paracrine control of cellular variation. *Nature* 2014;**510**(7505):363–9.
8. Hwang B, Lee JH, Bang D. Single-cell RNA sequencing technologies and bioinformatics pipelines. *Exp Mol Med* 2018;**50**(8):1–14.
9. AlJanahi AA, Danielsen M, Dunbar CE. An introduction to the analysis of single-cell RNA-sequencing data. *Mol Ther Methods Clin Dev* 2018;**10**:189–96.
10. Lawson DA, Kessenbrock K, Davis RT, *et al*. Tumour heterogeneity and metastasis at single-cell resolution. *Nat Cell Biol* 2018;**20**(12):1349–60.
11. Shaffer SM, Dunagin MC, Torborg SR, *et al*. Rare cell variability and drug-induced reprogramming as a mode of cancer drug resistance. *Nature* 2017;**546**(7658):431.
12. Cao J, Packer JS, Ramani V, *et al*. Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science* 2017;**357**(6352):661–7.
13. Regev A, Teichmann SA, Lander ES, *et al*. Science forum: the human cell atlas. *elife* 2017;**6**:e27041.
14. Elowitz MB, Levine AJ, Siggia ED, *et al*. Stochastic gene expression in a single cell. *Science* 2002;**297**(5584): 1183–6.
15. Marinov GK, Williams BA, McCue K, *et al*. From single-cell to cell-pool transcriptomes: stochasticity in gene expression and RNA splicing. *Genome Res* 2014;**24**(3):496–510.
16. Haque A, Engel J, Teichmann SA, *et al*. A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. *Genome Med* 2017;**9**(1):75.
17. Ziegenhain C, Vieth B, Parekh S, *et al*. Comparative analysis of single-cell RNA sequencing methods. *Mol Cell* 2017;**65**(4):631–43.
18. Macosko EZ, Basu A, Satija R, *et al*. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell* 2015;**161**(5):1202–14.
19. Klein AM, Mazutis L, Akartuna I, *et al*. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell* 2015;**161**(5):1187–201.

20. Fraction of mRNA transcripts captured per cell. https://kb.10xgenomics.com/hc/en-us/articles/360001539051-What-fraction-of-mRNA-transcripts-are-captured-per-cell.
21. Ramsköld D, Luo S, Wang Y-C, *et al*. Full-length mRNA-seq from single-cell levels of RNA and individual circulating tumor cells. *Nat Biotechnol* 2012;**30**(8):777.
22. Sheng K, Cao W, Niu Y, *et al*. Effective detection of variation in single-cell transcriptomes using MATQ-seq. *Nat Methods* 2017;**14**(3):267–70.
23. Jaitin DA, Kenigsberg E, Keren-Shaul H, *et al*. Massively parallel single-cell RNA-seq for marker-free decomposition of tissues into cell types. *Science* 2014;**343**(6172):776–9.
24. Hashimshony T, Wagner F, Sher N, *et al*. CEL-Seq: single-cell RNA-Seq by multiplexed linear amplification. *Cell Rep* 2012;**2**(3):666–73.
25. Rosenberg AB, Roco CM, Muscat RA, *et al*. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science* 2018;**360**(6385):176–82.
26. Pollen AA, Nowakowski TJ, Shuga J, *et al*. Low-coverage single-cell mRNA sequencing reveals cellular heterogeneity and activated signaling pathways in developing cerebral cortex. *Nat Biotechnol* 2014;**32**(10):1053.
27. Gierahn TM, Wadsworth MH, II, Hughes TK, *et al*. Seq-well: portable, low-cost RNA sequencing of single cells at high throughput. *Nat Methods* 2017;**14**(4):395–8.
28. Islam S, Zeisel A, Joost S, *et al*. Quantitative single-cell RNA-seq with unique molecular identifiers. *Nat Methods* 2014;**11**(2):163.
29. Ziegenhain C, Vieth B, Parekh S, *et al*. Comparative analysis of single-cell RNA sequencing methods. *Mol Cell* 2017;**65**(4):631–43.
30. Haque A, Engel J, Teichmann SA, *et al*. A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. *Genome Med* 2017;**9**(1):75.
31. Goh WWB, Wang W, Wong L. Why batch effects matter in omics data, and how to avoid them. *Trends Biotechnol* 2017;**35**(6):498–507.
32. Tung P-Y, Blischak JD, Hsiao CJ, *et al*. Batch effects and the effective design of single-cell gene expression studies. *Sci Rep* 2017;**7**:39921.
33. Tran HTN, Ang KS, Chevrier M, *et al*. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biol* 2020;**21**(1):12.
34. Zhu L, Lei J, Devlin B, *et al*. A unified statistical framework for single cell and bulk RNA sequencing data. *Ann Appl Stat* 2018;**12**(1):609.
35. Hou W, Ji Z, Ji H, *et al*. A systematic evaluation of single-cell RNA-sequencing imputation methods. *bioRxiv* 2020. doi: 10.1101/2020.01.29.925974.
36. Agarwal D, Wang J, Zhang NR, *et al*. Data denoising and post-denoising corrections in single cell RNA sequencing. *Stat Sci* 2020;**35**(1):112–28.
37. Lähnemann D, Köster J, Szczurek E, *et al*. Eleven grand challenges in single-cell data science. *Genome Biol* 2020;**21**(1):1–35.
38. Gong W, Kwak I-Y, Pota P, *et al*. DrImpute: imputing dropout events in single cell RNA sequencing data. *BMC Bioinformatics* 2018;**19**(1):220.
39. Tjaernberg A, Mahmood O, Jackson CA, *et al*. Optimal tuning of weighted kNN- and diffusion-based methods for denoising single cell genomics data. *bioRxiv* 2020. doi: 1101/2020.02.28.970202.
40. Ye P, Ye W, Ye C, *et al*. scHinter: imputing dropout events for single-cell RNA-seq data with limited sample size. *Bioinformatics* 2020;**36**(3):789–97.
41. Wagner F, Yan Y, Yanai I. K-nearest neighbor smoothing for high-throughput single-cell RNA-seq data. *bioRxiv* 2018. doi: 10.1101/217737.
42. Moussa M, Măndoiu II. Locality sensitive imputation for single cell RNA-seq data. *J Comput Biol* 2019;**26**(8):822–35.
43. Van Dijk D, Sharma R, Nainys J, *et al*. Recovering gene interactions from single-cell data using data diffusion. *Cell* 2018;**174**(3):716–29.
44. Ronen J, Akalin A. netSmooth: network-smoothing based imputation for single cell RNA-seq. *F1000Res* 2018; 7:8.
45. Jeong H, Liu Z. PRIME: a probabilistic imputation method to reduce dropout effects in single cell RNA sequencing. *Bioinformatics* 2020;**36**(13):4021–9.
46. Tracy S, Yuan G-C, Dries R. RESCUE: imputing dropout events in single-cell RNA-sequencing data. *BMC Bioinformatics* 2019;**20**(1):388.
47. Wu W, Dai Q, Liu Y, *et al*. G2S3: a gene graph-based imputation method for single-cell RNA sequencing data. *bioRxiv* 2020. doi: 10.1101/2020.04.01.020586.
48. Jin K, Ou-Yang L, Zhao X-M, *et al*. scTSSR: gene expression recovery for single-cell RNA sequencing using two-side sparse self-representation. *Bioinformatics* 2020;**36**(10):3131–8.
49. Leote AC, Wu X, Beyer A. Network-based imputation of dropouts in single-cell RNA sequencing data. *bioRxiv* 2019. doi: 10.1101/611517.
50. Elyanow R, Dumitrascu B, Engelhardt BE, *et al*. netNMF-sc: leveraging gene–gene interactions for imputation and dimensionality reduction in single-cell expression analysis. *Genome Res* 2020;**30**(2):195–204.
51. Wang J, Agarwal D, Huang M, *et al*. Data denoising with transfer learning in single-cell transcriptomics. *Nat Methods* 2019;**16**(9):875–8.
52. Peng T, Zhu Q, Yin P, *et al*. SCRABBLE: single-cell RNA-seq imputation constrained by bulk RNA-seq data. *Genome Biol* 2019;**20**(1):88.
53. Ye W, Ji G, Ye P, *et al*. scNPF: an integrative framework assisted by network propagation and network fusion for preprocessing of single-cell RNA-seq data. *BMC Genomics* 2019;**20**(1):347.
54. Badsha MB, Li R, Liu B, *et al*. Imputation of single-cell gene expression with an autoencoder neural network. *Quant Biol* 2020;1–17.
55. Zhu L, Lei J, Devlin B, *et al*. A unified statistical framework for single cell and bulk RNA sequencing data. *Ann Appl Stat* 2018;**12**(1):609.
56. Talwar D, Mongia A, Sengupta D, *et al*. AutoImpute: autoencoder based imputation of single-cell RNA-seq data. *Sci Rep* 2018;**8**(1):1–11.
57. Arisdakessian C, Poirion O, Yunits B, *et al*. DeepImpute: an accurate, fast, and scalable deep neural network method to impute single-cell RNA-seq data. *Genome Biol* 2019;**20**(1):1–14.
58. Eraslan G, Simon LM, Mircea M, *et al*. Single-cell RNA-seq denoising using a deep count autoencoder. *Nat Commun* 2019;**10**(1):390.
59. Zhang X-F, Ou-Yang L, Yang S, *et al*. EnImpute: imputing dropout events in single-cell RNA-sequencing data via ensemble learning. *Bioinformatics* 2019;**35**(22):4827–9.

60. Rao J, Zhou X, Lu Y, *et al*. Imputing single-cell RNA-seq data by combining graph convolution and autoencoder neural networks. *bioRxiv* 2020. doi: 10.1101/2020.02.05.935296.
61. Xu Y, Zhang Z, You L, *et al*. scIGANs: single-cell RNA-seq imputation using generative adversarial networks. *bioRxiv* 2020. doi: 10.1101/2020.01.20.913384.
62. Amodio M, Van Dijk D, Srinivasan K, *et al*. Exploring single-cell data with deep multitasking neural networks. *Nat Methods* 2019;**62**:1139–45.
63. Deng Y, Bao F, Dai Q, *et al*. Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning. *Nat Methods* 2019;**16**(4):311–4.
64. Lopez R, Regier J, Cole MB, *et al*. Deep generative modeling for single-cell transcriptomics. *Nat Methods* 2018;**15**(12):1053–8.
65. Mehtonen J, González G, Kramer R, *et al*. Semisupervised generative autoencoder for single-cell data. *J Comput Biol* 2019;**27**(8):1190–203.
66. Zhu K, Anastassiou D. 2DImpute: imputation in single-cell RNA-seq data from correlations in two dimensions. *Bioinformatics* 2020;**36**(11):3588–9.
67. Tran B, Tran D, Nguyen H, *et al*. Ria: a novel regression-based imputation approach for single-cell RNA sequencing. In: *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, 2019. pp. 1–9. New York City, NY, USA: IEEE.
68. Linderman GC, Zhao J, Kluger Y. Zero-preserving imputation of scRNA-seq data using low-rank approximation. *bioRxiv* 2018. doi: 10.1101/397588.
69. Wagner F, Barkley D, Yanai I. Accurate denoising of single-cell RNA-seq data using unbiased principal component analysis. *bioRxiv* 2019;**655365**. doi: 10.1101/655365.
70. Chen C, Wu C, Wu L, *et al*. scRMD: imputation for single cell RNA-seq data via robust matrix decomposition. *Bioinformatics* 2020;**36**(10):3156–61. 03.
71. Xu J, Cai L, Liao B, *et al*. CMF-Impute: an accurate imputation tool for single-cell RNA-seq data. *Bioinformatics* 2020;**36**(10):3139–47.
72. Mongia A, Sengupta D, Majumdar A. deepMc: deep matrix completion for imputation of single-cell RNA-seq data. *J Comput Biol* 2019;**27**(7):1011–9.
73. Mongia A, Sengupta D, Majumdar A. McImpute: matrix completion based imputation for single cell RNA-seq data. *Front Genet* 2019;**10**:9.
74. Zhang L, Zhang S. PBLR: an accurate single cell RNA-seq data imputation tool considering cell heterogeneity and prior expression level of dropouts. *bioRxiv* 2018. doi: 10.1101/379883.
75. Hu Y, Li B, Liu N, *et al*. WEDGE: recovery of gene expression values for sparse single-cell RNA-seq datasets using matrix decomposition. *bioRxiv* 2019;864488. https://doi.org/10.1101/864488.
76. Pierson E, Yau C. ZIFA: dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biol* 2015;**16**(1):241.
77. Aparicio L, Bordyuh M, Blumberg AJ, *et al*. A random matrix theory approach to denoise single-cell data. *Patterns* 2020;;**1**(3):100035.
78. Tang W, Bertaux F, Thomas P, *et al*. bayNorm: Bayesian gene expression recovery, imputation and normalization for single-cell RNA-sequencing data. *Bioinformatics* 2020;**36**(4):1174–81.
79. Azizi E, Prabhakaran S, Carr A, *et al*. Bayesian inference for single-cell clustering and imputing. *Genomics Comput Biol* 2017;**3**(1):e46–6.
80. Song F, Chan GM, Wei Y. Flexible experimental designs for valid single-cell RNA-sequencing experiments allowing batch effects correction. *Nat Commun* 2020;**11**:3274.
81. Lin P, Troup M, Ho JWK. CIDR: ultrafast and accurate clustering through imputation for single-cell RNA-seq data. *Genome Biol* 2017;**18**(1):59.
82. Yang MQ, Weissman SM, Yang W, *et al*. MISC: missing imputation for single-cell RNA sequencing data. *BMC Syst Biol* 2018;**12**(7):114.
83. Huang M, Wang J, Torre E, *et al*. SAVER: gene expression recovery for single-cell RNA sequencing. *Nat Methods* 2018;**15**(7):539.
84. Li WV, Li JJ. An accurate and robust imputation method scImpute for single-cell RNA-seq data. *Nat Commun* 2018;**9**(1):997.
85. Miao Z, Li J, Zhang X. scRecover: discriminating true and false zeros in single-cell RNA-seq data for imputation. *bioRxiv* 2019;665323. https://doi.org/10.1101/665323.
86. Zhang Y, Liang K, Liu M, *et al*. SCRIBE: a new approach to dropout imputation and batch effects correction for single-cell RNA-seq data. *bioRxiv* 2019;793463. https://doi.org/10.1101/793463.
87. Hu Z, Songpeng Z, Liu JS. SIMPLEs: a single-cell RNA sequencing imputation strategy preserving gene modules and cell clusters variation. *bioRxiv* 2020. doi: 10.1101/2020.01.13.904649.
88. Chen M, Zhou X. VIPER: variability-preserving imputation for accurate gene expression recovery in single-cell RNA sequencing studies. *Genome Biol* 2018;**19**(1):1–15.
89. Zhang X, Xu C, Yosef N. Simulating multiple faceted variability in single cell RNA sequencing. *Nat Commun* 2019;**10**(1):2611.
90. Zheng GXY, Terry JM, Belgrader P, *et al*. Massively parallel digital transcriptional profiling of single cells. *Nat Commun* 2017;**8**(1):1–12.
91. Tian L, Dong X, Freytag S, *et al*. Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments. *Nat Methods* 2019;**16**(6):479–87.
92. Segerstolpe Å, Palasantza A, Eliasson P, *et al*. Single-cell transcriptome profiling of human pancreatic islets in health and type 2 diabetes. *Cell Metab* 2016;**24**(4): 593–607.
93. Ho Y-J, Anaparthy N, Molik D, *et al*. Single-cell RNA-seq analysis identifies markers of resistance to targeted BRAF inhibitors in melanoma cell populations. *Genome Res* 2018;**28**(9):1353–63.
94. Andrews TS, Hemberg M. False signals induced by single-cell imputation. *F1000Res* 2018;**7**:1740.
95. Zhang L, Zhang S. Comparison of computational methods for imputing single-cell RNA-sequencing data. *IEEE/ACM Trans Comput Biol Bioinform* 2018;**17**(2):376–89.
96. Coifman RR, Lafon S. Diffusion maps. *Appl Comput Harmon Anal* 2006;**21**(1):5–30.
97. Goodfellow I, Bengio Y, Courville A. *Deep Learning*. Cambridge, MA, USA: The MIT Press, 2016.
98. Wang Y-X, Zhang Y-J. Nonnegative matrix factorization: a comprehensive review. *IEEE Trans Knowl Data Eng* 2012;**25**(6):1336–53.
99. Candès EJ, Recht B. Exact matrix completion via convex optimization. *Found Comput Math* 2009;**9**(6):717.
100. Eckart C, Young GM. The approximation of one matrix by another of lower rank. *Psychometrika* 1936;**1**:211–8.
101. Sun Y, Babu P, Palomar DP. Majorization-minimization algorithms in signal processing, communications, and

machine learning. *IEEE Trans Signal Process* 2016;**65**(3):816–794.

102. Livan G, Novaes M, Vivo P. *Introduction to Random Matrices: Theory and Practice*, Vol. 26. London, UK: Springer, 2018.
103. Hsu D. Kakade SM, Zhang T. Robust matrix decomposition with sparse corruptions. *IEEE Trans Inf Theory* 2011;**57**(11):7221–34.
104. Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 2002. pp. 849–56. Vancouver, BC, Canada: MIT press.
105. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J Comput Appl Math* 1987;**20**:53–65.
106. van der Maaten L, Hinton G. Visualizing data using t-SNE. *J Mach Learn Res* 2008;**9**:2579–605.
107. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol* 2018; **19**(1):15.
108. Butler A, Hoffman P, Smibert P, *et al*. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nat Biotechnol* 2018;**36**(5):411.
109. Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. *Sci Rep* 2019;**9**(1):1–12.
110. Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol Syst Biol* 2019;**15**(6):e8746.
111. Ramazzotti D, Angaroni F, Maspero D, *et al*. Longitudinal cancer evolution from single cells. *bioRxiv* 2020. doi: 10.1101/2020.01.14.906453.
112. Zhou Z, Xu B, Minn A, *et al*. DENDRO: genetic heterogeneity profiling and subclone detection by single-cell RNA sequencing. *Genome Biol* 2020;**21**(1):1–15.
113. Damiani C, Maspero D, Di Filippo M, *et al*. Integration of single-cell RNA-seq data into population models to characterize cancer metabolism. *PLoS Comput Biol* 2019;**15**(2):e1006733.
114. Graudenzi A, Maspero D, Damiani C. FBCA, a multiscale modeling framework combining cellular automata and flux balance analysis. *J Cell Autom* 2020;**15**:75–95.

A.2 Classifying Cancer Samples from Metabolic Networks

Contribution. In this chapter I will discuss the work done for the following articles:

[Mac+21] J. Machicao, F. Craighero, D. Maspero, F. Angaroni, C. Damiani, A. Graudenzi, M. Antoniotti, O. M. Bruno. "On the Use of Topological Features of Metabolic Networks for the Classification of Cancer Samples". In: *CURRENT GENOMICS* 22.2 (2021)

Summary. The recent surge in availability and reliability of -omics data encouraged the development of computational methods to investigate the metabolism in cancer [TK12; LA13]. In this regard, classifying cancer samples from metabolic properties represents a fundamental challenge towards a data-driven understanding of the disease. As done in [Gra+18; Dam+20], we first built a dataset by projecting transcriptomic data onto metabolic networks, allowing to derive an approximate activity value for each reaction, represented by the edges. Then, our contributions can be summarized as follows:

- *Metabolic Networks Pruning:* we proposed a pruning strategy to keep only the relevant edges (metabolic reactions) of the network.
- *Metabolic Network Topological Properties as Features:* we selected network metrics such as the average degree and the assortativity as relevant properties for the classification task.
- *Model Selection:* we evaluated multiple classifiers with a nested cross-validation to select the best performing one in our setting.

The experiments were performed using labelled expression profiles from the TCGA dataset [Cir+15], projected onto the Recon2.2 metabolic network [Jam+19]. The final results confirmed the goodness of metabolic networks' topological features to distinguish cancer samples, motivating additional studies on this topic.

Implementation. The experiments performed in the paper has been open-sourced on a Github repository[a].

[a]`https://github.com/BIMIB-DISCo/MET-NET-CLASSIFICATION`

RESEARCH ARTICLE

On the Use of Topological Features of Metabolic Networks for the Classification of Cancer Samples

Jeaneth Machicao[1,2,+,*], Francesco Craighero[3,+], Davide Maspero[3,4], Fabrizio Angaroni[3], Chiara Damiani[5,6,†], Alex Graudenzi[4,7,†,*], Marco Antoniotti[3,7,†] and Odemir M. Bruno[1,†,*]

[1]*São Carlos Institute of Physics, University of São Paulo, São Carlos, Brazil;* [2]*School of Engineering, University of São Paulo, São Paulo, Brazil;* [3]*Department of Informatics, Systems and Communication, University of Milan-Bicocca, Milan, Italy;* [4]*Institute of Molecular Bioimaging and Physiology, Consiglio Nazionale delle Ricerche (IBFM-CNR), Segrate, Milan, Italy;* [5]*Department of Biotechnology and Biosciences, University of Milan-Bicocca, Milan, Italy;* [6]*Sysbio Centre for Systems Biology, Milan, Italy;* [7]*Bicocca Bioinformatics, Biostatistics and Bioimaging Center (B4), University of Milan-Bicocca, Milan, Italy*

ARTICLE HISTORY

Received: July 07, 2020
Revised: December 16, 2020
Accepted: December 18, 2020

DOI:
10.2174/1389202922666210301084151

Abstract: ***Background***: The increasing availability of omics data collected from patients affected by severe pathologies, such as cancer, is fostering the development of data science methods for their analysis.

Introduction: The combination of data integration and machine learning approaches can provide new powerful instruments to tackle the complexity of cancer development and deliver effective diagnostic and prognostic strategies.

Methods: We explore the possibility of exploiting the topological properties of sample-specific metabolic networks as features in a supervised classification task. Such networks are obtained by projecting transcriptomic data from RNA-seq experiments on genome-wide metabolic models to define weighted networks modeling the overall metabolic activity of a given sample.

Results: We show the classification results on a labeled breast cancer dataset from the TCGA database, including 210 samples (cancer *vs*. normal). In particular, we investigate how the performance is affected by a threshold-based pruning of the networks by comparing Artificial Neural Networks, Support Vector Machines and Random Forests. Interestingly, the best classification performance is achieved within a small threshold range for all methods, suggesting that it might represent an effective choice to recover useful information while filtering out noise from data. Overall, the best accuracy is achieved with SVMs, which exhibit performances similar to those obtained when gene expression profiles are used as features.

Conclusion: These findings demonstrate that the topological properties of sample-specific metabolic networks are effective in classifying cancer and normal samples, suggesting that useful information can be extracted from a relatively limited number of features.

Keywords: Metabolic networks, cancer sample classification, machine learning, RNA-seq data, topological properties, network pruning.

