Università degli Studi di Trieste

Dipartimento di Scienze della Terra, Fluidodinamica e Matematica. Interazioni e Metodiche.

Settore scientifico INF-01

Ph.D. Thesis

Robustness and Interpretability of Neural Networks' Predictions under Adversarial Attacks

Ginevra Carbone [illegible] Stefano Maset

Candidata: **Ginevra Carbone**

Supervisore: **Prof. Luca Bortolussi**

Coordinatore: **Prof. Stefano Maset**

ANNO ACCADEMICO 2021-2022

Abstract

Deep Neural Networks (DNNs) are powerful predictive models, exceeding human capabilities in a variety of tasks. They learn complex and flexible decision systems from the available data and achieve exceptional performances in multiple machine learning fields, spanning from applications in artificial intelligence, such as image, speech and text recognition, to the more traditional sciences, including medicine, physics and biology. Despite the outstanding achievements, high performance and high predictive accuracy are not sufficient for real-world applications, especially in safety-critical settings, where the usage of DNNs is severely limited by their black-box nature. There is an increasing need to understand how predictions are performed, to provide uncertainty estimates, to guarantee robustness to malicious attacks and to prevent unwanted behaviours.

State-of-the-art DNNs are vulnerable to small perturbations in the input data, known as *adversarial attacks*: maliciously crafted manipulations of the inputs that are perceptually indistinguishable from the original samples but are capable of fooling the model into incorrect predictions [66, 90, 115, 34, 37]. In this work, we prove that such brittleness is related to the geometry of the data manifold and is therefore likely to be an intrinsic feature of DNNs' predictions. This negative condition suggests a possible direction to overcome such limitation: we study the geometry of adversarial attacks in the large-data, overparameterized limit for Bayesian Neural Networks and prove that, in this limit, they are immune to gradient-based adversarial attacks. Furthermore, we propose some training techniques to improve the adversarial robustness of deterministic architectures. In particular, we experimentally observe that ensembles of NNs trained on random projections of the original inputs into lower dimensional spaces are more resilient to the attacks.

Next, we focus on the problem of interpretability of NNs' predictions in the setting of *saliency-based explanations* [8]. We analyze the stability of the explanations under adversarial attacks on the inputs [63, 85, 4, 184] and we prove that, in the large-data and overparameterized limit, Bayesian interpretations are more stable than those provided by deterministic networks. We validate this behaviour in multiple experimental settings in the finite data regime.

Finally, we introduce the concept of adversarial perturbations of amino acid sequences for protein Language Models (LMs). Deep Learning models for protein structure prediction, such as AlphaFold2 [81], leverage *Transformer architectures* and their attention mechanism to capture structural and functional properties of

amino acid sequences. Despite the high accuracy of predictions, biologically small perturbations of the input sequences, or even single point mutations, can lead to substantially different 3D structures [80]. On the other hand, protein language models are insensitive to mutations that induce misfolding or dysfunction (e.g. missense mutations [23]). Precisely, predictions of the 3D coordinates do not reveal the structure-disruptive effect of these mutations. Therefore, there is an evident inconsistency between the biological importance of mutations and the resulting change in structural prediction. Inspired by this problem, we introduce the concept of adversarial perturbation of protein sequences in continuous embedding spaces of protein language models. Our method relies on attention scores to detect the most vulnerable amino acid positions in the input sequences. *Adversarial mutations* are biologically diverse from their references and are able to significantly alter the resulting 3D structures.

Contents

Chapter 1

Introduction

Deep Neural Networks (DNNs) are the core engine of the modern AI revolution. The growth in computational power and the increasing amount of available data are major contributors to the rapid development of deep learning algorithms: their universal approximation capabilities, coupled with advances in hardware and training algorithms, result in remarkably strong predictive performance in a variety of fields, from computer vision to natural language and bioinformatics. A few applications include medical diagnostics, image and speech recognition, artificial text generation, fraud and anomaly detection, product recommendation and customer service. Deep learning models are structured in multiple processing layers that can learn complex relationships between variables and reveal new statistical properties of the training data.

DNNs are an essential component towards the construction of AI-based technologies, yet many of the mathematical structures underlying their success are still poorly understood. Moreover, their usage is tempered with several drawbacks, which are somehow the natural flip side of dealing with extremely flexible and complex models: e.g. security issues in real-world tasks, bias induced by the learning process and lack of comprehension about the motivations behind predictions. In this thesis, we attempt to address a few of such weaknesses, which we briefly introduce in the next paragraphs.

Vulnerability to Adversarial Attacks Well-chosen infinitesimal changes in the inputs of deep learning systems can produce substantial changes in the outputs, leading to paradoxical predictions or unwanted behaviours [66, 156], with potentially serious consequences in any application to safety-critical systems. These small perturbations are known as *adversarial attacks* and are specifically designed to induce wrong decisions with high predictive confidence, harming even the most accurate DNNs. Adversarial perturbations are often unrecognisable by humans [90, 35], thus developing suitable defence strategies is crucial, especially in machine vision applications (e.g. traffic sign recognition for autonomous driving [54]).

Lack of Interpretability Another fundamental limitation to the usage of DNNs is their black-box nature, resulting in a lack of interpretability. Deep learning models achieve exceptional performances, but they do not provide any intuition about the possible explanations underlying the decisions. In several fields of application, such as medical imaging [54], there is an increasing need for humans to understand how the inputs are processed by the models to trust their predictions. Industries and governments are beginning to discuss the problem of the trustworthiness of AI systems. The European Union, for instance, even released a checklist of desired requirements for machine learning development[1]. Increased transparency in the decision-making process performed by DNNs would result in the growing usage of such algorithms, especially for sensitive applications like data-driven healthcare [126].

Limitations of protein structure prediction Lastly, we would like to touch on an important biological application of DNNs, i.e. the prediction of the three-dimensional structure of a protein from its primary sequence. Also in this setting, we analyse the robustness of structure prediction models to our notion of *adversarial mutations* on the input sequences. Thanks to the recent advances in sequencing, most of the naturally occurring proteins can be represented as sequences of amino acids. Understanding how such sequences fold in a 3D structure has substantial implications for human health, especially in drug design and in the development of disease therapies. It has been observed that state-of-the-art models for structure prediction are unable to predict the structural effect (e.g. misfolding) caused by the mutation of a few amino acids in the native sequence. Therefore, it is of fundamental importance to guarantee that the biological significance of mutations reflects on the predicted protein structures.

1.1 Thesis Contributions

Here we summarise the main contributions of this work to the aforementioned issues. In doing so, we introduce more technical terminology, which will be clarified in the next chapters. For ease of readability, we report notation, abbreviations and acronyms used throughout this work at the end of the manuscript.

1.1.1 Adversarial Robustness of Bayesian Neural Networks

We show a remarkable property of Bayesian Neural Networks: in a suitably defined large data limit, we prove that the gradients of the expected loss function of an infinitely-wide BNN with respect to the input vanish. Our analysis shows that adversarial attacks for highly accurate architectures arise from the low dimensional support of the data-generating distribution. By averaging over nuisance dimensions, BNNs achieve zero expected gradients of the loss and are thus provably immune to gradient-based adversarial attacks. Specifically, we

[1]`https://altai.insight-centre.org/`

first show that, for any neural network achieving zero loss, adversarial attacks arise in directions orthogonal to the data manifold. Then, we rely on the submanifold extension lemma [94] to show that in the limit of infinitely-wide layers, for any neural network and any weight set there exists another weight set achieving the same loss and with opposite loss gradients orthogonal to the data manifold on a given point. Under the assumption of infinitely many data and flat (i.e., uninformative) prior, we then show that by averaging over these sets of weights the expectation of the gradient w.r.t. the posterior distribution of a BNN vanishes. Crucially, our results guarantee that, in the limit, Bayesian posteriors are provably robust to gradient-based adversarial attacks, even though each neural network sampled from the posterior is still vulnerable.

We experimentally support our theoretical findings on various BNN architectures trained with Hamiltonian Monte Carlo (HMC) and with Variational Inference (VI) on MNIST, Fashion MNIST and the half-moon data set, empirically showing that the magnitude of the gradients decreases as more samples are taken from the BNN posterior, thus confirming our results also for the finite width and finite data setting. We then explore the robustness of BNNs to adversarial attacks. In particular, we conduct a large-scale experiment on thousands of different neural networks, empirically finding that for BNNs high accuracy correlates with high robustness to gradient-based adversarial attacks, contrary to what is observed for deterministic NNs trained via standard Stochastic Gradient Descent (SGD). Finally, we also investigate the robustness of BNNs to gradient-free adversarial attacks, showing that BNNs are substantially more robust than their deterministic counterparts even in this setting.

1.1.2 Improved adversarial robustness using random projections of the inputs

In the deterministic setting, we propose two training techniques to improve the robustness of pre-trained classifiers to adversarial examples. Both methods project the input data in multiple lower-dimensional spaces, each one determined by a random selection of directions in the space. In doing so, we aim at exploiting relevant geometrical features for adversarial robustness during training. *RP-Ensemble* trains a separate classifier in each subspace, using the corresponding projections of the data. Then, it performs an ensemble classification on the original high-dimensional data. *RP-Regularizer*, instead, is a regularization term for the training objective. It combines the norm of the loss gradients, intended as a measure of vulnerability, and the expectation over random projections of the inputs. The resulting models are comparable to SOTA adversarially trained models in terms of robustness, while improving generalization capabilities and, in some settings, computational efficiency.

1.1.3 Stability of Bayesian Interpretations

We argue theoretically and empirically that the black-box nature of DNNs and their vulnerability to the attacks are interlinked, and that therefore solutions that ameliorate resilience against adversarial attacks will also lead to more stable and reliable interpretations. We work within the framework of (pixel-wise) saliency explanations, which attempt to interpret post hoc DNN decisions by assigning a relevance score to each input feature of each data point. Specifically, we use the popular Layer-wise Relevance Propagation (LRP) [8], a method to assess the contribution of each pixel to the final classification score which backpropagates the prediction in the neural network until it reaches the input, using a set of suitable propagation rules.

We introduce a novel notion of LRP robustness under adversarial attacks. As previously observed in [63, 74, 46], our results confirm that the LRP robustness of deterministic DNN predictions is remarkably low even when the adversarial attack fails to change the overall classification of the data point, i.e. that LRP interpretations are less robust than actual classifications. Considerations on the geometry of LRP [5] suggest that the observed lack of robustness might be imputable to large gradients of the prediction function in directions orthogonal to the data manifold. Here we expand on such a point of view, integrating it with a theoretical analysis in a suitably defined large-data limit [141, 49, 107] about the robustness of BNNs to gradient-based adversarial attacks. Specifically, we prove that Bayesian training of the DNNs in the large-data and overparameterized limit induces a regularizing effect which naturally builds robust explanations. We emp Wirically validate this claim on the popular MNIST and Fashion MNIST benchmarks.

1.1.4 Structural Change Induced by Adversarial Mutations

Models for structure prediction (e.g. AlphaFold2 [81]) and Transformer based protein Language Models (LMs) [138, 132, 53] can recover biological properties of proteins by processing the amino acids in a sequence as words in a sentence. Despite the remarkable capabilities of protein language models, it has been observed that biologically small perturbations in amino acid sequences manage to induce radical changes in the 3D structures [80], similarly the phenomenon of adversarial attacks in computer vision, where an imperceptible alteration in a small set of pixels induces a misclassification. Moreover, protein LMs are often unable to predict misfolding caused by biologically meaningful single-point mutations [23]. This limitation is presumably due to the scarcity of structure disruptive mutations among the available PDBs [86], which makes protein language models unable to catch all the biological features causing structural variations. We introduce a set of mutations, which we refer to as *adversarial mutations*, whose goal is to: (1) alter a small subset of residues in the original sequences; (2) produce perturbations that are biologically distant from the native sequences (specifically, we use BLOSUM distance, presented in Section 7.2, as a metric of biological similarity); (3) induce misfolding with respect to wild-type reference structures. Such mutations could then be used to augment the available training datasets and to make structure prediction LMs more sensitive to disruptive mutations.

1.2 Thesis Structure

The content of this thesis is structured as follows.

In Chapter 3 we present the basic theoretical notions needed to understand our contributions. We start with the definition of deep neural network (Section 3.2) and clarify the notion of training over a dataset. We then generalise the definition of DNN to infinitely-wide architectures and report a fundamental theorem behind our analysis of the geometrical properties of the data manifold, namely the universal approximation theorem (Section 3.2.3). In Section 3.3 we introduce the reader to Bayesian neural networks, to some approximate inference techniques - precisely Variational Inference (Section 3.3.1) and Hamiltonian Monte Carlo (Section 3.3.2) - and to uninformative priors (Section 3.3.3). Next, we describe adversarial attacks (Section 3.4) and adversarial robustness, also in the Bayesian setting. In Section 3.5 we introduce saliency explanations, with a focus on layer-wise relevance propagation (Section 3.5.1). Finally, in Section 3.6 we provide a few fundamental notions about protein language models, from elementary concepts in structural biology (Section 3.6.2) to state of the art models for structure prediction (Section 3.6.3).

We follow up with our contributions.

In Chapter 4 we examine the robustness of Bayesian NNs to adversarial attacks. We first show that for highly accurate neural networks the gradient of the loss function is non-zero only in the orthogonal components w.r.t. the data manifold (Section 4.1). Then, in Section 4.1.1 we prove that for any neural network and weight set there exists another weight set for the same architecture, with the same loss and opposite orthogonal gradients to the data manifold. By averaging over these weight sets we prove that for BNNs the expected gradient of the loss is zero (Section 4.2), thus making them robust to adversarial attacks. Experiments in Section 4.3 show that BNNs are more robust to both gradient-based and gradient-free attacks than their deterministic counterparts and can resist the well-known accuracy/robustness trade-off.

Chapter 5 presents the two training techniques based on random projections of the data samples: RP-Ensemble model in Section 5.1 and RP-Regularizer in Section 5.2. Our assumptions are empirically supported by the experiments in Section 5.4, showing an improvement in adversarial robustness under multiple attack strategies, compared to the adversarially trained architectures.

Chapter 6 extends the analysis to the stability of saliency-based explanations under adversarial attacks. In Section 6.1 we introduce a novel robustness metric for LRP heatmaps subject to adversarial perturbations on the inputs (Section 6.1). We offer a theoretical analysis of the effects of saliency attacks on the interpretations and describe the improvements offered by a Bayesian treatment in Section 6.2. Based on the previous discussion from Chapter 4, we prove that, in the infinitely-wide layers and large data limit, Bayesian LRP heatmaps only preserve the components that are tangent to the data manifold, thus improving LRP robustness. Empirical results in Section 6.3 confirm our theoretical findings.

In Chapter 7 we present our adversarial mutations. We start by clarifying our strategy for the selection of target positions (Section 7.1.1) and target residues at the chosen positions (Section 7.1.2). Experiments in Section 7.3 show the impact of adversarial mutations on several evaluation metrics for biological and structural similarity, also comparing them to a database of known destabilizing mutations.

Finally, Chapter 8 discusses consequences, limitations and future directions of our results.

Chapter 2

Acknowledgements

This work is partially supported by the PRIN project "SEDUCE" n. 2017TWR-CNB and by AREA Science Park supercomputing platform ORFEO.

Thesis publications The content of this thesis builds on the following publications:

[27] Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks. *Advances in Neural Information Processing Systems*, 33:15602–15613, 2020.

[28] Ginevra Carbone, Guido Sanguinetti, and Luca Bortolussi. Random projections for improved adversarial robustness. In 2021 *International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2021. doi: 10.1109/I-JCNN52387.2021.9534346.

[29] Ginevra Carbone, Luca Bortolussi, and Guido Sanguinetti. Resilience of bayesian layer-wise explanations under adversarial attacks. In 2022 *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. doi: 10.1109/IJCNN55064.2022.9892788.

[30] Ginevra Carbone, Francesca Cuturello, Luca Bortolussi, and Alberto Cazzaniga. Adversarial attacks on protein language models. *Proceedings of the 17th Machine Learning in Computational Biology meeting (MLCB)*, 2022

Other contributions Here we list the additional contributions which were not included in this thesis:

[26] Ginevra Carbone and Gabriele Sarti. ETC-NLG: End-to-End Topic-Conditioned Natural Language Generation. *IJCoL. Italian Journal of Computational Linguistics*, 6(6-2):61–77, 2020

[25] Francesca Cairoli, Ginevra Carbone, and Luca Bortolussi. Abstraction of Markov Population Dynamics via Generative Adversarial Nets. In *International*

Conference on Computational Methods in Systems Biology, pages 19–35. Springer, 2021.

[19] Luca Bortolussi, Francesca Cairoli, Ginevra Carbone, Francesco Franchina, and Enrico Regolin. Adversarial Learning of Robust and Safe Controllers for Cyber-Physical Systems. *IFAC-PapersOnLine*, 54(5):223–228, 2021

[20] Luca Bortolussi, Francesca Cairoli, Ginevra Carbone, and Paolo Pulcini. Stochastic Variational Smoothed Model Checking. *arXiv preprint arXiv:2205.05398*, 2022.

[21] Luca Bortolussi, Ginevra Carbone, Luca Laurenti, Andrea Patane, Guido Sanguinetti, and Matthew Wicker. On the Robustness of Bayesian Neural Networks to Adversarial Attacks. *arXiv preprint arXiv:2207.06154*, 2022

Chapter 3

Background

Contents

Cognitive capabilities of the brain enable humans to process objects, memorize informations and perform decision making tasks despite the high complexity of the real world. We hardly realize how challenging it is to solve such problems by means of a computer program. Indeed, Machine Learning (ML) models try to reproduce our cognitive system without stating an explicit set of instructions to reach a certain goal, as it would be an unfeasible problem. Such algorithms typically learn to solve predictive tasks directly from the observed data or from past experience. They are *black box* models, meaning that they solve prediction problems without providing an analytical description of the relationship between the inputs and the outputs.

In what follows we clarify the main concepts in machine learning used throughout this work, starting from the basic principles and ending with a few fundamental models and learning paradigms. In doing so, we often rely on the example of image recognition to give a practical intuition of such concepts.

3.1 Learning from the Data Manifold

A *dataset* is a collection of numerical or categorical informations used by an ML system to learn how to solve a predictive task. It is usually split in two parts: the algorithm learns the statistical properties of the observed data from the *training set*, while the *test set* is used to evaluate the performances of the learned model on new data. Additionally, a *validation set* can be used to test the performances during the learning phase. The process of learning from training data is known as *training algorithm.* A single entry from a dataset is called a *data sample* and is composed by a set of *features*, which are analysed by the ML model to extract higher level informations and to acquire the statistical understanding that is needed to solve the prediction problem. A data sample may also include a *target*, i.e. the desired outcome of prediction. For instance, in the case of image classification the goal is to correctly label an input image (e.g. an image of a cat) among a set of possible labels (e.g. dogs, cats, birds). In this setting, an image is a data sample, pixel values are the input features and the correct class is the target.

Machine learning models are the learned algorithms that perform the desired task. As a first approximation, they are grouped in three categories, depending on the type of information available in the training dataset: supervised learning, unsupervised learning and reinforcement learning. In *supervised learning* problems the desired outcome is available in the training set and the model has to predict it from the observed data samples. The resulting model is a *regressor* if the output is a continuous numerical value (e.g. weather prediction, sales forecasting, stock prediction) or a *classifier* if the output is categorical (e.g. image classification, spam detection, speech recognition). In *unsupervised learning* the data samples are unlabelled, and the model has to learn geometric properties or patterns from the input features. An example of unsupervised learning is clustering, where the input samples are grouped in clusters according to some similarity measure. *Reinforcement Learning* (RL), instead, solves decision-making problems by learning an optimal set of actions to perform depending on the state of the environment. Such actions are learned from previous simulations based on the scores assigned by a reward function, which penalizes suboptimal actions. Examples of applications of RL are in board games and autonomous driving.

In this work, we mainly focus on image classification, a classical supervised learning setting, and protein structure prediction, introduced in Section 3.6, where models range from self-supervised to fully supervised. More specifically, structure prediction models such as AlphaFold2 [81] and Rosetta [139] are supervised since they learn from PDB data [86] of 3D coordinates. Language models for structure prediction such as ESM-1b [138] and MSA-Transformer [132], instead, are trained on a self-supervised masked language task.

3.1.1 Manifold Hypothesis

The number of data samples required to learn a ML model grows exponentially with the dimension of the space, i.e. the number of input features. Manifold learning faces the problem of nonlinear high-dimensionality reduction by means of the *manifold hypothesis*, stating that data points lie on low-dimensional manifolds embedded in high-dimensional spaces [160, 84]. The high codimension between the data manifold and the embedding space has a fundamental role in the generation of adversarial examples, as first highlighted by [84]. They noticed that adversarial perturbations usually arise in the directions that are normal to the data manifold (Figure 3.6), hence as the codimension increases there is an increasing number of orthogonal directions and, consequently, a decrease in adversarial robustness.

Informally, a *manifold* is a topological space that locally resembles a Euclidean space, hence this formalization allows us to define local coordinates in the neighbourhood of each data point as real-valued vectors. In particular, manifolds can be covered with coordinate charts, which are continuous mappings to Euclidean spaces.

Definition 3.1 (Coordinate chart). *A coordinate chart for a topological space $\mathcal{M}$ is a homeomorphism*[1] *Φ from an open subset of $\mathcal{M}$ to an open subset of a Euclidean space.*

Moreover, manifolds are equipped with a differentiable structure, which is of fundamental importance to prove our theoretical contributions in Chapters 4 and 6. We now report a formal definition of differentiable manifold and refer the reader to [94] for further details.

Definition 3.2 (Topological manifold). *An m-dimensional topological manifold $\mathcal{M}$ is a Hausdorff and second countable topological space that is locally homeomorphic to $\mathbb{R}^m$.*

[1]A bijective and continuous function between topological spaces, with continuous inverse.

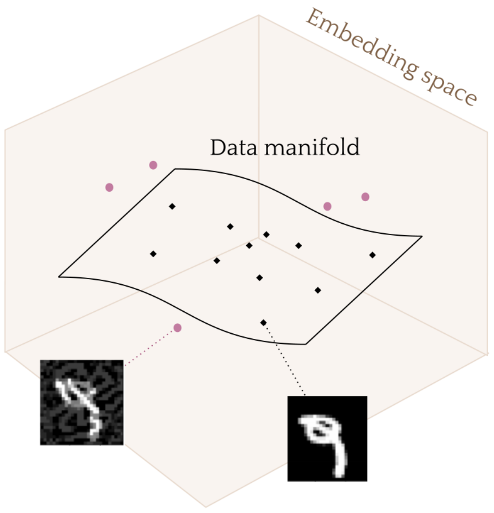


Figure 3.1: Low-dimensional manifold embedded in a high-dimensional space.

Definition 3.3 (Differentiable manifold)**.** *A $\mathcal{C}^k$-differentiable manifold is a topological manifold with an equivalence class of k-atlases on $\mathcal{M}$.*

Atlases are collections of coordinate charts such that overlapping charts are compatible. Intuitively, this means that the higher k is, the smoother $\mathcal{M}$ appears. A $\mathcal{C}^\infty$ differentiable manifold is called a *smooth manifold*. Throughout this work we often refer to *almost everywhere*[2] differentiable manifolds.

In the context of classification, the low-dimensional surfaces described by the manifold hypothesis correspond to the classification regions. Given K classes and an ambient space $\mathbb{R}^d$, the data points are sampled from K manifolds $\mathcal{M}_1, \dots, \mathcal{M}_K \subset \mathbb{R}^d$ and the whole data manifold is the union of all classification regions

$$\mathcal{M} = \bigcup_{1 \le j \le K} \mathcal{M}_j.$$

Let m be the dimension of $\mathcal{M}$, then $d - m$ is the codimension of $\mathcal{M}$ in the ambient space $\mathbb{R}^d$.

Lastly, let us define the tangent space at a point $\mathbf{x} \in \mathcal{M}$ on the data manifold.

Definition 3.4 (Tangent space)**.** *A tangent vector to an m-dimensional manifold $\mathcal{M}$ at a point $\mathbf{x} \in \mathcal{M}$ is an equivalence class of curves passing through $\mathbf{x}$ and such that their images in a chart Φ have the same tangent vector in $\mathbb{R}^m$ at $\Phi(\mathbf{x})$. The set of all tangent vectors to $\mathcal{M}$ at $\mathbf{x}$ is a real vector space called tangent space and denoted as $T_{\mathbf{x}}\mathcal{M}$.*

[2]Almost everywhere (a.e.) means everywhere except on a set with zero probability measure.

We point out that the dimension of $T_{\mathbf{x}}\mathcal{M}$ is m, the same as that of the manifold.

3.2 Deep Neural Networks

Going back to the example of image recognition, our intuition makes us decompose this problem into smaller tasks, performing what in image processing is known as *feature extraction*: we do not directly process an entire image, but first detect local shapes, edges or smaller objects, gradually building a multi-level representation of the original image. Inspired by the biological structure and functioning of the human brain, Deep Neural Networks (DNNs) are organized in multiple processing layers, structured as directed graphs (Figure 3.2). The nodes in this graph are called *neurons*, while the edges are representations of nonlinear functions, mapping a few input neurons into a single output neuron. This deeply layered structure allows DNNs to progressively extract higher-level features from the input data: the further you advance into the hidden layers, the more complex the features become, since they aggregate and recombine neurons from the previous layers. The presence of nonlinearities is necessary for the network to learn complex dependencies between the inputs and the outputs. Neurons are grouped into three categories: the *input layer* passes the input information to the hidden layers, without performing any calculation; the *hidden layers* perform all the computations involved in feature extraction and pass the results to the output layer; finally, the *output layer* outputs a complex representation of the input which is used to perform the desired task.

In our setup, we use a deep neural network f to approximate a target function $f^{true} : \mathbb{R}^d \to \mathbb{R}^K$, for some $d, K \in \mathbb{N}$. The target function is observed at points $\mathbf{x}$ drawn from a data distribution $p(D)$, whose support is the data manifold $\mathcal{M} \subset \mathbb{R}^d$. A DNN is a *parametric model*, meaning that the relationship between the inputs and the outputs depends on a set of parameters, namely the neural network's *weights*. The dimensionality of a DNN is determined by its *architecture*, i.e. by the set of mathematical formulas that define the function f. We focus on supervised learning problems, where the goal is to *fit* the parametric model $f(\cdot, \mathbf{w})$ with weights $\mathbf{w} \in \mathbb{R}^{n_{\mathbf{w}}}$ to a set of observed data points $\mathbf{x}$ with labels y. Learning the best fit to the data (or training the DNN) means searching for the most suitable set of parameters $\mathbf{w}$ to describe the relationship $f(\mathbf{x}, \mathbf{w}) = y$ between the observed $(\mathbf{x}, y)$ couples, among the set of all possible values of $\mathbf{w}$ in the parameter space $\mathbb{R}^{n_{\mathbf{w}}}$. Moreover, a high-quality model should also be able to guarantee an adequate *generalization*, that is a good performance on unseen data points. The model f depends on an additional set of fixed (non-learnable) parameters called *hyperparameters*, which could be either the number of weights and neurons in the architecture or additional parameters involved in the learning algorithm (e.g. the number of iterations, the size of a step in the parameter space, etc.).

3.2.1 Neural Network Architectures for Supervised Learning

Let $f^{true} : \mathcal{M} \to \mathbb{R}^K$ be a function defined on a smooth closed data manifold $\mathcal{M} \subset \mathcal{S} \subset \mathbb{R}^d$ with $\mathcal{S}$ being the ambient (or embedding) space. We consider the problem of approximating f^{true} via the learning of an $L+1$ layers neural network $f(\cdot, \mathbf{w})$, with $\mathbf{w} \in \mathbb{R}^{n_{\mathbf{w}}}$ being the aggregated vector of weights and biases. Formally, for $\mathbf{x} \in \mathcal{S}$, $f(\mathbf{x}, \mathbf{w})$ is defined iteratively over the number of layers as:

$$f_i^{(1)}(\mathbf{x}) = \sum_{j=1}^{d} w_{ij}^{(1)} x_j + b_i^{(1)} \tag{3.1}$$

$$f_i^{(l)}(\mathbf{x}) = \sum_{j=1}^{n_{l-1}} w_{ij}^{(l)} \phi(f_j^{(l-1)}(\mathbf{x})) + b_i^{(l)}, \quad \text{for } l = 2, \dots, L+1, \tag{3.2}$$

$$f(\mathbf{x}, \mathbf{w}) = f^{(L+1)}(\mathbf{x}), \tag{3.3}$$

for $i = 1, \dots, n_l$, where n_l is the number of neurons in the l-th layer, ϕ is a nonlinear *activation function*, which we assume to be a.e. smooth.

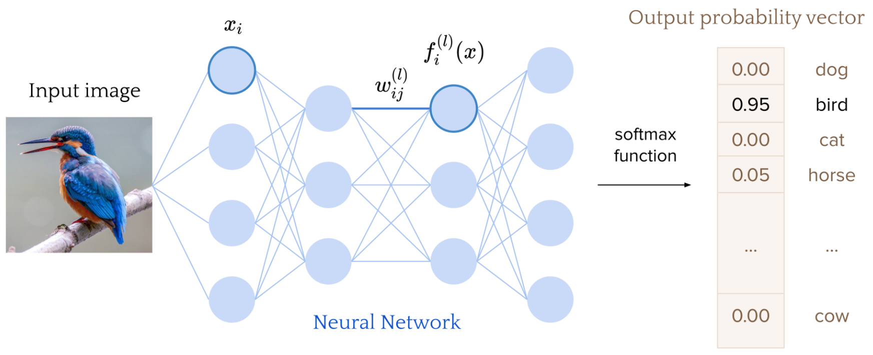


Figure 3.2: Visualization of a Neural Network architecture solving image classification.

The purpose of activation functions is to mimic the firing action of biological neurons in the brain; in practice, the presence of nonlinear activations allows neural network architectures to approximate any target smooth function on a compact set of weights, a result known as the *Universal Approximation Theorem* [40, 78]. In our experiments we mainly rely on the *ReLU* activation function $\phi(z) := \max(0, z)$, which sets a positive threshold on the input value. ReLU is an a.e. differentiable scale-invariant function, since it satisfies $\max(0, \alpha z) = \alpha \max(0, z)$ for any $\alpha > 0$. Other widely used nonlinear activations in the intermediate layers are sigmoid and tanh functions [16]. Activations in the output layer, instead, are task-dependent. For instance, typical choices in regression problems with a single output ($K = 1$ in our notation) are the linear function $\phi(z) = z$ for real-valued outputs and the sigmoid function $\phi(z) = \frac{1}{1+e^{-z}}$ for outputs in the $[0, 1]$ interval. In a classification problem with K classes, instead, the output is a probability vector, thus a reasonable choice is the softmax function

$$\phi_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}.$$

The selection of suitable activation functions in the intermediate layers and learning hyperparameters in the architecture is usually done during the *tuning phase*, by running multiple small learning trials with different combinations of hyperparameters.

3.2.2 Training via Stochastic Gradient Descent

The optimal weights of a NN $f(\cdot, \mathbf{w})$ are learned by means of a dataset $D_N = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathcal{M},\ y_i = f^{true}(\mathbf{x}_i),\ i = 1, \dots, N\}$ of size N. During training, data points are assumed to be drawn independently and identically distributed (i.i.d.) from the data manifold, to ensure that the algorithm learns from unbiased samples of the true data distribution. The discrepancy between f^{true} and $f(\cdot, \mathbf{w})$ is measured on the training dataset through a suitable *loss function* $\mathcal{L}(\mathbf{x}, \mathbf{w})$ of the form $\mathcal{L}(\mathbf{x}, \mathbf{w}) = \ell(f(\mathbf{x}, \mathbf{w}), f^{true}(\mathbf{x}))$, with $\ell(\cdot, \cdot)$ chosen accordingly to the semantic of the problem at hand.[3] Generally speaking, $\mathcal{L}(\mathbf{x}, \mathbf{w})$ is a convex function of the difference between observed and predicted values. For instance, a common choice for classification problems with softmax output activations is the *Cross-Entropy* (CE) loss function

$$\mathcal{L}(\mathbf{x}, \mathbf{w}) = -\sum_{i=1,\dots,N} y_i \log(f(\mathbf{x}_i, \mathbf{w})),$$

where $f(\mathbf{x}_i, \mathbf{w})$ is the predicted probability that $\mathbf{x}_i$ belongs to the each possible class and $y_i \in \mathbb{R}^K$ is a binary vector encoding the true class (i.e. it equals 1 on the index of the correct class and 0 otherwise).