1. INTRODUCTION

The development of automated strategies for the classification of cancer samples in distinct categories (*e.g.*, subtypes, risk groups, *etc.*) is one of the key challenges in current biosciences [1]. On the one hand, this might lead to the discovery of efficient, personalized diagnostic, prognostic, and therapeutic strategies for cancer patients. On the other hand, it could allow unraveling some of the still undeciphered mechanisms and processes underlying cancer development, leading to a data-driven understanding of the disease.

It is known that effective classification and clustering of cancer samples can be achieved by employing the information on expression data [2-7], genomic alteration profiles [8, 9], interaction networks [10], and even signaling pathways [11, 12]. In this work, however, we specifically focus on the metabolic properties that may distinguish cancer from normal samples. In fact, metabolic deregulation is one of the key hallmarks of cancer [13-15], even if its underlying mechanisms are still partially unknown. In this respect, in recent years, an increasing number of computational strate-

*Address correspondence to these authors at the São Carlos Institute of Physics, University of São Paulo, São Carlos, Brazil; Institute of Molecular Bioimaging and Physiology, Consiglio Nazionale delle Ricerche (IBFM-CNR), Segrate, Milan, Italy
E-mails: machicao@usp.br, alex.graudenzi@ibfm.cnr.it, bruno@ifsc.usp.br
[+]*Co-first authors*; [†]*Co-senior authors.*

gies have been devised, in order to take advantage of the growing availability and reliability of -omics data to investigate the alterations of metabolism in cancer [16-19]. Very often, such data have been employed in constraint-based models, such as Flux Balance Analysis (FBA), in which metabolic fluxes are simulated to compare different experimental scenarios [20-24].

Moreover, more recently, approaches coupling constraint-based metabolic modeling with supervised machine learning algorithms have been proposed [25]. In our case, we explore for the first time the possibility of employing the topological properties of metabolic networks as input features of classification algorithms. To this end, we rely on an approach firstly introduced in [26,27] in which transcriptomic data, such as RNA-seq, are employed to determine the approximate activity value of the reactions included in a given metabolic network.

More in detail, by introducing a relevance threshold on the metabolic activity level, we pruned the original metabolic network to define individual-specific networks in which only the significantly active reactions are preserved. The topological properties of such individual-specific networks are then used as features to perform a supervised classification task via various algorithmic strategies and, in particular, Multi-Layer Perceptrons (MLPs), Support Vector Machines (SVMs) and Random Forests (RFs).

To investigate our hypothesis, this work presents the classification results in a simple scenario in which the sample categories are known a priori - cancer *vs*. normal - concerning the TCGA-BRCA breast cancer dataset [28], which includes 210 total samples.

We show that noteworthy classification performance can be achieved by using a few key topological properties of metabolic networks, *i.e.*, average degree, average hierarchical degree, average geodesic path length and assortativity. Interestingly, a similar pruning threshold (in the range 0.01 – 0.1) is identified as optimal for all tested machine learning strategies, suggesting that it could be an effective choice to extract useful information from the "relevant" activity of metabolic networks, while discarding possible artifacts due to noisy observations. Overall, the best classification performance is obtained with SVMs and threshold 0.1, which exhibit 0.866 of (average) accuracy, 0.86 precision and 0.879 recall on the test set, after k-fold cross-validation and hyper-parameter estimation. Furthermore, we show that the best performing SVM classifier (with the optimal threshold) delivers similar classification performance with respect to an analogous classifier processing a reduced gene expression feature vector, as computed by selecting the 5 principal components on the list of 1673 metabolic genes from Recon2.2 [29].

These results prove that the projection of transcriptomic activity on metabolic networks provides useful information to efficiently classify cancer samples and might pave the way for the development of strategies for experimental hypothesis generation.

2. MATERIALS AND METHODS

2.1. Integration of RNA-seq and Metabolic Networks

As proposed earlier [26, 27], it is possible to project transcriptomic data onto human metabolic networks [30], to derive an approximate activity value for each metabolic reaction in any given sample.

We first employ an input metabolic network M such as the Human Metabolic Reaction (HMR) [31] or Recon [29, 32]. M is a bipartite-directed graph that includes two kinds of nodes: (*i*) metabolites (*i.e.*, substrates or products), and (*ii*) metabolic reactions. The edges in M connect either: (*i*) the substrates and the relative reaction, or (*ii*) a reaction and the relative products. The total number of nodes of M is N, whereas the total number of edges is E. Reaction nodes are associated with Gene-Protein-Reaction (GPR) rules, *i.e.*, logical formulas that describe the related catalyses *via* AND and OR logical operators. In particular, AND rules are employed when distinct genes encode different *subunits* of the same enzyme, whereas OR rules are used when distinct genes encode *isoforms* of the same enzyme.

RNA-seq data are then used to provide an approximate activity value to each reaction in the input network. In particular, our method takes as input a n (*genes*) × m (*samples*) matrix T in which each element $T_{g,s}$, $g = 1, \dots, n$, $s = 1, \dots, m$, includes the transcript level of gene g in sample s (the *Reads per Kilobase per Million* mapped reads – RPKM).

For each reaction in the input network $r \in G$ and for each sample $s = 1, \dots, m$, we define a *Reaction Activity Score* (RAS), by distinguishing two cases.

Reactions with GPR including an AND operator,

$$RAS_{r,s} = \min\left(T_{g,s} : g \in \mathcal{A}_r\right), \quad (1)$$

where $\mathcal{A}_r$ is the set of genes that encode the subunits of the enzyme catalyzing reaction r.

Reactions with GPR including an OR operator,

$$RAS_{r,s} = \sum_{g \in \mathcal{O}_r} T_{g,s}, \quad (2)$$

where $\mathcal{O}_r$ is the set of genes that encode isoforms of the enzyme that catalyzes reaction r.

In case of composite reactions, we respect the standard precedence of the two operators. The rationale underlying the definition of the RAS is that enzyme isoforms (OR) contribute *additively* to the overall activity of a certain reaction, whereas enzyme subunits (AND) *limit* its activity. RASs are finally normalized to obtain values in the range [0, 1] (with 0 meaning *no activity* and 1 meaning *maximum activity observed in the dataset*).

Even though this simplified approach neglects the heterogeneity of reaction kinetic constants, protein binding affinities and translation rates, it was proven effective in the investigation of cancer metabolic deregulation and in cancer sample stratification [26, 27].

2.2. Cancer Sample Classification *via* Metabolic Network Pruning

We define the sample-specific metabolic network of a given sample s as the weighted adjacency matrix W^s, which contains $N \times N$ elements, such that each element w_{ij}^s is equal to: (*i*) $RAS_{j,s}$ if i is a substrate of reaction j, (*ii*) $RAS_{i,s}$ if i is a reaction and j one of its products, (*iii*) 0 otherwise.

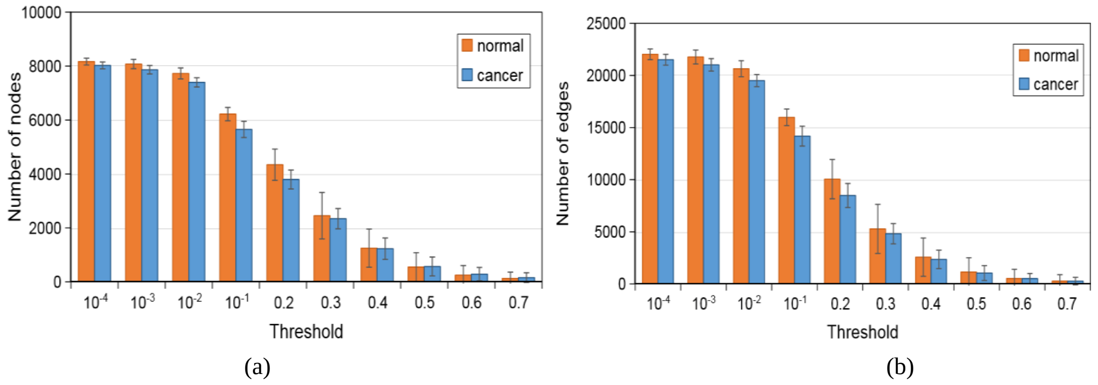


Fig. (1). Number of nodes $\langle N^{T_l,s}\rangle$ (averaged on all samples) (**a**) and number of edges $\langle E^{T_l,s}\rangle$ (averaged on all samples) (**b**) of the giant component $G^{T_l,s}$ of the sample-specific metabolic network (computed from the Recon2.2 network [29]), in addition to their standard deviation (error bar), defined by different threshold T_l values either on normal and cancer samples. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

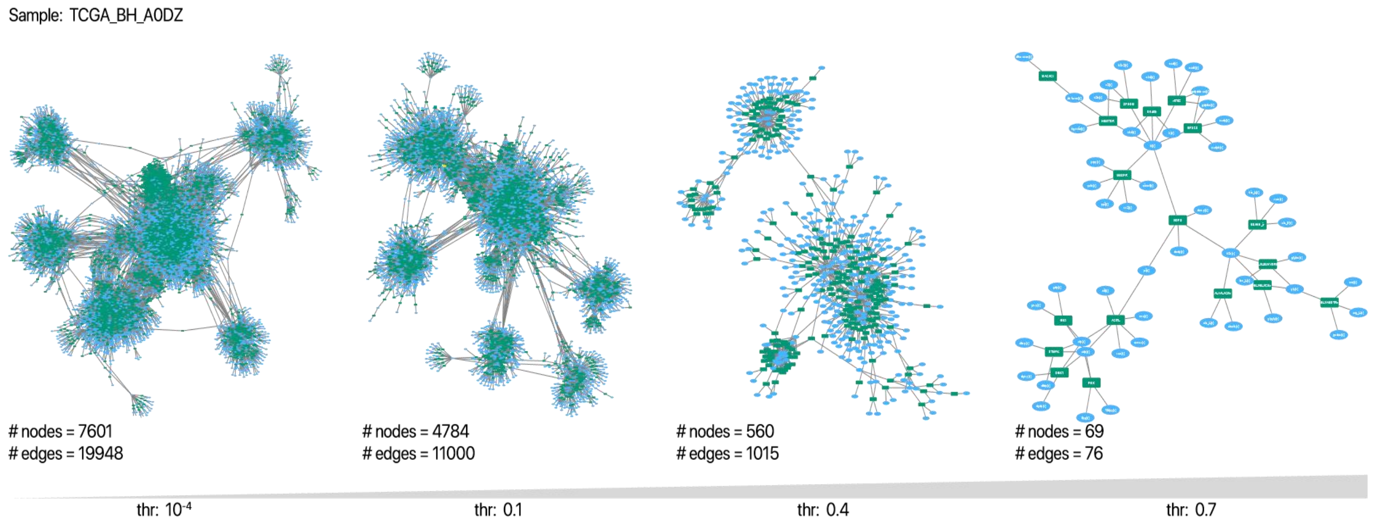


Fig. (2). The giant components of the metabolic network of the cancer sample of patient TCGA BH A0DZ obtained by projecting RNA-seq data on Recon2.2 metabolic network [29], are shown. 4 distinct giant components are shown, obtained with the following relevance thresholds: 10^{-4}, 0.1,0.4,0.7. Networks were drawn *via* Cytoscape [37]. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

Since we are interested in exploiting the topological properties of the "*giant component*" of the sample-specific metabolic network (as proposed, *e.g.*, in [33]), we employ a network pruning procedure to select the relevant metabolic reactions. This threshold criterion was employed earlier [34-36]. In detail, a threshold parameter $T_l \in [0,1]$ is used to obtain an unweighted and thresholded adjacency matrix $A^{T_l,s}$, the elements of which are defined as follows:

$$A_{ij}^{T_l,s} = \begin{cases} 1, \ if \ w_{ij}^s \geq T_l \\ 0, \ if \ w_{ij}^s < T_l \end{cases} \forall i,j = 1, \ldots, N. \tag{3}$$

It must be noted that we have focused on the *larger than* option, because we can hypothesize that only significantly active reactions (above the threshold) are responsible for the phenotypic/functional properties of cells. By scanning different values of the threshold, we can then evaluate the impact on the performance of classifiers that take as input certain topological measurements of the resulting giant component (see below), thus identifying an optimal threshold value.

Clearly the threshold parameter determines the size of the giant component, *i.e.*, the largest connected subgraph of the sample-specific metabolic network, which we define as $G^{T_l,s}$ and which includes $N^{T_l,s}$ nodes and $E^{T_l,s}$ edges.

For instance, in Fig. (**1**), one can see how the number of nodes and edges of the giant component of the sample-specific metabolic network (computed from the Recon2.2 network [29, 32]) is generally affected by the choice of distinct thresholds, regarding both cancer and normal samples. In greater detail, on the left side of Fig. (**1a**), smaller thresholds, such as $T_l \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, lead to a larger size of the giant component (on average), while on the right side, larger thresholds, such as $T_l \in \{0.2, 0.3, \ldots, 0.7\}$, lead to a radical network reduction, with a threshold $T_l = 0.7$ retaining 127 nodes on average, which represents approximately 1.45% of the total number of nodes of the original metabolic network. As a representative example, the shrinking of the giant component for a specific sample is visually represented in Fig. (**2**).

We also note that this behavior occurs similarly on both cancer and normal samples, even if the size of the giant component of the former ones tends to be slightly smaller. One may speculate that cancer subpopulations engage in a relatively lower number of metabolic functions with respect to normal cells, given that their main objective is "selfish" proliferation. Further investigations are needed to validate this interesting hypothesis [37].

2.3. Algorithmic Methods for Classification

In general, the choice of adequate network descriptors is crucial for pattern recognition purposes. Typically, the feature extraction is based on well-established network structural measures (see details in Section 2.3.1). The concurrent use of well-known measures such as *degree*, *mean degree*, *clustering coefficient*, *mean hierarchical degree*, *centrality*, and even *spectral measurements*, can identify global properties shared by a large majority of empirical and synthetic networks such as random, small-world, scale-free networks, and geographic networks models [38, 39].

2.3.1. Features Based on Network Structural Measures

Networks measurements falling in various categories (*e.g.*, connectivity-related, distance-related, spectral, degree correlation measures) can be effectively used to characterize the topological properties of real-world networks [38, 40]. In our case, we are interested in determining whether certain topological measurements of the giant component of the sample-specific metabolic network obtained from RNA-seq data projection, and after opportune threshold-based pruning, can be effectively employed as features to classify cancer samples. In particular, we selected the following measures.

Average Degree: Among the connectivity-related measurements, we here consider the degree (or connectivity) $k_i^{T_l,s}$ of node *i* of the giant component of sample *s*, given threshold T_l, as the number of neighbors of a node $i^{T_l,s}$ defined by:

$$k_i^{T_l,s} = \sum_{j=1}^{N^{T_l,s}} A_{ij}^{T_l,s}.$$

Accordingly, the average degree of the giant component is defined by Eq. (4), as follows:

$$\langle k^{T_l,s} \rangle = \frac{1}{N^{T_l,s}} \sum_{i=1}^{N^{T_l,s}} k_i^{T_l,s}. \tag{4}$$

Average Hierarchical Degree: The hierarchical degree $\mathrm{k_i^{T_l,s^h}}$ of node i can also be measured considering the connectivity of the neighboring nodes constrained to a hierarchical level h. As an example, in social networks, the hierarchical degree of level 2 of given node i, $\mathrm{k_i^2}$, is the sum of the degrees of the neighbors of its neighbors. Therefore, the mean hierarchical degree of the giant component of a sample-specific metabolic network is given by Eq. (5), as follows:

$$\langle k^{T_l,s^h} \rangle = \frac{1}{N^{T_l,s}} \sum_{i=1}^{N^{T_l,s}} k_i^{T_l,s^h}. \tag{5}$$

Average Geodesic Path Length: A path is defined as the sequence of nodes visited to go from node *i* to *j*. The distance between them is the number of edges within the path, and d_{ij} is defined as the geodesic path, *i.e.*, the smallest path length. When there is no path between *i* and j, $d_{ij} = 0$. The average geodesic path length of the giant component of the sample-specific metabolic network is given by:

$$\langle l^{T_l,s} \rangle = \frac{1}{N^{T_l,s}(N^{T_l,s}-1)} \sum_{i \neq j} d_{ij}, \tag{6}$$

where i and j are two nodes of the giant component and $\frac{1}{N^{T_l,s}(N^{T_l,s}-1)}$ corresponds to a normalization factor, considering a fully connected network [40].

Assortativity: The assortativity $\Gamma^{T_l,s}$ [41], *i.e.*, the Pearson correlation coefficient of degree among all pairs of linked nodes i and j of the giant component, quantifies the tendency of the nodes of a given degree k to connect to nodes with a similar degree and, in our case, it is defined as follows:

$$\Gamma^{T_l,s} = \frac{\left(\frac{1}{N^{T_l,s}}\right)\sum_{j>i}\left(k_i^{T_l,s}k_j^{T_l,s}A_{ij}^{T_l,s}\right) - \left[(1/N^{T_l,s})\sum_{j>i}(1/2)\left(k_i^{T_l,s}+k_j^{T_l,s}\right)A_{ij}^{T_l,s}\right]^2}{\left(\frac{1}{N^{T_l,s}}\right)\sum_{j>i}(1/2)\left(k_i^{T_l,s^2}+k_j^{T_l,s^2}\right)A_{ij}^{T_l,s} - \left[(1/N^{T_l,s})\sum_{j>i}(1/2)\left(k_i^{T_l,s}+k_j^{T_l,s}\right)A_{ij}^{T_l,s}\right]^2}, \tag{7}$$

$\Gamma^{T_l,s}$ is a value within the range $[-1, 1]$. Values closer to 1 indicate a positive correlation (nodes with high degree tend to connect to nodes with high degree), while values closer to -1, indicate a negative correlation (nodes with a high degree tend to connect to nodes with low degree), whereas values close to 0 indicates the absence of linear dependence.

In the following, we will show how to compose a feature vector by considering a set of topological measurements [35, 36, 38]. In this respect, the giant component of a sample-specific metabolic network $G^{T_l,s}$can be characterized by a tuple containing: *(i)* the average degree $\langle k^{T_l,s} \rangle$ (Eq. 4), *(ii)* the average hierarchical degree of level 2 $\langle k^{T_l,s} \rangle$ (Eq. 5), *(iii)* the average hierarchical degree of level 3 $\langle k^{T_l,s^3} \rangle$ (Eq. 5), *(iv)* the average geodesic path length $\langle l^{T_l,s} \rangle$ (Eq. 6) and *(v)* the assortativity $\Gamma^{T_l,s}$ (Eq. 7). The vector is given by:

$$\vec{\phi}(T_l, s) = [\langle k^{T_l,s} \rangle, \langle k^{T_l,s^2} \rangle, \langle k^{T_l,s^3} \rangle, \langle l^{\mathrm{T}_l,s} \rangle, \Gamma^{T_l,s}] \tag{8}$$

We notice that other measures such as the clustering coefficient might be employed as features. However, since in our case the input network is bipartite, there are no triangle neighborhoods and, accordingly, the clustering coefficient would always be 0. Since our framework is designed to be general, one can expect this feature to be relevant in different experimental scenarios, with distinct datasets and alternative representations of reaction graphs [42-44].

2.4. Classification Setup

Given any relevance threshold T_l, the feature vectors are extracted for the resulting giant component of each sample s, and the classification step can be performed. The main goal of this analysis is to evaluate the classification performance of various classifiers $\mathcal{M}$, *i.e.*, MLPs, SVMs and RFs on the feature vector $\vec{\phi}(T_l, s)$. Furthermore, we tested the same classifiers on a reduced feature vector, including the 5 first principal components of the expression profiles of the 1673 metabolic genes present in the Recon2.2 model [29], in order to provide a comparison on the same number of features employed in our approach.

Table 1. Hyperparameters grid search for the tested classifiers, *i.e.*, MLPs, SVMs and RFs, executed *via* the scikit-learn Python library. Parameter names are the sklearn arguments of the related functions (default was used for the other parameters).

Methods	Functions	Parameters	Grid Search Values
MLP	neuralnetwork.MLPClassifier	solver	[adam, lbfgs]
		hidden_layer_sizes	[(50,),(100,),(50,50)]
		batch_size	[16, 32, 64]
		learning_rate_init	[0.1, 0.01, 0.001]
		learning_rate	[constant, adaptative]
		max_iter	10000
RF	ensemble.RandomForestClassifier	max_depth	[10, 20, 40, None]
		max_features	[auto, sqrt]
		min_samples_leaf	[1, 2, 3]
		min_samples_split	[2, 3, 5]
		n_estimator	[100, 200, 500, 1000]
SVM	svm.SVC	C	[2^{-5}, 2^{-4},..., 2^{12}]
		gamma	[2^{-15} 2^{-14},..., 2^{4}]
		tol	[10^{-3}, 10^{-4}]
		kernel	[rbf, sigmoid, linear]

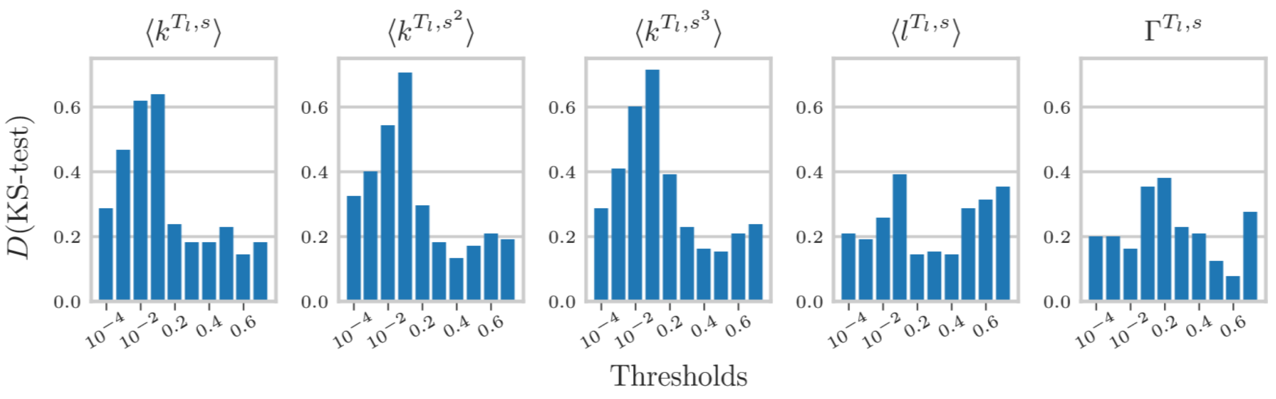


Fig. (3). Kolmogorov-Smirnov statistic (KS-test, [48]) between normal and cancer samples for each threshold and network topological measure: average degree $\langle k^{T_l,s} \rangle$, assortativity $\Gamma^{T_l,s}$ average hierarchical degree of level 2 $\langle k^{T_l,s^2} \rangle$ and 3 $\langle k^{T_l,s^3} \rangle$ and average geodesic path length $\langle l^{T_l,s} \rangle$. The higher the K-S test is, the more the distribution of the network measure is different between normal and cancer samples. The highest values are obtained with $\langle k^{T_l,s} \rangle$, $\langle k^{T_l,s^2} \rangle$, and $\langle k^{T_l,s^3} \rangle$ and thresholds equal to 10^{-2} and 0.1. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

In order to prevent over-optimistic results, we performed for each classifier a nested cross-validation as proposed earlier [45] and detailed as follows.

The original dataset, including cancer and normal samples, is split into 5 folds, ensuring the balance between classes. 5-fold outer cross-validation is executed by using: (*i*) one fold as the test set to assess the model performance and (*ii*) 4 folds in an inner 5-fold cross-validation procedure to select the optimal hyperparameters h of the model $\mathcal{M}(h)$ *via* grid search (Table **1**). The whole procedure is repeated 3 times to ensure robustness to the results. The performance of all classifiers is assessed on average accuracy, precision and recall with respect to ground-truth labels.

All the experiments described above were performed using the scikit-learn Python library [46].

2.5. Network Datasets

We tested our approach on the breast cancer dataset TCGA-BRCA published earlier [28]. We downloaded the dataset via the cBioPortal [47]. This dataset includes the expression profile (RNA Seq V2 RSEM) of biopsies taken from 817 patients. We selected the 105 patients for which the expression profiles of both cancer and normal tissues are provided, for a total of 210 samples used in our analysis.

RNA-seq data were projected on the Recon2.2 metabolic network [29, 32] to obtain a dataset in which a Reaction Activity Score is assigned to each metabolic reaction in each sample (see above). The RASs were then normalized by dividing each reaction score by the maximum value of all samples. Finally, normalized RAS profiles are used to weigh the metabolic network as described above.

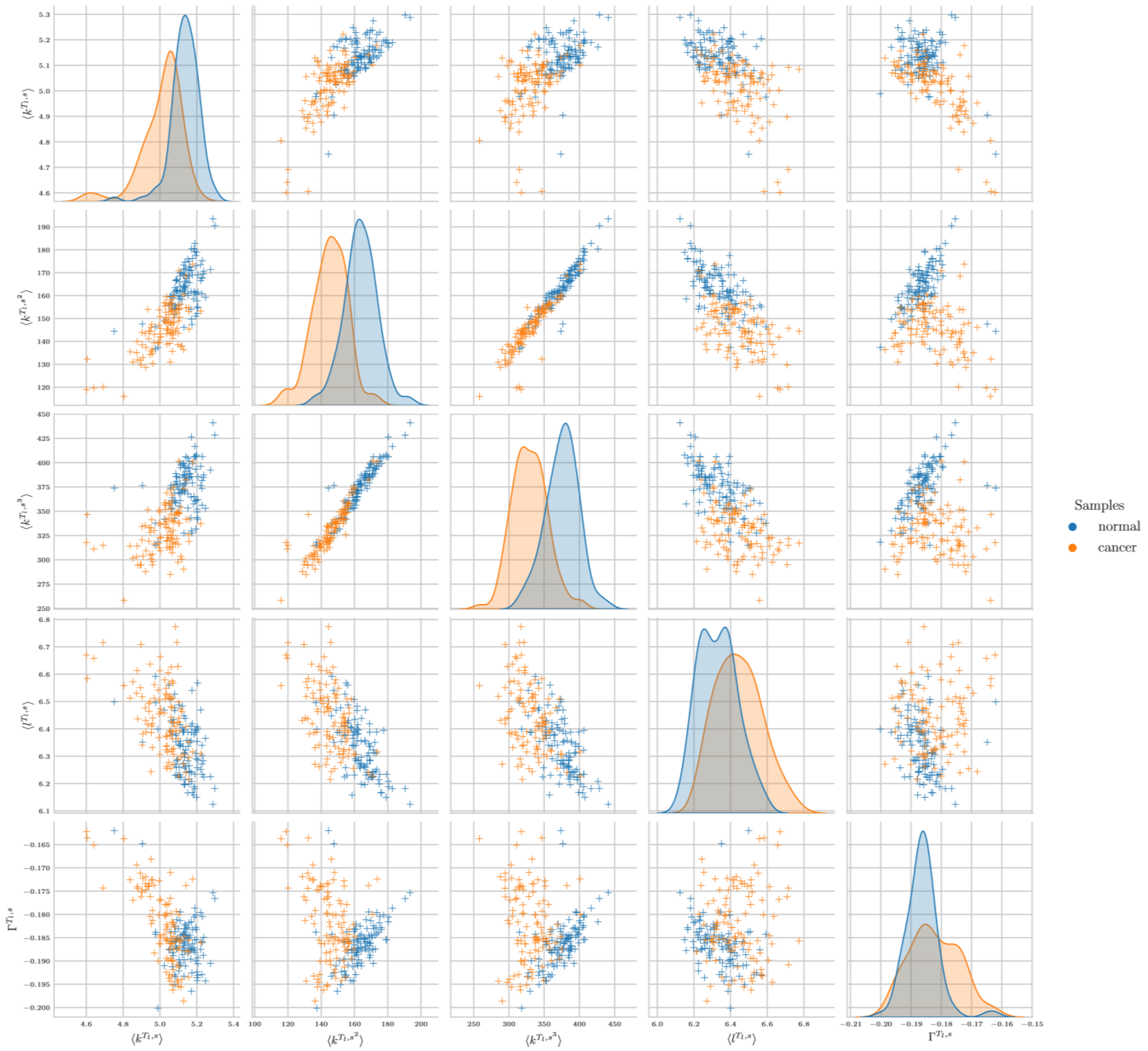


Fig. (4). Projection of cancer and normal samples on the space of topological measure pairs and (on the diagonal) the distribution for each measure and every sample category, for a selected threshold $T_l = 0.1$. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

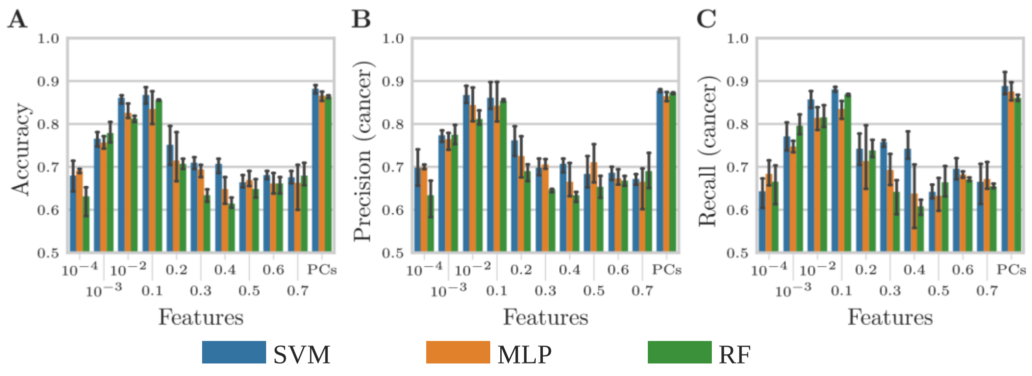


Fig. (5). From left to right: average accuracy (**A**), average precision on cancer samples (**B**) and average recall on cancer samples (**C**) with SVMs, MLPs and RFs. The average is computed on the test sets via a repeated nested cross-validation, for three different seeds, whereas the error bars represent the standard deviation (see Section 2.4 for additional details). The best thresholds are $T_l = 10^{-2}$ and $T_l = 0.1$. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

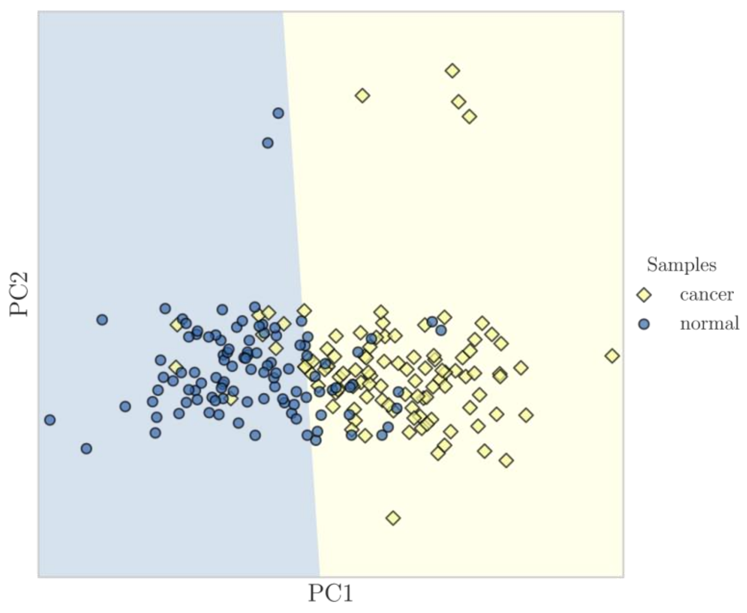


Fig. (6). Decision boundary of the SVM classifier with optimal hyperparameters and threshold $T_l = 0.1$ on the full dataset. The axes correspond to the first two principal components of the full feature vector $\vec{\phi}(T_l, s)$. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

3. RESULTS

3.1. RAS Threshold Analysis

A small T_l will result in larger giant components while, in contrast, higher values of T_l will result in smaller giant components. To choose the best classifier, we evaluated the performance obtained by the following distinct threshold values:

$$T_l \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 0.2, 0.3, \ldots, 0.7\}. \quad (9)$$

Thus, each feature vector $\vec{\phi}(T_l, s)$, contains the five topological measures defined above as descriptors (see Section 2.3.1).

To test the discrimination power of the feature vectors $\vec{\phi}(T_l, s)$, in Fig. (**3**), we computed the Kolmogorov-Smirnov statistic [48] between normal and cancer samples for each threshold and topological measure. The KS statistic D (KS-test) is the distance between the cumulative probability distributions; hence the higher is the value, the more the network measures are different between normal and cancer samples.

As a result, in our dataset, degree statistics, *i.e.*, $\langle k^{T_l,s} \rangle$, $\langle k^{T_l,s^2} \rangle$ and $\langle k^{T_l,s^3} \rangle$, achieve the highest D (KS-test), in particular for thresholds equal to 10^{-2} and 0.1. In Fig. (**4**), we plotted the distributions of all pairs of features in $\vec{\phi}(T_l, s)$, for $T_l = 0.1$. In accordance with the results of Fig. (**3**), the degree statistics distributions and, in particular, $\langle k^{T_l,s^2} \rangle$ and $\langle k^{T_l,s^3} \rangle$, have the sharpest difference among normal and cancer samples.

3.2. Classification Performance

The classification performance was assessed for all classifiers (*i.e.*, MLPs, SVMs and RFs) on the feature vector $\vec{\phi}(T_l, s)$, with regard to all relevance thresholds, via the nested cross-validation procedure described above (see Section 2.4). In addition, we employed as benchmark three analogous classifiers (*i.e.*, MLPs, SVMs and RFs), which were provided as input with a feature vector including the 5 first principal components (PCs) of the expression profiles of the 1673 metabolic genes.

In Fig. (**5**), we report the average accuracy, precision and recall for all tested classifiers, with respect to all relevance thresholds, as well as the benchmark classifiers on gene expression PCs, by employing the ground-truth cancer sample labels (the error bars represent the standard deviation).

Interestingly, the best performance is achieved for all classifiers with thresholds in the small range $T_l = 10^{-2}$ and $T_l = 0.1$, and points at the existence of an effective pruning strategy to maintain the “relevant” active metabolic pathways that discriminate cancer from normal samples, while limiting the confounding effects possibly due to noisy observations and biological variability.

More in detail, the best performing classifier is provided by SVMs, which reach an average accuracy of 0.86 and 0.87, a precision of 0.87 and 0.86 and a recall of 0.86 and 0.88, for $T_l = 10^{-2}$ and $T_l = 0.1$, respectively.

Interestingly, such performance is extremely similar to that obtained with SVMs on the vector of gene expression PCs (average accuracy = 0.88, precision = 0.88 and recall = 0.89) and slightly superior to that of MLPs and RFs on the same vector. This result suggests that the information extracted from the few selected topological measures on the giant component of the sample-specific metabolic network is effective in discriminating cancer from normal samples, similarly to benchmark approaches processing gene expression data (5).

Finally, in Fig. (**6**), the decision boundary of the best performing SVM classifier, *i.e.*, obtained with $T_l = 0.1$ and optimal hyperparameters is displayed on the first two PCs of the feature vector $\vec{\phi}(T_l, s)$, from which one can see that the method is able to correctly classify also the outliers of both categories.

CONCLUSION

In this work, we have introduced a new computational framework for the classification of cancer samples, which combines the integration of transcriptomic data and metabolic networks with state-of-the-art machine learning approaches. This task is of practical relevance in many biomedical contexts and might pave the way for the development of automated strategies for experimental hypothesis generation. In particular, the introduction of our framework contributes to the emerging field of approaches combining sample-specific metabolic modeling with machine learning to classify cancer samples and/or to predict drug response, as recently reviewed [49, 50].

More in detail, we here proved that the information on the metabolic activity of single samples, derived via integration of highly accessible RNA-seq data, can be effectively used to classify healthy and pathological states, a result that appears to be robust when the original networks are significantly pruned via a relevance threshold. All in all, this result would suggest that the useful information to determine possibly aberrant states in a given sample can be derived from the high-level (topological) properties of a relatively limited number of active processes. The identification and characterization of such processes deserve further investigation.

Regarding our machine learning approach, we here relied on classical topological measures, such as degree, hierarchical degrees, average geodesic path length and assortativity, to encode the structural information of the metabolic network. Additional experiments may employ recent graph representation learning techniques [51, 52], including graph kernels [53] and convolutional neural networks on graphs [54], to automatically extract a low-dimensional feature vector of the input network.

We finally remark that extensions of the framework are currently ongoing to test its applicability to more complex scenarios, involving, for instance, multiclass and multi-label classification with respect to cancer subtypes and risk categories.

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

Not applicable.

HUMAN AND ANIMAL RIGHTS

No animals/humans were used for studies that are the basis of this research.

CONSENT FOR PUBLICATION

Not applicable.

AVAILABILITY OF DATA AND MATERIALS

The datasets generated and analyzed for this study can be found at this link: https://github.com/BIMIB-DISCo/METNET-CLASSIFICATION.

FUNDING

Financial support from the Italian Ministry of University and Research (MIUR) through grant “Dipartimenti di Eccellenza 2017” to University of Milano-Bicocca, Department of Biotechnology and Biosciences is acknowledged. Partial support was also provided by the CRUK/AECC/AIRC Accelerator Award #22790, “Single-cell Cancer Evolution in the Clinic”. J.M. is grateful for the support from the National Council for Scientific and Technological Development (CNPq grant #155957/2018-0) and São Paulo Research Foundation (FAPESP grant #2020/03514-9).

O.M.B. acknowledges support from CNPq (Grant #307897/2018-4) and FAPESP (grant #2014/08026-1 and 2016/18809-9).

This work was also partially supported by a Bicocca 2020 Starting Grant to F.A.

CONFLICT OF INTEREST

The authors declare no conflict of interest, financial or otherwise.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] Kourou, K.; Exarchos, T.P.; Exarchos, K.P.; Karamouzis, M.V.; Fotiadis, D.I. Machine learning applications in cancer prognosis and prediction. *Comput. Struct. Biotechnol. J.*, **2014**, *13*, 8-17. http://dx.doi.org/10.1016/j.csbj.2014.11.005 PMID: 25750696

[2] Furey, T.S.; Cristianini, N.; Duffy, N.; Bednarski, D.W.; Schummer, M.; Haussler, D. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, **2000**, *16*(10), 906-914. http://dx.doi.org/10.1093/bioinformatics/16.10.906 PMID: 11120680

[3] Sotiriou, C.; Neo, S-Y.; McShane, L.M.; Korn, E.L.; Long, P.M.; Jazaeri, A.; Martiat, P.; Fox, S.B.; Harris, A.L.; Liu, E.T. Breast cancer classification and prognosis based on gene expression profiles from a population-based study. *Proc. Natl. Acad. Sci. USA*, **2003**, *100*(18), 10393-10398. http://dx.doi.org/10.1073/pnas.1732912100 PMID: 12917485

[4] Lu, J.; Getz, G.; Miska, E.A.; Alvarez-Saavedra, E.; Lamb, J.; Peck, D.; Sweet-Cordero, A.; Ebert, B.L.; Mak, R.H.; Ferrando, A.A.; Downing, J.R.; Jacks, T.; Horvitz, H.R.; Golub, T.R. MicroRNA expression profiles classify human cancers. *Nature*, **2005**, *435*(7043), 834-838. http://dx.doi.org/10.1038/nature03702 PMID: 15944708

[5] CP de Souto, M.; G Costa, I.; SA de Araujo, D.; B Ludermir, T.; Schliep, A. Clustering cancer gene expression data: a comparative study. *BMC Bioinformatics*, **2008**, *9*(1), 497. http://dx.doi.org/10.1186/1471-2105-9-497

[6] Vanneschi, L.; Farinaccio, A.; Mauri, G.; Antoniotti, M.; Provero, P.; Giacobini, M. A comparison of machine learning techniques for survival prediction in breast cancer. *BioData Min.*, **2011**, *4*(1), 12. http://dx.doi.org/10.1186/1756-0381-4-12 PMID: 21569330

[7] Curtis, C.; Shah, S.P.; Chin, S.F.; Turashvili, G.; Rueda, O.M.; Dunning, M.J.; Speed, D.; Lynch, A.G.; Samarajiwa, S.; Yuan, Y.; Gräf, S.; Ha, G.; Haffari, G.; Bashashati, A.; Russell, R.; McKinney, S.; Langerød, A.; Green, A.; Provenzano, E.; Wishart, G.; Pinder, S.; Watson, P.; Markowetz, F.; Murphy, L.; Ellis, I.; Purushotham, A.; Børresen-Dale, A.L.; Brenton, J.D.; Tavaré, S.; Caldas, C.; Aparicio, S. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*, **2012**, *486*(7403), 346-352. http://dx.doi.org/10.1038/nature10983 PMID: 22522925

[8] Caravagna, G.; Graudenzi, A.; Ramazzotti, D.; Sanz-Pamplona, R.; De Sano, L.; Mauri, G.; Moreno, V.; Antoniotti, M.; Mishra, B. Algorithmic methods to infer the evolutionary trajectories in cancer progression. *Proc. Natl. Acad. Sci. USA*, **2016**, *113*(28), E4025-E4034.

http://dx.doi.org/10.1073/pnas.1520213113 PMID: 27357673

[9] Caravagna, G.; Giarratano, Y.; Ramazzotti, D.; Tomlinson, I.; Graham, T.A.; Sanguinetti, G.; Sottoriva, A. Detecting repeated cancer evolution from multi-region tumor sequencing data. *Nat. Methods*, **2018**, *15*(9), 707-714. http://dx.doi.org/10.1038/s41592-018-0108-x PMID: 30171232

[10] Hofree, M.; Shen, J.P.; Carter, H.; Gross, A.; Ideker, T. Network-based stratification of tumor mutations. *Nat. Methods*, **2013**, *10*(11), 1108-1115. http://dx.doi.org/10.1038/nmeth.2651 PMID: 24037242

[11] Michael, L.G.; Joseph, E.L.; William, T.B.; Jong, W.K; Quanli, W.; Matthew, D.C; Michael, B.D; Michael, K.; Bernard Mathey, P.; Anil, P. A pathway-based classification of human breast cancer. *Proceedings of the National Academy of Sciences*, **2010**, *107*(15), 6994-6999.

[12] Graudenzi, A.; Cava, C.; Bertoli, G.; Fromm, B.; Flatmark, K.; Mauri, G.; Castiglioni, I. Pathway-based classification of breast cancer subtypes. *Front. Biosci.*, **2017**, *22*, 1697-1712. http://dx.doi.org/10.2741/4566 PMID: 28410140

[13] Hanahan, D.; Weinberg, R.A. Hallmarks of cancer: the next generation. *Cell*, **2011**, *144*(5), 646-674. http://dx.doi.org/10.1016/j.cell.2011.02.013 PMID: 21376230

[14] Cantor, J.R.; Sabatini, D.M. Cancer cell metabolism: one hallmark, many faces. *Cancer Discov.*, **2012**, *2*(10), 881-898. http://dx.doi.org/10.1158/2159-8290.CD-12-0345 PMID: 23009760

[15] Ward, P.S.; Thompson, C.B. Metabolic reprogramming: a cancer hallmark even warburg did not anticipate. *Cancer Cell*, **2012**, *21*(3), 297-308. http://dx.doi.org/10.1016/j.ccr.2012.02.014 PMID: 22439925

[16] Tomita, M.; Kami, K. Cancer. Systems biology, metabolomics, and cancer metabolism. *Science*, **2012**, *336*(6084), 990-991. http://dx.doi.org/10.1126/science.1223066 PMID: 22628644

[17] Teicher, B.A.; Linehan, W.M.; Helman, L.J. Targeting cancer metabolism. **2012**, *18*(20), 5537-5545.