Intuitively, minimisation of the loss function over the weight vector $\mathbf{w}$ leads to increasing fit of $f(\mathbf{x}, \mathbf{w})$ to $f^{true}(\mathbf{x})$, with zero-loss indicating that the fit is exact [17]. In practice, the problem of approximating f^{true} boils down to the problem of minimizing the loss function w.r.t. $\mathbf{w}$ in the parameter space $\mathbb{R}^{n_\mathbf{w}}$, using the training data D_N, also known as *Empirical Risk Minimization*:

$$f^{true} \approx f^* = \underset{\{f(\cdot,\mathbf{w}):\mathbf{w}\in\mathbb{R}^{n_\mathbf{w}}\}}{\arg\min} \underset{(\mathbf{x},y)\in D_N}{\mathbb{E}} [\ell(f(\mathbf{x}, \mathbf{w}), f^{true}(\mathbf{x}))].$$

This optimization problem is solved by *Gradient Descent* algorithm, where the weights $\mathbf{w}$ are initially set to a random value $\mathbf{w}_0 \in \mathbb{R}^{n_\mathbf{w}}$ and then are

[3]For simplicity of notation we omit the explicit dependence on the true function from the loss.

iteratively minimized in the direction of the gradient of the loss function (Figure 3.3)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \, \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t),$$

where

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{(\mathbf{x},y)\in D_N} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{x}, \mathbf{w})$$

denotes the expected gradient on the whole training set and the *learning rate* γ is the hyperparameter responsible for the amount of change (speed of learning) of the model at each step. For a sufficiently small step size γ and a non-pathological (convex or pseudoconvex) objective function the algorithm will *almost surely* (a.s.), i.e. with probability 1, converge to a local minimum in the space of weights [146]. This computation of the loss gradient, other than being inefficient as the number of data points N increases, leads to a noisy estimate of the true gradient. Therefore, Stochastic Gradient Descent (SGD) instead computes the gradient of the loss at each step on a random subset of the training points, known as *mini-batch*, resulting in a faster algorithm with a smoother convergence to the optimum. SGD provides a noisy estimate of the true gradient, due to the sampling noise coming from mini-batch sampling.

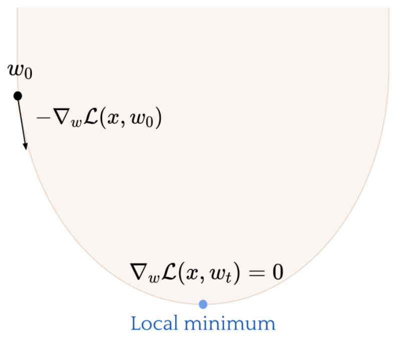

w_0
$-\nabla_w \mathcal{L}(x, w_0)$
$\nabla_w \mathcal{L}(x, w_t) = 0$
Local minimum

Figure 3.3: Loss minimization w.r.t. the weights $\mathbf{w}$.

The computation of partial derivatives of the loss is performed through the *backpropagation* algorithm, which leverages the chain rule to propagate the output loss across the neural network graph until reaching the input nodes. Precisely, referring to the notation from Section 3.2 and by omitting the explicit dependence on $\mathbf{x}$ and $\mathbf{w}$, the partial derivative of the loss in the l-th layer w.r.t. the weight w_{ij} is computed as

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} = \frac{\partial \mathcal{L}}{\partial f_i^{(l)}} \cdot \frac{\partial f_i^{(l)}}{\partial f_j^{(l-1)}} \cdot \frac{\partial f_j^{(l-1)}}{\partial w_{ij}^{(l)}}.$$

It is important to acknowledge that ReLU activations, as well as many other activations, suffer from the *vanishing gradient problem*, arising when the gradients decrease at each iteration until vanishing and resulting in large sets of weights that stop being updated by the learning algorithm. *Leaky ReLU* function solves this problem by setting a small slope in the negative part of the ReLU function

$$\phi(z) = \begin{cases} z & \text{if } z > 0 \\ 0.01\, z & \text{otherwise.} \end{cases}$$

3.2.3 Infinitely-wide Architectures and Universal Approximation Theorem

Our goal in this work is to analyse the adversarial robustness of NNs. For this purpose, we will rely on crucial results from Bayesian learning of neural networks and the properties of infinitely-wide neural networks, i.e. NNs with an infinite number of neurons.

Definition 3.5 (Infinitely-wide neural network)**.** *Consider a family of neural networks* $\{f(\cdot, \mathbf{w}_{n_\mathbf{w}}) : \mathbf{w}_{n_\mathbf{w}} \in \mathbb{R}^{n_\mathbf{w}}\}_{n_\mathbf{w}>0}$ *of Equations* (3.1)-(3.3)*, with a fixed number of neurons for* $l = 1, \ldots, L-1$ *and a variable number of neurons* n_L *in the last hidden layer. We say that*

$$f^\infty(\mathbf{x}) := \lim_{n_L \to \infty} f(\mathbf{x}, \mathbf{w}_{n_\mathbf{w}}) \tag{3.4}$$

is an infinitely wide neural network if the limit above exists $\forall \mathbf{x} \in \mathcal{S}$*, meaning that* $f(\cdot, \mathbf{w}_{n_\mathbf{w}})$ *converges to* f^∞ *pointwise, and if the resulting function defines a mapping from* $\mathcal{S}$ *to* $\mathbb{R}^K$*. Furthermore, we call* $\mathcal{F}$ *the set of such limit functions.*

The interest behind the set of infinitely-wide neural networks lies in the fact that they are universal approximators [40, 78].[4] More precisely, under the assumption that the true function f^{true} is continuous, we have that:

$$\forall \epsilon > 0,\ \exists f^* \in \mathcal{F} \text{ s.t. } \forall \mathbf{x} \in \mathcal{M},\ ||f^{true}(\mathbf{x}) - f^*(\mathbf{x})||_p < \epsilon, \tag{3.5}$$

under some p-norm. That is, $\mathcal{F}$ is dense in the space of continuous functions. Furthermore, any smooth function with bounded derivatives can be represented exactly by an infinitely wide NN with bounded weights norm (i.e. with a bounded sum of the squared Euclidean norm of the weights in the network) [122]. We will rely on these crucial properties of infinitely-wide neural networks to reason about their behaviour against adversarial attacks. For simplicity, as also common in the literature [141], throughout this work we will assume that the approximation error ϵ in Equation (3.5) is negligible, i.e. that the true function can be exactly expressed as an infinitely-wide neural network. More formally, we will assume that any smooth function $g : \mathcal{S} \to \mathbb{R}^K$ such that $g|_\mathcal{M} = f^{true}$ belongs to $\mathcal{F}$, where $|_\mathcal{M}$ denotes the restriction to the data manifold $\mathcal{M} \subset \mathcal{S}$.

[4]Notice that the limit in Definition 3.5 is taken only w.r.t. the last hidden layer. Similar results, albeit with additional care needed for the definition of the limiting sequence, can be obtained by taking the limit w.r.t. all the hidden layers [106].

We remark that, since the loss-landscape is in general non-convex, the existence of such a neural network does not necessarily imply that a gradient descent-based training will be able to retrieve it through training on a finite dataset. However, recent results [141] have shown that, in the limit of infinite points on data manifold, the loss function is a convex functional on the space of weights of an infinitely wide NN and hence the gradient flow of stochastic gradient descent will converge to a unique solution in the weight space. That is, given an infinitely wide NN f^{∞} and a sequence of datasets $\{D_N\}_N \subset \mathcal{M}$ of cardinality $N > 0$, we have that:

$$\lim_{N\to\infty} \ell\big(f^{\infty}_{D_N}(\mathbf{x}), f^{true}(\mathbf{x})\big) = 0, \quad \forall \mathbf{x} \in \mathcal{M} \tag{3.6}$$

where $f^{\infty}_{D_N}$ represents the infinitely wide NN trained on D_N until convergence.

3.3 Bayesian Neural Networks

Bayesian modelling captures the uncertainty of data-driven models by turning model parameters into random variables [10, 116]. In a nutshell, it places a prior measure[5] $p(\mathbf{w})$ over the weights $\mathbf{w}$ and models their fit to the observed data D through the likelihood $p(D|\mathbf{w})$ [16].[6] Bayesian inference then combines likelihood and prior via the Bayes' theorem [16] to obtain a *posterior* distribution over the NN's parameters

$$p\left(\mathbf{w}|D\right) \propto p\left(D|\mathbf{w}\right) p\left(\mathbf{w}\right). \tag{3.7}$$

Unfortunately, it is in general infeasible to compute the posterior distribution exactly for non-linear/non-conjugate models such as deep NNs, so approximate Bayesian inference methods are employed in practice. Asymptotically exact samples from the posterior distribution can be obtained via procedures such as Hamiltonian Monte Carlo (HMC) [117], while approximate samples can be obtained more cheaply via Variational Inference (VI) [18].

Irrespective of the posterior inference method of choice, Bayesian empirical predictions at a new input $\mathbf{x}$ are obtained from an ensemble of $M \in \mathbb{N}$ neural networks, each one with weights drawn from the learned posterior distribution $p(\mathbf{w}|D)$:

$$f\left(\mathbf{x}|D\right) = \mathbb{E}_{p(\mathbf{w}|D)}[f(\mathbf{x},\mathbf{w})] \simeq \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x},\mathbf{w}_i) \qquad \mathbf{w}_i \sim p\left(\mathbf{w}|D\right), \tag{3.8}$$

where $\mathbb{E}_{p(\mathbf{w}|D)}$ denotes the expectation w.r.t. the posterior distribution $p\left(\mathbf{w}|D\right)$.

[5] We employ the common notation of indicating density functions with p and their corresponding probability measures with P.

[6] Notice that in the Bayesian setting the likelihood is a transformation of the loss function used in deterministic settings. We use both terminologies, and the loss is not to be confused with that used in Bayesian decision theory [16].

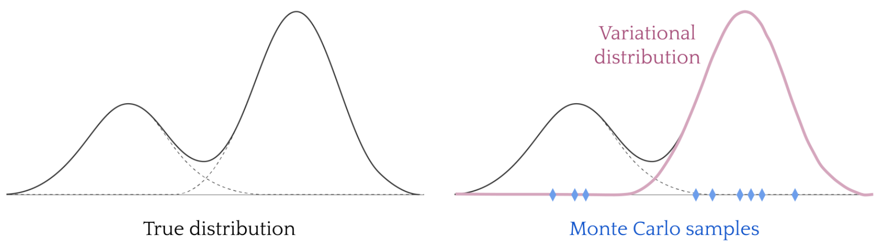


Figure 3.4: Hamiltonian Monte Carlo (sampling-based) vs Variational Inference (optimization-based) approximate inference techniques.

3.3.1 Variational Inference

Variational Inference (VI) [167] turns Bayesian inference into an optimization problem and provides an analytical approximation $q(\mathbf{w};\boldsymbol{\nu}) \approx p(\mathbf{w}|D)$ of the posterior distribution, namely the *variational distribution*, which belongs to a restricted family of known distributions (e.g. Gaussians). The variational parameters $\boldsymbol{\nu}$ are optimized by minimizing the dissimilarity between p and q. The measure of similarity is the Kullback-Leiber divergence

$$\mathrm{KL}(q||p) := -\sum_{\mathbf{w}} q(\mathbf{w};\boldsymbol{\nu}) \log\left(\frac{p(\mathbf{w}|D)}{q(\mathbf{w};\boldsymbol{\nu})}\right).$$

and the minimization problem

$$\boldsymbol{\nu}^* = \arg\min_{\boldsymbol{\nu}} \mathrm{KL}(q||p)$$

still requires the computation of the intractable term $p(\mathbf{w}|D)$. Notice that the evidence $\log p(D)$ is constant with respect to $\boldsymbol{\nu}$ and satisfies the equality

$$\log p(D) = \mathrm{KL}(q||p) + \mathbb{E}_q[\log p(\mathbf{w},D) - \log q(\mathbf{w};\boldsymbol{\nu})].$$

Therefore, the optimization problem above is equivalent to the minimization of the *Evidence Lower Bound* (ELBO) loss

$$\mathrm{ELBO}(\boldsymbol{\nu}) = \mathbb{E}_q[\log p(\mathbf{w},D) - \log q(\mathbf{w};\boldsymbol{\nu})].$$

The first term in the ELBO loss encourages q to place its probability mass on the MAP estimate, while the second favours entropy on the mass, to avoid its concentration in a single location.

3.3.2 Hamiltonian Monte Carlo

Markov Chain Monte Carlo (MCMC) [64] is a class of stochastic algorithms that allow sampling from an unknown high-dimensional probability distribution (in our case the posterior $p(\mathbf{w}|D)$) by building a Markov chain with the desired distribution as its equilibrium distribution. The efficiency of these methods in high-dimensional spaces is guaranteed by a proper exploration of the *typical set*, which is the region contributing the most to the computation of expectations w.r.t. the target density [13]. *Hamiltonian Monte Carlo* (HMC) [117] is a Markov Chain Monte Carlo technique that combines an approximate Hamiltonian dynamics simulation and a *Metropolis-Hastings* [109, 71] acceptance step: at each step, Metropolis-Hastings computes the probability of keeping the new samples $(\rho', \mathbf{w}')$

$$\min\{1, \exp(H(\rho, \mathbf{w}) - H(\rho', \mathbf{w}'))\}.$$

It is designed to generate efficient transitions in the parameter space and to reduce the problems of low acceptance rates and autocorrelation between consecutive samples. HMC introduces some momentum variables ρ and defines a Hamiltonian

$$H(\rho, \mathbf{w}) = -\log p(\rho|\mathbf{w}) - \log p(\mathbf{w}) = T(\rho|\mathbf{w}) + V(\mathbf{w}),$$

where $p(\rho, \mathbf{w})$ is the joint density $p(\rho|\mathbf{w})p(\mathbf{w})$. Starting from the initial values of ρ and $\mathbf{w}$, the system evolves according to Hamilton's equations

$$\begin{aligned} \frac{d\mathbf{w}}{dt} &= \frac{\partial H}{\partial \rho} = \frac{\partial V}{\partial \rho} \\ \frac{d\rho}{dt} &= -\frac{\partial H}{\partial \mathbf{w}} = -\frac{\partial T}{\partial \mathbf{w}} - \frac{\partial V}{\partial \mathbf{w}}, \end{aligned}$$

which are solved by means of *leapfrog integration* [77]. HMC build a chain of samples $\mathbf{w}_1, \dots, \mathbf{w}_M$, that are used to compute an empirical approximation of the posterior. In the limit of infinite samples, the distribution of the recorded samples exactly matches the posterior distribution.

3.3.3 Flat Priors

In this work, we rely on flat or uninformative priors [22, 39], that is wide priors resembling a uniform distribution. Notice that defining a distribution over the weights $p(\mathbf{w})$ naturally leads to the definition of a distribution over functions in $\mathcal{F}$, which we denote by $p(f(\cdot, \mathbf{w}))$. For simplicity of notation, in this section we consider the space $\mathcal{F}$ of real-valued continuous functions $f : \mathcal{S} \times \mathbb{R}^{n_\mathbf{w}} \to \mathbb{R}$, but the argument naturally extends to the multi-output case by treating each output value separately.

In particular, $p(f(\cdot, \mathbf{w}))$ leads to a probability measure P over the space $\mathcal{F}$, with a σ-algebra generated by sets of the form $\{f(\cdot, \mathbf{w}) \in \mathcal{F} : f(\mathbf{x}_1, \mathbf{w}) \in G_1, ..., f(\mathbf{x}_N, \mathbf{w}) \in G_N\}$ where N is an arbitrary integer, $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathcal{S}$, and $G_1, ..., G_N \subset \mathbb{R}$ are closed intervals [3]. According to the above definition of σ-algebra, any measurable set of functions $F \subseteq \mathcal{F}$ only depends on the behaviour of $f(\cdot, \mathbf{w})$ on a countable set of input points. Nevertheless, in our setting, this is not limiting. Indeed, properties involving an uncountable set of points can be reframed over a countable set of inputs by relying on the continuity of each function $f(\cdot, \mathbf{w}) \in \mathcal{F}$ [3]. Here we introduce the concept of *flat priors* over the space $\mathcal{F}$.

Definition 3.6 (Flat Prior)**.** *Let p be a prior distribution on the weight space $\mathbb{R}^{n_\mathbf{w}}$ and P the corresponding probability measure. Then, we say that p is flat if for any $N > 0$, any choice of N points $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathcal{S}$, any sequence of closed intervals $G_1, ..., G_N \subset \mathbb{R}$, and any $g_1, ..., g_N \in \mathbb{R}$ it holds that*

$$\begin{aligned}P(\{f(\cdot, \mathbf{w}) \in \mathcal{F} \,:\, f(\mathbf{x}_1, \mathbf{w}) \in G_1, ..., f(\mathbf{x}_K, \mathbf{w}) \in G_K\}) = \\ P(\{f(\cdot, \mathbf{w}) \in \mathcal{F} \,:\, f(\mathbf{x}_1, \mathbf{w}) \in G_1 + g_1, ..., f(\mathbf{x}_1, \mathbf{w}) \in G_K + g_K\})\end{aligned}$$

that is, if its finite-dimensional distributions are translation invariant.

According to Definition 3.6, a flat prior gives a uniform distribution on the outcomes of $f(\cdot, \mathbf{w})$ for any finite set of input points. Furthermore, as P is uniquely defined through its behaviour on a countable number of input points, Definition 3.6 implies that for any measurable set of functions $F \subset \mathcal{F}$, function $g \in \mathcal{F}$, and set $F_g = \{f \in \mathcal{F} : \exists f' \in F \, s.t. \, f = f' + g\}$, it holds that $P(f(\cdot, \mathbf{w}) \in F) = P(f(\cdot, \mathbf{w}) \in F_g)$, that is if we translate a set of function by a given function we obtain the same probability. Intuitively, the invariance principle implies that the prior distribution provides equal beliefs in any transformed version of the parameters. We should stress that, formally, a prior so defined is not a distribution but an *improper prior*, as the scaling factor for a translation invariant measure defined over an infinite support does not exist [22]. In practice, one can approximate a flat prior with a Gaussian distribution with a variance that tends to infinity.

Notice that while Definition 3.6 defines a prior over the space of functions induced by a neural network, in practice, it would be convenient to define a prior directly over the weights. In this context, we stress that, for an infinitely-wide BNN, a Gaussian prior with large (finite) variance over the weights induces a Gaussian prior with large variance over the space of functions [106]. Thus, assuming a prior $p(\mathbf{w})$ with large variance is in practice a good approximation of a flat prior as per Definition 3.6. In this work, we rely on flat priors in the large data setting, where the influence of a flat prior on the posterior is often limited [39]. In this context, a flat prior assumption is necessary to guarantee that all possible functions that the neural network can represent equally influence the posterior.

3.4 Adversarial Attacks

An *adversarial attack* (or *adversarial example*) against a DNN classifier f is a small perturbation $\tilde{\mathbf{x}}$ of an input point $\mathbf{x} \in \mathcal{M}$ that leads to a large change in the output prediction [66]; more precisely, a perturbation $\tilde{\mathbf{x}}$ is an attack if the prediction on $\tilde{\mathbf{x}}$ differs from the original prediction on $\mathbf{x}$.

Let us formally define an adversarial attack on the data manifold $\mathcal{M}$ with respect to a fixed p-norm $|| \cdot ||_p$ and using an *attack strength* $\epsilon \in \mathbb{R}$.

Definition 3.7 (ϵ-neighbour)**.** *The ϵ-neighbour of a manifold $\mathcal{M}$ is the set of*

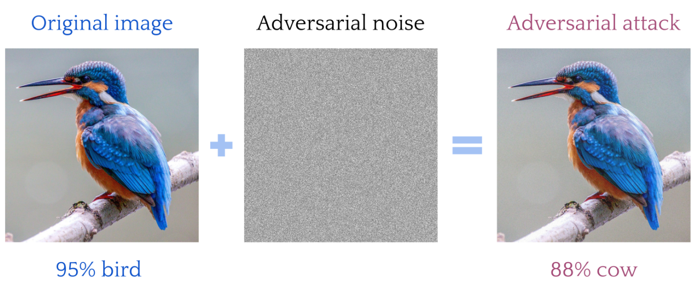


Figure 3.5: Example of misclassification induced by an adversarial attack.

points whose distance from $\mathcal{M}$ in the p-norm is lower than $\epsilon \in \mathbb{R}$:

$$\mathcal{M}_p(\epsilon) = \Big\{ \mathbf{x} \in \mathbb{R}^d : \inf_{\mathbf{x}' \in \mathcal{M}} ||\mathbf{x} - \mathbf{x}'||_p \leq \epsilon \Big\}.$$

Definition 3.8 (ϵ-adversarial attack)**.** *An ϵ-adversarial attack against a classifier $f(\cdot, \mathbf{w})$ on K classes at a point $\mathbf{x} \in \mathcal{M}$ is a perturbation $\tilde{\mathbf{x}} \in \mathcal{M}_p(\epsilon)$ such that*

$$\arg\max_{j=1,\dots,K} f_j(\mathbf{x}, \mathbf{w}) \neq \arg\max_{j=1,\dots,K} f_j(\tilde{\mathbf{x}}, \mathbf{w}).$$

As $f(\cdot, \mathbf{w})$ is a non-convex function, computing $\tilde{\mathbf{x}}$ is a non-convex optimisation problem for which several approximate solution methods have been proposed. Most attack strategies utilise information on the training loss function to detect an optimal perturbation direction, either through explicit knowledge of the loss (*white box*) or via point-wise estimations of the loss (*black box* or *query-based* attacks, generally weaker than their white box counterpart). In our theoretical discussion we primarily focus on *gradient-based attacks*, a specific category of white box attacks that employ the gradient of loss function w.r.t. $\mathbf{x}$ [14]. Our experiments include the following attacks: the gradient-based attacks Fast Gradient Sign Method [66], Projected Gradient Descent [90], DeepFool [115], Carlini and Wagner [34], all presented in the next section; and the black box attack ZOO [37], based on a finite-difference approximation of the gradient of the loss function.

We also extend the definition of adversarial attack to Bayesian architectures in Section 3.4.2 and discuss the evaluation of adversarial robustness in Section 3.4.3.

3.4.1 Gradient-based Attacks

Gradient-based attacks utilize gradient information on the training loss function $\mathcal{L}$ to determine the attack direction. They have complete knowledge of the target network $f(\cdot, \mathbf{w})$, yet they could also be effective on unknown models, thanks to their transferability properties [42]. Given an input point $\mathbf{x} \in \mathcal{M}$ and an attack

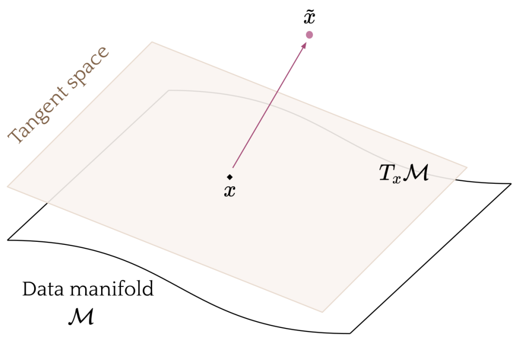


Figure 3.6: Adversarial examples usually arise in the directions that are normal to the data manifold $\mathcal{M}$.

strength $\epsilon > 0$, the worst-case adversarial perturbation can be defined as the point $\tilde{\mathbf{x}}$ in the ϵ-neighbourhood of $\mathbf{x}$ that maximises the training loss:

$$\tilde{\mathbf{x}} = \underset{\tilde{\mathbf{x}}:||\tilde{\mathbf{x}}-\mathbf{x}||_p \leq \epsilon}{\arg\max} \mathbb{E}_{p(\mathbf{w}|D)}[\mathcal{L}(\tilde{\mathbf{x}}, \mathbf{w})].$$

One of the most known gradient-based attacks is the Fast Gradient Sign Method (FGSM) [66], a computationally efficient single-step attack. FGSM takes a step of size ϵ in the direction of the sign of the gradient of the training loss function w.r.t. $\mathbf{x}$:

$$\tilde{\mathbf{x}} = \mathbf{x} + \epsilon \cdot \operatorname{sgn} \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{w}),$$

with the aim of inducing the highest change in the loss function towards a misclassification, with minimal attack strength ϵ. In the untargeted setting it does not push the misclassification towards a specific wrong class.

Projected Gradient Descent (PGD) [90] is a stronger, iterative version of FGSM. It starts from a random perturbation $\tilde{\mathbf{x}}_0$ in an ϵ-ball around $\mathbf{x}$ in the L_∞ norm, then at each iteration performs an FGSM attack with a smaller step size $\varepsilon < \epsilon$ and projects the perturbation back in the ϵ-ball:

$$\tilde{\mathbf{x}}_{t+1} = \operatorname{Clip}_{\varepsilon, L_\infty} \Big(\tilde{\mathbf{x}}_t + \varepsilon \cdot \operatorname{sgn} \nabla_{\mathbf{x}} \mathcal{L}(\tilde{\mathbf{x}}_t, \mathbf{w}) \Big).$$

The size of the final perturbation at a chosen timestep t is smaller than ϵ, resulting in a stronger attack.

DeepFool attack [115] searches for the nearest decision boundary to the data point $\mathbf{x}$ in the L_2 norm and pushes the perturbation $\tilde{\mathbf{x}}$ beyond this boundary. Specifically, it starts from the original data point $\mathbf{x}$ and iteratively minimizes the classifier f around the input point until it produces a misclassification, i.e. until

$$\underset{j=1,\dots,K}{\arg\max} f_j(\mathbf{x}_t, \mathbf{w}) \neq \underset{j=1,\dots,K}{\arg\max} f_j^{true}(\mathbf{x}_t).$$

Experiments using DeepFool show that for common classifiers almost all test samples are very close to the decision boundary, suggesting that DNNs are usually not robust to small perturbations [177].

Similarly to FGSM, Carlini & Wagner (C&W) attack [34] in the L_∞ norm searches for the minimal perturbation that causes a misclassification, that is

$$\min_{\tilde{\mathbf{x}}\in[0,1]^d} ||\tilde{\mathbf{x}}-\mathbf{x}||_\infty + \gamma\cdot\mathcal{L}(\tilde{\mathbf{x}},\mathbf{w},c)$$

where c is the target class for misclassification and $\mathcal{L}(\tilde{\mathbf{x}},\mathbf{w},c) := \max_{j\neq c}[f_j(\tilde{\mathbf{x}},\mathbf{w})-f_c(\tilde{\mathbf{x}},\mathbf{w})]$ is the margin loss, that penalizes output scores higher than $f_c(\tilde{\mathbf{x}},\mathbf{w})$ on the other classes. C&W is currently one of the strongest adversarial attacks available, but also one of the most computationally expensive.

3.4.2 Bayesian Adversarial Attacks

Bayesian adversarial attacks are crafted against a Monte Carlo approximation of the posterior distribution, that is against an ensemble of deterministic NNs sampled from the posterior. For instance, an FGSM attack with strength ϵ against a BNN with posterior distribution $p(\mathbf{w}|D)$ becomes

$$\tilde{\mathbf{x}} = \mathbf{x}+\epsilon\cdot\operatorname{sgn}\mathbb{E}_{p(\mathbf{w}|D)}\big[\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\mathbf{w})\big] \tag{3.9}$$

$$\simeq \mathbf{x}+\epsilon\cdot\operatorname{sgn}\sum_{i=1}^{M}\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\mathbf{w}_i) \qquad \mathbf{w}_i\sim p(\mathbf{w}|D). \tag{3.10}$$

Similarly, the definition of Bayesian attack can be extended to other perturbation strategies.

3.4.3 Adversarial Robustness

The robustness of classifiers to adversarial attacks is strongly related to the geometry of the learned decision boundaries since adversarially perturbed points always lie extremely close to these surfaces [50]. The evaluation of adversarial attacks can be either qualitative or quantitative. In the qualitative case, one simply observes whether an attack strategy with a certain strength is successful in changing the classification label of the given data point. Formally, adversarial robustness in an ϵ-neighbour of a data point is defined as follows.

Definition 3.9 (ϵ-adversarial robustness)**.** *A classifier $f(\cdot,\mathbf{w})$ is robust to ϵ-adversarial examples if $\mathcal{M}_p(\epsilon)$ is classified correctly.*

In practical terms, any robustness evaluation reports the fraction of successful attacks on the input dataset. The quantitative metric we use to evaluate network performances against adversarial attacks is *softmax robustness*, which computes the softmax difference between original and adversarial predictions as $1-||f(\mathbf{x})-f(\tilde{\mathbf{x}})||_\infty$ and is a real number between zero (maximal fragility) and one (complete robustness).

3.4.4 Defence Strategies

Defence research proposes a multitude of strategies to mitigate adversarial vulnerability. A first simple countermeasure against the attacks is *adversarial detection* [129], which attempts to check whether a sample is benign or malicious before feeding it to the model, but does not provide any improvement in robustness to the DNN.

Other techniques modify the input data at test time. Data compression, for instance, is an effective defence method against simpler attack techniques, like FGSM and DeepFool [52], although a large amount of compression causes a drastic decrease in test accuracy. Also resizing or adding random noise to test images sightly improves adversarial robustness [176, 168]. *Purification methods*, instead, learn generative models that are able to remove adversarial noise from the input images [181].

Robust optimization [177] refers to training techniques that improve adversarial robustness, typically leveraging some prior knowledge of the attacks. In this setting, the most popular method is *adversarial training* [89], where adversarial examples are crafted on the original model and added to the training set of a new model with the same architecture. One of the most significant limitations of adversarial training is that it protects against the chosen attack strategy, but the model can still be vulnerable to malicious perturbations in general; however, it has been shown that the transferability properties of NNs also guarantee some degree of robustness against other attacks or different architectures. Moreover, adversarial training is computationally expensive on complex attacks, such as Carlini & Wagner; for this reason, it is usually performed using FGSM attack, since it only requires one gradient evaluation at each input point. Robust optimization is also achieved by Lipschitz regularization of the loss function in *Parseval Networks* [155], whose goal is to penalize large instabilities in hidden representations of the inputs. *Provable defences*, instead, provide exact robustness bounds under a set of theoretical constraints which need to be satisfied during training [57, 177].

Gradient masking defences hide the gradient information provided by the model, which is used in most attack strategies [177]. An example is *defensive distillation* [76], which trains a smaller DNN on pre-softmax activations, also known as logits, to obtain smoother decision boundaries and less sensitivity to the adversaries; however, it is still vulnerable to some perturbations [34]. Another strategy that induces gradient masking is *pruning* [45], which randomly drops some neurons in the original architecture.

Lastly, several recent works provide empirical evidence of the robustness of Bayesian NNs [58, 182, 110, 123, 149, 161]. Earliest attempts to understand the robustness properties of BNNs appear in [11, 62]. In particular, [11] define Bayesian adversarial spheres and empirically show that, for BNNs trained with HMC, adversarial examples tend to have high uncertainty. Moreover, they show how basic ensembling strategies, like bootstrapping, are vulnerable to the attacks. Interestingly, [62] derive sufficient conditions for idealised BNNs to avoid adversarial examples; however, it is unclear how such conditions could be checked in practice, as it would require one to check that the BNN architecture is invariant under all the symmetries of the data. Because of the capabilities of BNNs to model uncertainty, which can be intuitively linked to their robustness properties, many recent works introduce both empirical [33, 95, 58, 133, 68] and formal verification methods [171, 12] to detect adversarial examples for BNNs. Other works perform adversarial training on BNNs [183, 179, 99, 172], together with robustness constraints or penalties added at training time. Empirical results obtained with such techniques highlight how, in the Bayesian settings, high accuracy and high robustness often are positively correlated with each other. In Chapter 4 we provide a theoretical and empirical analysis that further confirms the adversarial robustness of BNNs, as well as the aforementioned correlation.

3.5 Saliency Explanations

Explainable AI seeks to provide human-interpretable explanations that support DNNs' decisions. Most of the recent explanation methods provide post hoc interpretations of black box classifier, which generate visual explanations of the decisions performed on single input samples [2, 70, 104]. Among the variety of available techniques, gradient-based attribution methods rely on gradient information provided by the models to produce the explanations. We briefly mention a few of them in what follows.