[18] Hyduke, D.R.; Lewis, N.E.; Palsson, B.Ø. Analysis of omics data with genome-scale models of metabolism. *Mol. Biosyst.*, **2013**, *9*(2), 167-174. http://dx.doi.org/10.1039/C2MB25453K PMID: 23247105

[19] Lewis, N.E.; Abdel-Haleem, A.M. The evolution of genome-scale models of cancer metabolism. *Front. Physiol.*, **2013**, *4*, 237. http://dx.doi.org/10.3389/fphys.2013.00237 PMID: 24027532

[20] Orth, J.D.; Thiele, I.; Palsson, B.Ø. What is flux balance analysis? *Nat. Biotechnol.*, **2010**, *28*(3), 245-248. http://dx.doi.org/10.1038/nbt.1614 PMID: 20212490

[21] Machado, D.; Herrgård, M. Systematic evaluation of methods for integration of transcriptomic data into constraint-based models of metabolism. *PLOS Comput. Biol.*, **2014**, *10*(4), e1003580. http://dx.doi.org/10.1371/journal.pcbi.1003580 PMID: 24762745

[22] Jamialahmadi, O.; Hashemi-Najafabadi, S.; Motamedian, E.; Romeo, S.; Bagheri, F. A benchmark-driven approach to reconstruct metabolic networks for studying cancer metabolism. *PLOS Comput. Biol.*, **2019**, *15*(4), e1006936. http://dx.doi.org/10.1371/journal.pcbi.1006936 PMID: 31009458

[23] Damiani, C.; Di Filippo, M.; Pescini, D.; Maspero, D.; Colombo, R.; Mauri, G. popFBA: tackling intratumour heterogeneity with Flux Balance Analysis. *Bioinformatics*, **2017**, *33*(14), i311-i318. http://dx.doi.org/10.1093/bioinformatics/btx251 PMID: 28881985

[24] Damiani, C.; Maspero, D.; Di Filippo, M.; Colombo, R.; Pescini, D.; Graudenzi, A.; Westerhoff, H.V.; Alberghina, L.; Vanoni, M.; Mauri, G. Integration of single-cell RNA-seq data into population models to characterize cancer metabolism. *PLOS Comput. Biol.*, **2019**, *15*(2), e1006733. http://dx.doi.org/10.1371/journal.pcbi.1006733 PMID: 30818329

[25] Damiani, C.; Gaglio, D.; Sacco, E.; Alberghina, L.; Vanoni, M. Systems metabolomics: from metabolomic snapshots to design principles. *Curr. Opin. Biotechnol.*, **2020**, *63*, 190-199. http://dx.doi.org/10.1016/j.copbio.2020.02.013 PMID: 32278263

[26] Graudenzi, A.; Maspero, D.; Di Filippo, M.; Gnugnoli, M.; Isella, C.; Mauri, G.; Medico, E.; Antoniotti, M.; Damiani, C. Integration of transcriptomic data and metabolic networks in cancer samples reveals highly significant prognostic power. *J. Biomed. Inform.*, **2018**, *87*, 37-49. http://dx.doi.org/10.1016/j.jbi.2018.09.010 PMID: 30244122

[27] Damiani, C.; Rovida, L.; Maspero, D.; Sala, I.; Rosato, L.; Di Filippo, M.; Pescini, D.; Graudenzi, A.; Antoniotti, M.; Mauri, G. MaREA4Galaxy: Metabolic reaction enrichment analysis and visualization of RNA-seq data within Galaxy. *Comput. Struct. Biotechnol. J.*, **2020**, *18*, 993-999. http://dx.doi.org/10.1016/j.csbj.2020.04.008 PMID: 32373287

[28] Ciriello, G.; Gatza, M.L.; Beck, A.H.; Wilkerson, M.D.; Rhie, S.K.; Pastore, A.; Zhang, H.; McLellan, M.; Yau, C.; Kandoth, C.; Bowlby, R.; Shen, H.; Hayat, S.; Fieldhouse, R.; Lester, S.C.; Tse, G.M.; Factor, R.E.; Collins, L.C.; Allison, K.H.; Chen, Y.Y.; Jensen, K.; Johnson, N.B.; Oesterreich, S.; Mills, G.B.; Cherniack, A.D.; Robertson, G.; Benz, C.; Sander, C.; Laird, P.W.; Hoadley, K.A.; King, T.A.; Perou, C.M. Comprehensive molecular portraits of invasive lobular breast cancer. *Cell*, **2015**, *163*(2), 506-519. http://dx.doi.org/10.1016/j.cell.2015.09.033 PMID: 26451490

[29] Swainston, N.; Smallbone, K.; Hefzi, H.; Dobson, P.D.; Brewer, J.; Hanscho, M.; Zielinski, D.C.; Ang, K.S.; Gardiner, N.J.; Gutierrez, J.M.; Kyriakopoulos, S.; Lakshmanan, M.; Li, S.; Liu, J.K.; Martínez, V.S.; Orellana, C.A.; Quek, L.E.; Thomas, A.; Zanghellini, J.; Borth, N.; Lee, D.Y.; Nielsen, L.K.; Kell, D.B.; Lewis, N.E.; Mendes, P. Recon 2.2: from reconstruction to model of human metabolism. *Metabolomics*, **2016**, *12*(7), 109. http://dx.doi.org/10.1007/s11306-016-1051-4 PMID: 27358602

[30] Cazzaniga, P.; Damiani, C.; Besozzi, D.; Colombo, R.; Nobile, M.S.; Gaglio, D.; Pescini, D.; Molinari, S.; Mauri, G.; Alberghina, L.; Vanoni, M. Computational strategies for a system-level understanding of metabolism. *Metabolites*, **2014**, *4*(4), 1034-1087. http://dx.doi.org/10.3390/metabo4041034 PMID: 25427076

[31] Mardinoglu, A.; Agren, R.; Kampf, C.; Asplund, A.; Uhlen, M.; Nielsen, J. Genome-scale metabolic modelling of hepatocytes reveals serine deficiency in patients with non-alcoholic fatty liver disease. *Nat. Commun.*, **2014**, *5*, 3083. http://dx.doi.org/10.1038/ncomms4083 PMID: 24419221

[32] Thiele, I.; Swainston, N.; Fleming, R.M.; Hoppe, A.; Sahoo, S.; Aurich, M.K.; Haraldsdottir, H.; Mo, M.L.; Rolfsson, O.; Stobbe, M.D.; Thorleifsson, S.G.; Agren, R.; Bölling, C.; Bordel, S.; Chavali, A.K.; Dobson, P.; Dunn, W.B.; Endler, L.; Hala, D.; Hucka, M.; Hull, D.; Jameson, D.; Jamshidi, N.; Jonsson, J.J.; Juty, N.; Keating, S.; Nookaew, I.; Le Novère, N.; Malys, N.; Mazein, A.; Papin, J.A.; Price, N.D.; Selkov, E., Sr; Sigurdsson, M.I.; Simeonidis, E.; Sonnenschein, N.; Smallbone, K.; Sorokin, A.; van Beek, J.H.; Weichart, D.; Goryanin, I.; Nielsen, J.; Westerhoff, H.V.; Kell, D.B.; Mendes, P.; Palsson, B.Ø. A community-driven global reconstruction of human metabolism. *Nat. Biotechnol.*, **2013**, *31*(5), 419-425. http://dx.doi.org/10.1038/nbt.2488 PMID: 23455439

[33] Ma, H-W.; Zeng, A-P. The connectivity structure, giant strong component and centrality of metabolic networks. *Bioinformatics*, **2003**, *19*(11), 1423-1430. http://dx.doi.org/10.1093/bioinformatics/btg177 PMID: 12874056

[34] Backes, A.R.; Casanova, D.; Bruno, O.M. A complex network-based approach for boundary shape analysis. *Pattern Recognit.*, **2009**, *42*(1), 54-67. http://dx.doi.org/10.1016/j.patcog.2008.07.006

[35] Miranda, G.H.B.; Machicao, J.; Bruno, O.M. An optimized shape descriptor based on structural properties of networks. *Digit. Signal Process.*, **2018**, *82*, 216-229. http://dx.doi.org/10.1016/j.dsp.2018.06.010

[36] Machicao, J.; Filho, H.A.; Lahr, D.J.G.; Buckeridge, M.; Bruno, O.M. Topological assessment of metabolic networks reveals evolutionary information. *Sci. Rep.*, **2018**, *8*(1), 15918. http://dx.doi.org/10.1038/s41598-018-34163-7 PMID: 30374088

[37] Shannon, P.; Markiel, A.; Ozier, O.; Baliga, N.S.; Wang, J.T.; Ramage, D.; Amin, N.; Schwikowski, B.; Ideker, T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, **2003**, *13*(11), 2498-2504. http://dx.doi.org/10.1101/gr.1239303 PMID: 14597658

[38] Costa, L.D.F.; Boas, P.R.V.; Silva, F.N.; Rodrigues, F.A. A pattern recognition approach to complex networks. *J. Stat. Mech.*, **2010**, *2010*(11), P11015. http://dx.doi.org/10.1088/1742-5468/2010/11/P11015

[39] Banerjee, A.; Jost, J. Spectral plot properties: Towards a qualitative classification of networks. *NHM*, **2008**, *3*(2), 395-411. http://dx.doi.org/10.3934/nhm.2008.3.395

[40] Costa, L da F.; Francisco, A.; Rodrigues, G.T.; Villas Boas, P.R. Characterization of complex networks: A survey of measurements. *Adv. Phys.*, **2007**, *56*(1), 167-242. http://dx.doi.org/10.1080/00018730601170527

[41] Newman, M.E. Assortative mixing in networks. *Phys. Rev. Lett.*, **2002**, *89*(20), 208701. http://dx.doi.org/10.1103/PhysRevLett.89.208701 PMID: 12443515

[42] Filisetti, A.; Graudenzi, A.; Serra, R.; Villani, M.; De Lucrezia, D.; Rudolf, M. Füchslin, Stuart A Kauffman, Norman Packard, and Irene Poli. A stochastic model of the emergence of autocatalytic cycles. *J. Syst. Chem.*, **2011**, *2*(1), 2. http://dx.doi.org/10.1186/1759-2208-2-2

[43] Filisetti, A.; Graudenzi, A.; Serra, R.; Villani, M.; Füchslin, R.M.; Packard, N.; Kauffman, S.A.; Poli, I. A stochastic model of auto-catalytic reaction networks. *Theory Biosci.*, **2012**, *131*(2), 85-93. http://dx.doi.org/10.1007/s12064-011-0136-x PMID: 21979857

[44] Serra, R.; Filisetti, A.; Villani, M.; Graudenzi, A.; Damiani, C.; Panini, T. A stochastic model of catalytic reaction networks in protocells. *Nat. Comput.*, **2014**, *13*(3), 367-377. http://dx.doi.org/10.1007/s11047-014-9445-6

[45] Cawley, G.C.; Talbot, N.L.C. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.*, **2010**, *11*, 2079-2107.

[46] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, **2011**, *12*, 2825-2830.

[47] Cerami, E.; Gao, J.; Dogrusoz, U.; Gross, B.E.; Sumer, S.O.; Aksoy, B.A.; Jacobsen, A.; Byrne, C.J.; Heuer, M.L.; Larsson, E.; Antipin, Y.; Reva, B.; Goldberg, A.P.; Sander, C.; Schultz, N. The cBio cancer genomics portal: an open platform for exploring multi-dimensional cancer genomics data. *Cancer Discov.*, **2012**, *2*(5), 401-404.

[48] Hodges, J.L. The significance probability of the smirnov two-sample test. *Ark. Mat.*, **1958**, *3*, 469-486. http://dx.doi.org/10.1007/BF02589501

[49] Pacheco, M.P.; Bintener, T.; Sauter, T. Towards the network-based prediction of repurposed drugs using patient-specific metabolic models. *EBioMedicine*, **2019**, *43*, 26-27. http://dx.doi.org/10.1016/j.ebiom.2019.04.017 PMID: 30979684

[50] Zampieri, G.; Vijayakumar, S.; Yaneske, E.; Angione, C. Machine and deep learning meet genome-scale metabolic modeling. *PLOS Comput. Biol.*, **2019**, *15*(7), e1007084. http://dx.doi.org/10.1371/journal.pcbi.1007084 PMID: 31295267

[51] Cai, H.Y.; Zheng, V.W.; Chang, K.C.-C. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, **2018**, *30*(9), 1616-1637. http://dx.doi.org/10.1109/TKDE.2018.2807452

[52] Hamilton, W.L.; Ying, R.; Leskovec, J. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, **2017**, *40*(3), 52-74.

[53] Kriege, N.M.; Johansson, F.D.; Morris, C. A survey on graph kernels. *Appl. Network Sci.*, **2020**, *5*(1), 6. http://dx.doi.org/10.1007/s41109-019-0195-3

[54] Niepert, M.; Ahmed, M.; Kutzkov, K. Learning convolutional neural networks for graphs. *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016*, New York City, NY, USA. June 19-24, **2016**, *Volume 48*, pp. 2014-2023.

A.3 Deep Learning for Predicting Relative Fluxes in Reaction Systems

Contribution. In this chapter I will discuss the work done in:

[Pat+21] L. Patruno, F. Craighero, D. Maspero, A. Graudenzi, C. Damiani. "Combining Multi-Target Regression Deep Neural Networks and Kinetic Modeling to Predict Relative Fluxes in Reaction Systems". In: *Information and Computation* 281 (Dec. 2021)

Summary. A reaction system, such as metabolic networks [Jam+19], is characterized by reactions (fluxes) that transform one chemical into another. To understand the metabolomics of the systems, an important goal is to predict the variation of fluxes across steady-states. Unfortunately, this task is still very challenging given the hardness of measuring flux variations, even with the current technologies. One viable solution is to predict flux variations from relative chemical abundances, that can be tackled though constrained optimization [Saj+16]. However, this approach requires a lot of assumptions and simplifications. In this work, we propose a different method employing a deep model to predict flux variation from relative abundances. Our contributions can be summariezed as follows:

- *Dataset generation.* Given the hardness of measuring flux variations, we simulated them using kineting modeling [Dam+17].
- *Definition of a multi-target DNN.* We defined a multi-target DNN to predict the network's fluxes variations from chemical abundances and defined a cross-validated grid search for hyperparameter selection.
- *Evaluation under feature reduction.* To assess whether flux variations are affected by chemicals not directly involved into the reaction, we also investigated the effect of reducing the chemicals given in input to the model.

We evaluated the proposed model on a simulated dataset of a yeast metabolic network [Dam+17]. Results confirmed the validity of our approach, even with feature reduction, that is able to predict fluxes variations from relative abudances with no constraints a priori. Future work will investigate more complex simulations, for example by including also transcriptomic data.

Implementation. The experiments performed in the paper has been open-sourced on a Github repository[a].

[a] `https://github.com/BIMIB-DISCo/FLUX-PREDICT`

Information and Computation 281 (2021) 104798

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Combining multi-target regression deep neural networks and kinetic modeling to predict relative fluxes in reaction systems

Lucrezia Patruno [a,1], Francesco Craighero [a,1], Davide Maspero [a,b], Alex Graudenzi [b,c], Chiara Damiani [d,e,*]

[a] Department of Informatics, Systems and Communication, University of Milan-Bicocca, Milan, Italy
[b] Institute of Molecular Bioimaging and Physiology, Consiglio Nazionale delle Ricerche (IBFM-CNR), Segrate, Milan, Italy
[c] Bicocca Bioinformatics Biostatistics and Bioimaging Centre – B4, Milan, Italy
[d] Department of Biotechnology and Biosciences, University of Milano-Bicocca, Milan, Italy
[e] SYSBIO/ISBE.IT Centre for Systems Biology, Milan, Italy

A R T I C L E I N F O

Article history:
Received 13 October 2020
Received in revised form 20 May 2021
Accepted 29 August 2021
Available online 6 September 2021

Keywords:
ODE-based modelling
Monte Carlo sampling
Deep neural networks
Metabolomics
Metabolic fluxes

A B S T R A C T

The strong nonlinearity of large and highly connected reaction systems, such as metabolic networks, hampers the determination of variations in reaction fluxes from variations in species abundances, when comparing different steady states of a given system. We hypothesize that patterns in species abundance variations exist that mainly depend on the kernel of the stoichiometric matrix and allow for predictions of flux variations. To investigate this hypothesis, we applied a multi-target regression Deep Neural Network (DNN) to data generated via numerical simulations of an Ordinary Differential Equation (ODE) model of yeast metabolism, upon Monte Carlo sampling of the kinetic parameters. For each parameter configuration, we compared two steady states corresponding to different environmental conditions. We show that DNNs can predict relative fluxes impressively well even when a random subspace of input features is supplied, supporting the existence of recurrent variation patterns in abundances of chemical species, which can be recognized automatically.

1. Introduction

The determination of the rate at which a substance is transformed into another through a given reaction or pathway (i.e. the flux) on the basis of routine measurements of the abundance of chemical species, when the mechanistic dynamics of the systems is not fully characterized, is an important problem in different fields, from life [1] to environmental sciences [2]. Knowledge on relative fluxes is important, as it may translate into knowledge about the controllable mechanisms underlying the differences between two steady states of a system (e.g. pathological vs healthy state). This translation occurs more directly and successfully than in the case of abundances of chemicals, which provide a mere snapshot of the system [1]. Yet, fluxes are hardly measurable with current technologies, whereas abundances can be largely measured with high throughput methods.

* Corresponding author at: Department of Biotechnology and Biosciences, University of Milano-Bicocca, Milan, Italy.
E-mail address: chiara.damiani@unimib.it (C. Damiani).

1 Equal contribution.

https://doi.org/10.1016/j.ic.2021.104798

We investigate the problem of estimating relative fluxes from relative abundances, by means of both theoretical reasoning and simulations. We show that the problem can be solved analytically only if enzymatic kinetics are neglected. When enzymatic kinetics are taken into account, extra information is required, namely relative abundances of enzyme-substrate complexes when mass action rate law formulation is used, kinetic constants (i.e. the binding affinity of enzymes) when the Michaelis-Menten approximation is used. Both types of information are currently not measurable on a large scale. Moreover, analytical solutions require knowledge on the abundance of each single substrate involved in a reaction, whereas the current sensitivity of abundance quantification techniques (as e.g., mass spectrometry) typically allows detecting only a subset of them at a time.

To overcome this lack of knowledge, current approaches mainly rely on optimization subject to constraints to identify feasible solutions out of a very large set of candidates. Along with stoichiometric constraints and mass balance, such approaches incorporate constraints on relative abundances of metabolites in the form of constraints on relative fluxes. For example, iReMet-flux [3] seeks to minimize the distance between any pair of flux distributions in the feasible region, whose ratio between each flux is within upper and lower bounds, derived from the ratios of metabolic and enzyme abundances, according to the mass action formulation (see Section 2). Pandey et al. [4], instead, convert the variation in the abundance of a given metabolite into a constraint on the generic variation in the fluxes responsible of either its production or consumption and seek to maximize the consistency with such constraints, along with other constraints on relative fluxes assumed from relative gene expression data.

By requiring relative metabolic abundances to be incorporated in the form of constraints on relative fluxes, the above approaches require many assumptions and simplifications. Another limitation of these approaches is that they require the definition of an objective function. Moreover, it is difficult to find the optimal trade-off between narrow constraints leading to infeasible solutions and loose constraints leading to too wide feasible regions.

Here, we propose a different approach based on the combination of kinetic modeling and Machine Learning (ML). The combination of computational modeling, and in particular constraint-based modeling, with machine learning techniques is an emerging field which is revealing great potential. Recent approaches exploit the mechanistically linked information provided by context-specific models as the input of either supervised or unsupervised machine learning approaches, as reviewed in [5–7]. Although some of these studies have used neural network to predict e.g., individual fluxes from enzyme or gene expression data [8] or abundances of metabolites from other -omics data [9,10], to our knowledge, this is the first time that ML is used to predict overall flux variations from overall relative abundances.

The approach that we are proposing originates from the hypothesis that recurrent patterns resulting from stoichiometric and mass balance constraints exist. Hence, we can exploit information of the vector of abundance variations $\boldsymbol{\delta x}$ or, possibly, of a subspace of it, in order to predict with a good confidence level the vector of flux variations $\boldsymbol{\delta v}$. We expect these patterns to be learned and recognized by ML regression algorithms.

However, given that fluxes are hardly measurable in real-world scenarios, it is unrealistic to obtain a large and heterogeneous experimental training set of $(\boldsymbol{\delta x}, \boldsymbol{\delta v})$ pairs to properly train any ML algorithm. To overcome this limitation, we propose to simulate $(\boldsymbol{\delta x}, \boldsymbol{\delta v})$ pairs with kinetic modeling, namely via standard ODEs. Notice that reaction rate equations and constants are largely undetermined, otherwise it would suffice to directly simulate $\boldsymbol{\delta x}$ with a ODE model to predict $\boldsymbol{\delta v}$. Here, we assume that, in light of the steady state constraint, $f(\boldsymbol{\delta x}) = \boldsymbol{\delta v}$ is largely independent from the specific values of kinetic constants.

To investigate the validity of our assumption, we propose to randomly sample the space of kinetic parameters, as in [11–13]. For each sampled set of parameters, $\boldsymbol{\delta x}$ and $\boldsymbol{\delta v}$ can be collected, by comparing the state of the ODE model in two different environmental conditions in a time invariant condition (i.e., the steady state). In a preliminary phase, we employed the simulated dynamics of a previously published yeast metabolic network [12,11], undergoing nutritional perturbations, to train, validate and test different configurations of Deep Neural Networks (DNNs). We also evaluated the predictive performance of DNNs in the realistic scenario in which the abundance of metabolites can be measured for a limited subset of the model species.

2. Motivation and main assumptions

A biochemical reaction system is defined by a set $\mathcal{X} = \{X_1, \ldots, X_M\}$ of molecular species occurring in the system, and a set $\mathcal{R} = \{R_1, \ldots, R_N\}$ of chemical reactions taking place among the species. We define reactions as: $R_r : \sum_{q \in Q_r} \alpha_q X_q \Rightarrow \sum_{t \in T_r} \beta_t X_t$ where $\alpha_q, \beta_t \in \mathbb{Q}^+$ are stoichiometric coefficients associated, respectively, with the q-th reactant and the t-th product of the r-th reaction, and Q_r and T_r are the set of reaction substrates and products of reaction r, respectively. Let $[X_m](t)$, with $m = 1, \ldots, M$ be the abundance of X_m at a given time t in the system's evolution, either expressed as number of molecules or as concentration. Let V_r, with $r = 1, \ldots, N$ be the rate (or flux) through reaction R_r in a unit of time, i.e. the number of times R_r occurs in that unit of time. Such a system is said to be at steady state if $\partial[X_m](t)/\partial t = 0, \forall m$. Steady state is thus the condition in which fluxes may occur but the concentration of all species does not change in time. Let S be a $M \times N$ matrix, referred to as stoichiometric matrix, whose element $s_{m,r}$, takes value $-\alpha_{m,r}$ if species X_m is a substrate of reaction R_r (i.e., $X_m \in Q_r$), $\beta_{m,r}$ if species X_m is a product of reaction R_r (i.e., $X_m \in T_r$), 0 otherwise. Let $\boldsymbol{v} = (V_1, \ldots, V_N)$ be the vector of reaction fluxes, then a system is at steady state when $S\boldsymbol{v} = 0$.

It is worth mentioning that, if a reaction R_r is *reversible*, a reaction R_b exists such that $s_{m,r} = -s_{m,b}, \forall m$. Typically, life scientists use the term flux to indicate the net rate $V_r - V_b$, that is, the rate of the forward reaction minus the rate of the reverse reaction. However, in this work reversible reactions are represented with two distinct and complementary forward reactions, thus the terms flux and rate coincide.

Now let i and j be two different steady states of the system and $\boldsymbol{x^i} = ([X_1]^i, \ldots, [X_M]^i)$ be the vector of abundances of the chemical species in steady state i and $\boldsymbol{v^i} = ({V_1}^i, \ldots, {V_N}^i)$ be the vector of reaction fluxes in steady state j, and let $\boldsymbol{x^j} = ([X_1]^j, \ldots, [X_M]^j)$ be the vector of abundances of the chemical species in steady state j and $\boldsymbol{v^j} = ({V_1}^i, \ldots, {V_N}^j)$ be the vector of reaction fluxes in steady state j. We define the species abundance variation between state i and j as $\boldsymbol{\delta x^{i,j}} \equiv (\delta[X_1]^{i,j}, \ldots, \delta[X_M]^{i,j}) = \boldsymbol{x^j} - \boldsymbol{x^i}$, and the variation of reaction fluxes as $\boldsymbol{\delta v^{i,j}} \equiv (\delta V_1^{i,j}, \ldots, \delta V_N^{i,j}) = \boldsymbol{v^j} - \boldsymbol{v^i}$. In the following, $\boldsymbol{\delta x^{i,j}}$ and $\boldsymbol{\delta v^{i,j}}$ are also referred to as relative abundances and relative fluxes, respectively.

The aim of this work is to deduce flux variations from species abundance variations, that is, $\boldsymbol{\delta v^{i,j}}$ from $\boldsymbol{\delta x^{i,j}}$. In the following, we will make use of a very simple and specific example of reaction system to motivate, without loss of generality, the complexity of the problem and the non linearities that one may encounter when trying to deduce flux variations from species abundance variations.