[148] compute image-specific saliency maps using a single back-propagation step across the network for a chosen class. *Local Interpretable Model-agnostic Explanations* (LIME) [136] searches for the optimal explanation of a sample from a specified class of explanation models which are intrinsically interpretable [104]. The attribution method of *Integrated Gradients* [153] satisfies two specific axioms: *sensitivity* and *implementation invariance*. Sensitivity indicates that whenever an input and a baseline differ by a single feature and their predictions on that input are distinct, the attribution associated with the differing feature is non-zero. Two functionally equivalent networks satisfy implementation invariance if they associate identical attributions to the same input. Integrated gradients have been extended in several works [44, 79, 108]. *Deep learning important features* (DeepLIFT) [147] assigns attributions by comparing the activation of each neuron to a reference activation, i.e. the activation of a baseline input, which is task-dependent. It satisfies the sensitivity axiom but breaks implementation invariance [153]. *Shapley additive explanations* (SHAP) [100] generalizes all the explanation functions that can be expressed as a linear combination of binary variables, including LIME and DeepLIFT.

In this work we focus on Layer-wise Relevance Propagation (LRP) explanation method, introduced in Section 3.5.1, in the context of image classification. The reason for this choice is that LRP scales well with highly complex DNN architectures and can be easily extended to the Bayesian setting.

3.5.1 Layer-wise Relevance Propagation

Let $f : \mathcal{M} \times \mathbb{R}^{n_\mathbf{w}} \to \mathbb{R}^K$ be an image classifier on K classes, where $\mathcal{M}$ is the data manifold, $\mathbb{R}^{n_\mathbf{w}}$ is the space of learnable weights $\mathbf{w}$ and the vector $f(\mathbf{x}, \mathbf{w})$ denotes the probability that an image $\mathbf{x} \in \mathcal{M}$ belongs to each one of the possible classes. The idea of pixel-wise decomposition of a given image $\mathbf{x}$ is to understand how its pixels contribute to the prediction $f(\mathbf{x}, \mathbf{w})$. In particular, LRP associates with each pixel x_p a relevance score $R(x_p, \mathbf{w})$, which is positive when the pixel contributes positively to the classification, negative when it contributes negatively to the classification and zero when it has no impact on the classification. All the relevance scores for a given image $\mathbf{x}$ can be stored in a heatmap $R(\mathbf{x}, \mathbf{w}) = \{R(x_p, \mathbf{w})\}_p$, whose values quantitatively explain not only whether pixels contribute to the decision, but also by which extent.

One can leverage suitable propagation rules to ensure that the network output is fully redistributed through the network, namely that the relevance heatmap catches all the saliency features from the inputs [144]. For this purpose, the heatmap should be *conservative*, i.e. the sum of the assigned relevance values should correspond to the total relevance detected by the model: $f(\mathbf{x}, \mathbf{w}) = \sum_p R(x_p, \mathbf{w})$. In the multi-label setting $R(\mathbf{x}, \mathbf{w})$ is the heatmap associated with the correct classification label. Although the conservative property is not required to define a relevance heatmap, it has been empirically observed that conservative rules better support classification [144, 113]. Several propagation rules satisfy the conservative property, each of them leading to different relevance measures. In the next section, we report three practical propagation rules: the Epsilon rule, the Gamma rule and the Alpha-Beta rule.

[8] also presented LRP using a functional approach, i.e. independently of the network's topology. Then, [113] used *deep Taylor decomposition* to express any rule-based approach under the functional setting. Their method builds on the standard first-order Taylor expansion of a non-linear classifier at a chosen *root point* $\mathbf{x}^*$, i.e. a point on the data manifold such that $f_j(\mathbf{x}^*) = 0$ on the true class $j \in \{1, \dots, K\}$ of $\mathbf{x}$,

$$\begin{aligned} f(\mathbf{x}, \mathbf{w}) &= f(\mathbf{x}^*, \mathbf{w}) + \nabla_\mathbf{x} f(\mathbf{x}^*, \mathbf{w}) \cdot (\mathbf{x} - \mathbf{x}^*) + \gamma \\ &= \sum_{x_p} \frac{\partial f}{\partial x_p}\Big|_{\mathbf{x}=\mathbf{x}^*} \cdot (x_p - x_p^*) + \gamma, \end{aligned} \tag{3.11}$$

where γ denotes higher-order terms. The root point $\mathbf{x}^*$ represents a neutral image which is similar to $\mathbf{x}$, but does not influence classification, i.e. whose relevance is everywhere null. The nearest root point to the original image $\mathbf{x}$ can be obtained by solving an iterative minimization problem [113]. The resulting LRP heatmap is $R(\mathbf{x}, \mathbf{w}) = \nabla_\mathbf{x} f(\mathbf{x}^*, \mathbf{w}) \cdot (\mathbf{x} - \mathbf{x}^*)$.

In a similar way, it is possible to introduce a concept of *Bayesian explanations* for BNN predictions. Since the relevance score assigned to an input feature depends on the NN's weights, in the Bayesian setting it becomes a random variable and the LRP heatmap is computed in expectation under the posterior distribution over the weights:

$$\begin{aligned}R(\mathbf{x},\mathbf{w}) &= \mathbb{E}_{p(\mathbf{w}|D)}[\nabla_{\mathbf{x}} f(\mathbf{x}^*,\mathbf{w})] \cdot (\mathbf{x}-\mathbf{x}^*) \\ &= \sum_{i=1}^{M} \nabla_{\mathbf{x}} f(\mathbf{x}^*,\mathbf{w}_i) \cdot (\mathbf{x}-\mathbf{x}^*) \qquad \mathbf{w}_i \sim p(\mathbf{w}|D).\end{aligned}$$

3.5.2 LRP Rules

Following the notation in Section 3.2, for ease of reading let us denote a relevance heatmap $R(\mathbf{x},\mathbf{w})$ as R and the i-th activation at the l-th layer $f_i^{(l)}(\mathbf{x})$ during the forward pass as a_i^l. We recall that $w_{ij}^{(l)}$ is the weight connecting the neuron a_j^l to all neurons a_i^{l-1} in the previous layer, that is for $i = 1, \ldots, n_{l-1}$.

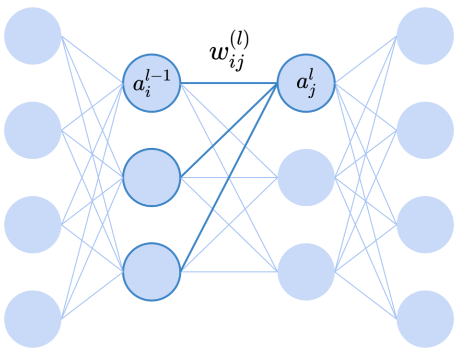


Figure 3.7: Visualization of a Neural Network architecture showing the indexing of neurons and weights.

A practical example of propagation rule is the *Epsilon rule* (ε-LRP) [114]. The computation starts with the relevance $R_j^L = 1$ attributed to the output classification neuron a_j^L, that is backpropagated through all the connected neurons until reaching the input features x_p. The resulting ε-LRP score for a chosen $\varepsilon > 0$ and neuron index $i = 1, \ldots, n_{l-1}$ at layer $l-1$ amounts to

$$R_i^{l-1} = \sum_{j=1}^{n_l} \frac{a_i^l w_{ij}^{(l)}}{\varepsilon + \sum_{k=0}^{n_{l-1}} a_k^l w_{kj}^{(l)}} R_j^l.$$

The *Gamma rule* (γ-LRP) favours positive contributions over negative contributions by a factor of γ. The score for a chosen $\gamma > 0$ is

$$R_i^{l-1} = \sum_{j=1}^{n_l} \frac{a_i^l \cdot \left(w_{ij}^{(l)} + \gamma \cdot \max\{0, w_{ij}^{(l)}\}\right)}{\sum_{k=0}^{n_{l-1}} a_k^l \cdot \left(w_{kj}^{(l)} + \gamma \cdot \max\{0, w_{kj}^{(l)}\}\right)} R_j^l.$$

Finally, the *Alpha-Beta rule* ($\alpha\beta$-LRP) computes

$$R_i^{l-1} = \sum_{j=1}^{n_l} \left(\alpha \cdot \frac{\max\{0, a_i^l w_{ij}^{(l)}\}}{\sum_{k=0}^{n_{l-1}} \max\{0, a_k^l w_{kj}^{(l)}\}} - \beta \cdot \frac{\min\{0, a_i^l w_{ij}^{(l)}\}}{\sum_{k=0}^{n_{l-1}} \min\{0, a_k^l w_{kj}^{(l)}\}} \right) R_j^l,$$

where the conservative property holds for any choice of α and β s.t. $\alpha - \beta = 1$.

We refer to [114] (Section 10.2.3) for a complete derivation of the propagation rules listed above within the deep Taylor decomposition framework [113]. Notice that for the $\alpha\beta$-LRP rule this generalization holds only when $\alpha = 1$ and $\beta = 0$, which are the values used in our experiments in Section 6.3.

3.5.3 Saliency Attacks

In this section, we define a few adversarial perturbations of the explanations used in our experiments (Section B.2), whose goal is to alter interpretations without affecting classifications.

[63] present a variety of attacks against feature importance methods, which iteratively maximize the diversity between original explanations $R(\mathbf{x}, \mathbf{w})$ and perturbed explanations $R(\tilde{\mathbf{x}}, \mathbf{w})$. At each step the image is perturbed in the direction of the gradient of a *dissimilarity function* $D(\mathbf{x}, \tilde{\mathbf{x}})$

$$\tilde{\mathbf{x}}_{t+1} = \text{Proj}\{\tilde{\mathbf{x}}_t + \alpha \cdot \text{sgn}\nabla_{\mathbf{x}} D(\mathbf{x}, \tilde{\mathbf{x}}_t)\}_{\epsilon,\infty},$$

where t indexes the current step in the iterative algorithm. We leverage two of the proposed techniques, with the following dissimilarity functions:

- $D(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{p \in \mathcal{A}_k} R(\tilde{x}_p, \mathbf{w})$ for the *target region* attack, where $\mathcal{A}_k$ is a pre-defined region of $k\%$ pixels;
- $D(\mathbf{x}, \tilde{\mathbf{x}}) = -\sum_{p \in \text{Top}_k(\mathbf{x})} R(\tilde{x}_p, \mathbf{w})$ for the *top-k* attack, where k indicates the chosen percentage of most relevant pixels.

In our experiments (Fig. B.8) we set $\alpha = 0.5$, $k = 20$, number of iterations $T = 10$ and the region of pixels was chosen randomly.

For each test image $\mathbf{x}$, *beta* attack by [46] builds a targeted perturbation $\tilde{\mathbf{x}}$ such that the classification is almost constant but the explanation resembles the target explanation of a randomly chosen test image $\hat{\mathbf{x}} \neq \mathbf{x}$. Specifically, it optimizes the loss function

$$||R(\tilde{\mathbf{x}}, \mathbf{w}) - R(\hat{\mathbf{x}}, \mathbf{w})||_2^2 + \gamma\, ||f(\tilde{\mathbf{x}}, \mathbf{w}) - f(\mathbf{x}, \mathbf{w})||_2^2$$

w.r.t. $\tilde{\mathbf{x}}$ and clamps $\tilde{\mathbf{x}}$ at each iteration to keep the image valid. Additionally, during the optimization phase, ReLU activations in the NN are replaced with softplus non-linearities

$$\text{softplus}_\beta(\mathbf{x}) = \frac{1}{\beta} \log\big(1 + e^{\beta \mathbf{x}}\big)$$

to avoid the problem of vanishing gradients. In our tests (Fig. B.9 (b)) we set $\gamma = 1$, iterations $T = 10$ and learning rate $\text{lr} = 0.01$.

3.6 Protein Language Models

Proteins are complex molecules that serve a variety of functions in our body's tissues and organs and are essential for life and reproduction. They consist of one or multiple chains of smaller units, called *amino acids*, which determine the three-dimensional structure and biological function of the protein. Naturally occurring proteins are composed of 20 different types of amino acids that can aggregate in a huge variety of unique combinations. In this large space of amino acid sequences, only a small fraction of proteins are expected to fold in a 3D structure [135]. The recent developments in sequencing techniques have enabled a large growth in the number of publicly available protein sequences; however, to understand the information encoded in amino acid chains it is essential to know how they fold in a three-dimensional structure. This is currently one of the main challenges in computational biology. Thanks to the development of powerful algorithms for structure prediction from the primary sequences, such as AlphaFold2 [81], it is now possible to predict 3D structures with high accuracy.

Structure prediction can be framed as a problem of *Natural Language processing* (NLP), a branch of computer science that deals with the analysis of human language and with automated tasks involving text generation (e.g. question answering, machine translation, etc.). *Language Model* (LM) architectures learn the probability distribution of characters in a sentence, which in the case of proteins translates into the probability distribution of amino acids in a sequence. LMs have recently emerged as a powerful tool in protein modeling, from the analysis of the relationship between amino acids [166], to the study of the evolutionary history of proteins [75, 72] and the generation of novel sequences [101, 82, 97, 135]. In particular, Transformer-based architectures [163] play a critical role in protein language modeling (and in NLP in general) and are the main focus of our work in Chapter 7.

3.6.1 Transformers

Transformers [163] are the current state-of-the-art models in NLP, solving tasks such as text classification, named entity recognition, question answering, text translation, text generation, with recent advances also in computer vision (e.g. image classification, object detection and segmentation) and signal processing (e.g. automatic speech recognition and audio classification). They solve sequence-to-sequence tasks by learning representations of large-scale sequence datasets that are aware of long-range dependencies. Unlike previous language model architectures, such as RNNs and LSTM, they have no recurrent connections and no sequential processing - i.e. they process an entire sequence simultaneously - therefore being suited for parallel computing on modern accelerators.

Masked language modeling The first step in any language model is *tokenization*, which is the process of breaking a sequence into individual components, e.g. words in a sentence or residues in a protein sequence. Transformers are trained to solve *masked language modeling*, which predicts missing *tokens* in the input sequence by learning the probability distribution of each masked token conditioned on the surrounding context, i.e. the remaining tokens in the unmasked part of the sequence.

Attention mechanism *Attention* is the basic concept in Transformer architectures, with the double goal of capturing long range-dependencies in a text and preserving coherence w.r.t. the surrounding context. It learns semantic dependencies between words in a text by processing an entire sequence, while also focusing on specific parts of it, similar to the mechanism of attention in the human mind [59, 163]. More specifically, attention computes correlations, named *self-attention scores*, between all couples of tokens in a sequence and uses them to control the impact of the contribution of each token to the hidden representations, therefore learning semantic dependencies between words.

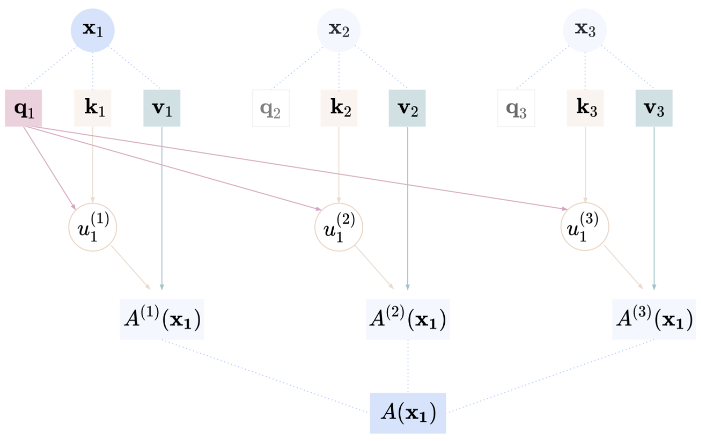


Figure 3.8: Computation of self-attention for a token $\mathbf{x}_1$ in a sequence $\mathbf{x}$ of length 3.

Self-attention for each token in a sentence $\mathbf{x}$ of length N is computed as follows. The sequence of tokens is first converted into a sequence of embedded representations called *word vectors*, which is the input of the first attention layer in the architecture. Precisely, let $\mathbf{x}_i \in \mathbb{R}^D$ denote the dense representation of a single token in $\mathbf{x}$, which is typically a one hot encoding over the reference alphabet summed up with a positional encoding. Given an input $\mathbf{x}_i$ an attention layer outputs three vector representations, namely *key* ($\mathbf{k}_i$), *query* ($\mathbf{q}_i$) and *value* ($\mathbf{v}_i$) vectors. The product between a query value $\mathbf{q}_i$ and the key value $\mathbf{k}_j$ for each token in the sentence ($j = 1, \ldots, N$) is divided by the square root of the dimension of $\mathbf{k}_j$ and passed through the softmax function

$$\mathbf{u}_i := \text{softmax}\left(\left[\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{\dim(\mathbf{k}_j)}}\right]_j\right) \in \mathbb{R}^N,$$

so that the total impact of token $\mathbf{x}_i$ on all other tokens $\mathbf{x}_j$ sums up to 1. Each softmax score $u_i^{(j)}$ in $\mathbf{u}_i$ is then multiplied by the j-th value vector $\mathbf{v}_j$, and finally, all the weighted scores are summed up to obtain the self-attention vector

$$A(\mathbf{x}_i) = \sum_{j=1}^{N} u_i^{(j)} \cdot \mathbf{v}_j \in \mathbb{R}^M,$$

representing the attention that token $\mathbf{x}_i$ places on itself and on all the other tokens in the surrounding context. Using multiple attention blocks, called *attention heads*, allows the Transformer to build several hidden representations of the input sequence, capturing different semantic information to improve local coherence.

In practice, self-attention is computed by matrix calculation, for faster processing. Given an input sequence of length N with dense representations of length D, represented in matrix form as $X \in \mathbb{R}^{N\times D}$, an embedding dimension M and the learnable key, query and vector weight matrices $W_k, W_q, W_v \in \mathbb{R}^{D\times M}$, each attention head h computes an $N \times M$ matrix:

$$A_h(W_k, W_q, W_v, X) = \text{softmax}\left(\frac{(X \cdot W_q) \cdot (X \cdot W_k)^T}{\sqrt{M}}\right) \cdot (X \cdot W_v).$$

Matrices from H attention heads are concatenated and multiplied by an additional matrix $W_o \in \mathbb{R}^{M\times M}$, to obtain a single multi-headed attention matrix, encoding the information provided by all the attention heads:

$$A(X) = \underset{h=1,\dots,H}{\text{concat}} (A_h) \cdot W_o.$$

Architecture and training Transformers are composed of multiple blocks of two types: an *encoder* block that combines a self-attention layer and a feed-forward layer; a *decoder* block that combines a self-attention layer, an encoder-decoder attention layer and a feed-forward layer (see Figure 3.9). The encoder-decoder layer processes the encoder's key and value vectors with the query vector from the decoder. A Transformer typically contains the same number of encoder and decoder modules (six in the original formulation). The decoder outputs a vector which is passed through a fully connected layer and a softmax layer, to return a probability distribution over words in the alphabet, i.e. to choose an output token that solves masked prediction. The learnable weights are initialized randomly and updated at each training step, using a random percentage of masked tokens for masked prediction. For further details on the architecture and the training procedure we refer the reader to [163].

3.6.2 Keywords in Structural Biology

The next paragraphs introduce some fundamental concepts in structural biology.

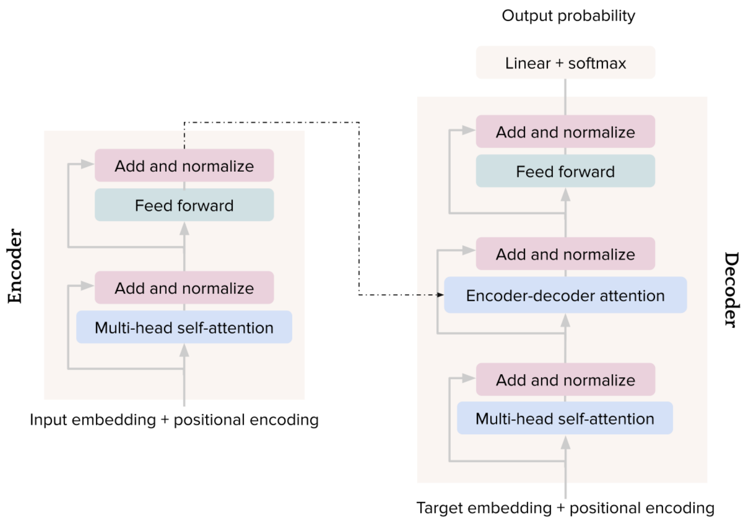


Figure 3.9: Example of Transformer architecture with one Encoder block and one Decoder block.

Amino acid An amino acid is a group of molecules that contains a central carbon (Cα) atom, an amino group, an acidic carboxyl group and an organic R group (or side chain), the latter being unique to each amino acid. Depending on the locations of the core functional groups, it is classified as α, β, γ or δ amino acid; for instance, a Cβ atom is the second carbon atom that attaches to a functional group. In proteins, amino-acid pairs that are far apart in a sequence are often spatially close in the 3D structure; this phenomenon, known as folding, ensures protein stability [135].

Protein structure A single protein has four orders of structure representations. The primary structure is the sequence of amino acids forming a polypeptide chain. The secondary structure describes local segments, such as alpha helices or beta sheets. The tertiary structure denotes the full 3D shape of a protein, which is determined by the interactions between groups of amino acids. Many proteins are composed of a single polypeptide chain, but others are made up of multiple chains, that together build its quaternary structure.

Contact map Contact maps provide a reduced representation of the tertiary structure of a protein. Precisely, they are 2-dimensional matrices containing the distances between all pairs of amino acids that are in contact but far apart (of at least 6 positions) in the sequence. Typically, two residues are considered to be in contact if their Cα atoms and their Cβ atoms are within 6-12Å. Contact maps are invariant to rotations and translations, and their diagonal represents the backbone of the protein. Interestingly, it is possible to retrieve the 3D coordinates of a protein from its contact map using distance geometry and stochastic optimization techniques [120, 164] or deep learning techniques [55, 128].

Binding site Binding sites describe the interaction of proteins with other molecules, i.e. their function. A binding site is a region that binds to a smaller external molecule, called *ligand*, eventually causing a conformational change that alters the protein's function.

Mutation A mutation is an alteration of an amino acid sequence, that can occur in both normal (e.g. evolution) and abnormal (e.g. cancer) biological processes. A mutation may alter the 3D structure (i.e. induce misfolding) of a protein without altering its function. However, some mutations at vulnerable sites, known as *missense* (or dysfunctional) mutations, can alter function, eventually causing diseases or influencing drug treatments [23, 150].

Sequence homology and Multiple Sequence Alignment Due to divergent evolution, biological organisms from the same common ancestor can evolve in different ways to serve several functions, resulting in homologous structures. Sequence homology quantifies the homology between proteins based on the evolutionary similarity between their amino acid sequences. For instance, percentage homology computes the percentage of conserved residues in the evolutionary history. A Multiple Sequence Alignment (MSA) is an alignment of multiple sequences according to their homologous regions.

BLOSUM matrix The BLOcks SUbstitution Matrix (BLOSUM) contains integer similarity scores between all couples of residues in a sequence and is used to score the alignments between evolutionarily divergent protein sequences. Scores in a BLOSUM-k matrix are based on replacement frequencies observed in known alignments with less than $k\%$ sequence similarity [80]. Therefore, it is a quantitative approach to determine whether an amino acid substitution is conservative (i.e. a substitution that preserves the biochemical properties of the chain) or nonconservative. In particular, the BLOSUM62 matrix has become a standard scoring matrix for a wide range of alignment programs. Positive BLOSUM scores for a substitution are classified as conservative, while negative values are classified as non-conservative.

3.6.3 NLP for Contact and Structure Prediction

The advancement of DNA sequencing has brought a rapid increase in the number of available protein sequences and several new algorithms that attempt to infer the biological function, the evolutionary history, the structural contacts and the full 3D structure of such sequences. Protein language models leverage state-of-the-art NLP architectures, and in particular, the attention mechanism, to learn rich representations of proteins from large databases of sequences.

In this work, we rely on two of the most known Transformer-based architectures for solving residue-residue contact prediction, namely *ESM-1b* [138] and *MSA Transformer* [132]. The former is trained on individual sequences of amino acids and computes classical row-attention on each sequence, while the latter takes as input an MSA of evolutionarily related sequences and combines row-attention with column-attention, which learns statistical properties among different positions in sequence alignments and identifies residues that are preserved during evolution. Similarly, AlphaFold2 model [81] is trained on MSAs and protein structures from the Protein Data Bank (PDB) [86] to predict 3D atomic coordinates of proteins from raw input sequences. In our experiments, we perform structure prediction through *ColabFold* [112], which optimizes structure prediction from AlphaFold2 by adding fast homology search of MMseqs2 [111].

3.6.4 Structural Similarity Scores

In our experiments, we rely on the following similarity measures [121] between two 3D protein structures:

- the *Root-Mean-Square-Deviation* computes the Euclidean distance between matching atoms after optimal superimposition

$$\mathrm{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2},$$

  where N is the number of equivalent atoms and d_i is the distance between couples of corresponding atoms. It is a global superimposition measure, ranging from 0 (same structure) to ∞;

- the *Local Distance Difference Test* [105] measures the percentage of preserved distances between all pairs of atoms in the target structure closer in space than a predefined cutoff, then it averages the resulting score using four different cutoffs (0.5, 1, 2 and 4Å)

$$\mathrm{LDDT} = \frac{1}{4}\cdot\frac{N_{0.5}+N_1+N_2+N_4}{M},$$

  where M is the number of atom pairs that do not belong to the same residue and are not further apart than 15Å in the target structure and N_k is the number of atom couples in the reference structure whose inter-atom distances deviate by no more than kÅ from the corresponding distances in the target structure. It is a score for local structural similarity, ranging from 0 (less similar structures) to 1 (more similar structures);

- the *TM-score* [185] computes the distances between all couples of Cα atoms after optimal superposition

$$\text{TM-score} = \max\left[\frac{1}{L_{\text{reference}}}\sum_{i=1}^{L_{\text{common}}}\frac{1}{1+(\frac{d_i}{d_0})^2}\right],$$

where $L_{\text{reference}}$ is the length of the reference sequence, L_{common} is the number of common residue pairs w.r.t. the target structure, d_i is the distance between corresponding residue pairs, $d_0 = 1.24\sqrt[3]{L_{\text{reference}} - 15} - 1.8$ length-dependent distance parameter that normalizes distances and the score is maximized w.r.t. superposition coordinates. It is a global superimposition score, ranging from 0 (less similar structures) to 1 (more similar structures).

Chapter 4

Adversarial Robustness of Bayesian Neural Networks

Contents

The robustness of BNNs to adversarial examples has already been observed empirically in various works [58, 182, 110, 123, 149, 161] and has been discussed in Section 3.4.4. However, while these works present empirical evidence on the robustness of BNNs, they do not give any theoretical motivation on the mechanisms that lead to BNN robustness. The framework we develop in this work further confirms and grounds findings from recent literature, by additionally providing a theoretical justification of such behaviour.

Indeed, we analyze the effect of dimensionality of the data manifold and the weight space on adversarial examples, proving that overparameterized BNNs are robust to gradient-based attacks in the limit of infinitely many data. Specifically, we first show that, for any neural network achieving zero loss, adversarial attacks arise in directions orthogonal to the data manifold. Then, we rely on the submanifold extension lemma [94] to show that in the limit of infinitely-wide layers, for any neural network and any set of weights there exists another set of weights achieving the same loss and with opposite loss gradients orthogonal to the data manifold on a given point. Under the assumption of infinitely many data and flat (i.e., uninformative) prior, we then show that by averaging over these weights sets the expectation of the gradient w.r.t. the posterior distribution of a BNN vanishes.

We conduct large-scale experiments on thousands of different neural networks, empirically finding that in the cases here analysed for BNNs high accuracy correlates with high robustness to gradient-based adversarial attacks, contrary to what is observed for deterministic NNs trained via standard Stochastic Gradient Descent (SGD). Finally, we also investigate the robustness of BNNs to gradient-free adversarial attacks, showing that BNNs are substantially more robust than their deterministic counterpart even in this setting.

4.1 Gradient-Based Adversarial Attacks for Neural Networks

Equation (3.9) in Chapter 3 defines adversarial attacks for Bayesian Neural Networks in terms of an expectation under the posterior distribution. This suggests a possible mechanism through which BNNs might acquire robustness against adversarial attacks: averaging under the posterior might lead to cancellations in the expectation of the gradient of the loss. In the next sections, we prove that this averaging property is closely related to the geometry of the data manifold $\mathcal{M}$. As a consequence, to study the expectation of the gradient of the loss for BNNs, we first introduce some results that link the geometry of $\mathcal{M}$ to adversarial attacks.

We start with a trivial, yet important observation: for a NN that achieves zero loss on the whole data manifold $\mathcal{M}$, the loss gradient is constant (and zero) along the data manifold for any $\mathbf{x}^* \in \mathcal{M}$. Therefore, in order to have adversarial examples the dimension of the data manifold $\mathcal{M}$ must necessarily be smaller than the dimension of the ambient space $\mathcal{S} \subset \mathbb{R}^d$, that is, $\dim(\mathcal{M}) < \dim(\mathcal{S})$, where $\dim(\mathcal{M})$ denotes the dimension of $\mathcal{M}$.

Lemma 4.1. *Assume that $\forall \mathbf{x} \in \mathcal{M}\ \mathcal{L}(\mathbf{x}, \mathbf{w}) = 0$, that is $f(\mathbf{x}, \mathbf{w})$ achieves zero loss on $\mathcal{M}$. Then, if f is vulnerable to gradient-based attacks at $\mathbf{x}^* \in \mathcal{M}$, $\dim(\mathcal{M}) < \dim(\mathcal{S})$ in a neighbourhood of $\mathbf{x}^*$, i.e. $\mathcal{M}$ is locally homeomorphic to a space of dimension smaller than the ambient space $\mathcal{S}$.*

Proof. By assumption $\forall \mathbf{x} \in \mathcal{M}, \mathcal{L}(\mathbf{x}, \mathbf{w}) = 0$, which implies that the gradient of the loss is zero along the data manifold. However, if f is vulnerable to gradient-based attacks at $\mathbf{x}^*$ then the gradient of the loss at $\mathbf{x}^*$ must be non-zero. Hence, there exists an open neighbourhood $\mathcal{B}$ of $\mathbf{x}^*$ such that $\mathcal{B} \nsubseteq \mathcal{M}$, which implies $\dim(\mathcal{M}) < \dim(\mathcal{S})$ locally around $\mathbf{x}^*$. $\square$

Lemma 4.1 confirms the widely held conjecture that adversarial attacks may originate from degeneracies of the data manifold [67, 56]. Indeed, it has been already empirically noticed [84] that adversarial perturbations often arise in directions normal to the data manifold. The suggestion that lower-dimensional data structures might be ubiquitous in NN problems is also corroborated by recent results [65] showing that the characteristic training dynamics of NNs are intimately linked to data lying on a lower-dimensional manifold. Notice that the implication is only one way; the data manifold can be low dimensional and still not vulnerable at many points.

We note that, as discussed in Section 3.2.3, at convergence of the training algorithm and in the limit of infinitely-many data, infinitely-wide neural networks are guaranteed to achieve zero loss on the data manifold, satisfying the assumption of Lemma 4.1. As a result, once an infinitely-wide NN is fully trained, for any $\mathbf{x} \in \mathcal{M}$ the gradient of the loss function is orthogonal to the data manifold as it is zero along the data manifold, i.e., $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{w}) = \nabla^{\perp}_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{w})$, where $\nabla^{\perp}_{\mathbf{x}}$ denotes the gradient projected into the normal subspace of $\mathcal{M}$ at $\mathbf{x}$. Note that for a given NN, $\nabla^{\perp}_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{w})$ is, in general, non-zero even if the network achieves zero loss on $\mathcal{M}$, thus explaining the existence of adversarial examples even for very accurate classifiers. Crucially, in Section 4.1.1 we show that for BNNs, when averaged w.r.t. the posterior distribution, the orthogonal gradient vanishes.

4.1.1 A Symmetry Property of Neural Networks

Before considering the BNN case, in Proposition 4.1 below we show a symmetry property of neural networks: given a neural network, we can always find an infinitely-wide NN that has the same loss but opposite orthogonal gradient. To prove this result, we first introduce Lemma 4.2, which is a generalization of the submanifold extension lemma and a key result we leverage. It proves that any smooth function defined on a submanifold $\mathcal{M}$ can be extended to the ambient space, in such a way that the choice of the derivatives orthogonal to the submanifold is arbitrary.