Example 1. Let us assume a very simple system composed of 4 chemical species $\mathcal{X} = \{A, B, C, D\}$ (e.g., metabolites) and 3 reactions $\mathcal{R} = \{R_1, R_2, R_3\}$ defined as follows:

$$R_1: \ A \Rightarrow B$$

$$R_2: \ C \Rightarrow B$$

$$R_3: \ B \Rightarrow D$$

In order for the system above to be able to reach a steady state, unbalanced reactions (also referred to as exchange reactions) must be included, for A and C – which must be fed into the system ($\emptyset \Rightarrow A$; $\emptyset \Rightarrow C$) – and for D – which must be depleted ($D \Rightarrow \emptyset$).

At the steady state, the rate of production and consumption of the species must balance. Hence, if any event (e.g., an external perturbation of the system) determines the increase of either V_1 and/or V_2, then V_3 must eventually increase to reach a new steady state. Consequently, when comparing two steady states of the system, if $\delta V_1 + \delta V_2 > 0$ then $\delta V_3 > 0$.

Let us now suppose that information on $\boldsymbol{\delta x^{i,j}}$ is given and, for instance, that an increase in $[B]$ ($\delta[B]^{i,j} > 0$) and an increase in $[D]$ ($\delta[D]^{i,j} > 0$) were observed. This must be imputed to one of the following cases:

1. an increase in V_3 and an increase in V_1,
2. an increase in V_3 and an increase in V_2,
3. an increase in V_3 and an increase in both V_1 and V_2.

Information on $\delta[A]$ and $\delta[C]$ does not let us to exclude case 1 or case 2. In fact, case 2 is compatible with both: (i) $\delta[C] > 0$, i.e., an increase in the reaction's substrate $[C]$ and (ii) $\delta[C] = 0$, if the higher depletion of C, resulting from an increase in $\delta V_2 > 0$, is compensated by a higher influx of C. Along similar lines, case 1 is compatible with both: $\delta[A] > 0$ and $\delta[A] = 0$.

Example 1 demonstrates the complexity of the problem of determining analytically flux variations from relative abundances. The complexity is expected to increase with the number of interconnected reactions and when reactions of higher order and/or feedback loops come into play, as it is typically observed in real-world scenarios.

However, the following assumptions would allow one to analytically estimate relative fluxes from relative abundances:

- for each reaction r in the system, the mass action law is assumed: $V_r = k_r * \prod_{q \in Q_r} [X_q]^{|S_{q,r}|}$, where k_r is the kinetic constant of reaction r, X_q is the q^{th} substrate of the set Q_r of substrates of reaction r, and $S_{q,r}$ is the stoichiometric coefficient of substrate X_q i.e., how many molecules of the substrate partake to the reaction;
- at (steady) states i and j, the kinetic constant k_r of any reaction r of the system is assumed to be identical.

Given such assumptions, the variation between the flux of an irreversible reaction r in two steady states i and j can be analytically determined as the ratio V_r^i / V_r^j:

$$\frac{V_r^i}{V_r^j} = \prod_{q \in Q_r} \left(\frac{[X_q]^i}{[X_q]^j} \right)^{|S_{q,r}|} \tag{1}$$

which does not depend on k_r.

The above situation completely neglects enzymatic kinetics, which are an important factor in the dynamics of chemical systems. Let us suppose, for instance, that reaction R_1 in the previous example is catalyzed by enzyme E^1. Hence, the series of steps through which reactants bind to specific enzymes before being transformed into products should be taken into account, as follows:

$R_1 : A + E^1 \Leftrightarrow AE^1 \Rightarrow B + E^1$

In principle, one can apply equation (1) to the last reaction step, but knowledge of the relative abundance of intermediate complexes (δAE^1) is required. Yet, owing such a level of detail of information is unrealistic with current technologies.

When dealing with cellular metabolic reactions, it is reasonable to assume that they are far from thermodynamic equilibrium and that substrates are in excess over enzyme-substrate complexes. Hence, the enzyme kinetics is generally approximated with Michaelis-Menten rate laws [14,15]. The Michaelis-Menten formulation does not explicitly take into account the abundance of enzymes, but it models saturation kinetics, by describing the variation of the rate of a reaction as a function of substrate's abundance. In the simplest scenario, the rate of a reaction R_r involving a single substrate X_r with unitary stoichiometric coefficient would be described as:

$$V_r = \frac{V_r^{MAX}[X_r]}{K_r^M + [X_r]} \tag{2}$$

where, briefly, V_r^{MAX} is the maximum rate of reaction r (when the enzyme is saturated with substrate), whereas K_r^M is the concentration of substrate that permits the enzyme to achieve half V_r^{MAX}, which depends inversely on the affinity of the enzyme for its substrate. In this scenario, the ratio V_r^i/V_r^j describing the variation between the flux of a irreversible reaction r in two steady states i and j is defined as follows:

$$\frac{V_r^i}{V_r^j} = \frac{[X_r]^i(K_r^M + [X_r]^j)}{[X_r]^j(K_r^M + [X_r]^i)} \tag{3}$$

which does depend on K_r^M. Given the incomplete knowledge of the value of K_r^M of metabolic reactions, $\boldsymbol{\delta v^{i,j}}$ cannot be analytically derived from $\boldsymbol{\delta x^{i,j}}$ in this scenario.

Moreover, both equations (1) and (3) require information on the variation of the abundance of each substrate partaking in reaction R_r. However, in real-life scenarios only a fraction of the species involved in a reaction system is detected by chemical quantification technologies. Hence, we here investigate the possibility of using information about variations in other species in the network to improve predictions of δV_r when information on the abundance of substrates of R_r is lacking.

Our hypothesis originates from the consideration that all the steady states of a biochemical reaction system abide by the constraint $S\boldsymbol{v} = 0$ and, therefore, relationships among $\boldsymbol{v}^i$ and $\boldsymbol{v}^j$ exist which are independent from specific rate laws and kinetic constants of reactions. Hence, we speculate that similar relationships between $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ may also exist, which do not depend on kinetic constants values. In this work, we investigate such hypothesis by means of simulation experiments and machine learning.

The general idea of the proposed approach is depicted in Fig. 1.

3. Methods

3.1. Synthetic dataset

To preliminarily investigate our hypothesis, we used a dataset previously generated via numerical simulations of an ODE model [11], in which elementary mass action law was assumed for every reaction rate. The model is defined by a set of $N = 48$ reactions and a set of $M = 34$ metabolites. The metabolic network model is available in SBML format at this link: github.com/BIMIB-DISCo/FLUX-PREDICT, and a graphical representation of it is shown in Fig. 1. For the sake of notation simplicity, in the following, we will refer to the name of specific reactions with the name of the first substrate and the name of the main product separated by the underscore symbol. For instance the reaction in the top left corner of the map ($Glc + ATP \Rightarrow G6P + ADP$) will be referred to as Glc_G6P. Reverse reactions are considered separately and are indicated with the suffix $\_reverse$. $P = 100\,000$ sets of kinetic constants $\mathcal{K}_1 = \{k_1, k_2, \ldots, k_N\}$, $\mathcal{K}_2 = \{k_1, k_2, \ldots, k_N\}$, ..., $\mathcal{K}_P = \{k_1, k_2, \ldots, k_N\}$ for the model reaction rates were generated randomly from a uniform distribution in [0,1). Initial abundances of metabolites were defined according to data in literature and are reported in [11]. For each parameter set $\mathcal{K}_p$, with $p = 1, \ldots, P$, we retrieved two steady states of the model corresponding to two different nutritional conditions: condition i – low glucose (2.8 mM), condition j – high glucose (25 mM). Glucose concentration is maintained fixed during the simulation. The model was simulated via integration of ODEs by means of the LSODA solver [16] for a simulated time of 50 seconds. The quasi-steady state condition was determined according to a small threshold (0.01) on the average standard deviation (σ) of the value of species concentration for the last 10% of time dynamics. For further details on the simulations the reader is referred to [12]. For each parameter set $\mathcal{K}_p$, with $p = 1, \ldots, P$, we obtained the vector of abundances at steady state of the chemical species in condition i $\boldsymbol{x_p^i} = ([X_1]_p^i, [X_2]_p^i, \cdots, [X_M]_p^i)$ and in condition j $\boldsymbol{x_p^j} = ([X_1]_p^j, [X_2]_p^j, \cdots, [X_M]_p^j)$ and the vectors of fluxes $\boldsymbol{v_p^i} = (V_{1,p}^i, V_{2,p}^i, \cdots, V_{N,p}^i)$ and $\boldsymbol{v_p^j} = (V_{1,p}^j, V_{2,p}^j, \cdots, V_{N,p}^j)$ and we computed the vector of variations of abundances $\boldsymbol{\delta x_p}^{\boldsymbol{i,j}} = \boldsymbol{x_p^j} - \boldsymbol{x_p^i}$ and of fluxes $\boldsymbol{\delta v_p^{i,j}} = \boldsymbol{v_p^j} - \boldsymbol{v_p^i}$ between conditions i and j. We decided to compute the difference rather than the ratio between conditions to avoid problems related to divisions by zero. We finally obtained 100 000 pairs of metabolites-flux variations ($\boldsymbol{\delta x_p}^{\boldsymbol{i,j}}, \boldsymbol{\delta v_p}^{\boldsymbol{i,j}}$), with $p = 1, 2, \cdots, 100\,000$.

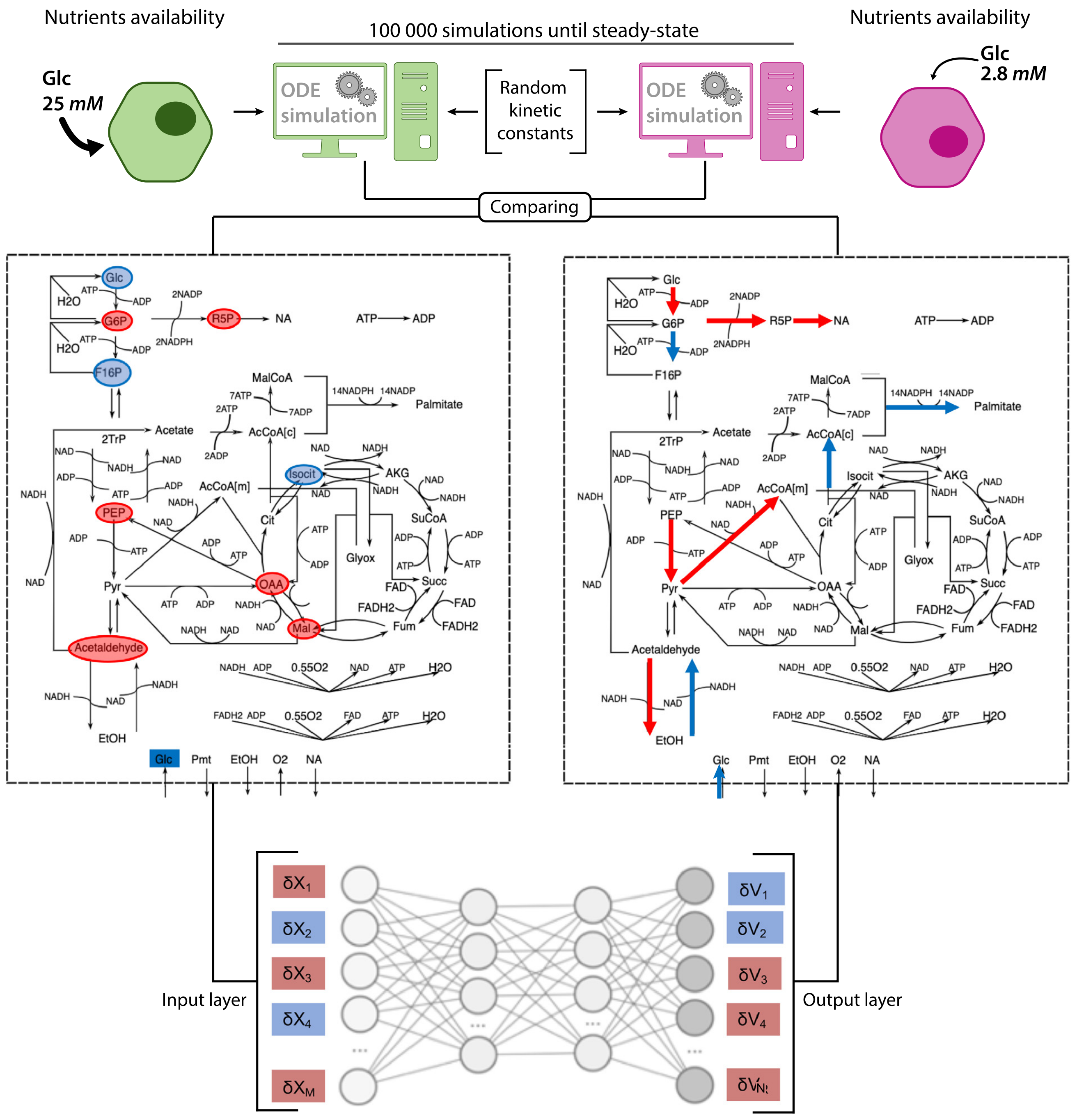


Fig. 1. Schematic representation of the proposed approach and diagram (adapted from [11]) of the yeast metabolic network used to generate the synthetic dataset. Blue/red represent positive/negative variations in species abundances and fluxes. The DNN diagram is for representative purposes only. In our setting, the input layer has less nodes ($M = 34$) that the output layer ($N = 48$). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

For the sake of simplicity, in the following we will refer to $\boldsymbol{\delta x} = (\delta x_1^{i,j}, \ldots, \delta x_p^{i,j})$ as the matrix of size $P \times M$, where $\boldsymbol{\delta x_p}$ is the vector of variation of abundances for constants $\mathcal{K}_p$ and $\boldsymbol{\delta x_{*,m}}$ is the vector of variation of abundances of metabolite m for all the P parameters. Similarly, we will refer to $\boldsymbol{\delta v} = (\delta v_1^{i,j}, \ldots, \delta v_p^{i,j})$ as the matrix of size $P \times N$, where $\boldsymbol{\delta v_p}$ is the vector of variation of fluxes for constants $\mathcal{K}_p$ and $\boldsymbol{\delta v_{*,r}}$ is the vector of variation of flux r for all the P parameters. In addition, we will refer to a specific metabolite variation $\boldsymbol{\delta x_{*,m}}$ and a specific flux variation $\boldsymbol{\delta v_{*,r}}$ with the name of the corresponding metabolite and reaction.

Data pre-processing Prior to training the neural network, the input dataset was pre-processed. We removed zero variance predictors [17] from the relative metabolites $\boldsymbol{\delta x}$ and we removed zero variance output features from the relative fluxes $\boldsymbol{\delta v}$. In detail, we removed 1 out of 34 relative metabolites (namely, $O2$) and 2 out of 48 relative fluxes (namely the exchange reactions of Glc and $O2$). Finally, in order to proceed with the training phase of our DNN model, we standardized the input variables (*i.e.* metabolite variations), by removing the mean and scaling to unit variance.

3.2. Model definition

In order to define a model that predicts the vectors of flux variations $\boldsymbol{\delta v_p}$ from the vectors of abundance variations $\boldsymbol{\delta x_p}$, we built a multi-target regression DNN. Such model is depicted in Fig. 1: its input layer contains as many nodes as the number of abundance variations (i.e., M), and the output layer is composed of a number of nodes equal to the number of flux variations that need to be predicted (i.e., N). As commonly done, networks are trained in order to minimize the Mean Squared Error (MSE) between true and predicted flux variations.

We considered a multi-target regression DNN rather than single-target to reduce computational costs and hopefully to exploit relationships among the output variables to improve the goodness of fit [18].

3.3. Cross-validated grid search

The selection of the hyperparameters that define the DNN was performed by a cross-validated grid search over a set $\mathcal{H}$ of 48 possible hyperparameters configurations. More in detail, for each configuration $h \in \mathcal{H}$ we estimate the performance on unseen data by cross-validation and then define our chosen model with the best performing h.

Train and test sets We split our dataset in training (*outer training set*) and *test set* with a percentage of 90% – 10%. The former partition is used to fit the neural network and to perform hyperparameter selection, while the latter partition is used to provide an unbiased evaluation of the prediction error, as it is used neither during the training phase, nor for hyperparameter optimization.

Hyperparameters The main aim of our grid search is to explore whether there is a need for deep networks or if wide networks with one single hidden layer may suffice, and to exclude configurations with low predictive power. To this aim, we varied the following hyperparameters:

- Hidden layers sizes, to take into account different widths (number of neurons for each layer) and depths (number of layers). In detail, we considered the following settings:

$$\{(100), (200), (500), (100, 100), (200, 200), (100, 100, 100)\},$$

  with each tuple indicating the number of neurons for each hidden layer.
- Optimization algorithms: $\{Adam, SGD\}$.
- Initial learning rate: $\{0.01, 0.001\}$.
- With and without the dropout regularization technique, that is commonly used to improve generalization. When used, we set the dropping rate to 50%.

In addition to varying the hyperparameters just listed, for each neural network configuration h we kept the following elements fixed: (i) batch normalization method to normalize the input of each activation function, with the aim of improving the stability of the training process. (ii) *ReLU* activation function. (iii) Early stopping heuristic to halt training if the model doesn't improve in 200 epochs, in order to prevent overfitting. (iv) Exponential decay schedule for the learning rate. (v) Batch size of 128. (vi) Mean Squared Error (MSE) as loss function. For a detailed explanation of all the techniques please refer to [19].

Cross validation The grid search procedure was combined with a 5-fold cross validation procedure (*i.e.* cross-validated grid search), see Fig. 2 for a schematized representation. In detail, the *outer training set* was split into 5 groups of equal size, the so-called *folds* (Step 1 in Fig. 2). Then, each neural network configuration $h \in \mathcal{H}$ was trained using 4 folds for the training process (*inner training set*) and the last one for performance evaluation (*valid set*). This procedure was repeated 5 times (CV Loop), so that each fold is employed once as validation set.

In order to have an unbiased estimation of when performing early stopping (i.e. without taking into account the *valid set*), for each iteration the *inner training set* was partitioned in two sets with a percentage of 90%-10% (Step 2*b*), using the former for training and the latter for early-stopping.

Then, when the training phase was concluded, the performance of the network was tested over the *valid set* (Step 2c). In detail, for our experiments we calculated the coefficient of determination R^2 for each flux r:

$$R^2(\delta v_{*,r}, \hat{\delta} v_{*,r}) = 1 - \frac{\sum_{f=1}^{F} (\delta v_{f,r} - \hat{\delta} v_{f,r})^2}{\sum_{f=1}^{F} (\delta v_{f,r} - \bar{\delta} v_{*,r})^2} \tag{4}$$

where F is the number of samples, $\delta v_{*,r}$ is the vector of variations for flux r, $\hat{\delta} v_{*,r}$ is the corresponding vector of predicted values, $\delta v_{f,r}$ and $\hat{\delta} v_{f,r}$ are the values for sample f and $\bar{\delta} v_{*,r}$ is the mean variation for flux r. R^2 measures the goodness

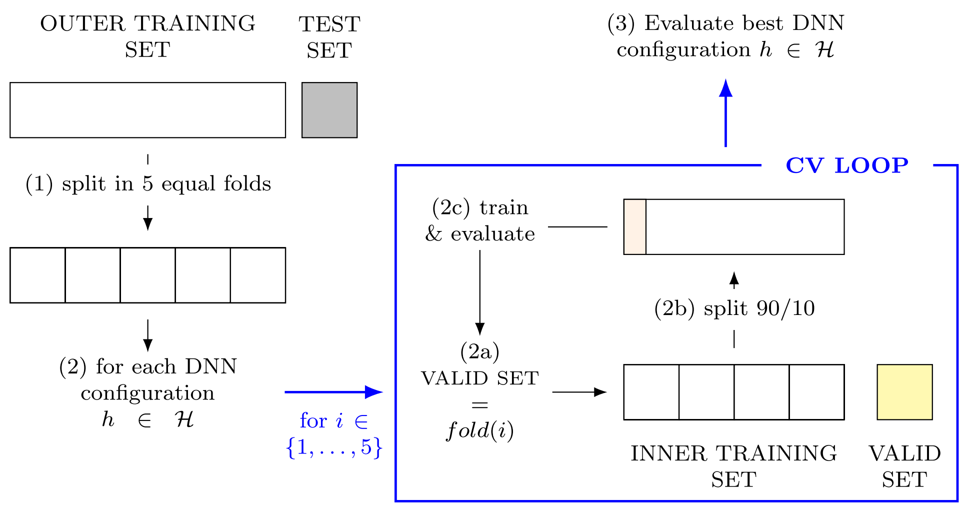

OUTER TRAINING SET
TEST SET
(1) split in 5 equal folds
(2) for each DNN configuration $h \in \mathcal{H}$
for $i \in \{1,\dots,5\}$
(3) Evaluate best DNN configuration $h \in \mathcal{H}$
CV LOOP
(2c) train & evaluate
(2b) split 90/10
(2a) VALID SET = $fold(i)$
INNER TRAINING SET
VALID SET

Fig. 2. Diagram of the 5 fold cross-validation procedure for hyperparameter selection with DNNs, where we take into account also an additional split for the early-stopping heuristic (step 2b).

of fit of the model by normalizing the sum of squared errors by the total deviance observed in ground truth data. After calculating R^2 for each flux r, we calculated the average R^2.

The final result of this cross-validated grid search consisted in 5 performance scores (average R^2) for each DNN configuration. Finally the model $h \in \mathcal{H}$ showing the best mean score was selected and evaluated on the *test set* (Step 3).

For details about the selected models see the Results section. See also Supplementary algorithm 1, for the pseudo-code of the Cross Validated Grid Search.

4. Results

4.1. Selected features

The main goal of this work is to investigate whether the prediction of the flux variation in a given reaction can benefit from information in the abundance variation of metabolites not directly involved in such reaction. To address this issue, we assessed the effect of reducing the number of input metabolites on the output prediction. To select a fraction g of the original features $\boldsymbol{\delta x_p}$ to be removed, with $g = 70\%$ and $g = 50\%$, at first instance, we followed the common practice of basing the choice on their redundancy. Firstly, we computed the pairwise Pearson correlation between metabolites. The heatmap in Fig. 3 shows the correlation of each pair. As already pointed out in [11], it can be observed that correlations between metabolites are not obvious. For example H_2O correlates strongly with $NADP$ even though they are not directly involved in the same reaction. Next, we ranked the pairs of metabolites by decreasing order of their absolute correlation. Starting from the most correlated one, we removed one feature from each pair until only a fraction g of the original metabolites was left.

4.2. Selected hyperparameters

We applied the overall methodology to train, cross-validate and test the best model, given simulated $(\boldsymbol{\delta x_p}, \boldsymbol{\delta v_p})$ pairs, with $p = 1, 2, \cdots, 100\,000$ (as illustrated in Section 3) for different fractions g of input metabolites.

We first applied the methodology by using the entire set of features, i.e., the variation of all the metabolites in the simulated network. In this case, the cardinality of $\boldsymbol{\delta x_p}$ coincides with the number of metabolites in the model $|\boldsymbol{\delta x_p}| = M$. The cross-validated grid-search procedure selected as best model the DNN with 2 hidden layers of 200 neurons each, i.e. $(200, 200)$, no dropout, learning rate $= 0.001$ and optimizer $=$ *Adam*. By analyzing the performances achieved by the different configurations (see Supplementary Table 1) we observed that models with one hidden layer provide similar performances regardless of their width. Thus, increasing the width is not enough to improve the performance and it is fundamental to explore deeper configurations. Indeed, an improvement in the predictive power of the model is observed when increasing the depth of the network. Finally, the best configuration provides an increase in the performance of $\approx 4\%$ with respect to the second best $(100, 100, 100)$. This result indicates that, by employing either wider or deeper models, the regression performance may be further improved. However, since the goal of this article is proving the potentiality of a Neural Network-based approach for predicting flux variations, exploring additional architectures is beyond the scope of this work. See Supplementary Table 1 for details about the performance of all the other DNN configurations tested.

We then tested the performance of the best model for $g = 70\%$ and $g = 50\%$ of features. On the one hand, the best DNN model selected by the cross-validated grid search procedure for $g = 70\%$ has 3 hidden layers with 100 neurons each, no dropout, learning rate $= 0.001$ and optimizer $=$ *Adam*. On the other hand, for $g = 50\%$ the best selected model has 2 hidden

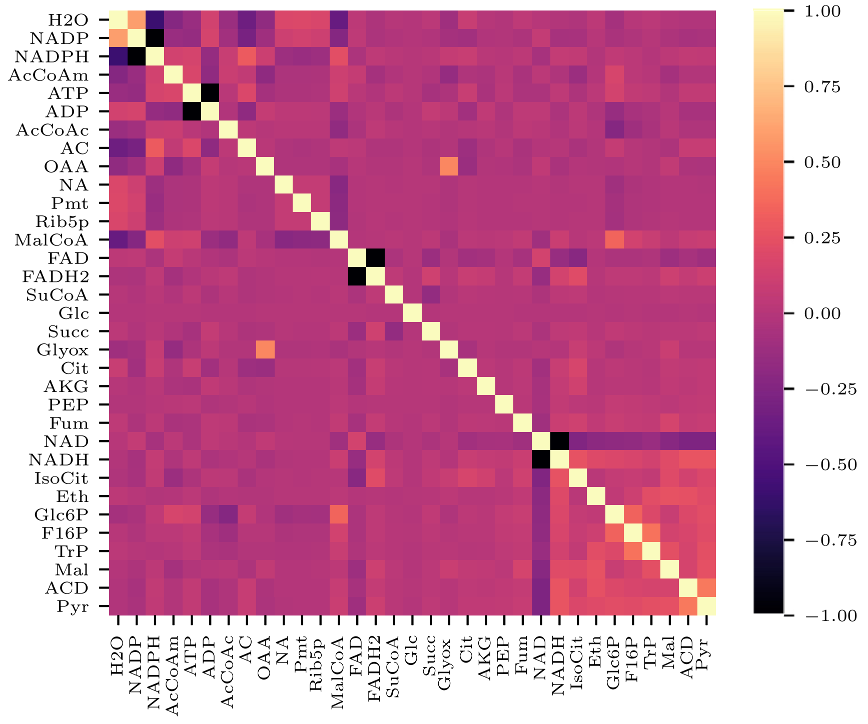


Fig. 3. Pearson correlation coefficient ρ for each pair of relative metabolites abundance variations $\boldsymbol{\delta x_{*,m}}$. The ρ correlations were computed by considering, for each metabolite, the vector of abundance variations over all samples in the dataset, with the aim of removing redundant features. For simplicity, we use metabolite names to refer to their δs, and for improved readability we sorted metabolites by their average absolute correlation $|\rho|$.

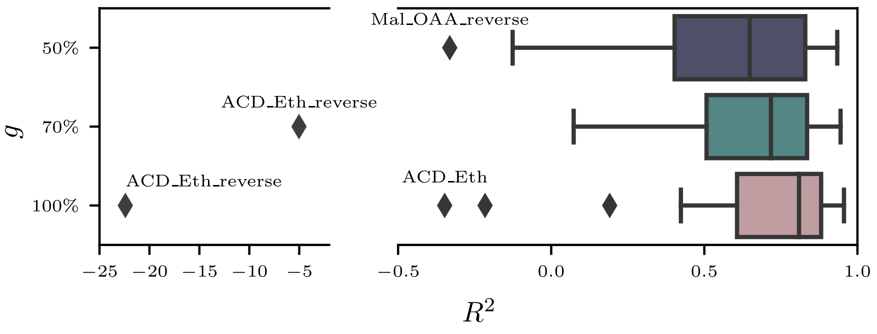


Fig. 4. Distribution of the R^2 coefficients calculated between true and predicted values of each flux r, and for different percentages g of features. The sensitivity of R^2 to outliers generates some negative outliers; as an example, ACD_Eth for $g = 100\%$, is plotted in Fig. 5. For simplicity, we use the fluxes names to refer to their δs.

layers with 200 neurons each, no dropout, learning rate $= 0.001$ and optimizer $=$ *Adam*. See Supplementary Table 2 for the MSE values of the best DNNs.

It is worth noticing that in all the experiments, we observed that configurations with no dropout, learning rate of 0.001 and *Adam* as optimizer outperformed the other possible combinations of those hyperparameters. This result indicates that we may rely upon this selection for downstream analyses, without repeating the hyperparameters selection procedure.

4.3. Performance evaluation

The three best configurations selected were retrained on the *outer training set* and, then, used to predict flux variations $\boldsymbol{\delta v_p}$ on the *test set*.

The performance was evaluated computing, for each output feature, the R^2 score, which is a measure of the amount of variance in the target values captured by the values predicted by the model (see Eq. (4)). The median value of R^2 obtained for $g = 100\%$ is 0.8, while for $g = 70\%$ it is $= 0.71$ and for $g = 50\%$ it is equal to 0.64. The distribution of R^2 values for the three cases is reported in Fig. 4. As expected, the R^2 decreases as the fraction g of selected input features gets smaller. Interestingly, the reduction in the number of features by 30% and 50% corresponds to a modest reduction of R^2 by 11% and 20%, respectively.

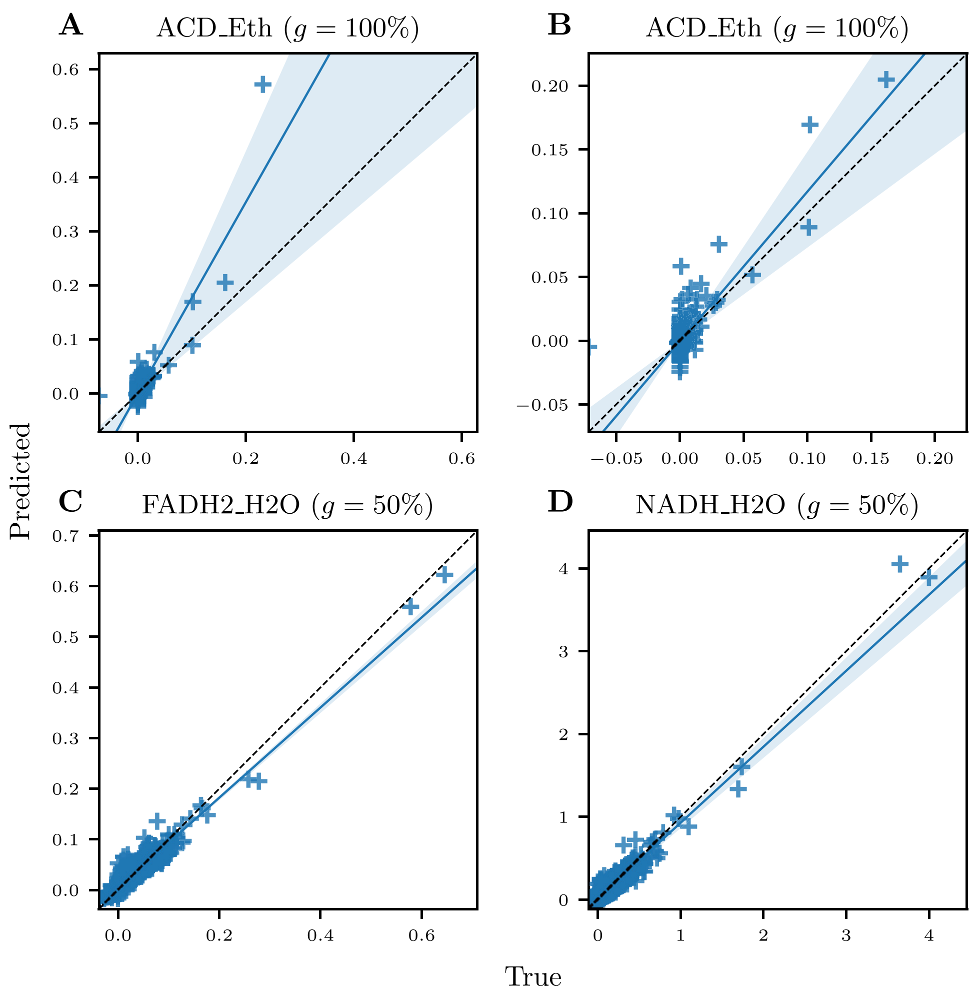


Fig. 5. Scatterplots of true and predicted values for selected fluxes. The 95% confidence interval of the regression line is visualized. For simplicity, we use the flux names to refer to their δs. (**A**) ACD_Eth for $g = 100\%$ with $R^2 = -0.35$, (**B**) ACD_Eth for $g = 100\%$ with the greatest outlier removed improves to $R^2 = 0.34$, (**C**-**D**) The two fluxes with best Pearson coefficient ($\rho = 0.971$) for $g = 50\%$.

Thus, results in terms of R^2 are overall good, except for a very few output features, such as the flux ACD_Eth in the $g = 100\%$ setting (see boxplots in Fig. 4). The existence of outlier reactions may be motivated with the choice of multi-target modeling. As motivated in Sec. 3.2, in our experiments we relied on MSE over all the predicted fluxes, and we weighed the error of each flux equally. As a result, the selected model corresponds to the one that performed better on average over all the fluxes, but the goodness of fit of different hyperparameters may vary across fluxes, as it can be observed in Supplementary Figure 1. Besides, low values of R^2 can be due to outliers in the error distribution of a single flux, especially when the variance of the true data is small. In fact, if we consider the predictions of the flux ACD_Eth (Fig. 5A), it stands out the presence of one single point for which the prediction error is many times higher than the deviance of the true values. The removal of this single outlier makes R^2 improve from $R^2 = -0.35$, up to 0.34 (Fig. 5B).

We also considered the Pearson correlation coefficient between true and predicted values (briefly ρ) to evaluate the models. It can be observed in Fig. 6 that the Pearson correlations are high. The median value of ρ is 0.90 for $g = 100\%$, 0.86 for $g = 70\%$ and 0.82 for $g = 50\%$.

To investigate in detail how the DNN performance is affected by a reduction of the number of features considered, we compared the ρ of each flux for the three g cases in Fig. 7. It can be observed that the worsening of the performance (decrease in ρ) is not homogeneously distributed among the different fluxes. On the contrary, the capability to predict many fluxes is nearly not affected by the change in g, whereas it dramatically worsens for a few fluxes.

It is natural to wonder whether the goodness of fit directly depends on the choice of the features that have been removed. The list of features that have been removed when $g = 50\%$ is: ACD, ATP, $AcCoAc$, Eth, $F16BP$, $FADH2$, $Glyox$, $H2O$, $MalCoA$, Mal, $NADH$, $NADPH$, $NADP$, NAD, Pyr, TrP, $O2$.

Surprisingly, the list includes most of the metabolites directly involved in the reactions with the best predictions in the $g = 50\%$ case, namely $FADH2\_H2O$ ($FADH2 + ADP + 0.55O_2 \Rightarrow FAD + ATP + H2O$) and $NADH\_H2O$ ($NADH2 + ADP + 0.55O_2 \Rightarrow NAD + ATP + H2O$). The ability of the DNN to predict well these two fluxes is also evident in the scatterplots

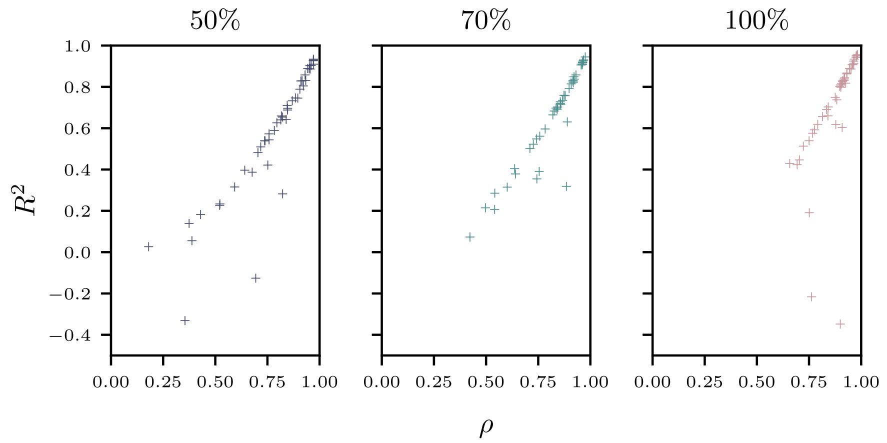


Fig. 6. Relation between Pearson and R^2 coefficients for true and predicted variation of each flux r, for different fractions of the original metabolites. We removed in advance the outliers for R^2 with a coefficient inferior to -5 (see Fig. 4).

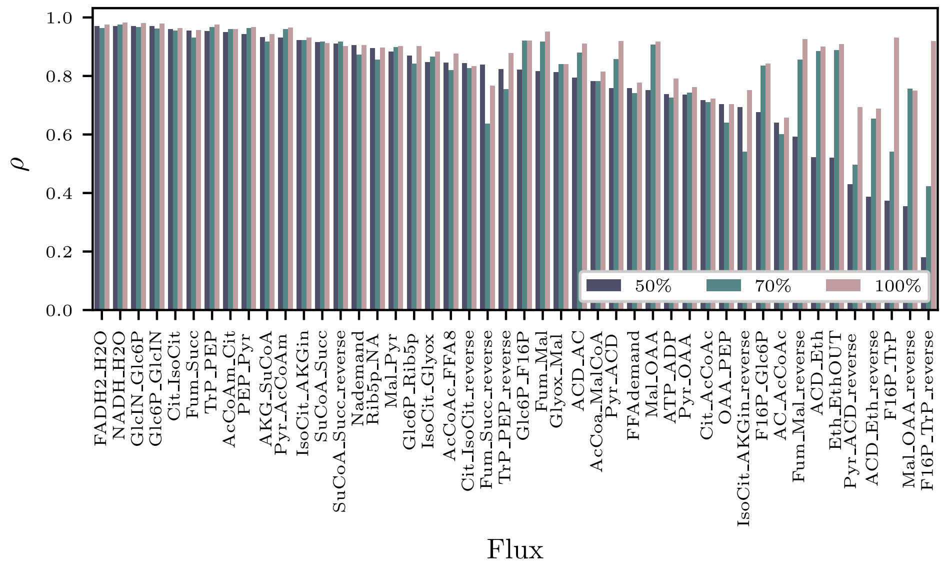


Fig. 7. Pearson correlation coefficient ρ for true and predicted variation of each flux r, for different fractions of the original metabolites. Results are sorted by decreasing ρ for the 50% fraction. For simplicity, we use the flux names to refer to their δs.

of true and predicted relative fluxes in Fig. 5B-C. This is a remarkable result, because it demonstrates that information on other metabolites in the network supports predictions in case of missing features.

However this consideration does not always hold. In fact, the list also includes all the metabolites directly involved in the reaction $F16P\_TrP\_reverse$ ($2TrP \Rightarrow F16BP$), and the substrate of the reaction $Mal\_OAA\_reverse$ ($MAL + NAD \Rightarrow OAA + NADH$), which display the worse prediction in the $g = 50\%$ case.

Taken together, the results of the performance evaluation confirm our hypothesis that patterns in metabolite abundance exist and that information in abundance variation can support the prediction of the flux variation even in reactions not directly involving those metabolites.

4.4. Performance is robust to feature reduction

Not all variation in the abundance of the different metabolites can be measured in a real system, and the variations that can be measured are hardly likely to coincide with our set of selected features. For this reason, it is relevant to investigate the effect of removing a random subset of features, instead of selecting them by looking at their pairwise correlation. To this aim, we evaluated the model performance for 10 randomly selected subsets of $g = 50\%$ and $g = 70\%$ metabolites with the overall best performing hyperparameters (i.e. hidden layers sizes (200,200), learning rate 0.001, optimizer *Adam* and no dropout). The results are reported in Fig. 8, where we show the distribution of the median R^2 of the test predictions for each random subset. It can be observed that by keeping $g = 50\%$ of the metabolites, the model performance showed greater variability (standard deviation = 0.058) than the case with $g = 70\%$ (standard deviation = 0.034). Interestingly, the model performance observed for our selected set of features falls within the first quartile and the median, in both cases.

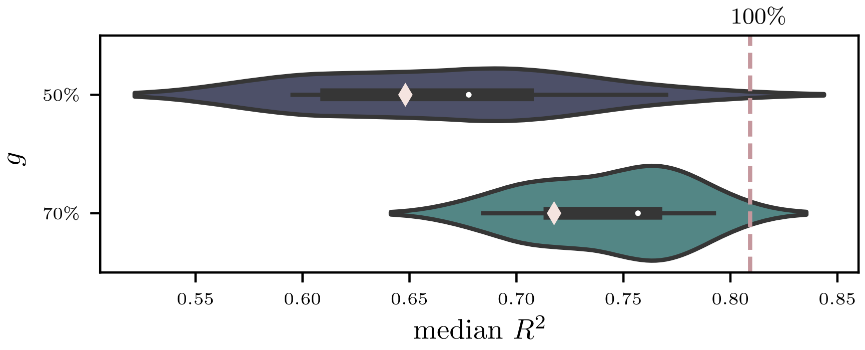


Fig. 8. Median test R^2 of 10 random subsets of $g = 50\%$ and $g = 70\%$ the original metabolites. The diamond indicates the median test R^2 obtained by our selection of features based on the absolute correlation and the dashed line corresponds to the median test R^2 achieved by keeping all the features, i.e. $g = 100\%$.

Remarkably, the median performance for R^2 drops by 6.4% only, when decreasing the cardinality of the set of features from $g = 100\%$ to $g = 70\%$, whereas it exhibits a drop of 16% when 50% of the features are removed.

It is also interesting to investigate whether the sensitivity to feature reduction differs across fluxes. In Supplementary Figure 2, the distribution of the reduction in the DNN prediction performance is reported for each flux. If we consider $g = 50\%$, it can be observed that some fluxes tend to be more sensitive to feature reduction, including the $F16P\_TrP\_reverse$ flux, which in fact resulted sensitive also in Fig. 7. On the contrary, flux $SuCoA\_Succ\_reverse$ does not seem to be particularly sensitive in Fig. 7, while displaying lower goodness of fit on average for random selections, suggesting that the low sensitivity of this flux observed in our selection was a result of the particular set of selected features.

Taken together, these results demonstrate that DNNs can predict flux variation well for most fluxes, regardless of the given subset of features. However, a few fluxes are intrinsically sensitive to feature reduction and would deserve further investigation.

4.5. *Availability and scalability*

The procedure described above was implemented using the `Python` libraries `Keras` [20] and `scikit-learn` [21]. The code and data used are available at github.com/BIMIB-DISCo/FLUX-PREDICT.

All tests were carried out on a machine with CPU 3.50 GHz Intel Xeon E3-1245 v5 and RAM 32 GB. The mean time required to train a configuration on the inner training set was 23.83 ± 11.58 minutes, while training the best model on the outer training set took 8.64 minutes.

Of course, the most computationally demanding step of our approach is the generation of the synthetic dataset by means of numerical simulations. In the specific case of the model used in this work, the total computational time to produce the data set was reasonable [11] (i.e., 5.5h to run ODEs simulations on a MacBookPro with CPU 2.6 GHz Intel Core i7, RAM 16 GB and to produce 268 Mb of data). Yet the computational time of this step depends on many factors, including the kinetic laws, the kinetic parameter values, the number of reactions and the number of simulations. An insufficient number of simulations, as well as the chosen variation range of each parameter, may impact on the goodness of fit of the ML model. However, given that we keep parameter values fixed when comparing two steady-states, the impact of under-sampling is expected to be limited. Furthermore, the computation of a large number of model trajectories may be reduced by exploiting GPU-accelerated algorithms.

5. Conclusions

We trained different configurations of deep neural networks to predict overall changes in the fluxes of a reaction system at the steady-state ($\boldsymbol{\delta v_p}$) from variations in the abundances of all or of some involved species ($\boldsymbol{\delta x_p}$). As training set, we used 100 thousands ($\boldsymbol{\delta x_p}, \boldsymbol{\delta v_p}$) pairs, obtained by sampling the parameter space and by simulating for each parameterization the steady state of a small metabolic network model under two different environmental conditions [12]. We have shown that DNNs can predict with a good level of confidence (median Pearson correlation between true and predicted values up to 0.9) changes in most reaction fluxes in the synthetic test dataset.

The fit remains good ($\rho = 0.82$) when up to 50% of the features are removed from the training set. When analyzing the goodness of fit for each output feature, we observed that the DNN predicts impressively well the variation in some fluxes, even when information on variation of the abundance of any species directly involved in the reaction is not given.

Our results indicate that patterns in relative abundances emerge from kinetic simulations of metabolic networks, with Monte Carlo generation of kinetic constants. These patterns reflect stoichiometric as well as mass balance constraints. The main advantage of using a DNN model to recognize such patterns *a posteriori*, instead of imposing constraints *a priori* as in constraint-based modeling, is the possibility to directly include information on relative abundances, instead of including them indirectly in the form of constraints on relative fluxes, which require limitations on admitted reaction rate laws (e.g., mass action) and are prone to feasibility problems. Analytical solutions, on the other hand, solve reactions individually, thus neglecting mass balance constraints, which are responsible to make predictions less sensitive to missing data.

A validation of the approach with experimental datasets and different metabolic models is of course desirable. Anyway, our approach has already the potential to pave the way for a systematic evaluation of alterations in metabolic fluxes, which is expected to guide drug target discovery, without the need for ad hoc laborious and expensive experiments and for explicit knowledge on kinetic parameters for dynamic simulations.

Our approach is not free from limitations. In order for fluxes to be predicted, the user must provide to the DNN deltas between two different conditions, whose difference must be controllable in order to be simulated (as e.g., difference in glucose availability). However, one may want to compare conditions whose triggering differences are not known *a priori*, as for instance pathological versus physiological conditions. A solution that we might envision and test in the future is to simulate many random perturbations to generate more heterogeneous ($\boldsymbol{\delta x_p}$, $\boldsymbol{\delta v_p}$) pairs and train a generic DNN.