Lemma 4.2 ([5]). *Let $T_{\mathbf{x}}\mathcal{M}$ be the tangent space of $\mathcal{M}$ at a point $\mathbf{x} \in \mathcal{M}$. Let $V = \sum_{i=\dim(\mathcal{M})+1}^{d} v^i \partial_i$ be a conservative vector field along $\mathcal{M}$ which assigns a vector in $T_{\mathbf{x}}\mathcal{M}^{\perp}$ for each $\mathbf{x} \in \mathcal{M}$. For any smooth function $f^{true} : \mathcal{M} \to \mathbb{R}$ there exists a smooth extension $F : \mathcal{S} \to \mathbb{R}$ such that*

$$F|_{\mathcal{M}} = f^{true},$$

where $F|_{\mathcal{M}}$ denotes the restriction of F to the submanifold $\mathcal{M}$, and such that the derivative of the extension F is

$$\nabla_{\mathbf{x}}F(\mathbf{x}) = (\nabla_1 f^{true}(\mathbf{x}), \ldots, \nabla_{\dim(\mathcal{M})} f^{true}(\mathbf{x}), v^{\dim(\mathcal{M})+1}(\mathbf{x}), \ldots, v^d(\mathbf{x}))$$

for all $\mathbf{x} \in \mathcal{M}$.

Notice that in Lemma 4.2, in $\nabla_{\mathbf{x}}F(\mathbf{x})$, we pick the local coordinates at $\mathbf{x} \in \mathcal{M}$, such that the first set of components parametrises the data manifold. We stress that as $\mathcal{M}$ is smooth, this is without any loss of generality [94]. Lemma 4.2, together with the universal approximation capabilities of NNs [78], is employed in Proposition 4.1 to show that for any possible value $\mathbf{v}$ of the orthogonal gradient to the data manifold in a point, there exists at least two (possibly not unique) different weight vectors that achieve zero loss and have orthogonal gradients respectively equal to $\mathbf{v}$ and $-\mathbf{v}$.

Proposition 4.1. *Consider a NN f with $M+1$ layers, last hidden layer of size n_M and an input $\mathbf{x} \in \mathcal{M}$. Then, in the limit of $n_M \to \infty$ (size of last hidden layer going to infinity), for any smooth function $f^{true} : \mathcal{M} \to \mathbb{R}$ and vector $\mathbf{v} \in \mathbb{R}^{\dim(\mathcal{S})-\dim(\mathcal{M})}$, there exist two sets of weights $\mathbf{w}_1, \mathbf{w}_2$ such that*

$$f(\cdot, \mathbf{w}_1)|_{\mathcal{M}} = f^{true} = f(\cdot, \mathbf{w}_2)|_{\mathcal{M}} \tag{4.1}$$

$$\nabla^{\perp}_{\mathbf{x}} f(\mathbf{x}, \mathbf{w}_1) = \mathbf{v} = -\nabla^{\perp}_{\mathbf{x}} f(\mathbf{x}, \mathbf{w}_2). \tag{4.2}$$

Proof. From Lemma 4.2 we know that there exist smooth extensions F^+ and F^- of f^{true} to the embedding space such that $\nabla^{\perp}_{\mathbf{x}} F^+(\mathbf{x}) = \mathbf{v} = -\nabla^{\perp}_{\mathbf{x}} F^-(\mathbf{x})$. As a consequence, to conclude the proof it suffices to apply Theorem 3 in [78] that guarantees that infinitely-wide neural networks are *uniformly 1-dense* on compacts in $\mathcal{C}^1(\mathcal{S})$, under the assumptions of smooth, bounded, and non-constant activation functions. Specifically, for any $F \in \mathcal{C}^1(\mathcal{S})$ and $\epsilon > 0$, for any compact $\mathcal{S}' \subseteq \mathcal{S}$ there exists a set of weights $\mathbf{w}$ s.t.

$$\max \left\{ \sup_{\mathbf{x} \in \mathcal{S}'} ||F(\mathbf{x}) - f(\mathbf{x}, \mathbf{w})||_{\infty}, \sup_{\mathbf{x} \in \mathcal{S}'} ||\nabla F(\mathbf{x}) - \nabla f(\mathbf{x}, \mathbf{w})||_{\infty} \right\} \leq \epsilon.$$

As $F^+, F^- \in \mathcal{C}^1(\mathcal{S})$, this concludes the proof. □

Note that by the chain rule, the gradient of the loss is proportional to the gradient of the NN. As a consequence, Proposition 4.1 guarantees that, for infinitely-wide NNs, for any set of weights achieving the minimum loss, then there exists another set of weights with the same loss and opposite orthogonal gradient of the loss w.r.t. the input. This suggests that by averaging over these configurations of the weights one can achieve a robust model that has a vanishing expected orthogonal gradient. In the next section, in Theorem 4.1 we show this is indeed the case. However, we should already emphasize that such a result does not hold by simply averaging the set of weights w.r.t. any distribution. Intuitively, for this result to hold, it is required that each set of weights achieving a given gradient value has the same measure as the set of weights with the same loss and opposite orthogonal gradient value.

4.2 Adversarial Robustness via Bayesian Averaging

We are now ready to state Theorem 4.1, where we show that under the assumption of a flat prior, the posterior of an infinitely-wide BNN achieving zero loss has vanishing orthogonal gradients.

Theorem 4.1. *Let $f(\mathbf{x}, \mathbf{w})$ be an infinitely-wide BNN trained on a data set D_N composed by N data points with true underlying function $f^{true} : \mathcal{M} \to \mathbb{R}$. Assume that:*

(1) The prior is flat.

(2) For $N \to \infty$ the posterior distribution of the BNN converges weakly to a distribution that achieves zero loss with probability 1, i.e.,

$$P(f(\cdot, \mathbf{w})|D_N) \xrightarrow{d} P(f \mid D_\infty),$$

where $P(f \mid D_\infty)$ is such that

$$P\Big(\sup_{\mathbf{x}\in\mathcal{M}} |f(\mathbf{x}) - f^{true}(\mathbf{x})| = 0 \,\Big|\, D_\infty\Big) = 1. \tag{4.3}$$

Then, for any $\mathbf{x} \in \mathcal{M}$ it holds that

$$\mathbb{E}_{p(f(\cdot,\mathbf{w})|D_N)}[\nabla^\perp_\mathbf{x} f(\mathbf{x}, \mathbf{w})] \to 0 \tag{4.4}$$

as $N \to \infty$.

Proof. Without any loss of generality assume that $\dim(\mathcal{S}) - \dim(\mathcal{M}) = 1$.[1] Consider the operator

$$\mathcal{G}_\mathbf{x} : f \mapsto \nabla^\perp_\mathbf{x} f(\mathbf{x}),$$

which is linear and bounded under the boundedness assumption on the derivatives of f. By the Portmanteau theorem, $\mathcal{G}_\mathbf{x}$ preserves weak convergence, hence as $N \to \infty$

$$\mathbb{E}_{p(f(\cdot,\mathbf{w})|D_N)}[\nabla^\perp_\mathbf{x} f(\mathbf{x}, \mathbf{w})] \to \mathbb{E}_{p(f|D_\infty)}[\nabla^\perp_\mathbf{x} f(\mathbf{x})]. \tag{4.5}$$

Then, to prove Equation (4.4) it is enough to show that

$$\mathbb{E}_{p(f|D_\infty)}[\nabla^\perp_\mathbf{x} f(\mathbf{x})] = 0.$$

To do that, we proceed as follows. By definition of expectation, we have that

$$\begin{aligned}\mathbb{E}_{p(f|D_\infty)}[\nabla^\perp_\mathbf{x} f(\mathbf{x})] &= \int\limits_{\mathbf{v}\in\mathbb{R}} \mathbf{v}\, p(\nabla^\perp_\mathbf{x} f(\mathbf{x}) = \mathbf{v}|D_\infty) d\mathbf{v} \\ &= \int\limits_{\mathbf{v}\in\mathbb{R}_{>0}} \mathbf{v}\, p(\nabla^\perp_\mathbf{x} f(\mathbf{x}) = \mathbf{v}|D_\infty) d\mathbf{v} + \int\limits_{\mathbf{v}\in\mathbb{R}_{<0}} \mathbf{v}\, p(\nabla^\perp_\mathbf{x} f(\mathbf{x}) = \mathbf{v}|D_\infty) d\mathbf{v}.\end{aligned} \tag{4.6}$$

If we now can show that for any $\mathbf{v} \in \mathbb{R}$ and $\epsilon \in \mathbb{R}_{>0}$

$$P(\nabla^\perp_\mathbf{x} f(\mathbf{x}) \in [\mathbf{v} - \epsilon, \mathbf{v} + \epsilon] \mid D_\infty) = \int_{[\mathbf{v}-\epsilon,\mathbf{v}+\epsilon]} p(\nabla^\perp_\mathbf{x} f(\mathbf{x}) = \mathbf{v} \mid D_\infty) dv$$

[1]In the case that $\dim(\mathcal{S}) - \dim(\mathcal{M}) > 1$, because of the linearity of the partial derivatives, any component of $\mathbb{E}_{p(f|D_\infty)}[\nabla^\perp_\mathbf{x} f(\mathbf{x})]$ can be treated analogously to the case $\dim(\mathcal{S}) - \dim(\mathcal{M}) = 1$.

is constant w.r.t. $\mathbf{v}$, then this guarantees that equation (4.6) is zero. In fact, by applying the fundamental theorem of calculus, the above implies that for any $\mathbf{v}$, $p(\nabla^{\perp}_{\mathbf{x}} f(\mathbf{x}) = \mathbf{v} \mid D_\infty) = p(\nabla^{\perp}_{\mathbf{x}} f(\mathbf{x}) = -\mathbf{v} \mid D_\infty)$ almost surely (i.e. with probability 1), hence terms in equation (4.6) cancel out. We prove that

$$P(\nabla^{\perp}_{\mathbf{x}} f(\mathbf{x}) \in [\mathbf{v} - \epsilon, \mathbf{v} + \epsilon] \mid D_\infty)$$

is constant w.r.t. $\mathbf{v}$ in what follows.

By assumption it holds that any $f \sim P(\cdot|D_\infty)$ is such that $\forall \bar{\mathbf{x}} \in \mathcal{M}, f(\bar{\mathbf{x}}) = f^{true}(\bar{\mathbf{x}})$ almost surely. Hence, to conclude, being our prior invariant to translations (see definition 3.6), it is enough to show that for any $\mathbf{v}, \mathbf{v}'$ sets

$$F_{\mathbf{v}} = \{g \in \mathcal{F} \ : \ \forall \bar{\mathbf{x}} \in \mathcal{M},\, g(\bar{\mathbf{x}}) = f^{true}(\bar{\mathbf{x}}) \wedge \nabla^{\perp}_{\mathbf{x}} g(\mathbf{x}) \in [\mathbf{v} - \epsilon, \mathbf{v} + \epsilon]\}$$

$$F_{\mathbf{v}'} = \{g \in \mathcal{F} \ : \ \forall \bar{\mathbf{x}} \in \mathcal{M},\, g(\bar{\mathbf{x}}) = f^{true}(\bar{\mathbf{x}}) \wedge \nabla^{\perp}_{\mathbf{x}} g(\mathbf{x}) \in [\mathbf{v}' - \epsilon, \mathbf{v}' + \epsilon]\}$$

are equal up to a translation. This can be shown as follows. Using Lemma 4.2 we extend f^{true} to a smooth function $f_{\mathbf{v}}$ in $\mathcal{S}$ such that its orthogonal gradient in $\mathbf{x} \in \mathcal{M}$ equals $\mathbf{v}$, i.e. $\nabla^{\perp}_{\mathbf{x}} f_{\mathbf{v}}(\mathbf{x}) = \mathbf{v}$. Then, because of the linearity of the gradient operator, any other function g on X that is equal to f^{true} when restricted to $\mathcal{M}$ and such that $\nabla^{\perp}_{\mathbf{x}} g(\mathbf{x}) \in [\mathbf{v} - \epsilon, \mathbf{v} + \epsilon]$ can be written as

$$g(\mathbf{x}) = f_{\mathbf{v}}(\mathbf{x}) + \bar{k}(\mathbf{x}),$$

where $\bar{k}$ is a function that has the orthogonal gradient in $\mathbf{x}$ within $[-\epsilon, \epsilon]$ and that is 0 on $\mathcal{M}$, i.e.,

$$k \in \mathcal{K} = \{\bar{k} \ : \ \forall \bar{\mathbf{x}} \in \mathcal{M}, \bar{k}(\bar{\mathbf{x}}) = 0 \wedge \nabla^{\perp}_{\mathbf{x}} \bar{k}(\mathbf{x}) \in [-\epsilon, \epsilon]\}. \tag{4.7}$$

Let's consider a function $f_{\mathbf{v}'}$ built similarly as $f_{\mathbf{v}}$, then we obtain that all $g'(\mathbf{x})$ equal to the true function and such that $\nabla^{\perp}_{\mathbf{x}} g'(\mathbf{x}) \in [\mathbf{v}' - \epsilon, \mathbf{v}' + \epsilon]$ can be written as

$$g'(\mathbf{x}) = f_{\mathbf{v}'}(\mathbf{x}) + \bar{k}(\mathbf{x}),$$

with $\bar{k} \in \mathcal{K}$, where $\mathcal{K}$ is as in equation (4.7). It then follows that the sets $F_{\mathbf{v}}$ and $F_{\mathbf{v}'}$ are such that $F_{\mathbf{v}} = F_{\mathbf{v}'} + (f_{\mathbf{v}} - f_{\mathbf{v}'})$. Hence,

$$P(\nabla^{\perp}_{\mathbf{x}} f(\mathbf{x}) \in [\mathbf{v} - \epsilon, \mathbf{v} + \epsilon] \mid D_\infty) = P(\nabla^{\perp}_{\mathbf{x}} f(\mathbf{x}) \in [\mathbf{v}' - \epsilon, \mathbf{v}' + \epsilon] \mid D_\infty).$$

Thus, concluding the proof. □

The proof of Theorem 4.1 relies on two main assumptions: (1) the prior is flat, which guarantees that probabilities are invariant to translations, thus allowing for perfect cancellation in expectation; (2) the limiting distribution of an infinitely-wide BNN has zero loss with probability 1, which allows us to only focus on the set of functions that are a.s. equal to the true functions on the data manifold. While we will discuss the implication of the flat prior assumption in Section 8.1, we would like to stress that because of the universal approximation properties of neural networks, the second assumption in Theorem 4.1 is a standard assumption to require, as it guarantees that our posterior converges to the true function with infinitely many data. For example, under mild assumptions, a similar requirement has been explicitly shown to hold for infinitely-wide feed-forward Bayesian neural networks with one hidden layer [93].

4.2.1 Multimodality of Posterior Weight Distribution

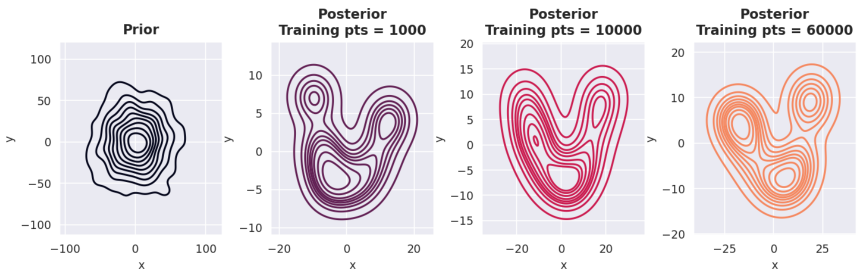


Figure 4.1: Prior (first column) and posterior (columns 1-3) distributions for models trained with an increasing number of training points. Models are trained on MNIST dataset with HMC. Samples are projected on the first two principal components (here x and y) using PCA.

A key aspect in the proof of Theorem 4.1 is that when the data manifold has a dimension smaller than the ambient space, then there are multiple weight sets minimizing the loss (see Lemma 4.2) and the BNN posterior averages between them achieving vanishing orthogonal gradients. Note that, if there were just one set of minimizing weights, we could have employed standard results of asymptotic statistics, such as the Bernstein von Mises Theorem to study the asymptotic behaviour of the BNN [162]. However, a standard application of Bernstein von Mises Theorem would have led to convergence to a Gaussian distribution centred in the maximum likelihood parameters and with vanishing variance: this would have implied that a BNN is robust if and only if also the correspondent deterministic NN with fixed (deterministic) weights is robust, which we know not to be the case.

Figure 4.1 empirically confirms this phenomenon by showing that the posterior distributions of the weights are multimodal, thus Bernstein von Mises theorem does not apply to BNNs.

4.2.2 Comparison with other Randomization Strategies

While the Bayesian posterior ensemble may not be the only randomization to provide protection, it is clear that some simpler randomization techniques such as bootstrap will be ineffective, as noted empirically by [11]. This is because bootstrap resampling introduces variability along the data manifold, rather than in orthogonal directions. In this sense, bootstrap clearly cannot be considered a Bayesian approximation, especially when the data distribution has zero measure support w.r.t. the ambient space. Similarly, we do not expect gradient smoothing approaches [7] or adaptive attacks [159] to be successful, since the type of smoothing performed by Bayesian inference is specifically informed by the geometry of the data manifold. In Section 4.3.5 we also run an empirical comparison with Deep Ensembles [91] - i.e. ensembles of deterministic NNs - and

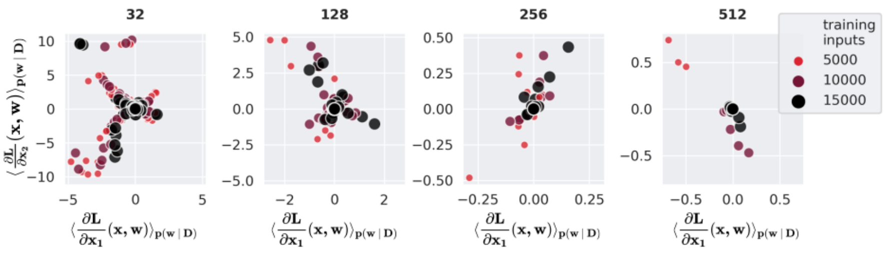


Figure 4.2: Expected loss gradients components on 100 two-dimensional test points from the Half Moons dataset [142] (both partial derivatives of the loss function are shown). We used a collection of HMC BNNs, by varying the number of hidden neurons (a different number in each subplot) and training points (different colours and sizes of dots). Each dot represents a different NN. Only models with test accuracy greater than 80% were taken into account.

show that ensemble averaging does not provide an improvement in robustness. However, in Chapter 5 we propose two randomization methods to improve the adversarial robustness of pre-trained deterministic architectures, both based on the computation of random projections of the training data. We show that such techniques contribute to the robustness of the networks, but, to our knowledge, do not provide any certifiable robustness in the limit. Lastly, it is worth noticing that Gaussian Processes [173] are equivalent to infinitely wide BNNs, therefore Theorem 4.1 provides theoretical backing to recent empirical observations of their adversarial robustness [32, 69, 125].

4.3 Experimental Results

In this section, we empirically investigate our theoretical findings on different BNNs. We train a variety of BNNs on the popular MNIST and Fashion MNIST [174] datasets. Both data sets are composed of 60.000 training images belonging to ten classes: in the MNIST case, these are hand-written digits, while the Fashion MNIST data set consists of stylized Zalando images of clothing items. While MNIST is considered a relatively trivial data set, with accuracies over 99% being regularly reported, Fashion MNIST is considerably more complex, and the best architectures report accuracies around 95%. We infer the posterior distributions using HMC and VI approximate inference methods, presented in Section 3.3. In Section 4.3.1, we experimentally verify the validity of the zero-averaging property of gradients implied by Theorem 4.1, and discuss its implications on the behaviours of FGSM and PGD attacks (Section 3.4.1) on BNNs in Section 4.3.2. In Section 4.3.3 we analyse the relationship between robustness and accuracy on thousands of different NN architectures, comparing the results obtained by Bayesian and by deterministic training. Furthermore, in Section 4.3.4 we investigate the robustness of BNNs on a gradient-free adversarial attack [7]. Details on the experimental settings and BNN training parameters are reported in Appendix A.

Simulations[2] are conducted on a machine with 34 single core Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz processors and 200GB of RAM. We make an extensive use of PyTorch [124] and Pyro [15] libraries.

4.3.1 Evaluation of the Gradient of the Loss for BNNs

We investigate the vanishing behaviour of input gradients - established by Theorem 4.1 for the infinite limit regime - in the finite, practical settings, that is with a finite number of training data and with finite-width BNNs. Specifically, for each inference method, we perform hyperparameter tuning and select the architecture achieving the highest test accuracy: we train a two hidden layers BNN (with 1024 neurons per layer) with HMC and a three hidden layers BNN (512 neurons per layer) with VI. These achieve approximately 95% test accuracy on MNIST and 89% on Fashion MNIST when trained with HMC; as well as 95% and 92%, respectively, when trained with VI.

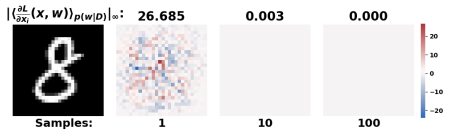


Figure 4.3: The expected loss gradients of BNNs exhibit a vanishing behaviour when increasing the number of samples from the posterior predictive distribution. Here we show an example image from MNIST (first column) and heatmaps (second to fourth column) of the expected loss gradients w.r.t. BNNs trained with SVI.

Figure 4.3 depicts anecdotal evidence on the behaviour of the component-wise expectation of the loss gradient as more samples from the posterior distribution are incorporated into the predictive distribution of the BNN. As the number of samples taken from the posterior distribution of $\mathbf{w}$ increases, all the components of the gradient approach zero. Notice that the gradients of the individual NNs (that is those with just one sampled weight vector), are far away from being zero. As shown in Theorem 4.1, it is only through Bayesian averaging of the ensemble predictors that the gradients cancel out.

This is confirmed in Figures 4.4 and 4.5, where we provide a systematic analysis of the aggregated gradient convergence properties on 1k test images for MNIST and Fashion MNIST. Each dot shown in the plots represents a component of the expected loss gradient from each one of the images, for a total of 784k points used to visually approximate the empirical distribution of the component-wise expected loss gradient. For both HMC and VI the magnitude of the gradient components drops as the number of samples increases and tends to stabilize

[2]Code is available at: `https://github.com/ginevracoal/robustBNNs`.

around zero already with 100 samples drawn from the posterior distribution, suggesting that the conditions laid down in Theorem 4.1 are approximately met by the BNN analysed here. Notice that the gradients computed on HMC-trained networks drop more quickly and towards a smaller value compared to VI-trained networks, most likely because VI introduces additional approximations in the Bayesian posterior computation.

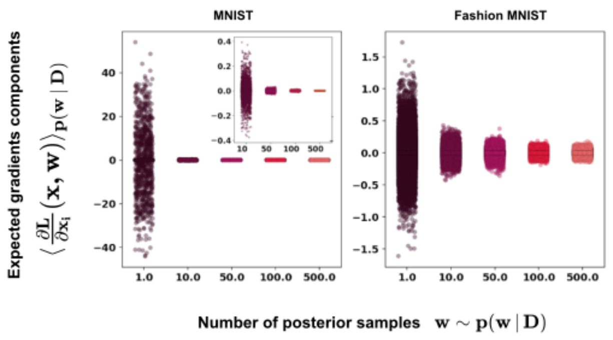


Figure 4.4: The components of the expected loss gradients approach zero as the number of samples from the posterior distribution increases. For each number of samples, the figure shows 784 gradient components for 1k different test images, from both the MNIST and Fashion MNIST datasets. The gradients are computed on HMC-trained BNNs.

4.3.2 Gradient-Based Attacks for BNNs

The fact that gradient cancellation occurs in the limit does not directly imply that BNNs' predictions are robust to gradient-based attacks in the finite case. For example, FGSM attacks are crafted such that the direction of the manipulation is given only by the sign of the expectation of the loss gradient and not by its magnitude. Thus, even if the entries of the expectation drop to an infinitesimal magnitude but maintains a meaningful sign, then FGSM could potentially produce effective attacks. To test the implications of vanishing gradients on the robustness of the posterior predictive distribution against gradient-based attacks, we compare the behaviour of FGSM and PGD[3] attacks to a randomly devised attack.

Namely, the random attack mimics a randomised version of FGSM in which the sign of the attack is sampled at random. In practice, we perturb each component of a test image by a random value in $\{-\epsilon, \epsilon\}$. In Table 4.1 we compare the effectiveness of FGSM, PGD and the random attack and report the adversarial robustness for 500 images. For each image, we compute the expected gradient using 250 posterior samples. The attacks were performed with $\epsilon = 0.1$

[3]With 15 iterations and 1 restart.

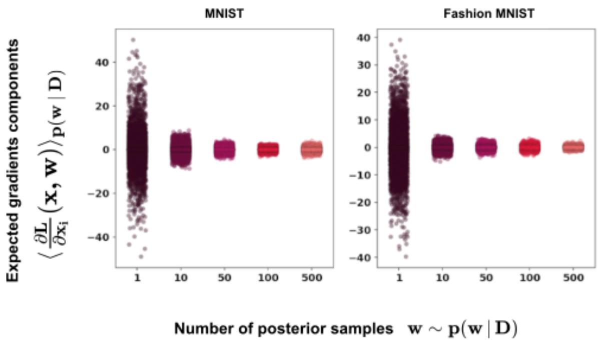


Figure 4.5: The components of the expected loss gradients approach zero as the number of samples from the posterior distribution increases. For each number of samples, the figure shows 784 gradient components for 1k different test images, from both the MNIST and Fashion MNIST datasets. The gradients are computed on VI-trained BNNs.

Dataset/Method	**Rand**	**FGSM**	**PGD**
MNIST/HMC	**0.850**	0.960	0.970
MNIST/VI	0.956	**0.936**	0.938
Fashion/HMC	**0.812**	0.848	0.826
Fashion/VI	**0.744**	0.834	0.916

Table 4.1: Adversarial robustness of BNNs trained with HMC and VI with respect to the *random attack* (Rand), FGSM attack and PGD attack. Robustness is computed on 500 images with 250 posterior samples. Attacks are performed using $\epsilon = 0.1$.

and using the categorical cross-entropy loss function. In almost all cases, we see that the random attack is stronger than the gradient-based attacks in the Bayesian setting, showing how the vanishing behaviour of the gradient makes FGSM and PGD attacks ineffective.

4.3.3 Robustness Accuracy Analysis in Deterministic and Bayesian Neural Networks

Theorem 4.1 holds in a specific thermodynamic limit, however, we expect the averaging effect of BNN gradients to still provide considerable protection in conditions where the network architecture and the data amount lead to high accuracy and strong expressivity. In practice, high accuracy might be a good indicator of robustness for BNNs. An empirical confirmation of this result is shown in Figure 4.2, where we examine the impact of the assumptions made in Theorem 4.1: by exploiting a setting in which we have access to the data-generating distribution, the half-moons dataset [142]. Precisely, we show that the magnitude of the expectation of each component of the gradient shrinks as we increase the number of the network's parameters and the number of training points.

In this section, we analyze more than 1000 BNNs with different hyper-parameters trained with HMC and VI on MNIST and Fashion-MNIST. We experimentally evaluate their accuracy/robustness trade-off on FGSM attacks as compared to that of the same neural network architectures trained via standard (i.e., non-Bayesian) SGD method. For the robustness evaluation, we consider the average difference in the softmax prediction between the original test points and the crafted adversarial input, as this provides a quantitative and smooth measure of adversarial robustness that is closely related to misclassification ratios [31]. That is, for a collection of N test points, we compute

$$\frac{1}{N}\sum_{j=1}^{N}\left|\mathbb{E}_{p(\mathbf{w}|D)}[f(\mathbf{x}_j,\mathbf{w})] - \mathbb{E}_{p(\mathbf{w}|D)}[f(\tilde{\mathbf{x}}_j,\mathbf{w})]\right|_{\infty}. \tag{4.8}$$

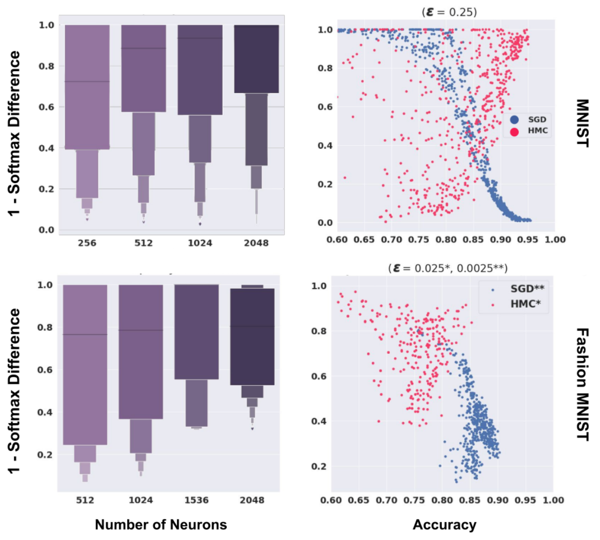


Figure 4.6: Robustness-Accuracy trade-off on MNIST (first row) and Fashion MNIST (second row) for BNNs trained with HMC (red dots) and SGD (blue dots). The boxplots show the correlation between model capacity and robustness. Different attack strengths (ϵ) are used for the three methods depending on their average robustness.

The results of the analysis are plotted in Figures 4.6 and 4.7 for MNIST and Fashion MNIST datasets. Each dot in the scatter plots represents the results

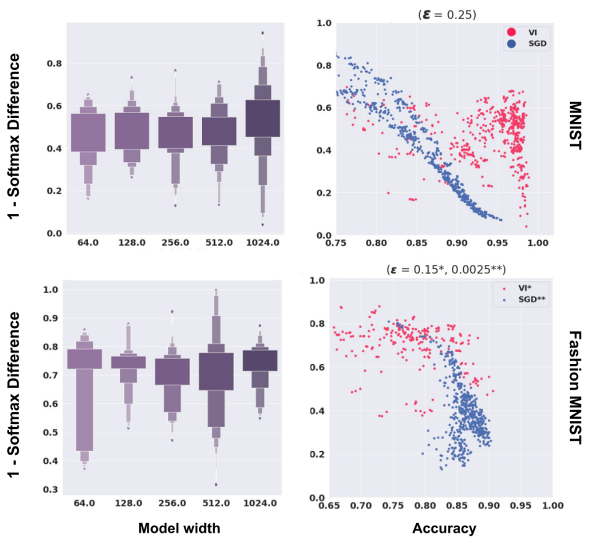


Figure 4.7: Robustness-Accuracy trade-off on MNIST (first row) and Fashion MNIST (second row) for BNNs trained with VI (red dots) and SGD (blue dots). The boxplots show the correlation between model capacity and robustness. Different attack strengths (ϵ) are used for the three methods depending on their average robustness.

obtained for a specific network architecture trained with SGD (blue dots), HMC (pink dots in Figure 4.6) and VI (pink dots in Figure 4.7). As already reported in the literature [152] we observe a marked trade-off between predictive accuracy and robustness (i.e., 1 - softmax difference) for high-performing deterministic networks. Interestingly, this trend is fully reversed for BNNs trained with HMC (Figure 4.6) where we find that as networks become more accurate, they also become more robust to FGSM attacks. We further examine this trend in the boxplots that represent the effect of the network's width on the robustness of the resulting posterior. We observe an increasing trend in robustness as the number of neurons in the network grows. This is in line with our theoretical findings, indeed, as the BNN approaches the infinite width limit, the conditions for Theorem 4.1 are approximately met and the network is protected against gradient-based attacks.

On the other hand, the trade-off behaviours are less obvious for the BNNs trained with VI. In particular, in Figure 4.7 we find that, similarly to the deterministic case, also for BNNs trained on Fashion-MNIST with VI robustness seems to have a slightly negative correlation with accuracy. Furthermore, only in the case of VI trained on MNIST we observe an increasing trend in robustness w.r.t. the growing size of the model, but this could also lead to poor robustness, which may be indicative of mode collapse.

4.3.4 Gradient-Free Adversarial Attacks

We empirically evaluate the most accurate BNN posteriors on MNIST and Fashion MNIST from Figures 4.6 and 4.7 against gradient-free adversarial attacks. In particular, we select ZOO attack [37] because it is effective even when tested on networks that purposefully obfuscate their gradients or have vanishing gradients [7]. In Figure 4.8 we observe that, similarly to gradient-based methods, ZOO is substantially less effective on BNNs compared to deterministic NNs in all cases, with BNNs again achieving both high accuracy and high robustness simultaneously. Furthermore, once again, HMC is more robust to the attacks compared to VI, which is in turn substantially more robust than deterministic NNs. This suggests how, similarly to what was observed in the previous subsections, a more accurate posterior distribution may lead to a more robust model, also in the case of gradient-free adversarial attacks.