In the future, we will also test our approach when more complex enzymatic kinetics are simulated. Difference in enzyme activities may be also taken into account by including proteomics or transcriptomics data. Finally, alternatives to DNNs, such as multi-target regression trees could also be evaluated.

CRediT authorship contribution statement

LP: Methodology, Software, Formal Analysis, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization; **FC**: Methodology, Software, Formal Analysis, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization; **DM**: Methodology, Formal Analysis, Writing - Review & Editing, Visualization; **AG**: Methodology, Formal Analysis, Writing - Review & Editing; **CD**: Conceptualization, Methodology, Formal Analysis, Writing - Original Draft, Writing - Review & Editing, Visualization, Supervision.

Declaration of competing interest

The authors declare that they have no competing interests.

Acknowledgments

We warmly thank Dario Pescini for providing the dataset and Giancarlo Mauri for the useful suggestions provided during the revision process.

Funding

The institutional financial support to SYSBIO.ISBE.IT within the Italian Roadmap for ESFRI Research Infrastructures and the FLAG-ERA grant ITFoC is gratefully acknowledged. Financial support from the Italian Ministry of University and Research (MIUR) through grant ‘Dipartimenti di Eccellenza 2017’ to University of Milano Bicocca is also greatly acknowledged. Support was also provided by the CRUK/AIRC Accelerator Award #22790: “Single-cell Cancer Evolution in the Clinic”.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ic.2021.104798.

References

[1] C. Damiani, D. Gaglio, E. Sacco, L. Alberghina, M. Vanoni, Systems metabolomics: from metabolomic snapshots to design principles, Curr. Opin. Biotechnol. 63 (2020) 190–199.
[2] S. Galmarini, J.V.-G. de Arellano, J. Duyzer, Fluxes of chemically reactive species inferred from mean concentration measurements, Atmos. Environ. 31 (15) (1997) 2371–2374.
[3] M. Sajitz-Hermstein, N. Töpfer, S. Kleessen, A.R. Fernie, Z. Nikoloski, IReMet-flux: constraint-based approach for integrating relative metabolite levels into a stoichiometric metabolic models, Bioinformatics 32 (17) (2016), i755–i762.
[4] V. Pandey, N. Hadadi, V. Hatzimanikatis, Enhanced flux prediction by integrating relative expression and relative metabolite abundance into thermodynamically consistent metabolic models, PLoS Comput. Biol. 15 (5) (2019) e1007036.
[5] P. Rana, C. Berry, P. Ghosh, S.S. Fong, Recent advances on constraint-based models by integrating machine learning, Curr. Opin. Biotechnol. 64 (2020) 85–91.
[6] G. Zampieri, S. Vijayakumar, E. Yaneske, C. Angione, Machine and deep learning meet genome-scale metabolic modeling, PLoS Comput. Biol. 15 (7) (2019) e1007084.
[7] M. Cuperlovic-Culf, Machine learning methods for analysis of metabolic data and metabolic pathway modeling, Metabolites 8 (1) (2018) 4.
[8] A. Ajjolli Nagaraja, N. Fontaine, M. Delsaut, P. Charton, C. Damour, B. Offmann, B. Grondin-Perez, F. Cadet, Flux prediction using artificial neural network (ann) for the upper part of glycolysis, PLoS ONE 14 (5) (2019) e0216178.
[9] A. Zelezniak, J. Vowinckel, F. Capuano, C.B. Messner, V. Demichev, N. Polowsky, M. Mülleder, S. Kamrad, B. Klaus, M.A. Keller, et al., Machine learning predicts the yeast metabolome from the quantitative proteome of kinase knockouts, Cell Syst. 7 (3) (2018) 269–283.
[10] Z. Costello, H.G. Martin, A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data, NPJ Syst. Biol. Appl. 4 (1) (2018) 1–14.

[11] C. Damiani, R. Colombo, M. Di Filippo, D. Pescini, G. Mauri, Linking alterations in metabolic fluxes with shifts in metabolite levels by means of kinetic modeling, in: Italian Workshop on Artificial Life and Evolutionary Computation, Springer, 2016, pp. 138–148.

[12] R. Colombo, C. Damiani, G. Mauri, D. Pescini, Constraining mechanism based simulations to identify ensembles of parametrizations to characterize metabolic features, in: International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics, Springer, 2016, pp. 107–117.

[13] R. Colombo, C. Damiani, D. Gilbert, M. Heiner, G. Mauri, D. Pescini, Emerging ensembles of kinetic parameters to characterize observed metabolic phenotypes, BMC Bioinform. 19 (7) (2018) 45–59.

[14] A. Cornish-Bowden, One hundred years of Michaelis–Menten kinetics, in: Proceedings of the Beilstein ESCEC Symposium - Celebrating the 100th Anniversary of Michaelis Menten-Kinetics, Perspect. Sci. 4 (2015) 3–9, https://doi.org/10.1016/j.pisc.2014.12.002.

[15] M.A. Savageau, Michaelis-Menten mechanism reconsidered: implications of fractal kinetics, J. Theor. Biol. 176 (1) (1995) 115–124.

[16] L. Petzold, Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations, SIAM J. Sci. Stat. Comput. 4 (1) (1983) 136–148, https://doi.org/10.1137/0904010.

[17] M. Kuhn, K. Johnson, et al., Applied Predictive Modeling, vol. 26, Springer, 2013.

[18] H. Borchani, G. Varando, C. Bielza, P. Larrañaga, A survey on multi-output regression, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 5 (5) (2015) 216–233, https://doi.org/10.1002/widm.1157.

[19] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[20] F. Chollet, et al., Keras, https://keras.io, 2015.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

Bibliography

[Qui86] J. R. Quinlan. "Induction of Decision Trees". In: *Machine Learning* 1.1 (Mar. 1986), pp. 81–106. ISSN: 0885-6125, 1573-0565. DOI: `10.1007/BF00116251` (cit. on p. 22).

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. ISSN: 0893-6080. DOI: `10.1016/0893-6080(89)90020-8` (cit. on p. 71).

[McL99] Goeffrey J McLachlan. "Mahalanobis Distance". In: *Resonance* 4.6 (1999), pp. 20–26 (cit. on p. 73).

[Sch+99] Bernhard Schölkopf et al. "Support Vector Method for Novelty Detection". In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. Ed. by Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller. The MIT Press, 1999, pp. 582–588 (cit. on pp. 4, 35, 38, 68, 95).

[Jap01] Nathalie Japkowicz. "Concept-Learning in the Presence of Between-Class and Within-Class Imbalances". In: *Advances in Artificial Intelligence*. Ed. by Gerhard Goos et al. Vol. 2056. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 67–77. ISBN: 978-3-540-42144-3 978-3-540-45153-2. DOI: `10.1007/3-540-45153-6_7` (cit. on pp. 70, 89).

[Tax01] DMJ Tax. "One-class classification; concept-learning in the absence of counter-examples". PhD thesis. Delft University of Technology, 2001. ISBN: 90-75691-05-X (cit. on p. 38).

[DG06] Jesse Davis and Mark Goadrich. "The Relationship between Precision-Recall and ROC Curves". In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. Ed. by William W. Cohen and Andrew W. Moore. Vol. 148. ACM International Conference Proceeding Series. ACM, 2006, pp. 233–240. DOI: `10.1145/1143844.1143874` (cit. on p. 43).

[BW08] Peter L. Bartlett and Marten H. Wegkamp. "Classification with a Reject Option Using a Hinge Loss". In: *Journal of Machine Learning Research* 9 (2008), pp. 1823–1840. DOI: 10.5555/1390681.1442792 (cit. on pp. 13, 16).

[Car+08] Gunnar E. Carlsson et al. "On the Local Behavior of Spaces of Natural Images". In: *International Journal of Computer Vision* 76.1 (2008), pp. 1–12. DOI: 10.1007/s11263-007-0056-x (cit. on p. 70).

[vH08] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data Using T-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605 (cit. on pp. 53, 54, 77, 79).

[Ben+09] Yoshua Bengio et al. "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. Montreal, Quebec, Canada: ACM Press, 2009, pp. 1–8. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553380 (cit. on p. 93).

[Den+09] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 27).

[Kri09] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. 2009, p. 60 (cit. on pp. 35, 67, 69, 76, 81, 82, 89, 92).

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST Handwritten Digit Database". In: *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010) (cit. on pp. 17, 24, 27, 31, 92).

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. Ed. by Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 315–323 (cit. on pp. 16–18).

[KBQ11] Tomer Kalisky, Paul Blainey, and Stephen R Quake. "Genomic Analysis at the Single-Cell Level". In: *Annual review of genetics* 45 (2011) (cit. on p. 97).

[Kri+11] Hans-Peter Kriegel et al. "Interpreting and Unifying Outlier Scores". In: *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*. SIAM / Omnipress, 2011, pp. 13–24. DOI: 10.1137/1.9781611972818.2 (cit. on p. 73).

[Net+11] Yuval Netzer et al. “Reading Digits in Natural Images with Unsupervised Feature Learning”. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. 2011 (cit. on pp. 35, 69, 76, 82, 84, 85, 87, 92).

[BSJ12] Colin Bellinger, Shiven Sharma, and Nathalie Japkowicz. “One-Class versus Binary Classification: Which and When?” In: *2012 11th International Conference on Machine Learning and Applications*. Vol. 2. Dec. 2012, pp. 102–106. DOI: `10.1109/ICMLA.2012.212` (cit. on p. 68).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a Meeting Held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett et al. 2012, pp. 1106–1114 (cit. on p. 1).

[QS12] Jing Qian and Venkatesh Saligrama. “New Statistic in P-Value Estimation for Anomaly Detection”. In: *2012 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE. 2012, pp. 393–396 (cit. on pp. 73, 74, 95).

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a Meeting Held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett et al. 2012, pp. 2960–2968 (cit. on pp. 38, 44).

[TK12] Masaru Tomita and Kenjiro Kami. “Systems Biology, Metabolomics, and Cancer Metabolism”. In: *Science (New York, N.Y.)* 336.6084 (2012), pp. 990–991 (cit. on p. 118).

[Goo+13] Ian J. Goodfellow et al. “Maxout Networks”. In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 1319–1327 (cit. on p. 19).

[LA13] Nathan E Lewis and Alyaa M Abdel-Haleem. “The Evolution of Genome-Scale Models of Cancer Metabolism”. In: *Frontiers in physiology* 4 (2013), p. 237 (cit. on p. 118).

[Lóp+13] Victoria López et al. “An Insight into Classification with Imbalanced Data: Empirical Results and Current Trends on Using Data Intrinsic Characteristics”. In: *Information Sciences* 250 (Nov. 2013), pp. 113–

141. ISSN: 0020-0255. DOI: 10.1016/j.ins.2013.07.007 (cit. on p. 70).

[MHN+13] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. “Rectifier Nonlinearities Improve Neural Network Acoustic Models”. In: *Proc. Icml*. Vol. 30. Atlanta, Georgia, USA. 2013, p. 3 (cit. on p. 19).

[Mon+14] Guido F. Montúfar et al. “On the Number of Linear Regions of Deep Neural Networks”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani et al. 2014, pp. 2924–2932 (cit. on pp. 16, 18, 21, 31).

[PMB14] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. “On the Number of Response Regions of Deep Feed Forward Networks with Piece-Wise Linear Activations”. In: *arXiv:1312.6098 [cs]* (Feb. 2014). arXiv: 1312.6098 [cs] (cit. on p. 18).

[Sri+14] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958 (cit. on pp. 32, 94).

[Sze+14] Christian Szegedy et al. “Intriguing Properties of Neural Networks”. In: *International Conference on Learning Representations*. 2014 (cit. on pp. 2, 3, 13, 34).

[ZAR14] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. “Capturing Long-Tail Distributions of Object Subcategories”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2014, pp. 915–922. DOI: 10.1109/CVPR.2014.122 (cit. on pp. 2, 9).

[Blu+15] Charles Blundell et al. “Weight Uncertainty in Neural Networks”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML’15. Lille, France: JMLR.org, 2015, pp. 1613–1622 (cit. on p. 12).

[Cir+15] Giovanni Ciriello et al. “Comprehensive Molecular Portraits of Invasive Lobular Breast Cancer”. In: *Cell* 163.2 (2015), pp. 506–519 (cit. on p. 118).

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015 (cit. on pp. 2, 3, 7, 9, 10, 13, 34–36, 44, 73, 92).

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv:1502.03167 [cs]* (Feb. 2015). arXiv: `1502.03167 [cs]` (cit. on pp. 32, 94).

[LBA15] Daniel J. Laydon, Charles R. M. Bangham, and Becca Asquith. "Estimating T-Cell Repertoire Diversity: Limitations of Classical Estimators and a New Approach". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 370.1675 (Aug. 2015), p. 20140291. ISSN: 0962-8436. DOI: `10.1098/rstb.2014.0291` (cit. on p. 76).

[Lin+15] Carsten Linnemann et al. "High-Throughput Epitope Discovery Reveals Frequent Recognition of Neo-Antigens by CD4+ T Cells in Human Melanoma". In: *Nature Medicine* 21.1 (Jan. 2015), pp. 81–85. ISSN: 1546-170X. DOI: `10.1038/nm.3773` (cit. on p. 77).

[Plo15] Plotly Technologies Inc. *Collaborative Data Science*. https://plot.ly. Montreal, QC, 2015 (cit. on p. 28).

[RSS15] Jason A. Reuter, Damek Spacek, and Michael P. Snyder. "High-Throughput Sequencing Technologies". In: *Molecular cell* 58.4 (May 2015), pp. 586–597. ISSN: 1097-2765. DOI: `10.1016/j.molcel.2015.05.004` (cit. on pp. 69, 77).

[Spr+15] Jost Tobias Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015 (cit. on pp. 1, 13, 48).

[Yu+15] Fisher Yu et al. "LSUN: Construction of a Large-Scale Image Dataset Using Deep Learning with Humans in the Loop". In: (2015). DOI: `10.48550/ARXIV.1506.03365` (cit. on pp. 69, 82, 85, 87).

[Amo+16] Dario Amodei et al. *Concrete Problems in AI Safety*. July 2016. DOI: `10.48550/arXiv.1606.06565`. arXiv: `1606.06565 [cs]` (cit. on pp. 1, 2, 7, 12, 13).

[Boj+16] Mariusz Bojarski et al. "End to End Learning for Self-Driving Cars". In: (2016). DOI: `10.48550/ARXIV.1604.07316` (cit. on p. 1).

[FMN16] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. "Testing the Manifold Hypothesis". In: *Journal of the American Mathematical Society* 29.4 (Oct. 2016), pp. 983–1049. ISSN: 0894-0347, 1088-6834. DOI: `10.1090/jams/852` (cit. on p. 70).

[GG16] Yarin Gal and Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 1050–1059 (cit. on pp. 2, 12, 73).

[He+16] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016 (cit. on pp. 35, 92).

[KKK16] Been Kim, Oluwasanmi Koyejo, and Rajiv Khanna. “Examples Are Not Enough, Learn to Criticize! Criticism for Interpretability”. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee et al. 2016, pp. 2280–2288 (cit. on pp. 13, 16).

[MFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2574–2582 (cit. on pp. 35–37, 44, 92).

[Pap+16] Nicolas Papernot et al. “Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks”. In: *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 582–597. DOI: 10.1109/SP.2016.41 (cit. on pp. 13, 34).

[Saj+16] Max Sajitz-Hermstein et al. “iReMet-Flux: Constraint-Based Approach for Integrating Relative Metabolite Levels into a Stoichiometric Metabolic Models”. In: *Bioinformatics (Oxford, England)* 32.17 (2016), pp. i755–i762 (cit. on p. 129).

[Sil+16] David Silver et al. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nat.* 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961 (cit. on p. 1).

[AG17] Mahdieh Abbasi and Christian Gagné. “Robustness to Adversarial Examples through an Ensemble of Specialists”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017 (cit. on pp. 13, 34, 35).

[AS17] Charu C. Aggarwal and Saket Sathe. *Outlier Ensembles - an Introduction*. Springer, 2017. ISBN: 978-3-319-54764-0. DOI: 10.1007/978-3-319-54765-7 (cit. on p. 35).

[BBS17] Alexander Bagnall, Razvan Bunescu, and Gordon Stewart. “Training Ensembles to Detect Adversarial Examples”. In: arXiv:1712.04006 (Dec. 2017). arXiv: `1712.04006 [cs]` (cit. on pp. 13, 34, 35).

[Bau+17] David Bau et al. “Network Dissection: Quantifying Interpretability of Deep Visual Representations”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 3319–3327. DOI: `10.1109/CVPR.2017.354` (cit. on pp. 17, 31).

[CW17a] Nicholas Carlini and David Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. May 2017, pp. 39–57. DOI: `10.1109/SP.2017.49` (cit. on pp. 36, 37).

[CW17b] Nicholas Carlini and David A. Wagner. “Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*. Ed. by Bhavani M. Thuraisingham et al. ACM, 2017, pp. 3–14. DOI: `10.1145/3128572.3140444` (cit. on pp. 35, 44, 50, 92).

[Car+17] Fabio Carrara et al. “Detecting Adversarial Example Attacks to Deep Neural Networks”. In: *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing, CBMI 2017, Florence, Italy, June 19-21, 2017*. ACM, 2017, 38:1–38:7. DOI: `10.1145/3095713.3095753` (cit. on p. 11).

[Dam+17] Chiara Damiani et al. “Linking Alterations in Metabolic Fluxes with Shifts in Metabolite Levels by Means of Kinetic Modeling”. In: *Italian Workshop on Artificial Life and Evolutionary Computation*. Springer. 2017, pp. 138–148 (cit. on p. 129).

[Die17] Thomas G. Dietterich. “Steps Toward Robust Artificial Intelligence”. In: *AI Magazine* 38.3 (Oct. 2017), pp. 3–24. ISSN: 2371-9621. DOI: `10.1609/aimag.v38i3.2756` (cit. on pp. 1, 2, 7, 12).

[Fac+17] Elena Facco et al. “Estimating the Intrinsic Dimension of Datasets by a Minimal Neighborhood Information”. In: *Scientific Reports* 7.1 (Sept. 2017), p. 12140. ISSN: 2045-2322. DOI: `10.1038/s41598-017-11873-y` (cit. on pp. 70, 74, 75, 94).

[Fei+17] Reuben Feinman et al. “Detecting Adversarial Samples from Artifacts”. In: *CoRR* abs/1703.00410 (2017). arXiv: `1703.00410` (cit. on pp. 11, 12).

[GWW17] Wilson Wen Bin Goh, Wei Wang, and Limsoon Wong. “Why Batch Effects Matter in Omics Data, and How to Avoid Them”. In: *Trends in biotechnology* 35.6 (2017), pp. 498–507 (cit. on p. 98).

[Gro+17] Kathrin Grosse et al. “On the (Statistical) Detection of Adversarial Examples”. In: *CoRR* abs/1702.06280 (2017). arXiv: `1702.06280` (cit. on p. 11).

[Haq+17] Ashraful Haque et al. “A Practical Guide to Single-Cell RNA-Sequencing for Biomedical Research and Clinical Applications”. In: *Genome medicine* 9.1 (2017), pp. 1–12 (cit. on p. 98).

[He+17] Warren He et al. “Adversarial Example Defense: Ensembles of Weak Defenses Are Not Strong”. In: *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. 2017 (cit. on pp. 50, 52).

[HG17a] Dan Hendrycks and Kevin Gimpel. “A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017 (cit. on pp. 11, 12, 56).

[HG17b] Dan Hendrycks and Kevin Gimpel. “Early Methods for Detecting Adversarial Images”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017 (cit. on pp. 11, 12).

[Hua+17] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4700–4708 (cit. on pp. 35, 92).

[KG17] Alex Kendall and Yarin Gal. “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5574–5584 (cit. on p. 12).

[KGB17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. “Adversarial Examples in the Physical World”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017 (cit. on pp. 35–37, 44, 92).

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 6402–6413 (cit. on pp. 2, 12, 72, 77).

[Lei+17] Christian Leibig et al. “Leveraging Uncertainty Information from Deep Neural Networks for Disease Detection”. In: *Scientific Reports* 7.1 (Dec. 2017), p. 17816. ISSN: 2045-2322. DOI: `10.1038/s41598-017-17876-z` (cit. on pp. 1, 2, 13, 31, 93).

[Lit+17] Geert Litjens et al. “A Survey on Deep Learning in Medical Image Analysis”. In: *Medical Image Analysis* 42 (Dec. 2017), pp. 60–88. ISSN: 1361-8415. DOI: `10.1016/j.media.2017.07.005` (cit. on p. 1).

[LIF17] Jiajun Lu, Theerasit Issaranon, and David A. Forsyth. “SafetyNet: Detecting and Rejecting Adversarial Examples Robustly”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, 2017, pp. 446–454. DOI: `10.1109/ICCV.2017.56` (cit. on pp. 11, 56).

[MC17] Dongyu Meng and Hao Chen. “MagNet: A Two-Pronged Defense against Adversarial Examples”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017.* Ed. by Bhavani M. Thuraisingham et al. ACM, 2017, pp. 135–147. DOI: `10.1145/3133956.3134057` (cit. on p. 11).

[Met+17] Jan Hendrik Metzen et al. “On Detecting Adversarial Perturbations”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.* OpenReview.net, 2017 (cit. on p. 11).

[OMS17] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. “Feature Visualization”. In: *Distill* (2017). DOI: `10.23915/distill.00007` (cit. on pp. 17, 31, 93).

[Rag+17] Maithra Raghu et al. “On the Expressive Power of Deep Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017.* Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 2847–2854 (cit. on p. 18).

[Sah+17] Ugur Sahin et al. “Personalized RNA Mutanome Vaccines Mobilize Poly-Specific Therapeutic Immunity against Cancer”. In: *Nature* 547.7662 (July 2017), pp. 222–226. ISSN: 1476-4687. DOI: `10.1038/nature23003` (cit. on p. 76).

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. “Learning Important Features Through Propagating Activation Differences”. In: *Proceedings of the 34th International Conference on Machine Learning.* PMLR, July 2017, pp. 3145–3153 (cit. on pp. 1, 13, 48).

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, July 2017, pp. 3319–3328 (cit. on pp. 1, 13, 48).

[Aro+18] Raman Arora et al. “Understanding Deep Neural Networks with Rectified Linear Units”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018 (cit. on p. 17).

[ACW18] Anish Athalye, Nicholas Carlini, and David Wagner. “Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples”. In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, July 2018, pp. 274–283 (cit. on pp. 37, 50, 52).

[CM18] Vasileios Charisopoulos and Petros Maragos. “A Tropical Approach to Neural Networks with Piecewise Linear Activations”. In: (2018). DOI: 10.48550/ARXIV.1805.08749 (cit. on p. 17).