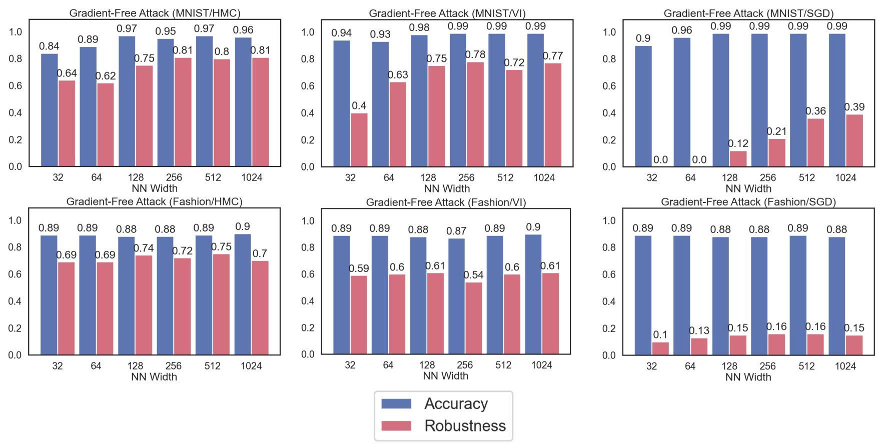


Figure 4.8: Gradient-free adversarial attacks on BNNs display similar behavior to gradient-based attacks. We evaluate the more accurate networks from Figures 4.6 and 4.7 with ZOO attacks on MNIST (first row) and Fashion MNIST (second row) for BNNs trained with HMC (left column), VI (center column), and SGD (right column).

4.3.5 Empirical comparison with Deep Ensembles

[91] presented *Deep Ensembles* as a computationally cheap alternative to Bayesian NNs. Deep ensembles are an ensemble of deterministic neural networks trained with different random initial conditions, whose outputs are averaged at prediction time. As already discussed in this chapter, adversarial attack strategies identify directions of high variability in the loss function by evaluating the gradient w.r.t. to the neural network input [67, 103]. We showed how this variability is linked to uncertainty in predictions and how Bayesian Neural Networks can guarantee

robustness to gradient-based attacks, even in the finite data regime. Therefore, it is important to run a comparison between deterministic NNs, deep ensembles and Bayesian NNs in terms of adversarial robustness.

In Table 4.2 we consider the same architectures used to perform the experiments in Section 4.3.2 and perform FGSM and PGD adversarial attacks on the three models under comparison. Training hyper-parameters are reported in Table A.3 in the appendix. As expected, Bayesian NNs are more robust than deterministic ones. Moreover, we find that deep ensembles and deterministic NNs are comparable in terms of robustness, suggesting that simply averaging predictions for different weight initialization and mini-batching is not enough to achieve a robust model.

Model	**Test accuracy**	**FGSM accuracy**	**PGD accuracy**
Deterministic NN	97.69	21.19	1.45
Ensemble NN	99.4	20.6	0.3
Bayesian NN	96.1	90.0	89.8

Table 4.2: FGSM and PGD attacks on the network employed in Section 4.3.2. We compare a deterministic NN, a deep ensemble NN (of size 100), and a BNN (trained with VI). Attacks are performed on 1k test points from the MNIST dataset. We observe that VI-trained network achieves better robustness against PGD and FGSM.

4.4 Final Considerations

Our results guarantee that, in the overparameterized and infinite data limit, BNNs' posteriors are robust to gradient-based adversarial attacks even if each neural network sampled from the posterior is vulnerable to such attacks.

We believe that the fact that Bayesian ensembles of NNs can evade a broad class of adversarial attacks will be of great relevance. While promising, this result comes with some significant limitations. First, our theoretical results hold in a thermodynamic limit, which is never realised in practice. More worryingly, we currently have no rigorous diagnostics to understand how near we are to the limit case, and can only reason about this empirically. Moreover, learning accurate BNNs on more complex datasets is extremely challenging, which makes the Bayesian scheme currently not suitable for large-scale applications. This suggests the need for further investigations on such matters, especially on sufficiently accurate and scalable approximate inference methods for BNNs [170]. However, in our experiments cheaper approximations, such as VI, also enjoyed a degree of adversarial robustness.

Chapter 5

Defending through Random Projections

Contents

Random projections of the input samples into lower-dimensional spaces have been extensively used for dimensionality reduction, but we are primarily interested in using them as a defence strategy against adversarial attacks. Randomization has been proven effective as a defence [73, 6], detection [48] and regularization [51] strategy against adversarial attacks. [175] apply two random transformations to the input images, [98] add random noise between the layers of the architecture, [45] randomly prune activations between the layers, [178] and [96] propose input denoising and feature denoising methods. The reasoning behind these techniques is that NNs are usually robust to random perturbations [134], thus incorporating them in the models might weaken adversarial perturbations.

We also explore the geometrical properties related to randomization in a high-dimensional setting and further investigate the role of codimension in the generation of adversarial examples, which was already examined from a Bayesian perspective in Chapter 4. We recall that [84] suggests that adversarial perturbations mainly arise in the directions that are normal to the data manifold, hence as the codimension in the embedding space increases there is a higher number of directions in which one could build adversarial perturbations. Our framework is strongly influenced by this finding, as it suggests that by randomly selecting directions it should be more likely to catch features that are significant in the adversarial context. Moreover, we experimentally observe that projected versions of the original data are easier to learn and lie in less complex regions of the space.

We propose two training techniques to improve the robustness of deterministic neural networks to adversarial attacks. Both methods are independent of the chosen attack and leverage random projections of the original inputs, to exploit both dimensionality reduction and some characteristic geometrical properties of adversarial perturbations. The first technique is called *RP-Ensemble* and consists of an ensemble of networks trained on multiple projected versions of the original inputs. The second one, namely *RP-Regularizer*, leverages random projections to build a regularization term for the training objective. We experimentally observe that both techniques can catch relevant features of the inputs in terms of adversarial robustness and improve the adversarial robustness of deterministic architectures trained on MNIST and Fashion MNIST datasets via stochastic gradient descent.

5.1 Random Projections Ensemble

Random Projection Ensemble (RP-Ensemble) model builds a neural network ensemble upon a pre-trained model and can be interpreted as a fine-tuning technique for adversarial robustness. RP-Ensemble projects the input data in multiple lower dimensional spaces, each one determined by a random selection of directions in the space, to exploit input features that are able to guarantee robustness to the adversaries. Then, it trains a classifier on each projected dataset and it classifies the original data by aggregating the individual predictions from the ensemble and the original model.

We take into account N training examples from the data manifold $\mathcal{M} \subset \mathbb{R}^d$, represented in matrix form as $X \in \mathbb{R}^{N \times d}$. Let $g : \mathcal{M} \times \mathbb{R}^{n_{\mathbf{w}}} \to \mathbb{R}^K$ be a pre-trained neural network architecture with weights $\mathbf{w} \in \mathbb{R}^{n_{\mathbf{w}}}$, solving a classification problem on K classes. First of all, we project the whole dataset X into p different subspaces of dimension $k_j \leq d$, for $k_j = 1, \ldots, p$, using Gaussian random projection matrices [41]. Each projection matrix $Q_j \in \mathbb{R}^{k_j \times d}$ maps the input data matrix X into its k_j-dimensional projected version $\mathcal{P}_j(X) = XQ_j^T \in \mathbb{R}^{N \times k_j}$, using k_j randomly selected directions. The elements of each random matrix Q_j are independently drawn from a $\mathcal{N}(0, 1/k_j)$ distribution. This particular choice is motivated by the Johnson-Lindenstrauss lemma (Lemma 5.1), ensuring that the Euclidean distance between any two points in the new low-dimensional space is approximately equal to the distance between the same points in the original high-dimensional space [41]. Any two independently randomly chosen vectors in a high dimensional space are almost orthogonal and nearly have the same length [83]; therefore, for any given number of sample points N, the k_j-dimensional columns of a random matrix Q_j generated from a Gaussian distribution are approximately orthogonal to each other. This procedure yields a projection in the k_j-dimensional subspace generated by the columns of Q_j.

Lemma 5.1 (Johnson-Lindenstrauss lemma)**.** *Given* $0 < \epsilon < 1/2$*, a set* D_N *of* N *data points from the data manifold* $\mathcal{M} \subset \mathbb{R}^d$ *and an integer* $k > 8 \log N/\epsilon^2$*, let* $Q \in \mathbb{R}^{k \times d}$ *be a matrix whose entries have been sampled independently from* $\mathcal{N}(0, 1/k)$*. For any couple of points* $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{M}$ *the following inequality holds*

$$P\Big[(1-\epsilon)||\mathbf{x}_1 - \mathbf{x}_2||^2 \leq \big|\big|Q\mathbf{x}_1 - Q\mathbf{x}_2\big|\big|^2 \leq (1+\epsilon)||\mathbf{x}_1 - \mathbf{x}_2||^2\Big] \geq 1 - 2e^{-(\epsilon^2 - \epsilon^3)k/4}.$$

Next, we train a classifier $\psi_j : \mathbb{R}^{k_j} \times \mathbb{R}^{n_{\mathbf{w}_j}} \to \mathbb{R}^K$ in each projected subspace on the corresponding data projections $\mathcal{P}_j(X)$, where $n_{\mathbf{w}_j}$ is the size of the j-th weight space. The architecture of the ψ_j-s mirrors that of the pre-trained network g, except for the first layer, whose size is adapted to the new input dimension k_j. We emphasize that the ψ_j-s do not share their weights nor the inputs, thus backpropagation algorithm during the training phase stops at the projected dataset $\mathcal{P}_j(X)$.

Finally, we perform an ensemble classification on the original high dimensional data, by averaging the predictions from the projected classifiers ψ_j together with the predictions from the pre-trained classifier g. Let $p_{\mathbf{w}_j}, p_g$ be the probability mass functions for the classifiers ψ_j and g. The classification of an input sample $\mathbf{x} \in D_N$ is given by

$$\underset{y=1,\dots,K}{\arg\max} \Bigg(\sum_{j=1,\dots,p} p_{\mathbf{w}_j}(y|\mathcal{P}_j(\mathbf{x})) + p_g(y|\mathbf{x}) \Bigg).$$

Notice that the number of projections p and the size of projections k_j for $j = 1, \dots, p$ are additional hyperparameters of the model.

5.2 Random Projections Regularizer

[60] show that regularization of the loss gradient on the inputs improves adversarial robustness. In particular, they notice that the *Total Variation* regularization [143] can be interpreted as the regularization induced by a single step of adversarial training on gradient-based attacks.

Random Projections Regularizer (RP-Regularizer) is a variant of Total Variation (TV) regularization [143], a well-known denoising approach in image processing, already used in [60] as a regularization term to improve adversarial robustness. The general formulation of total variation for a real-valued continuously differentiable function f involves integration over an open subset of $\mathbb{R}^d$; Therefore, to guarantee its tractability, it is usual practice to compute instead a numerical approximation on a finite set of points. In the image classification setup, we consider the approximation

$$\mathrm{TV}(f) \approx \frac{1}{N} \sum_{i=1}^{N} \left|\nabla_{\mathbf{x}} \mathcal{L}(f(\mathbf{x}_i, \mathbf{w}), y_i)\right|,$$

on a labeled dataset $D_N = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,N}$, where $\mathcal{L}$ is the training loss function (e.g. cross-entropy loss) and N is the number of training samples.

Our regularization term builds upon TV regularization and random projections of the input points according to the following procedure. First, we sample the components of the random matrices $R_j \in \mathbb{Q}^{k_j \times d}$ from a Gaussian distribution, with a randomly chosen projection sizes k_j, for $j = 1, \dots, p$. Next, we project the input data matrix X in the k_j-dimensional subspaces

$$\mathcal{P}_j(X) = XQ_j^T \in \mathbb{R}^{N \times k_j},$$

and we map the projections $\mathcal{P}_j(X)$ back into the original d-dimensional space by means of the *Moore-Penrose pseudo-inverse* [127] $R_j^\dagger$ of R_j. Specifically, we apply it to the projected points

$$\mathcal{P}_j^\dagger(\mathcal{P}_j(X)) = XQ_j^T(Q_j^\dagger)^T \in \mathbb{R}^{N \times d}.$$

The pseudo-inverse is a generalized inverse matrix. It exists and is unique for any given real rectangular matrix and the resulting composition $Q_j^T(Q_j^\dagger)^T$ is an orthogonal projection operator on $\mathbb{R}^d$. Figure 5.1 shows an example of this procedure on the MNIST dataset [92]. In a nutshell, it builds on the two projection operators

$$\mathcal{P} : \mathbb{R}^{N \times d} \longrightarrow \prod_{j=1}^{p} \mathbb{R}^{N \times k_j}$$
$$\mathcal{P}^\dagger : \prod_{j=1}^{p} \mathbb{R}^{N \times k_j} \longrightarrow \prod_{j=1}^{p} \mathbb{R}^{N \times d}.$$

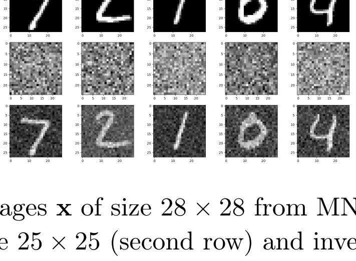


Figure 5.1: Original images $\mathbf{x}$ of size 28×28 from MNIST dataset (first row), projections $\mathcal{P}_j(\mathbf{x})$ of size 25×25 (second row) and inverse projections $\mathcal{P}_j^\dagger \mathcal{P}_j(\mathbf{x})$ (third row).

The last step allows us to define the penalty term for the objective function in terms of the network's weights at the current training step. Specifically, given the training loss function $\mathcal{L}$, we propose two distinct formulations for the regularization term $\mathcal{R}(\mathbf{w})$ of the objective $J(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \lambda \cdot \mathcal{R}(\mathbf{w})$ on a set of weights $\mathbf{w}$. The first one, namely $\mathcal{R}_{\text{v1}}$, adds a penalty which is proportional to the expected norm of loss gradients computed on the projected data.

$$\begin{aligned}\mathcal{R}_{\text{v1}} &= \mathbb{E}_{\mathbf{x}}\Big[\mathbb{E}_{\mathcal{P}}\Big[\big|\big|\nabla_{\mathbf{x}}\mathcal{L}(f(\mathcal{P}^{\dagger}\mathcal{P}(\mathbf{x}), \mathbf{w}), y)\big|\big|_2^2\Big]\Big] \\ &\approx \frac{1}{Np}\sum_{\substack{i=1,\dots,N \\ j=1,\dots,p}} \Big|\Big|\nabla_{\mathbf{x}}\mathcal{L}(f(\mathcal{P}_j^{\dagger}\mathcal{P}_j(\mathbf{x}_i), \mathbf{w}), y_i)\Big|\Big|_2^2. \end{aligned} \tag{5.1}$$

The natural interpretation of $\mathcal{R}_{\text{v1}}$ is that it allows us to minimize loss variation across the inverse projections $\mathcal{P}_j^{\dagger}\mathcal{P}_j(\mathbf{x}_i)$-s. $\mathcal{R}_{\text{v2}}$, instead, minimizes the variation of the loss gradients on the original inputs in randomly chosen projected subspaces:

$$\begin{aligned}\mathcal{R}_{\text{v2}} &= \mathbb{E}_{\mathbf{x}}\Big[\mathbb{E}_{\mathcal{P}}\Big[\big|\big|\mathcal{P}(\nabla_{\mathbf{x}}\mathcal{L}(f(\mathbf{x}, \mathbf{w}), y))\big|\big|_2^2\Big]\Big] \\ &\approx \frac{1}{Np}\sum_{\substack{i=1,\dots,N \\ j=1,\dots,p}} \Big|\Big|\mathcal{P}_j\Big(\nabla_{\mathbf{x}}\mathcal{L}(f(\mathbf{x}_i, \mathbf{w}), y_i)\Big)\Big|\Big|_2^2. \end{aligned} \tag{5.2}$$

At each training step we perform a finite approximation of the expectations on mini-batches of data, by randomly sampling the directions, the dimension of the projected subspace and the number of projections.

5.2.1 Equivalence of the Regularizers

In this section we prove that the two regularization terms $\mathcal{R}_{\text{v1}}$ and $\mathcal{R}_{\text{v2}}$ are equivalent as $k \to \infty$.

Theorem 5.1. *Let $\mathcal{R}_{v1}$ and $\mathcal{R}_{v2}$ be the regularization terms defined by Equation 5.1 and Equation 5.2, where $\mathcal{P} : \mathbb{R}^d \to \mathbb{R}^k$ is a random projection such that the elements of the orthogonal random matrix Q are sampled from $\mathcal{N}(0, 1/k)$. If $k \in O(d)$ then $\mathcal{R}_{v1} \approx \mathcal{R}_{v2}$ as $k \to \infty$.*

Let $Q : \mathbb{R}^d \to \mathbb{R}^k$ be a random projection matrix, $Q^{\dagger} : \mathbb{R}^k \to \mathbb{R}^d$ its pseudo-inverse and $\mathbf{x}^{\dagger} := Q^{\dagger}Q\mathbf{x} \in \mathbb{R}^d$ for $\mathbf{x} \in \mathbb{R}^d$. For any given couple $(\mathbf{x}, y) \in \mathbb{R}^d \times \mathbb{R}^K$ and projection matrix Q let us define

$$\begin{aligned}\mathcal{T}_1(R) &:= \big|\big|\nabla_{\mathbf{x}}\mathcal{L}\big(f(\mathbf{x}^{\dagger}, \mathbf{w}), y\big)\big|\big|_2^2 \\ \mathcal{T}_2(R) &:= \big|\big|Q \cdot \nabla_{\mathbf{x}}\mathcal{L}\big(f(\mathbf{x}, \mathbf{w}), y\big)\big|\big|_2^2.\end{aligned}$$

Proposition 5.1. *Let Q be a random projection matrix whose elements are sampled from $\mathcal{N}(0,1/k)$ and whose columns are orthogonal. Suppose that $\mathbf{x} \in (\ker Q)^{\perp}$ for all $\mathbf{x} \in \mathbb{R}^d$. Then*

$$\mathbb{E}_Q\big[\mathcal{T}_2(Q)\big] = \frac{d}{k}\,\mathbb{E}_Q\big[\mathcal{T}_1(Q)\big].$$

Proof. A random matrix Q and its pseudo-inverse $Q^{\dagger}$ induce the direct sum decompositions

$$\begin{aligned}\mathbb{R}^d &= (\ker Q)^{\perp} \oplus \ker Q\\ \mathbb{R}^k &= \text{rank } Q \oplus (\text{rank}\, Q)^{\perp},\end{aligned}$$

where $\ker Q$ is the kernel space of Q and $\text{rank}\, Q$ is the rank space of Q. Moreover, $Q\big|_{(\ker Q)^{\perp}}$ is an isomorphism with inverse $Q^{\dagger}\big|_{\text{rank}\, Q}$ and $Q^{\dagger}\big|_{(\text{rank}\, Q)^{\perp}} \equiv 0$. If $\mathbf{x} \in (\ker Q)^{\perp}$, then $Q\mathbf{x} \in \text{rank } Q$ and $\mathbf{x}^{\dagger} = Q^{\dagger}Q\mathbf{x} = \mathbf{x}$, therefore

$$\mathcal{T}_1(Q) = \big|\big|\nabla_{\mathbf{x}}\mathcal{L}\big(f(\mathbf{x},\mathbf{w}),y\big)\big|\big|_2^2$$

for all $\mathbf{x} \in (\ker Q)^{\perp}$.

Let $q_i \in \mathbb{R}^k$ be the orthogonal columns of Q. Then $\mathbb{E}_Q\big[||Q||^2\big] = d/k$ and

$$\mathbb{E}_Q\big[||Q\mathbf{x}||_2^2\big] = \sum_{i,j=1}^{d} \mathbb{E}_Q\big[q_i^T q_j\big]\mathbf{x}_i\mathbf{x}_j = \sum_{i=1}^{d} \mathbb{E}_Q\big[q_i^T q_i\big]\mathbf{x}_i^2 = \frac{d}{k}||\mathbf{x}||_2^2$$

for any $\mathbf{x} \in \mathbb{R}^d$. In particular $\mathbb{E}_Q\big[\mathcal{T}_2(Q)\big] = \frac{d}{k}\mathcal{T}_1(Q)$.

□

Notice that when $\mathbb{R}^k$ is high dimensional we can assume that the columns of any random matrix are orthogonal [165].

We now prove that Proposition 5.1 holds for an arbitrary $\mathbf{x} \in \mathbb{R}^d$.

Proposition 5.2. *Let $\pi_R : \mathbb{R}^d \to (\ker Q)^{\perp}$ be an orthogonal projection and $k = \dim(\ker Q)^{\perp}$. Then, for any $\mathbf{x} \in \mathbb{R}^d$ and $\epsilon > 0$*

$$P\big(||\pi_Q(\mathbf{x}) - \mathbf{x}||_2^2 > \epsilon\, ||\mathbf{x}||_2^2\big) \le \Big(1 - \frac{\epsilon}{\pi}\Big)^k.$$

Proof. Suppose that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subset \mathbb{R}^d$ is a basis for $(\ker Q)^{\perp}$, i.e. that $(\ker Q)^{\perp} = \text{span}(\mathbf{v}_1, \dots, \mathbf{v}_k)$. Then any $\mathbf{x} \in \mathbb{R}^d$ can be decomposed as $\mathbf{x} = \pi_Q(\mathbf{x}) + \mathbf{u}$, where $\pi_Q(\mathbf{x}) \in (\ker Q)^{\perp}$ and $\mathbf{u} \in \ker Q$.

Let α_i be the angle between $\mathbf{v}_i$ and $\mathbf{x}$. First, we observe that the projection $\pi_Q(\mathbf{x})$ is smaller than any other projection on a single direction $\mathbf{v}_i$

$$\begin{aligned}||\pi_Q(\mathbf{x}) - \mathbf{x}||_2^2 &\le \min_i ||\pi_{\mathbf{v}_i}(\mathbf{x}) - \mathbf{x}||_2^2\\ &= \min_i \big(||\mathbf{x}||_2^2 |\sin\alpha_i|\big)\\ &= ||\mathbf{x}||_2^2 \min_i |\sin\alpha_i|.\end{aligned}$$

For any choice of $\epsilon > 0$

$$\begin{aligned}
P\big(||\pi_Q(\mathbf{x}) - \mathbf{x}||_2^2 > \epsilon\, ||\mathbf{x}||_2^2\big) &\leq P\big(||\mathbf{x}||_2^2 \min_i |\sin \alpha_i| > ||\mathbf{x}||_2^2\, \epsilon\big) \\
&= P(\min_i |\sin \alpha_i| > \epsilon) \\
&= \prod_i P(|\sin \alpha_i| > \epsilon) \\
&\leq \prod_i P(|\alpha_i| > \epsilon) \\
&= \Big(1 - \frac{\epsilon}{\pi}\Big)^k.
\end{aligned}$$

□

Notice that $1 - \frac{\epsilon}{\pi} < 1$, so $\big(1 - \frac{\epsilon}{\pi}\big)^k \to 0$ as k goes to ∞. Therefore, from 5.2 we get $\mathbf{x}^\dagger = \pi_Q(\mathbf{x}) \approx \mathbf{x}$ and

$$\nabla_{\mathbf{x}} \mathcal{L}\big(f(\mathbf{x}^\dagger, \mathbf{w}), y\big) \approx \nabla_{\mathbf{x}} \mathcal{L}\big(f(\mathbf{x}, \mathbf{w}), y\big)$$

as $k \to \infty$. This proves that Proposition 5.1 is true for an arbitrary $\mathbf{x} \in \mathbb{R}^d$ in the limit. Assuming that $k = O(d)$ as $k \to \infty$, e.g. $\frac{d}{k} \to M > 0$, the two regularization terms differ by a positive constant in the limit, i.e. they are equivalent if weighted w.r.t. M. This proves Theorem 5.1.

Notice that this punctual property on $\mathcal{R}_{v1}$ and $\mathcal{R}_{v2}$ also holds in expectation over the training data when $\mathbf{x}$ is uniformly sampled from a compact subset of $\mathbb{R}^d$. Therefore, the equivalence between the two regularization terms holds in practice on mini-batches of training data.

5.3 Transferability of RP Defense Strategies

One of the simplest and most effective approaches to learn robust deterministic NNs is *adversarial training* [66]. This process consists in training a classifier by including adversarial examples in the training data, thus allowing to convert any attack into a defense. The biggest limitation of this procedure is that it tends to overfit the chosen attack, meaning that the adversarially trained model has better performances against the attacks which they learned to defend from, but might show poor results against other threats. This problem is easily tackled by our methods, which are completely unaware of the nature of the attack, yet able to improve robustness. Furthermore, adversarial training can be roughly interpreted as a form of data augmentation, with significant differences w.r.t. the more traditional approaches: instead of applying transformations that are expected to occur in the test set (translations, rotations, etc.), only the most unlikely examples are added to the training dataset. This produces a dilation of the data manifold: adversarial examples are learned in a halo around the surface, which makes the manifold smoother [50]. In this regard, we emphasize that RP-Ensemble does not perform any data augmentation in the original high dimensional space, since the projected data samples lie in new subspaces. RP-Regularizer, instead, produces new high-dimensional examples, which could be intended as perturbations of the original samples, but not generated through a specific attack methodology.

Moreover, the robustness of classifiers is strongly related to the geometry of the learned decision boundaries. Indeed, to learn robust decision boundaries a model has to correctly classify all the input points lying in a neighbourhood of the data manifold. In particular, adversarially perturbed points always lie extremely close to the decision boundaries [50], but robustness conditions change under different p-norms, meaning that no single decision boundary can be optimally robust in all norms [84]. For instance, if a classifier is trained to be robust under L_∞ norm, poor robustness under the L_2 norm should be expected. In general, no distance metric can be considered a perfect measure of similarity [34], so one of the strengths of our RP methods is that they are independent of the norm chosen for the attacks.

5.4 Experimental Results

We evaluate the proposed methods on neural networks for image classification with 10 classes, using MNIST [92] and CIFAR-10 [88] datasets. Our baseline models are Convolutional Neural Networks with ReLU activation functions. We achieve 99.13% accuracy on MNIST and 76.52% accuracy on CIFAR-10. Architectures and hyperparameters are reported in Table A.7 in the appendix. The adversarial attacks in our tests are Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), DeepFool, and Carlini and Wagner (C&W) in the L_∞ norm, presented in Section 3.4. Across all the experiments, the attack strength is set to $\epsilon = 0.3$ and $\mathcal{L}$ is the cross-entropy loss function.

Simulations[1] are conducted on a machine with 34 single core Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz processors and 200GB of RAM. We make an extensive use of Tensorflow [1] and IBM adversarial-robustness-toolbox [118] libraries.

5.4.1 Improved Adversarial Robustness

To assess the improvement in robustness achieved by our techniques we perform the following steps: (1) we build several adversarial attacks against a baseline model; (2) we use the attacked training sets to perform adversarial training on the baseline model; (3) we test the baseline model, the adversarially trained model, our RP-Ensemble model and our models trained with RP-Regularizers model against the original adversarial attacks. Employing this procedure we can investigate the generalization capabilities of our methods, which are completely unaware of the chosen attacks, and compare them to models that should exhibit ideal performances against the adversaries, i.e. the adversarially trained ones.

We train multiple versions of RP-Ensemble, using different combinations of the number of projections and the size of projections, as described in Table 5.1.

[1]Code is available at `https://github.com/ginevracoal/adversarial_examples`.

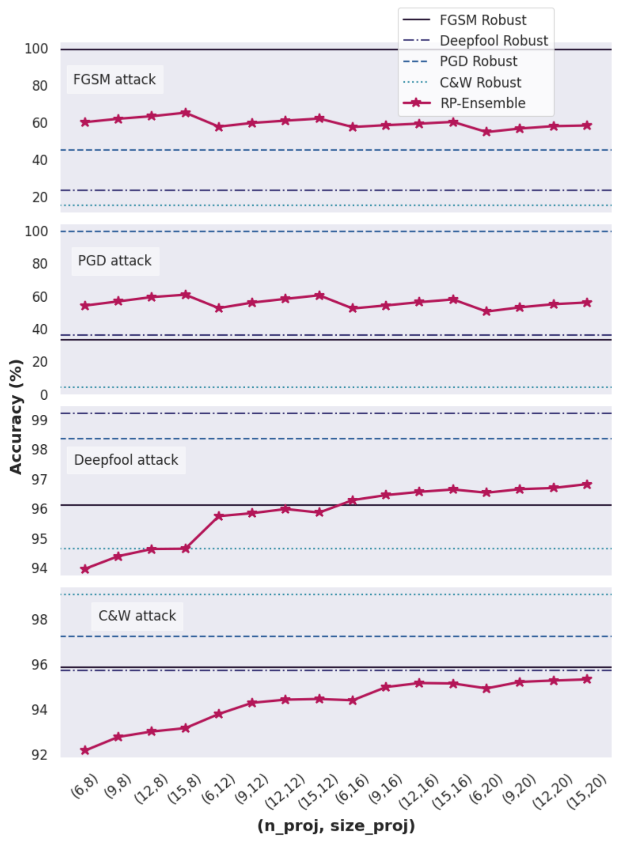


Figure 5.2: Test accuracy of RP-Ensemble model and the adversarially trained models against adversarial attacks crafted on MNIST dataset. The robust models are the result of adversarial training performed on the perturbed training sets. Multiple RP-Ensemble models are trained on different combinations of the number of projections and the size of each projection. The evaluations are performed on multiple adversarially perturbed versions of the test set, using FGSM, PGD, DeepFool and Carlini & Wagner attacks.

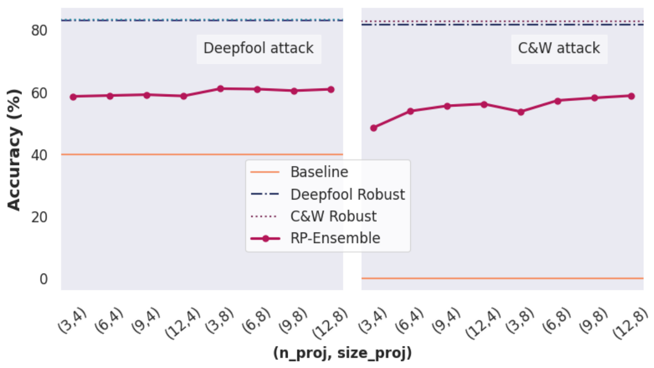


Figure 5.3: Test accuracy of RP-Ensemble model, the baseline model and the adversarially trained models against adversarial attacks crafted on MNIST dataset. The robust models are the result of adversarial training performed on the perturbed training sets. Multiple RP-Ensemble models are trained on different combinations of the number of projections and the size of each projection (x axis). The evaluations are performed on multiple adversarially perturbed versions of the test set, using DeepFool and Carlini & Wagner attacks.

In our experiments, each classifier ψ_j is indexed by the seed j used to sample the projection matrix Q_j. We train an RP-Regularized model on each dataset, by uniformly sampling the number of projections and the size of projections at each training step, as reported in Table B.4. To ensure numerical stability, we compute the pseudo-inverse matrix $Q_j^\dagger$ by means of the SVD decomposition of Q_j.

Dataset	**Number of projections**	**Projection size**
MNIST	$6, 9, 12, 15$	$8, 12, 16, 20$
CIFAR-10	$3, 6, 9, 12$	$4, 8$

Table 5.1: Number of projections and size of each projection used in RP-Ensemble model.

Dataset	**Number of projections**	**Projection size**
MNIST	`n_proj` $\sim \mathcal{U}(2, 8)$	`size_proj`$\sim \mathcal{U}(15, 25)$
CIFAR-10	`n_proj`$= 1$	`size_proj`$\sim \mathcal{U}(5, 10)$

Table 5.2: Number of projections and size of each projection used in RP-Regularized models.

We craft adversarial perturbations on the original test set using the baseline model, then test the robustness of RP-Ensemble to the adversaries in terms of prediction accuracy. We perform the experiments on $10k$ test images from MNIST (Figure 5.2) and CIFAR-10 (Figure 5.3) datasets. In Figure 5.2 we do not report the prediction accuracy of the baseline model against the attacks, since the very low values would affect the readability of the figure. However, we report all the exact numerical values for the prediction accuracy in Tables B.1 and B.2 of the appendix. RP-Ensemble brings a general improvement in the adversarial robustness of the baseline model. Adversarially trained robust models show great results on their target attacks (i.e. the ones on which they were trained) but perform poorly on the other ones, while RP-Ensemble preserves its robustness across the different attacks.

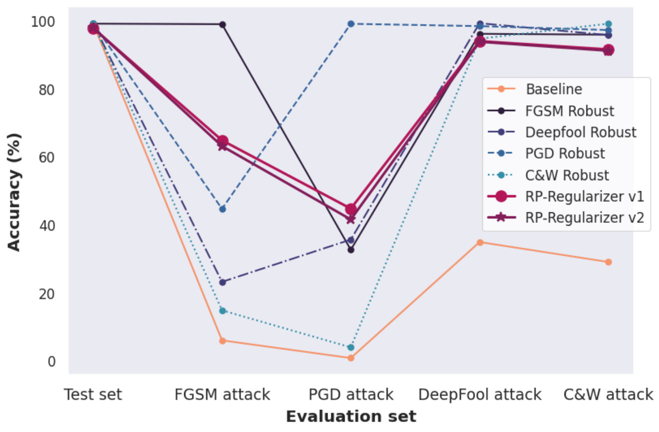


Figure 5.4: Test accuracy of models trained with RP-Regularizer on MNIST. We compare the baseline model, the adversarially trained robust models and the models trained with the two RP-Regularizers, namely $\mathcal{R}_{v1}$ (Eq. 5.1) and $\mathcal{R}_{v2}$ (Eq. 5.2). Adversarial perturbations are produced on the baseline model using FGSM, PGD, Deepfool and Carlini & Wagner attacks.

Models trained with RP-Regularizers are comparable to SOTA adversarially trained models against the attacks performed on MNIST (Figure 5.4). Prediction accuracies are higher on DeepFool and Carlini & Wagner attacks than on FGSM and PGD, suggesting that this method performs better on algorithms which are optimized to produce perturbations that are closer to the original samples (e.g. C&W), rather than faster in computation (e.g. FGSM). The results are less striking on CIFAR-10 (Fig. 5.5), but we stress that the robustness of RP-Regularized models improves as the number of projections increases and that in the case of CIFAR-10 we compromised on the choice of this parameter, and always computed a single projection, intending to achieve good adversarial robustness while also preserving computational efficiency. The trade-off between these two objectives needs to be further explored.

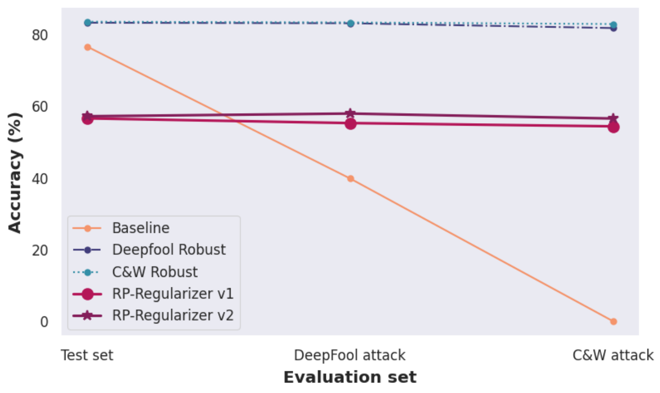


Figure 5.5: Test accuracy of models trained with RP-Regularizer on CIFAR-10. We compare the baseline model, the adversarially trained robust models and the models trained with the two RP-Regularizers, namely $\mathcal{R}_{\text{v1}}$ (Eq. 5.1) and $\mathcal{R}_{\text{v2}}$ (Eq. 5.2). Adversarial perturbations are produced on the baseline model using FGSM, PGD, Deepfool and Carlini & Wagner attacks.

5.4.2 Computational Efficiency of RP-Ensemble

The classifiers ψ_j from PR-Ensemble are defined upon independent projected subspaces, thus their training is completely parallelizable. This ensures the computational efficiency of RP-Ensemble, indeed we observe a training time close to that of the baseline, or even lower when the number of projections is sufficiently small (Figure 5.6).

We emphasise that RP-Ensemble preserves high adversarial robustness despite the heavy dimensionality reduction on the inputs. This is probably because, thanks to Johnson-Lindenstrauss Lemma, pairwise distances between the projected points are nearly preserved in the projected subspaces, therefore the projection classifiers ψ_j are able to maintain features which turn out to be a significant defence strategy against the attacks.

5.5 Final Considerations

We empirically show that random projections of the training data act as attack-independent adversarial features, that can be used to provide better resilience to adversarial perturbations. We propose a fine-tuning method and a regularization method, both based on the computation of random projections of the inputs. In future work, we plan to improve the computational cost of RP-Regularizer and to compare the performances of our methods to that of other attack-independent defence strategies. We believe that further exploration of the connections between random projections and the geometrical characterization of adversarial regions could bring valuable insights to adversarial defence research.

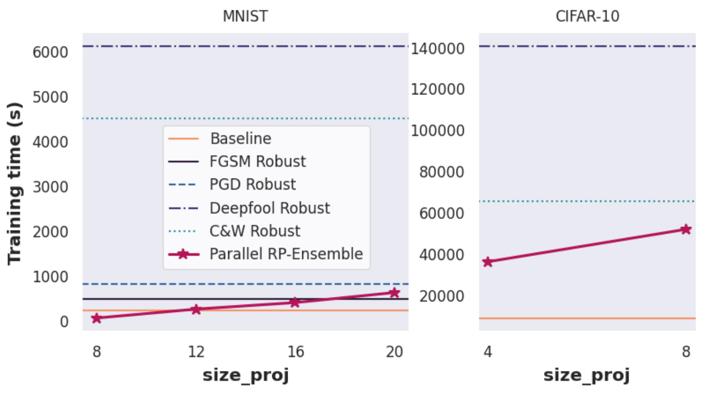


Figure 5.6: The training time of RP-Ensemble on MNIST and CIFAR-10. We compare its efficiency to that of the baseline models and the adversarially trained robust models.

Chapter 6

Stability of Bayesian Saliency Map Explanations

Contents

Adversarial machine learning has been extensively studied from the perspective of interpretability. Many works focus on adversarial manipulations of the explanations [74, 63, 46], which consist in altering the explanations without affecting models' predictions. Interestingly, [137] notice that aggregations of multiple explanation methods are less vulnerable to adversarial attacks. It has been observed that there is a strong connection between adversarial robustness and explainability of neural networks [157]. For instance, [47] use an adversarial training [66, 102] to improve the interpretability of the representations, while [140] present a regularization technique based on gradient smoothing, which favours adversarial robustness and interpretability simultaneously. Adversarial training is one of the most effective defence methods against attacks, but it has also been used to improve the sensitivity of the explanations to input perturbations [180]. In contrast to the previous studies, we provide a new concept of robustness of the explanations, where we compare the interpretation of an input to that of an adversarial attack in terms of common *most relevant* pixels (Section 6.1).

We focus on LRP saliency interpretations, which are well known to be unstable under perturbations of the inputs [63, 85, 4, 184]. A recent work [24] investigates how the stability of the interpretations could be improved by adding stochasticity to the model weights. Their *NoiseGrad* method relies on a tempered Bayes posterior [170] and aggregates the explanations provided by independent samples to produce a Bayesian explanation. Our idea that Bayesian neural networks could provide more stable explanations, instead, is motivated by the adversarial robustness of BNNs to gradient-based attacks, proved in the previous chapter. Hence, we focus on gradient-based attacks and also provide a formal proof for the stability of LRP explanations (Corollary 6.1), which holds w.l.o.g. for any other gradient-based attribution method. Furthermore, we empirically evaluate our findings using VI and HMC approximate inference methods, presented in Section 3.3.

Here we consider the problem of the stability of saliency-based explanations of neural network predictions under adversarial attacks in a classification task. Saliency interpretations of deterministic neural networks are remarkably brittle even when the attacks fail, i.e. for attacks that do not change the classification label. We empirically show that interpretations provided by Bayesian neural networks are considerably more stable under adversarial perturbations of the inputs and even under direct attacks to the explanations. By leveraging recent results, we also provide a theoretical explanation of this result in terms of the geometry of the data manifold. Additionally, we discuss the stability of the interpretations of high-level representations of the inputs in the internal layers of a network. Our results demonstrate that Bayesian methods, in addition to being more robust to adversarial attacks, have the potential to provide more stable and interpretable assessments of neural networks' predictions.

6.1 LRP Robustness

We define the *k-LRP robustness* of relevance heatmaps to adversarial attacks and use this measure to assess how adversarial perturbations of the inputs affect the explanations.

Definition 6.1. *Let $\mathbf{x}$ be an image with relevance heatmap $R(\mathbf{x},\mathbf{w})$ and let $\tilde{\mathbf{x}}$ be an adversarial perturbation with relevance heatmap $R(\tilde{\mathbf{x}},\mathbf{w})$. Let $Top_k(R)$ denote the pixel indexes corresponding to the top $k\%$ most relevant pixels in the absolute value of a heatmap R. The* k-LRP robustness *of $\mathbf{x}$ w.r.t. the attack $\tilde{\mathbf{x}}$ is*

$$k\text{-}LRP(\mathbf{x},\tilde{\mathbf{x}},\mathbf{w}) := \frac{|\,Top_k(R(\mathbf{x},\mathbf{w})) \cap Top_k(R(\tilde{\mathbf{x}},\mathbf{w}))|}{k}. \tag{6.1}$$

In other words, the $\text{Top}_k(R)$ pixels have a strong positive or negative impact on classification and k-LRP$(\mathbf{x},\tilde{\mathbf{x}},\mathbf{w})$ is the fraction of common most relevant pixels for $\mathbf{x}$ and $\tilde{\mathbf{x}}$ in the top $k\%$. Figure 6.1 gives an intuition of this computation. Notice that the LRP robustness of a point depends only implicitly on the strength ϵ of the attack, through the attacked point $\tilde{\mathbf{x}}$.

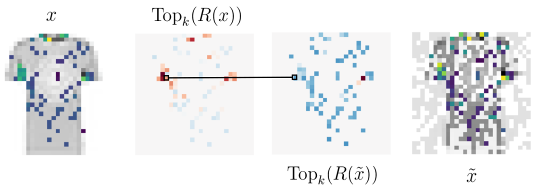


Figure 6.1: Top_k pixels in an image $\mathbf{x}$ from Fashion MNIST dataset and an FGSM adversarial perturbation $\tilde{\mathbf{x}}$.

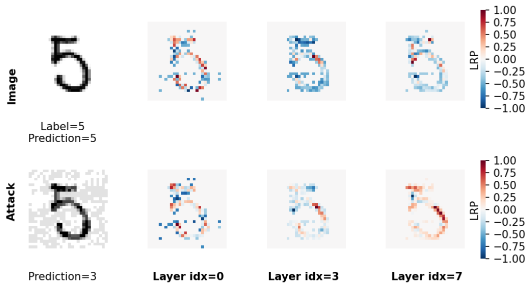


Figure 6.2: LRP heatmaps of an image $\mathbf{x}$ and an FGSM adversarial perturbation $\tilde{\mathbf{x}}$. Explanations are computed using the epsilon rule w.r.t. the learnable layers of a deterministic network trained on the MNIST dataset. For each layer, we show the 30% most relevant pixels for both the original image and its adversarial counterpart, i.e. the ones selected for the computation of k-LRP$(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{w}, l)$ from Equation (6.1).

Inner Layers Explanations We analyse the behaviour of LRP representations in the internal layers of the network, thus we also extend the computation of LRP heatmaps to any feature representation of the input $\mathbf{x}$ at a learnable layer $l \in \mathbb{N}$. We denote it by $R(\mathbf{x}, \mathbf{w}, l)$, where $l \leq L$ and L is the total number of layers available in the architecture. The corresponding LRP robustness will be denoted by k-LRP$(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{w}, l)$. In such case, the robustness does not refer anymore to explanations in the classification phase (pre-softmax layer), but rather to the explanations in the learning phases, hence it gives an idea of the most relevant pixels determining an internal representation.

Figure 6.2 shows an example of internal LRP heatmaps on a deterministic NN with learnable layers indexed by $l \in [0, 3, 7]$. For illustrative purposes, heatmaps appearing on the same row are normalized in $[-1, 1]$ range before selecting the Top_k pixels, since numeric scales are significantly different across the different internal representations.

Bayesian LRP Robustness The notion of LRP robustness can be naturally generalised to the Bayesian setting using the concept of Bayesian model averaging introduced in Section 3.3. Hence, the LRP heatmap of a BNN is computed as the average of all the deterministic heatmaps from the ensemble: $\mathbb{E}_{p(\mathbf{w}|D)}[k\text{-LRP}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{w}, l)]$. In this regard, we emphasise that Bayesian interpretations are affected by the chosen number of posterior samples drawn from the learned distribution.

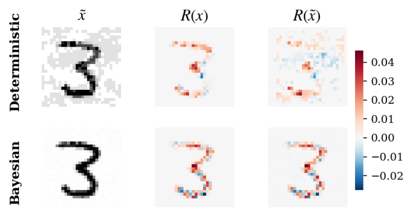


Figure 6.3: LRP heatmaps of an image $\mathbf{x}$ (second column) and an FGSM adversarial perturbation $\tilde{\mathbf{x}}$ (third column) which fails on a deterministic network and a Bayesian network. The two models have the same fully connected architecture and are both trained on the MNIST dataset. Explanations are computed using the epsilon rule on the pre-softmax layer, i.e. layer index = 5. Bayesian LRP is computed using 100 posterior samples. For each heatmap, we only show the 30% most relevant pixels. The LRP robustness amounts to 0.46 in the deterministic case and to 0.68 in the Bayesian case.

Results from Chapter 4 show that Bayesian neural networks are adversarially robust to gradient-based attacks in the overparameterized and infinite data limit. It is therefore of interest to investigate whether such robustness also extends to the learned explanations. To do so, we compare the explanations of deterministic NNs to that of Bayesian NNs against such attacks. Fig. 6.3 shows an example of failed FGSM attack for a deterministic network and a Bayesian network with the same architecture. Although the attack did not manage to change the overall classification, we can see immediately a large difference between the deterministic LRP explanation of the original image, $R(\mathbf{x})$, and of the adversarial image, $R(\tilde{\mathbf{x}})$ (top row). On the other hand, in the Bayesian case (bottom row), the saliency maps before and after the attack are essentially identical. We provide both a theoretical explanation of this phenomenon (Section 6.2) and systematically substantiate empirically (Section 6.3) the robustness of Bayesian explanations to adversarial attacks.

6.2 Geometric Meaning of Adversarial Interpretations

To better conceptualise the impact of a Bayesian treatment on LRP robustness, it is convenient to consider the thermodynamic limit of infinite data and infinite expressivity of the network, as formalised in Chapter 4. We recall that, for our discussion, the main ingredients are the data manifold $\mathcal{M}$, a piecewise smooth submanifold of the input space where the data lie, and the target function $f^{true} : \mathcal{M} \to \mathbb{R}^K$, which is assumed to be smooth and hence representable through an infinitely-wide DNN (Section 3.2.3). Practically, this limit might be well approximated on large data sets where the networks achieve high accuracy. In this limit, it is proved [49, 107, 141] that the DNN $f(\mathbf{x}, \mathbf{w})$ trained via SGD will converge to the true underlying function $f^{true}(\mathbf{x})$ over the whole data manifold $\mathcal{M}$. As in Chapter 4, w.l.o.g., we will consider a target function with values in $\mathbb{R}$. Because the data manifold is assumed to be piecewise smooth, it is possible to define a tangent space to the data manifold almost everywhere, and therefore to define two operators $\nabla^{\perp}_{\mathbf{x}}$ and $\nabla^{\parallel}_{\mathbf{x}}$, representing the gradients along the normal and tangent directions to the data manifold $\mathcal{M}$ at a point $\mathbf{x}$, for any function defined over the whole input space.

LRP and gradient-based adversarial strategies both share a reliance on gradient information. In the case of adversarial attacks, one evaluates the gradient of the loss function which, by the chain rule is given by

$$\nabla_{\mathbf{x}} \mathcal{L}(f, f^{true}) = \frac{\delta \mathcal{L}(f, f^{true})}{\delta f} \frac{\partial f}{\partial \mathbf{x}}. \tag{6.2}$$

The tangent components of the gradient of the prediction function $f(\mathbf{x}, \mathbf{w})$ will coincide with the gradients of the true function $f^{true}(\mathbf{x})$, and therefore represent directions of true sensitivity of the decision function, which are correctly recognised as relevant. However, such directions might be confounded or dwarfed by normal gradient components, which create directions of apparent relevance that, by construction, are targeted by gradient-based adversarial attacks.

From the discussion in Section 4.2 we trivially obtain that BNNs in the thermodynamic limit will only retain components that are tangent to the data manifold.

Corollary 6.1. Let $f(\mathbf{x}, \mathbf{w})$ be an infinitely-wide BNN trained on a data set D_N composed by N data points with true underlying function $f^{true} : \mathcal{M} \to \mathbb{R}$. Under the assumptions (1) and (2) in Theorem 4.1, for any $\mathbf{x} \in \mathcal{M}$ it holds that

$$\mathbb{E}_{p(\mathbf{w}|D)}[R(\mathbf{x}, \mathbf{w})] = \mathbb{E}_{p(\mathbf{w}|D)}[\nabla^{\parallel}_{\mathbf{x}} f(\mathbf{x}^*, \mathbf{w})] \cdot (\mathbf{x} - \mathbf{x}^*)$$

as $N \to \infty$.

Proof. The main result in Theorem 4.1 was to show that the orthogonal component of the loss gradient has expectation zero under the posterior weight distribution, therefore showing that BNNs are robust against adversarial attacks. In the LRP setup, we instead consider gradients of the prediction function, as opposed to the loss; nevertheless, the insight remains valid. From Theorem 4.1 we know that, under the assumption of a flat prior and for an infinitely wide neural network f, for any $\mathbf{x} \in \mathcal{M}$

$$\mathbb{E}_{p(\mathbf{w}|D)}[\nabla^{\perp}_{\mathbf{x}} f(\mathbf{x}, \mathbf{w})] \to 0.$$

Moreover, as described in Section 3.5, from the deep Taylor decomposition on a root point $\mathbf{x}^* \in \mathcal{M}$ we obtain the expected LRP heatmap

$$\mathbb{E}_{p(\mathbf{w}|D)}[R(\mathbf{x}, \mathbf{w})] = \mathbb{E}_{p(\mathbf{w}|D)}[\nabla_{\mathbf{x}} f(\mathbf{x}^*, \mathbf{w})] \cdot (\mathbf{x} - \mathbf{x}^*).$$

Therefore, the orthogonal component of the gradient of the prediction function vanishes in expectation under the posterior weight distribution and Bayesian averaging of the relevance heatmaps naturally builds explanations in the tangent space $T_{\mathbf{x}}\mathcal{M}$. $\square$

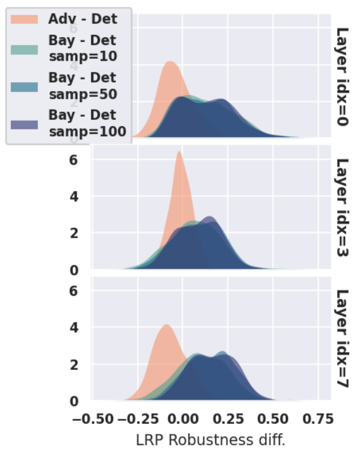
It should be noticed that LRP heatmaps at layer l involve partial derivatives w.r.t. $\mathbf{x}$ of the subnetwork $f^l(\cdot, \mathbf{w})$ of f, which associates to an input $\mathbf{x}$ the l-th activation from $f(\cdot, \mathbf{w})$. Consequently, the same vanishing property of the gradients holds for explanations in the internal layers, which are therefore more robust in the Bayesian case.

6.3 Experimental Results

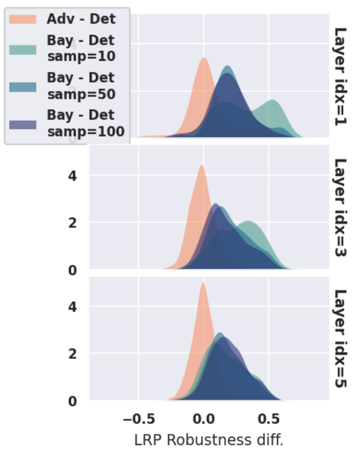
We corroborate the insights described in Section 6.1 with an experimental evaluation, training fully connected and convolutional NNs on MNIST [92] and Fashion MNIST [174] benchmark data sets. We also extend the experiments to a ResNet20 architecture trained on 3-channel images from CIFAR-10 [88], which consists of 50.000 3-channels training images from ten classes. We do not examine more complex data sets, such as ImageNet [43], because of the high computational costs and the problems in convergence to the posterior distribution for running Bayesian inference on deep networks trained on very large data sets[1]. We train multiple DNNs and BNNs using both HMC and VI, which allows us to contrast the effect of a locally Gaussian approximation to the posterior against the asymptotically exact (but computationally more expensive) approximation provided by HMC. Because we require high accuracy to approximate the asymptotic conditions described in Section 6.1, different architectures were used on the three data sets and between VI and HMC. In all cases, however, the BNN is compared with a DNN with the same architecture, to ensure the fairness of the comparisons. Details about the architectures are reported in Tables A.10 and A.11 of the appendix. Adversarial attacks in our tests are Fast Gradient Sign Method and Projected Gradient Descent (Section 3.4), with a maximum perturbation size of 0.2. Saliency attacks are *target region* and *top-k* attacks [63] and *beta* attacks [46], presented in Section 3.5.3.

In Figure 6.5 and Figures B.4-B.9 in the appendix we also compare the robustness distributions for adversarially trained and Bayesian Neural Networks using Mann-Whitney U rank test. The asterisk notation (Table B.5 in the appendix) denotes statistically significant p-values in favour of the alternative hypothesis that the the distribution from the adversarially trained model is stochastically lower than the distribution from the Bayesian model. For significant

[1]We do not experiment with scalable Monte Carlo dropout methods [61] here since there is no guarantee that their uncertainty estimates can capture the full posterior [149].



(a)



(b)

Figure 6.4: LRP robustness differences for FGSM (a) and PGD (b) attacks computed on 500 test points from Fashion MNIST dataset using the epsilon rule. NNs in (a) have a convolutional architecture (Table A.12 in the appendix) and the BNN is trained with VI. NNs in (b) have a fully connected architecture (Table A.13 in the appendix) and the BNN is trained with HMC. BNNs are tested using an increasing number of samples $(10, 50, 100)$. Layer indexes refer to the learnable layers in the architectures.

p-values the Bayesian model is significantly more robust than the deterministic model.

Simulations[2] are conducted on a machine with 36 cores, Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz processors and 192GB of RAM. We use PyTorch [124] and Pyro [15] libraries. We also rely on TorchLRP library[3] for the computation of LRP explanations and set $\epsilon = 0.1$ in the Epsilon rule, $\gamma = 0.1$ in the Gamma rule, $\alpha = 1$ and $\beta = 0$ in the Alpha-Beta rule.

6.3.1 Resilience of Bayesian Interpretations

Our first significant result is that Bayesian explanations are more robust under gradient and saliency-based attacks than deterministic architectures. For multiple data sets, attacks, training techniques (deterministic training, adversarial training, Bayesian inference) and approximate inference methods, LRP robustness scores are considerably higher than their deterministic counterparts. In Figure 6.4 and in Figures B.1, B.2, B.3 in the appendix we show the distribution of point-wise differences w.r.t. the deterministic baselines between LRP robustness scores for MNIST and Fashion MNIST data sets, using the

[2]Code is available at: `https://github.com/ginevracoal/BayesianRelevance`.
[3]`https://github.com/fhvilshoj/TorchLRP`

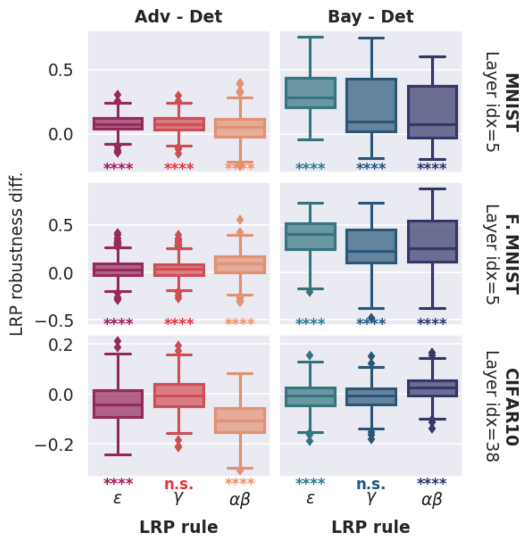


Figure 6.5: LRP robustness differences for FGSM attack attacks computed on 500 test points from MNIST (top row), Fashion MNIST (middle) and CIFAR-10 datasets (bottom) using epsilon, gamma and alpha-beta rules (columns) on the Top$_{20}$ pixels, for adversarially trained networks (left block) and BNNs (right block). NNs for MNIST and Fashion MNIST have a fully connected architecture (Table A.13) and the BNNs are trained with HMC. NNs for CIFAR-10 have a ResNet20 architecture from Bayesian Torch library [87] and the BNN is trained with VI. BNNs are tested using 100 posterior samples. Layer indexes refer to the last learnable layer in each architecture. Parameters are described in the main text.

epsilon rule. The bottom row of the figures is the standard LRP (computed from the pre-softmax layer), while the top row is the initial feature representation (after the first non-linear layer), and the middle row is the LRP of an internal layer. We test Bayesian representations using an increasing number of posterior samples, i.e. $10, 50, 100$. We attack 500 randomly selected test images, whose choice is balanced w.r.t. the available classes. The first notable observation is that adversarially trained networks have low LRP robustness compared to BNNs: this confirms empirically the conjecture of Section 6.2 that the components of the gradient that are normal to the data manifold - and are therefore the ones likely to be changed in an attack - are often major contributors to the relevance in DNN. Conversely, the Bayesian averaging process greatly reduces the expected relevance of such direction.

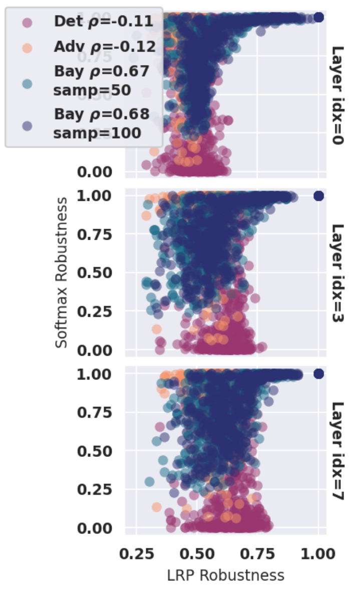
In Figures 6.5 and B.4-B.7 in the supplementary material we test the stability of deterministic and Bayesian interpretations w.r.t. FGSM and PGD attacks on 500 test inputs from MNIST, Fashion MNIST and CIFAR-10 datasets. In Figure B.8 and Figure B.9 in the appendix, we also test LRP stability against top-k, target region and beta saliency attacks (Sec. 3.5.3). As the saliency attacks are specifically designed to harm the interpretations, we consider them as a proxy for the worst-case robustness of the interpretations in a neighbor of the attacked point. Additionally, in Figure 6.5 and Section B.2 of the appendix, we compare the robustness distributions for adversarially trained and Bayesian NNs using the Mann-Whitney U rank test. We compute the relevance heatmaps using gamma, epsilon and alpha-beta rules. BNNs are trained with HMC and VI and evaluated using 100 samples from the posterior. The experiments confirm that Bayesian explanations are more stable across multiple LRP rules, gradient-based adversarial attacks and saliency attacks, also in the internal layers. Experiments on CIFAR-10 images in Figure 6.5 (bottom row) with a ResNet20 architecture from `bayesian_torch` library [87] show only a modest improvement in adversarial and LRP robustness, probably due to the conditions of Corollary 6.1 not being satisfied in this data set with higher complexity but relatively small size.

6.3.2 Bayesian LRP Robustness Increases with Softmax Robustness

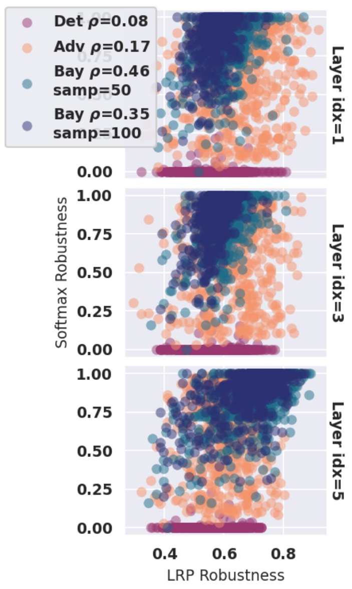
A simple explanation for the improved LRP robustness of BNNs lies in the fact that BNNs are provably immune to gradient-based attacks (Chapter 4). Therefore, one might argue that the stability of the LRP is a trivial byproduct of the empirical stability of the classifications. To explore this question more in depth, we consider the relationship between the LRP robustness of a test point (stability of the explanation, Section 6.1) and its softmax robustness (resilience of the classification against an attack, Section 3.4). Figure 6.6 and Figure B.10 in the appendix show scatterplots of these two quantities. An immediate observation is that deterministic explanations are weak against adversarial perturbations even when their softmax robustness is close to 1. Therefore, even in the cases where the classification is unchanged, deterministic saliency heatmaps are fragile. Indeed, there are no significant changes in LRP robustness between data points that are vulnerable to attacks and data points that are robust to attacks. Bayesian models, instead, show a strong positive correlation between LRP and softmax robustness, especially as the number of posterior samples increases. While it is immediately evident that Bayesian predictions are robust to adversarial attacks (since most data points have softmax robustness greater than 0.5), it is also clear from this correlation that attacks which are more successful (i.e. lower softmax robustness) also alter more substantially the interpretation, and are likely to represent genuine directions of change of the true underlying decision function along the data manifold.

6.4 Final Considerations

We harness the geometric perspective to adversarial attacks introduced in Chapter 4 to study the resilience of layer-wise relevance propagation heatmaps to



(a)



(b)

Figure 6.6: LRP vs softmax robustness of deterministic, adversarially trained and Bayesian NNs trained on MNIST dataset and tested against FGSM (a) and PGD (b) attacks. ρ denotes the Spearman correlation coefficient. LRP Robustness is computed with the epsilon rule on the 20% most relevant pixels. BNNs are trained with VI (a) and HMC (b) and are tested using an increasing number of samples $(50, 100)$. Layer indexes refer to the learnable layers in the convolutional (a) and fully connected (b) architectures (Tables A.12 and A.13 in the appendix).

adversarial attacks. The geometric analysis suggests a fundamental link between the fragility of DNNs against adversarial attacks and the difficulties in understanding their predictions: because of the unconstrained nature of classifiers defined on high dimensional input spaces but trained on low dimensional data, gradients of both the loss function and the prediction function tend to be dominated by directions which are orthogonal to the data manifold. These directions both give rise to adversarial attacks and provide spurious explanations which are orthogonal to the natural parametrization of the data manifold. In the limit of infinite data, a Bayesian treatment remedies the situation by averaging out irrelevant gradient directions in expectation. Not only BNN interpretations are considerably more robust than deterministic DNN, but we also observe a correlation between softmax (adversarial) robustness and LRP robustness which suggests that indeed Bayesian interpretations capture the relevant parametrization of the data manifold.

We point out the presence of theoretical and practical limitations. The strong assumptions in Theorem 6.1, which restrict the geometrical considerations to fully trained BNNs in the limit of an infinite amount of weights and training data, do not prevent us from observing the desired behaviour in practice, even when using cheap approximate inference techniques (VI). Indeed, Bayesian interpretations are considerably more robust than deterministic and adversarially trained networks. However, performing Bayesian inference in large non-linear models is extremely challenging. Nevertheless, we believe that the insights provided by a geometric interpretation will be helpful towards a better understanding of both the strengths and the weaknesses of deep learning.