[FKH18] Stefan Falkner, Aaron Klein, and Frank Hutter. “BOHB: Robust and Efficient Hyperparameter Optimization at Scale”. In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1436–1445 (cit. on p. 55).

[Gra+18] Alex Graudenzi et al. “Integration of Transcriptomic Data and Metabolic Networks in Cancer Samples Reveals Highly Significant Prognostic Power”. In: *Journal of Biomedical Informatics* 87 (Nov. 2018), pp. 37–49. ISSN: 1532-0480. DOI: 10.1016/j.jbi.2018.09.010 (cit. on p. 118).

[KLS18] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. “Optimal Whitening and Decorrelation”. In: *The American Statistician* 72.4 (2018), pp. 309–314. DOI: 10.1080/00031305.2016.1277159. eprint: https://doi.org/10.1080/00031305.2016.1277159 (cit. on p. 38).

[Kol+18] Bojan Kolosnjaji et al. “Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables”. In: *26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, September 3-7, 2018*. IEEE, 2018, pp. 533–537. DOI: 10.23919/EUSIPCO.2018.8553214 (cit. on p. 34).

[Lee+18] Kimin Lee et al. “A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks”. In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8,* *2018, Montréal, Canada*. Ed. by Samy Bengio et al. 2018, pp. 7167–7177 (cit. on pp. 2, 4, 11, 34, 35, 38, 39, 41, 44, 47, 51, 52, 55, 63–65).

[LLS18] Shiyu Liang, Yixuan Li, and R. Srikant. "Enhancing the Reliability of Out-of-Distribution Image Detection in Neural Networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018 (cit. on pp. 12, 41, 56, 82).

[Ma+18] Xingjun Ma et al. "Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018 (cit. on pp. 4, 11, 34, 35, 38, 42, 44, 47, 51, 52, 55, 63–65).

[Nov+18] Roman Novak et al. "Sensitivity and Generalization in Neural Networks: An Empirical Study". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018 (cit. on pp. 10, 18, 93).

[PM18] Nicolas Papernot and Patrick McDaniel. "Deep K-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning". In: *arXiv:1803.04765 [cs, stat]* (Mar. 2018). arXiv: `1803.04765 [cs, stat]` (cit. on pp. 2, 11).

[Ruf+18] Lukas Ruff et al. "Deep One-Class Classification". In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, July 2018, pp. 4393–4402 (cit. on p. 95).

[STR18] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. "Bounding and Counting Linear Regions of Deep Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4565–4573 (cit. on pp. 18, 93).

[Str+18] Thilo Strauss et al. "Ensemble Methods as a Defense to Adversarial Perturbations Against Deep Neural Networks". In: arXiv:1709.03423 (Feb. 2018). arXiv: `1709.03423 [cs, stat]` (cit. on pp. 13, 34, 35).

[Wen+18] Zeyi Wen et al. "ThunderSVM: A Fast SVM Library on GPUs and CPUs". In: *Journal of Machine Learning Research* 19 (2018), pp. 797–801 (cit. on p. 55).

[XEQ18] Weilin Xu, David Evans, and Yanjun Qi. “Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks”. In: *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018 (cit. on p. 11).

[ZH18] Zhihao Zheng and Pengyu Hong. “Robust Detection of Adversarial Attacks by Modeling the Intrinsic Properties of Deep Neural Networks”. In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al. 2018, pp. 7924–7933 (cit. on p. 11).

[AD19] Jonathan Aigrain and Marcin Detyniecki. “Detecting Adversarial Examples and Other Misclassifications in Neural Networks by Introspection”. In: *CoRR* abs/1905.09186 (2019). arXiv: `1905.09186` (cit. on p. 12).

[Ans+19] Alessio Ansuini et al. “Intrinsic Dimension of Data Representations in Deep Neural Networks”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 6109–6119 (cit. on pp. 69, 70, 82, 83, 94).

[Bal+19] Randall Balestriero et al. “The Geometry of Deep Networks: Power Diagram Subdivision”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 15806–15815 (cit. on p. 17).

[BL19] Jonathon Byrd and Zachary Lipton. “What Is the Effect of Importance Weighting in Deep Learning?” In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, pp. 872–881 (cit. on pp. 71, 82).

[CEP19] Nicholas Carlini, Úlfar Erlingsson, and Nicolas Papernot. *Distribution Density, Tails, and Outliers in Machine Learning: Metrics and Applications*. Oct. 2019. arXiv: `1910.13427 [cs, stat]` (cit. on p. 10).

[FAK19] Sara Fotouhi, Shahrokh Asadi, and Michael W. Kattan. “A Comprehensive Data Level Analysis for Cancer Diagnosis on Imbalanced Data”. In: *Journal of Biomedical Informatics* 90 (Feb. 2019), p. 103089. ISSN: 1532-0464. DOI: `10.1016/j.jbi.2018.12.003` (cit. on pp. 4, 68, 69, 94).

[HR19a] Boris Hanin and David Rolnick. “Complexity of Linear Regions in Deep Networks”. In: *International Conference on Machine Learning*. 2019, pp. 2596–2604 (cit. on p. 18).

[HR19b] Boris Hanin and David Rolnick. “Deep ReLU Networks Have Surprisingly Few Activation Patterns”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 359–368 (cit. on pp. 16, 18, 28, 93).

[Hoo+19] Sara Hooker et al. “A Benchmark for Interpretability Methods in Deep Neural Networks”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 9734–9745 (cit. on p. 48).

[HWW19] Bo Huang, Yi Wang, and Wei Wang. “Model-Agnostic Adversarial Detection by Random Perturbations”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 4689–4696. DOI: `10.24963/ijcai.2019/651` (cit. on p. 11).

[Jam+19] Oveis Jamialahmadi et al. “A Benchmark-Driven Approach to Reconstruct Metabolic Networks for Studying Cancer Metabolism”. In: *PLoS computational biology* 15.4 (2019), e1006936 (cit. on pp. 118, 129).

[Ma+19] Shiqing Ma et al. “NIC: Detecting Adversarial Samples with Neural Network Invariant Checking”. In: *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019 (cit. on pp. 11, 35, 38).

[Mon+19] João Monteiro et al. “Generalizable Adversarial Examples Detection Based on Bi-Model Decision Mismatch”. In: *2019 IEEE International Conference on Systems, Man and Cybernetics, SMC 2019, Bari, Italy, October 6-9, 2019*. IEEE, 2019, pp. 2839–2844. DOI: `10.1109/SMC.2019.8913861` (cit. on p. 12).

[Mös+19] Anja Mösch et al. “Machine Learning for Cancer Immunotherapies Based on Epitope Recognition by T Cell Receptors”. In: *Frontiers in Genetics* 10 (2019). ISSN: 1664-8021 (cit. on pp. 68, 77).

[Mur+19] W. James Murdoch et al. “Definitions, Methods, and Applications in Interpretable Machine Learning”. In: *Proceedings of the National Academy of Sciences* 116.44 (Oct. 2019), pp. 22071–22080. DOI:

`10.1073/pnas.1900654116`. (Visited on 04/03/2023) (cit. on pp. 1, 2, 12).

[Nor+19] Harsha Nori et al. "InterpretML: A Unified Framework for Machine Learning Interpretability". In: *arXiv preprint arXiv:1909.09223* (2019). arXiv: `1909.09223` (cit. on pp. 12, 13, 16).

[Qin+19] Yao Qin et al. "Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5231–5240 (cit. on p. 34).

[RKH19] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. "The Odds Are Odd: A Statistical Test for Detecting Adversarial Examples". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5498–5507 (cit. on pp. 11, 56).

[Sno+19] Jasper Snoek et al. "Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty under Dataset Shift". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 13969–13980 (cit. on p. 12).

[Ton+19] Mariya Toneva et al. "An Empirical Study of Example Forgetting during Deep Neural Network Learning". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019 (cit. on pp. 2, 10, 22, 25, 31).

[Vie+19] Beate Vieth et al. "A Systematic Evaluation of Single Cell RNA-Seq Analysis Pipelines". In: *Nature communications* 10.1 (2019), pp. 1–11 (cit. on p. 97).

[Vin+19] Oriol Vinyals et al. "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning". In: *Nature* 575.7782 (Nov. 2019), pp. 350–354. ISSN: 1476-4687. DOI: `10.1038/s41586-019-1724-z` (cit. on p. 1).

[ZXY19] Xiuwei Zhang, Chenling Xu, and Nir Yosef. "Simulating Multiple Faceted Variability in Single Cell RNA Sequencing". In: *Nature communications* 10.1 (2019), pp. 1–16 (cit. on p. 98).

[Agg20] Charu C. Aggarwal. *Linear Algebra and Optimization for Machine Learning - A Textbook*. Springer, 2020. ISBN: 978-3-030-40343-0. DOI: `10.1007/978-3-030-40344-7` (cit. on p. 39).

[Ala+20] Shamshe Alam et al. "One-Class Support Vector Classifiers: A Survey". In: *Knowl. Based Syst.* 196 (2020), p. 105754. DOI: `10.1016/j.knosys.2020.105754` (cit. on p. 40).

[Arr+20] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI". In: *Inf. Fusion* 58 (2020), pp. 82–115. DOI: `10.1016/j.inffus.2019.12.012` (cit. on pp. 12, 91).

[Bag+20] Dmitry V. Bagaev et al. "VDJdb in 2019: Database Extension, New Analysis Infrastructure and a T-Cell Receptor Motif Compendium". In: *Nucleic Acids Research* 48.Database-Issue (2020), pp. D1057–D1062. DOI: `10.1093/nar/gkz874` (cit. on p. 77).

[Bro+20] Tom Brown et al. "Language Models Are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901 (cit. on p. 1).

[CSG20] Gilad Cohen, Guillermo Sapiro, and Raja Giryes. "Detecting Adversarial Samples Using Influence Functions and Nearest Neighbors". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 14441–14450. DOI: `10.1109/CVPR42600.2020.01446` (cit. on p. 11).

[Cra+20a] Francesco Craighero et al. "Investigating the Compositional Structure of Deep Neural Networks". In: *Machine Learning, Optimization, and Data Science*. Ed. by Giuseppe Nicosia et al. Cham: Springer International Publishing, 2020, pp. 322–334. ISBN: 978-3-030-64583-0 (cit. on pp. 3, 15, 56, 91, 92).

[Cra+20b] Francesco Craighero et al. "Understanding Deep Learning with Activation Pattern Diagrams". In: *Proceedings of the Italian Workshop on Explainable Artificial Intelligence Co-Located with 19th International Conference of the Italian Association for Artificial Intelligence, XAI.it@AIxIA 2020, Online Event, November 25-26, 2020*. Ed. by Cataldo Musto et al. Vol. 2742. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 119–126 (cit. on pp. 3, 15, 91, 92).

[Dam+20] Chiara Damiani et al. "MaREA4Galaxy: Metabolic Reaction Enrichment Analysis and Visualization of RNA-Seq Data within Galaxy". In: *Computational and structural biotechnology journal* 18 (2020), pp. 993–999 (cit. on p. 118).

[Din+20] Jennifer N. Dines et al. "The ImmuneRACE Study: A Prospective Multicohort Study of Immune Response Action to COVID-19 Events with the ImmuneCODE™ Open Access Database". In: *medRxiv : the preprint server for health sciences* (2020). DOI: `10.1101/2020.08.`

17.20175158. eprint: https://www.medrxiv.org/content/early/2020/08/21/2020.08.17.20175158.1.full.pdf (cit. on p. 77).

[Fel20] Vitaly Feldman. “Does Learning Require Memorization? A Short Tale about a Long Tail”. In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing.* 2020, pp. 954–959 (cit. on pp. 2, 9).

[Gao+20] Long Gao et al. “Handling Imbalanced Medical Image Data: A Deep-Learning-Based One-Class Classification Approach”. In: 108 (2020), p. 101935. DOI: 10.1016/j.artmed.2020.101935 (cit. on p. 69).

[Gri+20] Sorin Grigorescu et al. “A Survey of Deep Learning Techniques for Autonomous Driving”. In: *Journal of Field Robotics* 37.3 (2020), pp. 362–386. ISSN: 1556-4967. DOI: 10.1002/rob.21918 (cit. on p. 1).

[KK20] Ryo Kamoi and Kei Kobayashi. “Why Is the Mahalanobis Distance Effective for Anomaly Detection?” In: *CoRR* abs/2003.00402 (2020). arXiv: 2003.00402 (cit. on pp. 11, 39).

[Kok+20] Narine Kokhlikyan et al. “Captum: A Unified and Generic Model Interpretability Library for PyTorch”. In: (2020). arXiv: 2009.07896 [cs.LG] (cit. on p. 48).

[LC20] Julia Lust and Alexandru Paul Condurache. “GraN: An Efficient Gradient-Norm Based Detector for Adversarial and Misclassified Examples”. In: *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020.* 2020, pp. 7–12 (cit. on p. 12).

[Pat+20] Lucrezia Patruno et al. “A Review of Computational Strategies for Denoising and Imputation of Single-Cell Transcriptomic Data”. In: *Briefings in Bioinformatics* 22.4 (Oct. 2020). ISSN: 1477-4054. DOI: 10.1093/bib/bbaa222. eprint: https://academic.oup.com/bib/article-pdf/22/4/bbaa222/39136488/patruno\textbackslash_et\textbackslash_al\textbackslash_sm\textbackslash_bbaa222.pdf (cit. on pp. 4, 97).

[SR20] Thiago Serra and Srikumar Ramalingam. “Empirical Bounds on Linear Regions of Deep Rectifier Networks”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020.* AAAI Press, 2020, pp. 5628–5635 (cit. on p. 18).

[Sot+20] Angelo Sotgiu et al. “Deep Neural Rejection against Adversarial Examples”. In: *EURASIP J. Inf. Secur.* 2020 (2020), p. 5. DOI: `10.1186/s13635-020-00105-y` (cit. on p. 11).

[Tra+20] Florian Tramèr et al. “On Adaptive Attacks to Adversarial Example Defenses”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual.* Ed. by Hugo Larochelle et al. 2020 (cit. on p. 50).

[Abu+21] Ahmed Abusnaina et al. “Adversarial Example Detection Using Latent Neighborhood Graph”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* Oct. 2021, pp. 7687–7696 (cit. on p. 11).

[And+21] McKane Andrus et al. “What We Can’t Measure, We Can’t Understand: Challenges to Demographic Data Procurement in the Pursuit of Fairness”. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency.* FAccT ’21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 249–260. ISBN: 978-1-4503-8309-7. DOI: `10.1145/3442188.3445888` (cit. on p. 9).

[Bac+21] Jonathan Bac et al. “Scikit-Dimension: A Python Package for Intrinsic Dimension Estimation”. In: *Entropy. An International and Interdisciplinary Journal of Entropy and Information Studies* 23.10 (2021). ISSN: 1099-4300. DOI: `10.3390/e23101368` (cit. on pp. 70, 75).

[BMN21] Robert J. N. Baldock, Hartmut Maennel, and Behnam Neyshabur. “Deep Learning through the Lens of Example Difficulty”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual.* Ed. by Marc’Aurelio Ranzato et al. 2021, pp. 10876–10889 (cit. on p. 10).

[Cra+21] Francesco Craighero et al. “Unity Is Strength: Improving the Detection of Adversarial Examples with Ensemble Approaches”. In: (2021) (cit. on pp. 4, 11, 33, 91, 92).

[Gho+21] Soumya Ghosh et al. “Uncertainty Quantification 360: A Holistic Toolkit for Quantifying and Communicating the Uncertainty of AI”. In: (2021). arXiv: `2106.01410 [cs.AI]` (cit. on pp. 67, 74).

[HW21] Eyke Hüllermeier and Willem Waegeman. “Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods”. In: *Machine Learning* 110.3 (2021), pp. 457–506 (cit. on pp. 12, 69, 71, 72).

[Jia+21] Ziheng Jiang et al. "Characterizing Structural Regularities of Labeled Data in Overparameterized Models". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 5034–5044 (cit. on p. 10).

[Jum+21] John Jumper et al. "Highly Accurate Protein Structure Prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589 (cit. on p. 1).

[Kau+21] Ramneet Kaur et al. "Detecting OODs as Datapoints with High Uncertainty". In: *CoRR* abs/2108.06380 (2021). arXiv: `2108.06380` (cit. on p. 35).

[Lia+21] Bin Liang et al. "Detecting Adversarial Image Examples in Deep Neural Networks with Adaptive Noise Reduction". In: *IEEE Trans. Dependable Secur. Comput.* 18.1 (2021), pp. 72–85. DOI: `10.1109/TDSC.2018.2874243` (cit. on p. 11).

[Ma+21] Xingjun Ma et al. "Understanding Adversarial Attacks on Deep Learning Based Medical Image Analysis Systems". In: *Pattern Recognition* 110 (2021), p. 107332. DOI: `10.1016/j.patcog.2020.107332` (cit. on p. 93).

[Mac+21] Jeaneth Machicao et al. "On the Use of Topological Features of Metabolic Networks for the Classification of Cancer Samples". In: *CURRENT GENOMICS* 22.2 (2021), pp. 88–97. DOI: `10.2174/1389202922666210301084151` (cit. on pp. 4, 118).

[Mor+21] Pieter Moris et al. "Current Challenges for Unseen-Epitope TCR Interaction Prediction and a New Perspective Derived from Image Classification". In: *Briefings in Bioinformatics* 22.4 (July 2021), bbaa318. ISSN: 1467-5463, 1477-4054. DOI: `10.1093/bib/bbaa318` (cit. on p. 77).

[Pat+21] Lucrezia Patruno et al. "Combining Multi-Target Regression Deep Neural Networks and Kinetic Modeling to Predict Relative Fluxes in Reaction Systems". In: *Information and Computation* 281 (Dec. 2021), p. 104798. ISSN: 0890-5401. DOI: `10.1016/j.ic.2021.104798` (cit. on pp. 4, 129).

[Pet+21] Vitali Petsiuk et al. "Black-Box Explanation of Object Detectors via Saliency Maps". In: *Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on p. 93).

[Pop+21] Phillip Pope et al. "The Intrinsic Dimension of Images and Its Impact on Learning". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021 (cit. on pp. 69–71, 81).

[Rag+21] Jayaram Raghuram et al. “A General Framework For Detecting Anomalous Inputs to DNN Classifiers”. In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, July 2021, pp. 8764–8775 (cit. on pp. 11, 56, 95).

[Ruf+21] Lukas Ruff et al. “A Unifying Review of Deep and Shallow Anomaly Detection”. In: *Proc. IEEE* 109.5 (2021), pp. 756–795. DOI: 10.1109/JPROC.2021.3052449 (cit. on pp. 1, 2, 68).

[Shu+21] Yang Shu et al. “Open Domain Generalization with Domain-Augmented Meta-Learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 9624–9633. DOI: 10.1109/CVPR46437.2021.00950 (cit. on p. 9).

[Var+21] Raj Vardhan et al. “ExAD: An Ensemble Approach for Explanation-Based Adversarial Detection”. In: arXiv:2103.11526 (Mar. 2021). arXiv: 2103.11526 [cs] (cit. on pp. 11, 35, 44, 48).

[WBR21] Anna Weber, Jannis Born, and María Rodriguez Martínez. “TITAN: T-Cell Receptor Specificity Prediction with Bimodal Attention Networks”. In: *Bioinformatics (Oxford, England)* 37.Supplement_1 (2021), pp. i237–i244 (cit. on pp. 4, 69, 76–78, 89, 93).

[ZZ21] Fei Zuo and Qiang Zeng. “Exploiting the Sensitivity of L2 Adversarial Examples to Erase-and-Restore”. In: *ASIA CCS ’21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June 7-11, 2021*. Ed. by Jiannong Cao et al. ACM, 2021, pp. 40–51. DOI: 10.1145/3433210.3437529 (cit. on p. 11).

[ADH22] Chirag Agarwal, Daniel D’souza, and Sara Hooker. “Estimating Example Difficulty Using Variance of Gradients”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10368–10378 (cit. on pp. 2, 10, 16, 93).

[Cai+22] Michael Cai et al. “ATM-TCR: TCR-Epitope Binding Affinity Prediction Using a Multi-Head Self-Attention Model”. In: *Frontiers in Immunology* 13 (2022). ISSN: 1664-3224 (cit. on p. 77).

[Ji+22a] Xu Ji et al. *Test Sample Accuracy Scales with Training Sample Density in Neural Networks*. July 2022. DOI: 10.48550/arXiv.2106.08365. arXiv: 2106.08365 [cs, stat] (cit. on pp. 10, 18, 93, 95).

[Ji+22b] Yuanfeng Ji et al. “DrugOOD: Out-of-Distribution (OOD) Dataset Curator and Benchmark for AI-Aided Drug Discovery – a Focus on Affinity Prediction Problems with Noise Annotations”. In: *CoRR* (2022). DOI: 10.48550/ARXIV.2201.09637 (cit. on pp. 9, 94).

[KHS22] Sara Kaviani, Ki Jin Han, and Insoo Sohn. "Adversarial Attacks and Defenses on AI in Medical Imaging Informatics: A Survey". In: *Expert Systems with Applications* 198 (July 2022), p. 116815. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.116815 (cit. on p. 93).

[KKL22] Daniel Kienitz, Ekaterina Komendantskaya, and Michael Lones. "The Effect of Manifold Entanglement and Intrinsic Dimensionality on Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.7 (June 2022), pp. 7160–7167. ISSN: 2374-3468. DOI: 10.1609/aaai.v36i7.20676 (cit. on pp. 70, 71, 81, 85, 89, 95).

[Roy+22] Abhijit Guha Roy et al. "Does Your Dermatology Classifier Know What It Doesn't Know? Detecting the Long-Tail of Unseen Conditions". In: *Medical Image Anal.* 75 (2022), p. 102274. DOI: 10.1016/j.media.2021.102274 (cit. on p. 9).

[Sap+22] Nicolae Sapoval et al. "Current Progress and Open Challenges for Applying Deep Learning across the Biosciences". In: *Nature Communications* 13.1 (Apr. 2022), p. 1728. ISSN: 2041-1723. DOI: 10.1038/s41467-022-29268-7 (cit. on p. 1).

[Sun+22] Yiyou Sun et al. "Out-of-Distribution Detection with Deep Nearest Neighbors". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 20827–20840 (cit. on p. 74).

[Yan+22] Jingkang Yang et al. "Generalized Out-of-Distribution Detection: A Survey". In: (Aug. 2022). DOI: 10.48550/arXiv.2110.11334. arXiv: 2110.11334 [cs] (cit. on pp. 1, 2, 7–9, 11, 69, 94).

[Moh+23] Sina Mohseni et al. "Taxonomy of Machine Learning Safety: A Survey and Primer". In: *ACM Comput. Surv.* 55.8 (2023), 157:1–157:38. DOI: 10.1145/3551385 (cit. on pp. 1, 2, 7, 12, 13).

[Hea+] Tim Head et al. *Scikit-Optimize: Sequential Model-Based Optimization in Python.* DOI: 10.5281/zenodo.1157319 (cit. on p. 40).