Chapter 7

Adversarial Attacks on Protein Language Models

Contents

As introduced in Section 1, we search for biologically plausible perturbations of a few amino acids in the original sequences that are maximally different in the 3D structures (Figure 7.1). It is natural to interpret this reasoning through the lens of adversarial attacks [67], where the goal is to craft small perturbations of the inputs while inducing the most significant change in predictions. A similar approach is presented in [80], where the authors already propose a notion of “adversarial mutation” on amino acid sequences and use the perturbations to assess the reliability of structure prediction on the RoseTTAFold [9] model. Specifically, they use BLOSUM similarity to identify the space of similar protein sequences and search for maximally distant perturbations in this space. Moreover, they define a robustness measure for structure prediction based on the computation of the inverse RMSD (Section 7.2) between original and perturbed coordinates and use it to evaluate adversarial robustness. The fundamental novelty in our approach is that the adversarial perturbations do not require direct knowledge of the 3D coordinates, but only leverage the hidden representations provided by language models (Section 3.6).

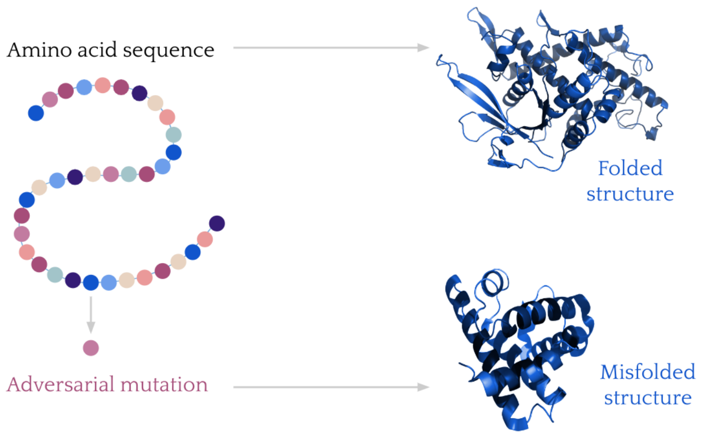


Figure 7.1: Adversarial mutations are alterations of a small set of amino acids that induce a significant structural change.

In the large space of all possible 3D structures, only a small portion contains biologically meaningful proteins. Moreover, high confident scores in structure prediction using AlphaFold2 model do not necessarily guarantee the plausibility of the native sequences [97]. Generative models [101, 82, 135, 97] tackle this problem by learning the data distribution in the space of sequences. Hence, they catch new evolutionary dependencies between the amino acids and generate new samples from the learned distribution. Our method does not directly rely on a generative architecture, but rather explores the space of hidden representations while guaranteeing that a set of desired conditions are met. In what follows, we discuss the approach behind our choices of target positions and mutant residues.

7.1 Methodology

In this section we describe the strategy behind our choice of target positions in the original sequences and target residues in the alphabet.

7.1.1 Target Positions

Several recent works show that Transformer language models are able to recover functional properties of protein sequences [131, 166, 138]. [166], in particular, observe that attention scores capture structural information and that most of the attention is directed to binding sites, i.e. amino acids that bind with other molecules, such as proteins, natural ligands, and small-molecule drugs. We use attention scores to identify token positions having the highest impact on

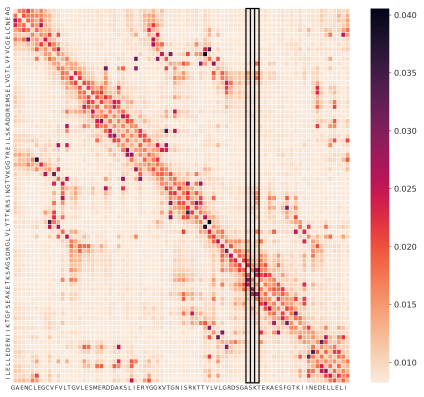


Figure 7.2: Average attention matrix $\mathbb{E}_{h,l}[A_{h,l}(\mathbf{x})] \in \mathbb{R}^{D \times D}$ over all attention heads and layers for an input sequence from domain PF00627. Scores are computed by MSA Transformer.

the surrounding context (i.e. on the remaining amino acids) and use them as target positions for adversarial mutations. Let $\mathbf{x} = (x_1, \dots, x_D) \in \mathcal{A}^D$ be a protein sequence, where $\mathcal{A}$ is the 25-character alphabet of amino acids[1] and $D \in \mathbb{N}$ is the length of the sequence. Given a fixed number of token substitutions $n < D$, each *attention head* $h \in \{1, \dots, H\}$ in a Transformer LM computes a set of attention scores $A_{h,l}(\mathbf{x}) \in \mathbb{R}^{D \times D}$ for each layer $l \in \{1, \dots, L\}$ in the architecture. We stress that the computation of the attention scores in model-dependent, as explained later in Section 7.3. Figure 7.2 shows an example of average attention heatmap over all heads and layers, where the i-th column is the average attention $\mathbb{E}_{h,l}[A_{h,l}(x_i)] \in \mathbb{R}^D$ for a single token x_i and represents the average importance attributed to the i-th token from all the other tokens in the sequence. The resulting target tokens are the first n positional indexes maximizing the Euclidean norm of the average attention across all layers and heads, i.e. the first n values in $\arg\text{sort}_{i=1,\dots,D} ||\mathbb{E}_{h,l}[A_{h,l}(x_i)]||_2$.

7.1.2 Target Residues

Once target token indexes are fixed, we use Block Substitutions Matrices (BLOSUM) (Section 7.2) [158] to assess the set of allowed substitutions of residues at each position. The selection of amino acids at a given position is restricted to residues with non-null BLOSUM62 scores, i.e. those with non-null frequency in the reference alignments, to avoid biologically meaningless substitutions. We also use BLOSUM62 matrix at evaluation time to assess the biological similarity between perturbed and original sequences. The choice of target residues is performed through a search over all plausible token substitutions that satisfy

[1]20 characters for the standard amino acids and 5 characters for non-standard or unknown amino acids [130].

a desired "adversarial" property. Given a fixed set of target positions I in a

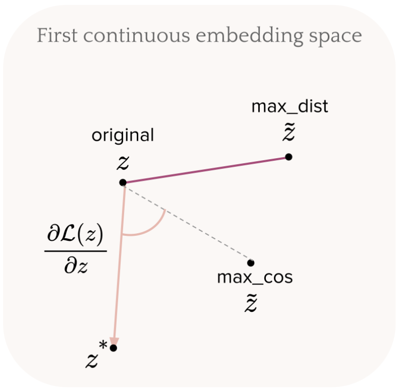

First continuous embedding space
max_dist
$\tilde{z}$
original
z
$\frac{\partial \mathcal{L}(z)}{\partial z}$
max_cos
$\tilde{z}$
z^*

Figure 7.3: Visualization of maximum embedding distance (max_dist) and maximum cosine similarity (max_cos) attacks in the first embedding space of a protein language model.

wild-type sequence $\mathbf{x}$, we propose multiple attack strategies to craft an adversarial perturbation $\tilde{\mathbf{x}}$, built upon different hidden representations of the original sequence but with the common goal of causing a significant structural change:

- Let $\mathbf{z}$ and $\tilde{\mathbf{z}}$ respectively denote the continuous embeddings of $\mathbf{x}$ and $\tilde{\mathbf{x}}$ in the first embedding space, i.e. the positional embeddings. **Maximum distance perturbations** indicate token substitutions $\tilde{\mathbf{z}}$ that maximize the L1 distance from $\mathbf{z}$ in the first embedding space, i.e. $\arg\max_{\tilde{\mathbf{z}}}(||\mathbf{z} - \tilde{\mathbf{z}}||_1)$;

- Protein LMs are trained to solve a *masked prediction task*, meaning that part of the input sequence is masked at random positions, and the model has to predict missing residues from the surrounding context. Therefore, given single candidate residue at a target position i, the LM outputs a *pseudo-likelihood* score $p(\mathbf{x}|\hat{\mathbf{x}}_i)$, denoting an approximate likelihood of the full sequence with the chosen residue, where $\hat{\mathbf{x}}_i$ is the sequence masked at position i. We define a loss function $\mathcal{L}(\mathbf{z}) = \max_{i \in I} p(\mathbf{z}|\hat{\mathbf{z}}_i)$ that penalizes the highest pseudo-likelihood score attributed to the first embedding of the original sequence $\mathbf{z}$ and use it to build a perturbation inspired by classical gradient-based attacks in continuous spaces [67]. The goal of this loss function is to increase uncertainty in masked prediction at the target token positions in I. **Maximum cosine similarity perturbations** search for residues that maximize the cosine similarity w.r.t. the loss gradient direction in the first embedding space, i.e. they build perturbations in the direction of greatest change in the loss function. More precisely, given a gradient-based attack

$$\mathbf{z}^* = \mathbf{z} + \epsilon \cdot \frac{\partial \mathcal{L}(\mathbf{z})}{\partial \mathbf{z}}$$

in the first embedding space, they search for a perturbation $\tilde{\mathbf{x}}$ such that the first embedding $\tilde{\mathbf{z}}$ maximizes

$$\cos\text{similarity}(\mathbf{z}^* - \mathbf{z}, \tilde{\mathbf{z}} - \mathbf{z}) = \cos\text{similarity}\Big(\frac{\partial \mathcal{L}(\mathbf{z})}{\partial \mathbf{z}}, \tilde{\mathbf{z}} - \mathbf{z}\Big).$$

Notice that this definition does not depend on the intensity ϵ of the attack;

- Protein LMs provide a reduced representation of the 3D structure, known as *contact map* (Section 3.6.2), which consists of a heatmap of estimated distances between all residue pairs in the 3D structure [169]. **Maximum contact map distance perturbations** maximize the L2 distance between original and perturbed contact maps:

$$\arg\max_{\tilde{\mathbf{x}}} ||\text{cmap}(\mathbf{x}) - \text{cmap}(\tilde{\mathbf{x}})||_2;$$

- Additionally, we introduce **maximum entropy perturbations** for MSA Transformer only. Given an input sequence $\mathbf{x}$ with an associated MSA (Section 3.6.2) and a set of amino-acid substitutions $\{c_{m_i} \in \mathcal{A}\}_{i \in I}$ at target sites I, we use the substitution frequency $p_i(c_{m_i})$ of a residue c_{m_i} at position $i \in I$ in the MSA to compute the entropy of that substitution. Then, we search for a perturbation that maximizes the entropy across all positions:

$$\max_{\{c_{m_i} \in \mathcal{A} : i \in I\}} \left[-\sum_{i \in I} p_i(c_{m_i}) \log_2 p_i(c_{m_i}) \right].$$

7.2 Evaluation Scores for Adversarial Mutations

Since protein LMs for structure prediction could be insensitive to single point mutations [23], we do not only rely on evaluation metrics on the structure [121], but also examine several evaluation metrics on continuous embeddings of amino acid sequences. The first natural evaluation metric for sequence similarity is the L1 distance between original and perturbed embeddings in the first layer of a protein LM, $||\mathbf{z} - \tilde{\mathbf{z}}||_1$, providing a preliminary geometric interpretation of the distribution of continuous representations in the first embedding space. We point out that the choice of the first embedding space to evaluate distances provides a natural baseline for comparison (i.e. maximum distance perturbations), but it would be interesting to extend this analysis to multiple layers of hidden representations. Secondly, we leverage BLOSUM62 matrix to compute a biological sequence similarity measure known as BLOSUM distance:

$$\text{BLOSUM}(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{i=1,\dots,D} (B_{x_i,x_i} - B_{x_i,\tilde{x}_i}),$$

where B_{r_i,r_j} is the entry associated to a couple of residues (r_i, r_j) in BLOSUM62 matrix. BLOSUM distance is zero when $\mathbf{x} = \tilde{\mathbf{x}}$. Another fundamental information provided by protein LMs is the predicted contact map, thus we also compute the L2 distance between contact maps of original and perturbed sequences. It is usual practice to examine upper submatrices in a contact map and look at the distances between long-range contacts. Therefore, we use an index $k \in \mathbb{N}_{>0}$ to denote the diagonal index of an upper triangular submatrix in a full contact map ($k = 0$), i.e. we select contacts that are at least k positions apart, and compute a range of distances $||\text{cmap}_k(\mathbf{x}) - \text{cmap}_k(\tilde{\mathbf{x}})||_2$ as k increases.

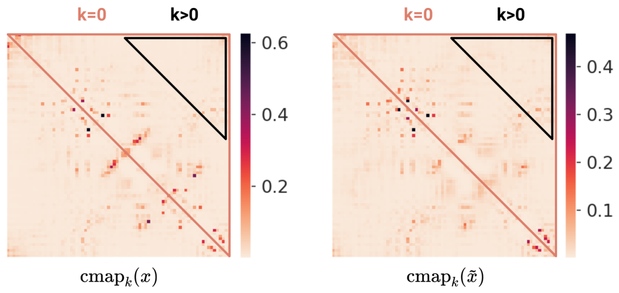


Figure 7.4: Upper triangular submatrices of original (left) and perturbed (right) full contact maps.

On the structural side, let s and $\tilde{s}$ respectively denote original and perturbed 3D structures. We evaluate their similarity using both local and global similarity measures to capture the relative orientation of the deviations as well as the global superimposition between the structures. Let us briefly recall a few measures already defined in Section 7.2. Local Distance Difference Test (LDDT) [105] is a local score that measures the percentage of preserved distances between all pairs of atoms in the target structure closer in space than a predefined cutoff. In particular, it computes the mean fraction of preserved distances using four different thresholds $(0.5, 1, 2, 4\,\mathring{A})$. Then, we use two popular global scores: Root-Mean-Square-Deviation (RMSD) computes the average Euclidean distance between matching atoms in the two structures; TM-score, instead, computes the degree of match between corresponding Cα atoms, scaled by a length-dependent distance parameter.

7.3 Experimental Results

Our experiments involve two Transformer models trained on Uniref50 database [154], namely ESM-1b [138] and MSA Transformer [132]. ESM-1b takes as inputs protein sequences and computes pairwise attention scores between all couples of amino acids. The set of attention scores (Section 3.6.1) $R_{h,l}(\mathbf{x}) \in \mathbb{R}^D$ from an attention head h in the l-th layer is called *row attention*, where $\mathbf{x}$ is the input sequence and D is the length of the sequence. The resulting attention score used to recover target positions is $A_h(\mathbf{x}) := \mathbb{E}_l[R_{h,l}(\mathbf{x})]$, the average attention across all layers for head h. MSA Transformer, instead, works on MSAs [36], i.e. on sets of aligned protein sequences. In this setting, self-attention mechanism also computes *column attention* scores $C_{h,l}(\mathbf{x}) \in \mathbb{R}^D$ from the input MSA and we select positions based on a weighted sum of the two scores: $A_h(\mathbf{x}) := \gamma\,\mathbb{E}_l[R_{h,l}(\mathbf{x})] + (1-\gamma)\,\mathbb{E}_l[C_{h,l}(\mathbf{x})]$, where $\gamma \in \mathbb{R}$. In the experiments, we weight the two contributions equally by setting $\gamma = 0.5$. We use the hhfilter method from HH-suite tool [151] to select a subset of most diverse sequences in the alignment by means of a sequence similarity score based on the degree of homology among sequences. We build a filtered MSA for each selected sequence to be used as an input for MSA Transformer. Then, we craft the set of adversarial perturbations presented in Section 7.1.2.

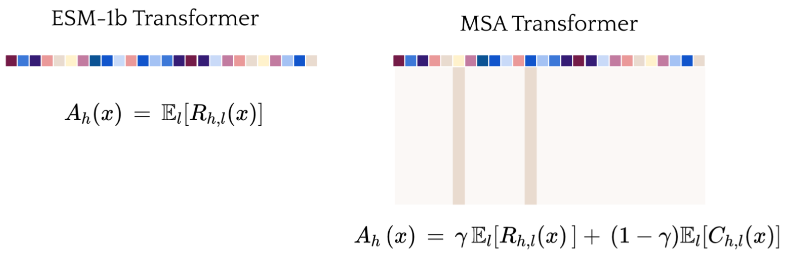

ESM-1b Transformer
$A_h(x) = \mathbb{E}_l[R_{h,l}(x)]$
MSA Transformer
$A_h(x) = \gamma\mathbb{E}_l[R_{h,l}(x)] + (1-\gamma)\mathbb{E}_l[C_{h,l}(x)]$

Figure 7.5: Visualization of row attention scores for ESM-1b Transformer model (left) and combined row and column attention scores for MSA-Transformer model (right).

In our first set of experiments, we build adversarial mutations on protein families PF00533 and PF00627, both containing examples of structure disruptive mutations not detected by AlphaFold2 model [23]. For the sake of brevity, we only report the experiments performed on domain PF00627 in this chapter, where we use MSA Transformer model to build 3 sites mutations on 100 native sequences with input MSAs of depth 100. We observe a similar experimental behaviour on domain PF00533 and include the additional experiments in Appendix B.3. Next, we compare adversarial perturbations to those provided by ProTherm database [119], containing more than $4k$ single-point mutations associated with high changes in stability. Protein stability denotes the capability of a protein to retain the native conformation under a stress condition (e.g. change in temperature or pressure). Since high changes in stability are often associated with disease-causing mutations and unfolding, we want to analyse their statistical properties against those of adversarial mutations. To analyse the change in stability we focus on ProTherm values of *change in free energy* $\Delta\Delta G$, which is negative for spontaneous reactions and positive for non-spontaneous reactions. Specifically, we select the most stabilizing (highest $\Delta\Delta G$) and destabilizing (lowest $\Delta\Delta G$) mutations by setting a threshold of $1kcal/mol$ on the minimum absolute value of $\Delta\Delta G$. In this case, we identify the reference sequences belonging to Pfam families and build the associated filtered MSAs accordingly.

We run the experiments[2] on a machine with 48 cores, Intel(R) Xeon(R) Gold 6226 CPU @ 2.70GHz processors, 263GB of RAM and two Tesla V100-PCIE-32GB GPUs. Our tests rely on HH-suite tool [151] for MSA filtering and on ColabFold library [112] for structure prediction.

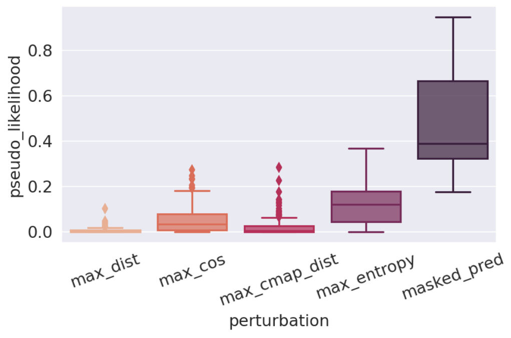
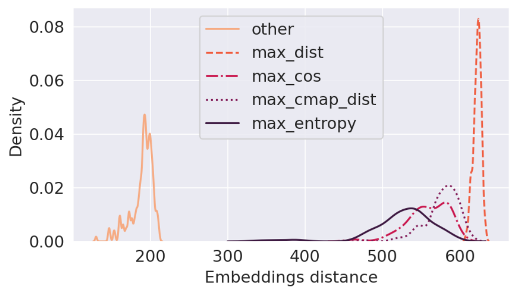


Figure 7.6: Pseudo-likelihood of adversarial (columns 1-4) and masked prediction (column 5) mutations at target token indexes. Values refer to 3 sites mutations obtained from MSA Transformer on 100 sequences from domain PF00627.

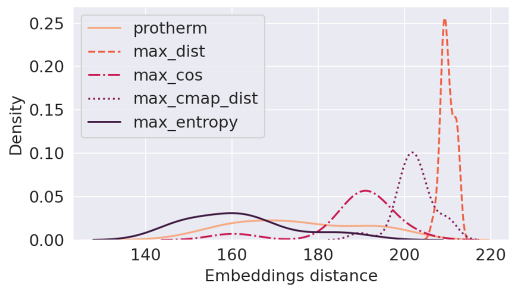
7.3.1 Impact of Adversarial Mutations on Wild Type Sequences

We analyse the effect of adversarial perturbations using the evaluation metrics described in Section 7.1.2. First of all, Figure 7.6 shows how adversarial perturbations are overall less likely compared to the residues selected by masked predictions, suggesting that our method detects rare substitutions at fixed mutant sites. This is an important argument in favour of the generation of new and diverse disruptive mutations. We compare adversarial perturbations to the ones obtained from all the discarded plausible token substitutions at the chosen target positions, which we refer to as "other" in Figures 7.7a and 7.8a. Figure 7.7 reports the L1 distances $||\mathbf{z} - \tilde{\mathbf{z}}||_1$ between original and perturbed first layer embeddings, while Figure 7.8 shows BLOSUM distances between original and perturbed sequences. Adversarial perturbations significantly depart from the other plausible perturbations both in terms of embedding distance (Figure 7.7a) and BLOSUM distance (Figure 7.8a), approaching perturbations at maximum embedding distance in the first case. Adversarial embeddings are at least as far from the reference as the most stabilizing and destabilizing embeddings in ProTherm (Figure 7.7b), while BLOSUM distances are comparable (Figure 7.8b). Moreover, [80] observed that larger BLOSUM distances between original and perturbed sequences lead to higher RMSD in predicted structures, therefore we expect adversarial perturbations to produce a significant structural

[2]Code is available at `https://github.com/ginevracoal/adversarial-protein-sequences`.



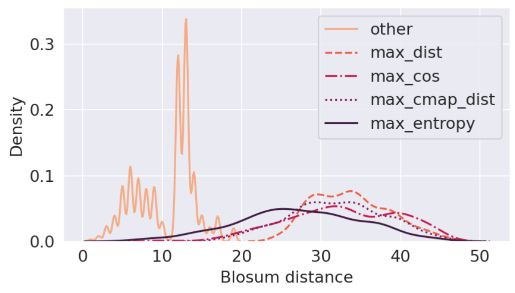
(a)



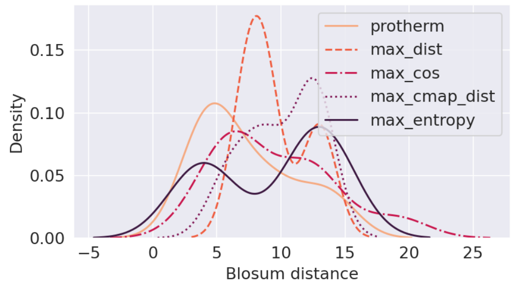
(b)

Figure 7.7: L1 distances $||\mathbf{z} - \tilde{\mathbf{z}}||_1$ between the embeddings of original and perturbed sequences in the first embedding space. Perturbations are computed using MSA Transformer model. In (a) we report mutations of 3 sites on domain PF00627, while Figure (b) shows single mutations on ProTherm database. "Other" refers to adversarial perturbations obtained from all the discarded plausible token substitutions at the chosen target positions.

change. We stress that adversarial mutant positions for ProTherm are those that maximize the attention scores and, in most cases, differ from ProTherm mutation sites. Indeed, Figure 7.9 reports attention-based importance ranks attributed to ProTherm token positions and shows that only a fraction of the selected adversarial positions (i.e. those with zero rank) match ProTherm ones, while in the other cases (higher rank) ProTherm indexes have lower average attention scores. Nonetheless, adversarial sequences are able to significantly alter embedding distances, BLOSUM distances and contact maps distances (see the appendix for additional results), suggesting that our attention-based selection method catches new relevant mutant positions.



(a)

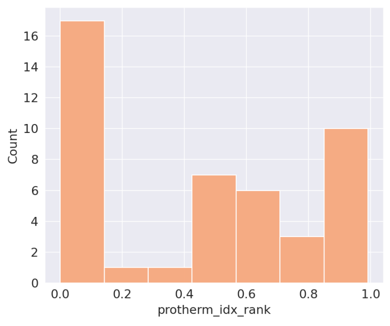


(b)

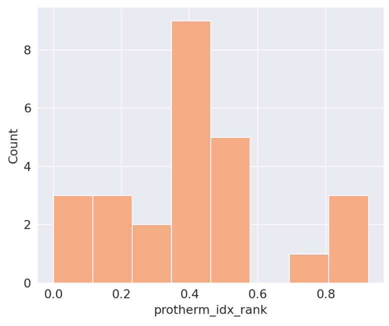
Figure 7.8: Blosum distances $\mathrm{BLOSUM}(\mathbf{x}, \tilde{\mathbf{x}})$ between original and perturbed sequences. Perturbations are computed using MSA Transformer model. In (a) we report mutations of 3 sites on domain PF00627, while (b) shows single mutations on ProTherm database. "Other" refers to adversarial perturbations obtained from all the discarded plausible token substitutions at the chosen target positions.

7.3.2 Predicted Adversarial Structures are Substantially Altered and Highly Confident

We analyze the effect of adversarial perturbations on 3D structures predicted by ColabFold [112], an extension of AlphaFold2 model. First, we select the 100 most diverse sequences from domain PF00627 using hhfilter method, as explained at the beginning of this chapter. Since our goal is to generate adversarial mutations that are able to "fool" structure prediction models, the latter should be highly confident in predictions. Therefore, we rely on the pLDDT confidence score for structure prediction provided by AlphaFold2 model, and among the 100 original sequences we select the ones whose average (over residues) pLDDT is greater than 80%, for a total of 37 sequences. Then, we build the adversarial perturbations, predict their 3D structures and compare them to the original predicted structures. Figure 7.10 reports LDDT, TM-score and RMSD (presented in Section 7.1)



(a)



(b)

Figure 7.9: Attention-based importance ranks attributed to ProTherm positions using ESM-1b (a) and MSA-Transformer (b) models. The most important tokens, i.e. those maximizing the average attention, are those with zero rank.

between original and perturbed 3D structures on among the selected sequences. Figure B.16 in the appendix reports the resulting adversarial pLDDT scores, as well as another accuracy measure for structure prediction, namely the pTM-score. Adversarial mutations exhibit high confidence in structure prediction and produce structures that are significantly distant from their references in terms of global distance. In particular, low TM-scores and high RMSD scores, consistently across the several attack techniques, indicate that adversarial structures deviate from the original folding. As a matter of comparison, it is important to observe that two structures with 50% sequence identity (i.e. the percentage of corresponding amino acids) align within approximately $1\mathring{A}$ RMSD [38] in the 3D structures, and two proteins with even 40% sequence identity and at least 35 aligned residues (w.r.t. optimal superimposition of the 3D structures) align with approximately $2.5\mathring{A}$ RMSD [145]. Higher LDDT scores in Figure 7.10 instead show that local atomic interactions in the original structure happen to be preserved in adversarial structures, especially for maximum entropy perturbations.

7.4 Final Considerations

We showed how adversarial perturbations on protein language models produce substantial changes according to several geometric, biological, and structural evaluation scores compared to the reference sequences. Additionally, they introduce a new efficient attention-based method for the selection of target positions in the reference sequences. Nonetheless, we point out the presence of some limitations, which we plan to address in future versions of this work. Precisely, we intend to: (1) propose a more extensive empirical evaluation on several protein families and include structure prediction on databases of dysfunctional mutations; (2) fine-tune protein LMs on adversarial sequences and test their sensitivity to a set

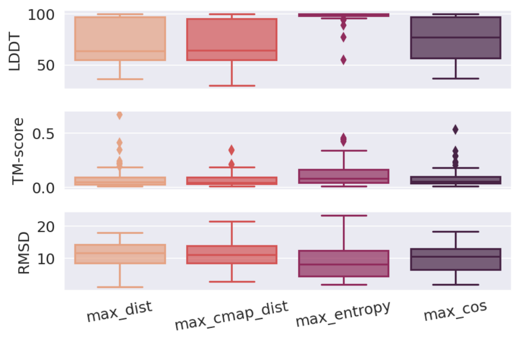


Figure 7.10: LDDT, TM and RMSD scores between original and perturbed structures for 3 sites mutations on 100 sequences from domain PF00627. Adversarial perturbations are computed by MSA Transformer model, while structure predictions are performed by ColabFold model.

of known missense mutations.

Chapter 8

Conclusions

Adversarial examples show that many of the modern machine learning algorithms can be fooled in unexpected ways. Both in terms of attacks and defences, many theoretical problems remain open. Moreover, the most effective defence techniques in the deterministic setting, e.g. adversarial training, are still too computationally expensive or are not transferable to multiple attack strategies. From a practical point of view, no one has yet designed a powerful defence algorithm which could be suitable against a variety of attacks, with different degrees of knowledge about the models under attack and their predictions.

In this work, we first analyse the geometry of adversarial attacks in the large-data, overparameterized limit for Bayesian neural networks. We show that, in the limit, vulnerability to gradient-based attacks arises as a result of degeneracy in the data distribution, i.e., when the data lies on a lower-dimensional submanifold of the ambient space. As a direct consequence, we demonstrate that in this limit BNN posteriors are robust to gradient-based adversarial attacks. In particular, we prove that the expected gradient of the loss with respect to the BNN posterior distribution vanishes, even when each neural network sampled from the posterior is vulnerable to gradient-based attacks. Experimental results in the finite data regime with approximate Bayesian inference techniques support this line of argument, showing that BNNs can display both high accuracies on clean data and robustness to gradient-based and gradient-free adversarial attacks.

Next, we leverage the high codimension between the data manifold and the embedding space to build low-dimensional representations of the input data - precisely random projections of the inputs - that preserve relevant adversarial features. Our defence methods are attack-independent, i.e. they have no knowledge about the chosen attacks, yet they are comparable to state of the art models in terms of adversarial robustness and still benefit from dimensionality reduction on the computational side.

Additionally, we examine the stability of saliency interpretations under targeted adversarial attacks that aim to change the classification under perturbations of the input. We observe that, in the overparameterized regime, the DNN function coincides with the true underlying function almost everywhere on the data manifold and therefore the tangent gradient of the loss function is (a.e.) identically zero. The normal gradient of the loss, however, is unconstrained by the data, and, particularly in a high dimensional setting, might achieve very high values along certain directions, creating therefore weaknesses that may be exploited by an adversarial attacker. Hence, we prove a trivial consequence of the robustness of BNNs to gradient-based attacks: in the thermodynamic limit, BNNs will only retain relevant directions along the data manifold, which correspond to genuine directions of high relevance.

Lastly, we introduce adversarial mutations against protein language models, designed with the twofold goal of altering the smallest amount of amino acids in the original sequence while inducing a substantial change in the 3d structure. Our experiments on single-sequence and MSA-based protein LMs examine the effect how such mutations on the hidden representations of the LMs, according to several geometrical and biological evaluation metrics, and on multiple protein families. The resulting adversarial structures predicted in ColabFold [112] significantly depart from their original counterparts, in terms of both local and global distance measures, even in the case of single-point mutations.

8.1 Limitations and future directions

Theorem 4.1 has the natural consequence of protecting BNNs against all gradient-based attacks, due to the vanishing average of the expectation of the gradients in the limit. While promising, this result comes with some significant limitations; indeed, the theorem holds under a set of strong theoretical assumptions, which are also required for the robustness of saliency explanations under gradient-based attacks (Corollary 6.1). First, the assumption of flat priors is needed for perfect gradient cancellation; but in practice, unless the priors are too informative with restrictive support, we do not expect a major deviation from the idealised case. This is confirmed both from our experimental results and by the fact that in the limit of infinite data the posterior is less influenced by the choice of the prior [162]. Secondly, the averaging property holds when the ensemble is drawn from the true posterior; nevertheless cheaper approximate Bayesian inference methods which retain ensemble predictions, such as VI, may still in practice provide good protection. Lastly, Theorem 4.1 and Corollary 6.1 only guarantee protection against gradient-based adversarial attacks, hence it is not clear whether the robustness properties of BNNs also extend to any other adversarial threat. However, in Section 4.3.4 we empirically show that the vanishing gradient properties of BNNs also guarantee robustness to gradient-free attacks.

Furthermore, and perhaps more importantly, performing Bayesian inference on complex datasets and large architectures is extremely challenging, therefore we restricted our empirical analysis on BNNs to a set of fairly simple settings, compared to the current research in image processing. To this end, we hope that our work will spark interest in the development of efficient Bayesian inference algorithms. Nonetheless, we point out that our random projections-based ensemble method could also be applied to more complex scenarios.

Finally, our analysis of the behaviour of protein LMs for structure prediction against adversarial mutations is promising, yet still preliminary. First of all, to assess the biological implications of our results we would need to perform a thorough comparison with known mutations causing misfolding or dysfunction. Secondly, it would be interesting to fine-tune state of the art protein LMs on our adversarial mutations and then check if they can correctly predict the 3d structures of some known disruptive mutations which are currently undetected by the models [80]. Despite such limitations, we believe that adversarial mutations could help to improve the sensitivity of protein LMs to dysfunctional mutations, therefore opening a new path for a deeper understanding of the connection between protein stability and variations in the 3d structures.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[3] Robert J Adler. *The geometry of random fields.* SIAM, 2010.

[4] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*, 2018.

[5] Christopher Anders, Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller, and Pan Kessel. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, pages 314–323. PMLR, 2020.

[6] Alexandre Araujo, Laurent Meunier, Rafael Pinot, and Benjamin Negrevergne. Robust neural networks using randomized adversarial training. *arXiv preprint arXiv:1903.10219*, 2019.

[7] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[8] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise

explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[9] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[10] David Barber. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[11] Artur Bekasov and Iain Murray. Bayesian adversarial spheres: Bayesian inference and adversarial examples in a noiseless setting. *arXiv preprint arXiv:1811.12335*, 2018.

[12] Leonard Berrada, Sumanth Dathathri, Robert Stanforth, Rudy Bunel, Jonathan Uesato, Sven Gowal, M Pawan Kumar, et al. Verifying probabilistic specifications with functional lagrangians. *arXiv preprint arXiv:2102.09479*, 2021.

[13] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[14] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[15] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *CoRR*, abs/1810.09538, 2018. URL `http://arxiv.org/abs/1810.09538`.

[16] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[17] Christopher M Bishop et al. *Neural networks for pattern recognition.* Oxford university press, 1995.

[18] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[19] Luca Bortolussi, Francesca Cairoli, Ginevra Carbone, Francesco Franchina, and Enrico Regolin. Adversarial learning of robust and safe controllers for cyber-physical systems. *IFAC-PapersOnLine*, 54(5):223–228, 2021.

[20] Luca Bortolussi, Francesca Cairoli, Ginevra Carbone, and Paolo Pulcini. Stochastic variational smoothed model checking. *arXiv preprint arXiv:2205.05398*, 2022.

[21] Luca Bortolussi, Ginevra Carbone, Luca Laurenti, Andrea Patane, Guido Sanguinetti, and Matthew Wicker. On the robustness of bayesian neural networks to adversarial attacks. *arXiv preprint arXiv:2207.06154*, 2022.

[22] George EP Box and George C Tiao. *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

[23] Gwen R Buel and Kylie J Walters. Can alphafold2 predict the impact of missense mutations on structure?, 2022. URL `https://pubmed.ncbi.nlm.nih.gov/35046575/`.

[24] Kirill Bykov, Anna Hedström, Shinichi Nakajima, and Marina M-C Höhne. Noisegrad: enhancing explanations by introducing stochasticity to model weights. *arXiv preprint arXiv:2106.10185*, 2021.

[25] Francesca Cairoli, Ginevra Carbone, and Luca Bortolussi. Abstraction of markov population dynamics via generative adversarial nets. In *International Conference on Computational Methods in Systems Biology*, pages 19–35. Springer, 2021.

[26] Ginevra Carbone and Gabriele Sarti. ETC-NLG: End-to-end topic-conditioned natural language generation. *IJCoL. Italian Journal of Computational Linguistics*, 6(6-2):61–77, 2020.

[27] Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks. *Advances in Neural Information Processing Systems*, 33:15602–15613, 2020.

[28] Ginevra Carbone, Guido Sanguinetti, and Luca Bortolussi. Random projections for improved adversarial robustness. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2021. doi: 10.1109/IJCNN52387.2021.9534346.

[29] Ginevra Carbone, Luca Bortolussi, and Guido Sanguinetti. Resilience of bayesian layer-wise explanations under adversarial attacks. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. doi: 10.1109/IJCNN55064.2022.9892788.

[30] Ginevra Carbone, Francesca Cuturello, Luca Bortolussi, and Alberto Cazzaniga. Adversarial attacks on protein language models. *Proceedings of the 17th Machine Learning in Computational Biology meeting*, 2022.

[31] Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of bayesian neural networks. In *IJCAI*, 2019.

[32] Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, and Andrea Patane. Robustness guarantees for bayesian inference with gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7759–7768, 2019.

[33] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.

[34] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016. URL `http://arxiv.org/abs/1608.04644`.

[35] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J. Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *CoRR*, abs/1902.06705, 2019. URL `http://arxiv.org/abs/1902.06705`.

[36] Maria Chatzou, Cedrik Magis, Jia-Ming Chang, Carsten Kemena, Giovanni Bussotti, Ionas Erb, and Cedric Notredame. Multiple sequence alignment modeling: methods and applications. *Briefings in bioinformatics*, 17(6): 1009–1023, 2016.

[37] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[38] Cyrus Chothia and Arthur M Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO journal*, 5(4):823–826, 1986.

[39] Ronald Christensen, Wesley Johnson, Adam Branscum, and Timothy E Hanson. *Bayesian ideas and data analysis: an introduction for scientists and statisticians*. CRC press, 2010.

[40] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[41] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical report, 1999.

[42] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX security symposium (USENIX security 19)*, pages 321–338, 2019.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[44] Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. How important is a neuron? *arXiv preprint arXiv:1805.12233*, 2018.

[45] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

[46] Ann-Kathrin Dombrowski, Maximilian Alber, Christopher J Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *arXiv preprint arXiv:1906.07983*, 2019.

[47] Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv preprint arXiv:1708.05493*, 2017.

[48] Nathan Drenkow, Neil Fendley, and Philippe Burlina. Random projections for adversarial attack detection. *arXiv preprint arXiv:2012.06405*, 2020.

[49] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[50] Simant Dube. High dimensional spaces, deep learning and adversarial examples. *CoRR*, abs/1801.00634, 2018. URL `http://arxiv.org/abs/1801.00634`.

[51] Robert J Durrant and Ata Kabán. Random projections as regularizers: learning a linear discriminant from fewer observations than dimensions. *Machine Learning*, 99(2):257–286, 2015.

[52] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[53] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Wang Yu, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. doi: 10.1109/TPAMI.2021.3095381.

[54] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, 2017. URL `http://arxiv.org/abs/1707.08945`.

[55] Piero Fariselli, Osvaldo Olmea, Alfonso Valencia, and Rita Casadio. Prediction of contact maps with neural networks and correlated mutations. *Protein engineering*, 14(11):835–843, 2001.

[56] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems*, pages 1178–1187, 2018.

[57] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, 2018.

[58] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[59] Noelia Ferruz and Birte Höcker. Towards controllable protein design with conditional transformers. *arXiv preprint arXiv:2201.07338*, 2022.

[60] Chris Finlay, Adam M. Oberman, and Bilal Abbasi. Improved robustness to adversarial examples using lipschitz regularization of the loss, 2019.

[61] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059. PMLR, 2016.

[62] Yarin Gal and Lewis Smith. Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with bayesian neural networks. *arXiv preprint arXiv:1806.00667*, 2018.

[63] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of AAAI*, volume 33, pages 3681–3688, 2019.

[64] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice.* CRC press, 1995.

[65] Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modelling the influence of data structure on learning in neural networks. *arXiv preprint arXiv:1909.11500*, 2019.

[66] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015. URL `http://arxiv.org/abs/1412.6572`.

[67] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[68] Kathrin Grosse, David Pfaff, Michael Thomas Smith, and Michael Backes. The limitations of model uncertainty in adversarial settings. *arXiv preprint arXiv:1812.02606*, 2018.

[69] Kathrin Grosse, Michael T Smith, and Michael Backes. Killing four birds with one gaussian process: the relation between different test-time attacks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4696–4703. IEEE, 2021.

[70] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *CSUR*, 51(5):1–42, 2018.

[71] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[72] Liang He, Shizhuo Zhang, Lijun Wu, Huanhuan Xia, Fusong Ju, He Zhang, Siyuan Liu, Yingce Xia, Jianwei Zhu, Pan Deng, et al. Pre-training co-evolutionary protein representation via a pairwise masked language model. *arXiv preprint arXiv:2110.15527*, 2021.

[73] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 588–597, 2019.

[74] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. *arXiv preprint arXiv:1902.02041*, 2019.

[75] Brian L Hie, Kevin K Yang, and Peter S Kim. Evolutionary velocity with protein language models predicts evolutionary dynamics of diverse proteins. *Cell Systems*, 13(4):274–285, 2022.

[76] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[77] Roger W Hockney and James W Eastwood. *Computer simulation using particles*. crc Press, 2021.

[78] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[79] Anupama Jha, Joseph K Aicher, Matthew R Gazzara, Deependra Singh, and Yoseph Barash. Enhanced integrated gradients: improving interpretability of deep learning models using splicing codes as a case study. *Genome biology*, 21(1):1–22, 2020.

[80] Sumit Kumar Jha, Arvind Ramanathan, Rickard Ewetz, Alvaro Velasquez, and Susmit Jha. Protein folding neural networks are not robust, 2021. URL `https://arxiv.org/abs/2109.04460`.

[81] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[82] Mostafa Karimi, Shaowen Zhu, Yue Cao, and Yang Shen. De novo protein design for novel folds using guided conditional wasserstein generative adversarial networks. *Journal of chemical information and modeling*, 60 (12):5667–5681, 2020.

[83] Samuel Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. volume 1, pages 413 – 418 vol.1, 06 1998. ISBN 0-7803-4859-1. doi: 10.1109/IJCNN.1998.682302.

[84] Marc Khoury and Dylan Hadfield-Menell. On the geometry of adversarial examples. *CoRR*, abs/1811.00525, 2018. URL `http://arxiv.org/abs/1811.00525`.

[85] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[86] Andrei Kouranov, Lei Xie, Joanna de la Cruz, Li Chen, John Westbrook, Philip E Bourne, and Helen M Berman. The rcsb pdb information portal for structural genomics. *Nucleic acids research*, 34(suppl_1):D302–D305, 2006.

[87] Ranganath Krishnan and Piero Esposito. Bayesian-torch: Bayesian neural network layers for uncertainty estimation, 2020.

[88] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5), 2014.

[89] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016. URL `http://arxiv.org/abs/1611.01236`.

[90] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[91] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017.

[92] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

[93] Herbert KH Lee. Consistency of posterior distributions for neural networks. *Neural Networks*, 13(6):629–642, 2000.

[94] John M Lee. Smooth manifolds. In *Introduction to smooth manifolds*, pages 1–31. Springer, 2013.

[95] Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2052–2061. JMLR. org, 2017.

[96] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[97] Zeming Lin, Tom Sercu, Yann LeCun, and Alexander Rives. Deep generative models create new and diverse protein structures. In *Machine Learning for Structural Biology Workshop, NeurIPS*, 2021.

[98] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385, 2018.

[99] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. *arXiv preprint arXiv:1810.01279*, 2018.

[100] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st NeurIPS*, pages 4768–4777, 2017.

[101] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

[102] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[103] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[104] Ričards Marcinkevičs and Julia E Vogt. Interpretability and explainability: A machine learning zoo mini-tour. *arXiv preprint arXiv:2012.01805*, 2020.

[105] Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. lddt: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, 29(21):2722–2728, 2013.

[106] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

[107] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[108] John Merrill, Geoff Ward, Sean Kamkar, Jay Budzik, and Douglas Merrill. Generalized integrated gradients: A practical method for explaining diverse ensembles. *arXiv preprint arXiv:1909.01869*, 2019.

[109] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[110] Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7344–7350. IEEE, 2020.

[111] Milot Mirdita, Martin Steinegger, and Johannes Söding. MMseqs2 desktop and local web server app for fast, interactive sequence searches. *Bioinformatics*, 35(16):2856–2858, 01 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty1057. URL `https://doi.org/10.1093/bioinformatics/bty1057`.

[112] Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. *Nature Methods*, pages 1–4, 2022.

[113] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[114] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209, 2019.

[115] 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*, pages 2574–2582, 2016.

[116] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[117] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[118] Maria-Irina Nicolae, Mathieu Sinn, Tran Ngoc Minh, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Ian M. Molloy, and Benjamin Edwards. Adversarial robustness toolbox v0.2.2. *CoRR*, abs/1807.01069, 2018. URL `http://arxiv.org/abs/1807.01069`.

[119] Rahul Nikam, A Kulandaisamy, K Harini, Divya Sharma, and M Michael Gromiha. Prothermdb: thermodynamic database for proteins and mutants revisited after 15 years. *Nucleic acids research*, 49(D1):D420–D424, 2021.

[120] Michael Nilges, G Marius Clore, and Angela M Gronenborn. Determination of three-dimensional structures of proteins from interproton distance data by hybrid distance geometry-dynamical simulated annealing calculations. *FEBS letters*, 229(2):317–324, 1988.

[121] Kliment Olechnovič, Bohdan Monastyrskyy, Andriy Kryshtafovych, and Česlovas Venclovas. Comparative analysis of methods for evaluation of protein models against native structures. *Bioinformatics*, 35(6):937–944, 2019.

[122] Greg Ongie, Rebecca Willett, Daniel Soudry, and Nathan Srebro. A function space view of bounded norm infinite width relu nets: The multivariate case. In *International Conference on Learning Representations*, 2020.

[123] Yutian Pang, Sheng Cheng, Jueming Hu, and Yongming Liu. Evaluating the robustness of bayesian neural networks against different types of attacks. *arXiv preprint arXiv:2106.09223*, 2021.

[124] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[125] Andrea Patane, Arno Blaas, Luca Laurenti, Luca Cardelli, Stephen Roberts, and Marta Kwiatkowska. Adversarial robustness guarantees for gaussian processes. *Journal of Machine Learning Research*, 23:1–55, 2022.

[126] Urja Pawar, Donna O'Shea, Susan Rea, and Ruairi O'Reilly. Explainable ai in healthcare. In *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–2. IEEE, 2020.

[127] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955. doi: 10.1017/S0305004100030401.

[128] Gianluca Pollastri and Pierre Baldi. Prediction of contact maps by giohmms and recurrent neural networks using lateral propagation from all four cardinal corners. *Bioinformatics*, 18(suppl_1):S62–S70, 2002.

[129] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences*, 9(5):909, 2019.

[130] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32, 2019.

[131] Roshan Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. Transformer protein language models are unsupervised structure learners. *Biorxiv*, 2020.

[132] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*, pages 8844–8856. PMLR, 2021.

[133] Ambrish Rawat, Martin Wistuba, and Maria-Irina Nicolae. Adversarial phenomenon in the eyes of bayesian deep learning. *arXiv preprint arXiv:1711.08244*, 2017.

[134] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

[135] Donatas Repecka, Vykintas Jauniskis, Laurynas Karpus, Elzbieta Rembeza, Irmantas Rokaitis, Jan Zrimec, Simona Poviloniene, Audrius Laurynenas, Sandra Viknander, Wissam Abuajwa, et al. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence*, 3(4):324–333, 2021.

[136] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[137] Laura Rieger and Lars Kai Hansen. A simple defense against adversarial attacks on heatmap explanations. *arXiv preprint arXiv:2007.06381*, 2020.

[138] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021. doi: 10.1073/pnas.2016239118. URL `https://www.pnas.org/doi/abs/10.1073/pnas.2016239118`.

[139] Carol A Rohl, Charlie EM Strauss, Kira MS Misura, and David Baker. Protein structure prediction using rosetta. In *Methods in enzymology*, volume 383, pages 66–93. Elsevier, 2004.

[140] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI*, 2018.

[141] Grant M Rotskoff and Eric Vanden-Eijnden. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

[142] Alessandro Rozza, Mario Manzo, and Alfredo Petrosino. A novel graph-based fisher kernel method for semi-supervised learning. In *Proceedings of the 2014 22nd International Conference on Pattern Recognition*, ICPR '14, page 3786–3791, USA, 2014. IEEE Computer Society. ISBN 9781479952090. doi: 10.1109/ICPR.2014.650. URL `https://doi.org/10.1109/ICPR.2014.650`.

[143] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4): 259–268, 1992.

[144] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[145] Chris Sander and Reinhard Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Bioinformatics*, 9(1):56–68, 1991.

[146] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning*, pages 71–79. PMLR, 2013.

[147] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *ICML*, pages 3145–3153. PMLR, 2017.

[148] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[149] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

[150] Shannon Stefl, Hafumi Nishi, Marharyta Petukh, Anna R Panchenko, and Emil Alexov. Molecular mechanisms of disease-causing missense mutations. *Journal of molecular biology*, 425(21):3919–3936, 2013.

[151] Martin Steinegger, Markus Meier, Milot Mirdita, Harald Vöhringer, Stephan J Haunsberger, and Johannes Söding. Hh-suite3 for fast remote homology detection and deep protein annotation. *BMC bioinformatics*, 20 (1):1–15, 2019.

[152] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[153] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML*, pages 3319–3328. PMLR, 2017.

[154] Baris E Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H Wu. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

[155] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[156] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[157] Richard Tomsett, Amy Widdicombe, Tianwei Xing, Supriyo Chakraborty, Simon Julier, Prudhvi Gurram, Raghuveer Rao, and Mani Srivastava. Why the failure? how adversarial examples can provide insights for interpretable machine learning. In *2018 21st FUSION*, pages 838–845. IEEE, 2018.

[158] Joo Chuan Tong. *BLOcks SUbstitution Matrix (BLOSUM)*, pages 152–152. Springer New York, New York, NY, 2013. ISBN 978-1-4419-9863-7. doi: 10.1007/978-1-4419-9863-7_942. URL `https://doi.org/10.1007/978-1-4419-9863-7_942`.

[159] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020.

[160] Claudio Turchetti and Laura Falaschetti. A manifold learning approach to dimensionality reduction for modeling data. *Information Sciences*, 491: 16–29, 2019.

[161] Adaku Uchendu, Daniel Campoy, Christopher Menart, and Alexandra Hildenbrandt. Robustness of bayesian neural networks to white-box adversarial attacks. In *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 72–80. IEEE, 2021.

[162] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 2000. URL `https://EconPapers.repec.org/RePEc:cup:cbooks:9780521784504`.

[163] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[164] Michele Vendruscolo, Edo Kussell, and Eytan Domany. Recovery of protein structure from contact maps. *Folding and Design*, 2(5):295–306, 1997.

[165] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[166] Jesse Vig, Ali Madani, Lav R Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. Bertology meets biology: interpreting attention in protein language models. *arXiv preprint arXiv:2006.15222*, 2020.

[167] Martin J Wainwright and Michael Irwin Jordan. *Graphical models, exponential families, and variational inference.* Now Publishers Inc, 2008.

[168] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, Alexander G Ororbia II, Xinyu Xing, Xue Liu, and C Lee Giles. Learning adversary-resistant deep neural networks. *arXiv preprint arXiv:1612.01401*, 2016.

[169] Sheng Wang, Siqi Sun, Zhen Li, Renyu Zhang, and Jinbo Xu. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLoS computational biology*, 13(1):e1005324, 2017.

[170] Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

[171] Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for bayesian neural networks. In *Conference on Uncertainty in Artificial Intelligence*, pages 1198–1207. PMLR, 2020.

[172] Matthew Wicker, Luca Laurenti, Andrea Patane, Zhuotong Chen, Zheng Zhang, and Marta Kwiatkowska. Bayesian inference with certifiable adversarial robustness. In *International Conference on Artificial Intelligence and Statistics*, pages 2431–2439. PMLR, 2021.

[173] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[174] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[175] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[176] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1369–1378, 2017.

[177] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17 (2):151–178, 2020.

[178] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[179] Nanyang Ye and Zhanxing Zhu. Bayesian adversarial learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6892–6901. Curran Associates Inc., 2018.

[180] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Suggala, David I Inouye, and Pradeep K Ravikumar. On the (in) fidelity and sensitivity of explanations. *NeurIPS*, 32:10967–10978, 2019.

[181] Jongmin Yoon, Sung Ju Hwang, and Juho Lee. Adversarial purification with score-based generative models. In *International Conference on Machine Learning*, pages 12062–12072. PMLR, 2021.

[182] Matthew Yuan, Matthew Wicker, and Luca Laurenti. Gradient-free adversarial attacks for bayesian neural networks. *arXiv preprint arXiv:2012.12640*, 2020.

[183] Jiaru Zhang, Yang Hua, Zhengui Xue, Tao Song, Chengyu Zheng, Ruhui Ma, and Haibing Guan. Robust bayesian neural networks by spectral expectation bound regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3815–3824, 2021.

[184] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable deep learning under fire. In *29th USENIX*, 2020.

[185] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

Appendix A

Architectures and Hyperparameters

Half moons grid search

Posterior samples	{250}
HMC warmup samples	{100, 200, 500}
Training inputs	{5000, 10000, 15000}
Hidden size	{32, 128, 256, 512}
Nonlinear activation	Leaky ReLU
Architecture	2 fully connected layers

Table A.1: Hyperparameters for training BNNs in Figure 4.2

HMC training hyperparameters for Chapter 4

Dataset	MNIST	Fashion MNIST
Training inputs	60k	60k
Architectures	Fully connected	Fully connected
Hidden size	512	1024
Nonlinear activation	Leaky ReLU	Leaky ReLU
Warmup samples	100	100
Numerical Integrator Stepsize	0.5	0.5
Number of steps for Numerical Integrator	10	10

Table A.2: Hyperparameters for training BNNs using HMC in Figures 4.4 and 4.5.

VI training hyperparameters for Chapter 4

Dataset	MNIST	Fashion MNIST
Training inputs	60k	60k
Architecture	Convolutional	Convolutional
Hidden size	512	1024
Nonlinear activations	Leaky ReLU	Leaky ReLU
Training epochs	5	15
Learning rate	0.01	0.001

Table A.3: Hyperparameters for training BNNs using VI in Figures 4.3, 4.4 and 4.5.

HMC MNIST/Fashion MNIST grid search for Chapter 4

Posterior samples	{250, 500, 750*}
Numerical Integrator Stepsize	{0.01, 0.005*, 0.001, 0.0001}
Numerical Integrator Steps	{10*, 15, 20}
Hidden size	{128, 256, 512*}
Nonlinear activation	{relu*, tanh, sigmoid}
Architecture	{1*,2,3} fully connected layers

Table A.4: Hyperparameters for training BNNs with HMC in Figure 4.6. * denotes the parameters used in Table 1 of the main text.

SVI MNIST/Fashion MNIST grid search for Chapter 4

Learning Rate	{0.001*}
Minibatch Size	{128, 256*, 512, 1024}
Hidden size	{64, 128, 256, 512, 1024*}
Nonlinear activation	{relu*, tanh, sigmoid}
Architecture	{1*,2,3} fully connected layers
Training epochs	{3,5*,7,9,12,15} epochs

Table A.5: Hyperparameters for training BNNs with SVI in Figure 4.7. * denotes the parameters used in Table 1 of the main text.

SGD MNIST/Fashion MNIST grid search for Chapter 4

Learning Rate	{0.001, 0.005, 0.01, 0.05}
Hidden size	{64, 128, 256, 512}
Nonlinear activation	{relu, tanh, sigmoid}
Architecture	{2, 3, 4, 5} fully connected layers
Training epochs	{5, 10, 15, 20, 25} epochs

Table A.6: Hyperparameters for training BNNs with SGD in Figures 4.6 and 4.7.

SGD training hyperparameters for Chapter 5

Dataset	MNIST	CIFAR-10
Training inputs	60k	60k
Architectures	Convolutional	Convolutional
N. of learnable layers	4	8
Nonlinear activations	ReLU	ReLU

Table A.7: Hyperparameters for training deterministic NNs in Chapter 5. Architectures are reported in Tables A.8 and A.9

MNIST architecture for Chapter 5

Layer	`in_channels`	`out_channels`	`kernel_size`
2D Conv.	784	32	3
2D Conv.	32	64	3
F.c.	64	128	-
F.c.	128	10	-

Table A.8: Learnable layers for the Keras architecture trained on MNIST in Chapter 5. Training hyperparamters are reported in Table A.7.

Fashion MNIST architecture for Chapter 5

Layer	`in_channels`	`out_channels`	`kernel_size`
2D Conv.	784	32	3
2D Conv.	32	32	3
2D Conv.	32	64	3
2D Conv.	64	64	3
2D Conv.	64	128	3
2D Conv.	128	128	3
F.c.	128	128	-
F.c.	128	10	-

Table A.9: Learnable layers for the Keras architecture trained on Fashion MNIST in Chapter 5. Training hyperparamters are reported in Table A.7.

HMC hyperparameters for Chapter 6

Dataset	MNIST	Fashion MNIST
Training inputs	60k	60k
Architecture	Fully Connected	Fully Connected
Hidden size	1024	1024
Nonlinear activation	ReLU	ReLU
Warmup samples	100	100
Posterior Samples	500	500
Numerical Integrator Stepsize	0.002	0.001
Number of steps for Numerical Integrator	10	10

Table A.10: Hyperparameters for training BNNs using HMC in Figures 4.3 and 4.4.

VI hyperparameters for Chapter 6

Dataset	MNIST	Fashion MNIST
Training inputs	60k	60k
Architecture	Convolutional	Convolutional
Hidden size	512	1024
Nonlinear activation	Leaky ReLU	Leaky ReLU
Training epochs	5	10
Learning rate	0.01	0.001

Table A.11: Hyperparameters for training BNNs using VI in Figures 4.3 and 4.5.

Convolutional architectures in Chapter 6

Idx	Layer	Parameters
0	2D Conv.	`in_channels` = 784 `out_channels` = 32 `kernel_size` = 5
3	2D Conv.	`in_channels` = 32 `out_channels` = hidden size `kernel_size` = 5
7	F.c.	`in_features` = hidden size `out_features` = 10

Table A.12: Learnable layers and corresponding indexes in PyTorch for the convolutional architecture used in Chapter 6. Hidden size is reported in Tables A.10 and A.11.

Fully connected architectures in Chapter 6

Idx	Layer	Parameters
1	F. c.	`in_features` = 784 `out_features` = hidden size
3	F. c.	`in_features` = hidden size `out_features` = hidden size
5	F. c.	`in_features` = hidden size `out_features` = 10

Table A.13: Learnable layers and corresponding indexes in PyTorch for the fully connected architecture used in Chapter 6. Hidden size is reported in Tables A.10 and A.11.

Appendix B

Additional Experiments

In this section we present additional empirical results from Chapters 5-7. Precisely, we test:

- the prediction accuracy of RP-Ensemble model against FGSM, PGD, DeepFool and C&W attacks (Section B.1);
- the LRP robustness SVI and HMC-trained Bayesian NNs on MNIST and Fashion MNIST datasets under FGSM and PGD attacks, with LRP parameters set to $\epsilon = 0.1, \gamma = 0.1, \alpha = 1, \beta = 0$ (Section B.2);
- the change in hidden representations and 3D coordinates induced by adversarial mutations using ESM-1b and MSA Transformer models on domains PF00533 and PF00627 and on ProTherm database (Section B.3).

B.1 Prediction accuracy of RP-Ensemble

Prediction accuracy (%)	Test set	FGSM	PGD	DeepFool	C&W
Baseline model					
	99.13	5.91	0.71	34.83	28.96
Adversarially trained models					
FGSM	99.13	**98.91**	32.55	96.08	95.82
PGD	99.10	44.60	**99.02**	98.34	97.19
DeepFool	99.03	23.11	35.55	**99.20**	95.70
C&W	99.10	14.74	3.85	94.63	**99.06**
RP-Ensemble on (n_proj, size_proj) combinations					
$(6, 8)$	97.66	59.71	53.68	93.94	92.12
$(9, 8)$	97.57	61.55	56.23	94.37	92.73
$(12, 8)$	97.45	62.93	58.85	94.61	92.97
$(15, 8)$	97.47	**64.82**	**60.30**	94.62	93.12
$(6, 12)$	98.12	57.29	52.12	95.72	93.75
$(9, 12)$	98.02	59.30	55.52	95.82	94.25
$(12, 12)$	97.97	60.54	57.77	95.96	94.39
(15 , 12)	97.91	61.65	59.95	95.84	94.42
$(6, 16)$	98.22	57.09	51.90	96.26	94.36
$(9, 16)$	98.33	58.10	53.77	96.43	94.95
$(12, 16)$	98.32	58.93	55.78	96.54	95.13
$(15, 16)$	98.26	59.85	57.42	96.62	95.11
$(6, 20)$	98.49	54.37	50.02	96.51	94.89
$(9, 20)$	98.42	56.28	52.60	96.63	95.18
$(12, 20)$	98.40	57.56	54.52	96.67	95.24
$(15, 20)$	98.40	57.91	55.57	**96.80**	**95.29**

Table B.1: Test accuracy of the baseline, its robust versions and RP Ensemble model on MNIST dataset. The robust models are the result of adversarial training on the perturbed training sets. RP Ensemble model is been trained on multiple combinations of number of projections and size of each projection. The evaluations are performed on the original test set and its adversarially perturbed versions.

Prediction accuracy (%)	Test set	DeepFool attack	C&W attack
Baseline model			
	76.52	39.77	0.00
Adversarially trained models			
DeepFool training set	83.16	**83.01**	81.67
C&W training set	83.44	83.23	**82.79**
RP-Ensemble model on (n_proj, size_proj) combinations			
$(3, 4)$	**67.93**	58.52	48.48
$(6, 4)$	64.59	58.81	53.78
$(9, 4)$	63.15	59.06	55.48
$(12, 4)$	61.93	58.65	56.07
$(3, 8)$	67.66	**61.00**	53.61
$(6, 8)$	64.83	60.86	57.21
$(9, 8)$	63.36	60.35	58.03
$(12, 8)$	62.99	60.81	**58.75**

Table B.2: Prediction accuracy (%) of the baseline, its robust versions and RP-Ensemble model on CIFAR-10 dataset. The robust models are the result of adversarial training on the perturbed training sets. RP-Ensemble model is been trained on multiple combinations of number of projections and size of each projection. The evaluations are performed on the original test set and its adversarially perturbed versions.

Prediction accuracy (%)	Test set	FGSM	PGD	DeepFool	C&W
			Baseline model		
	99.13	5.91	0.71	34.83	28.96
			Adversarially trained models		
FGSM	99.13	**98.91**	32.55	96.08	95.82
PGD	99.10	44.60	**99.02**	98.34	97.19
DeepFool	99.03	23.11	35.55	**99.20**	95.70
C&W	99.10	14.74	3.85	94.63	**99.06**
			RP-Regularizer model		
$\mathcal{R}_{\text{v1}}$, $\lambda = 0.4$	97.92	62.34	38.96	93.24	90.75
$\mathcal{R}_{\text{v2}}$, $\lambda = 0.4$	97.82	63.25	42.37	93.86	91.56
$\mathcal{R}_{\text{v1}}$, $\lambda = 0.5$	97.53	**69.12**	**52.39**	**94.44**	**91.96**
$\mathcal{R}_{\text{v2}}$, $\lambda = 0.5$	98.06	60.64	36.28	93.92	91.05
$\mathcal{R}_{\text{v1}}$, $\lambda = 0.6$	97.80	62.78	42.61	94.05	91.73
$\mathcal{R}_{\text{v2}}$, $\lambda = 0.6$	97.69	65.25	45.77	93.45	90.70

Table B.3: Test accuracy of RP-Regularizer on MNIST. We compare the baseline model, the adversarially trained robust models and two different versions of RP-Regularizer model, namely $\mathcal{R}_{\text{v1}}$ (5.1) and $\mathcal{R}_{\text{v2}}$ (5.2). Adversarial perturbations are produced on the baseline model using FGSM, PGD, Deepfool and Carlini & Wagner attacks.

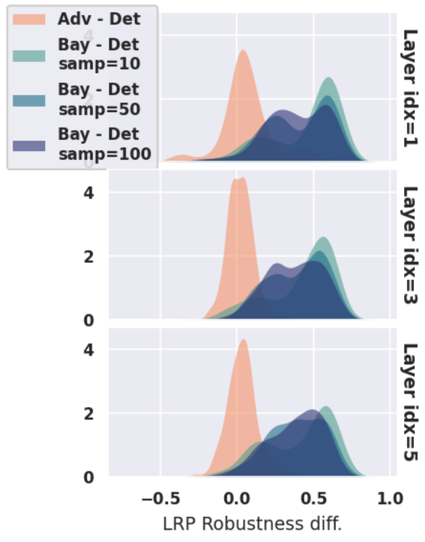
Prediction accuracy (%)	Test set	DeepFool	C&W
Adversarially trained models			
	76.52	39.77	0.00
Adversarially trained models			
DeepFool	83.16	83.01	81.67
C&W	83.44	83.23	82.79
RP-Regularizer model, $\lambda = 0.5$			
$\mathcal{R}_{\mathrm{v1}}$	56.51	55.20	54.29
$\mathcal{R}_{\mathrm{v2}}$	57.10	57.85	56.47

Table B.4: Test accuracy of RP-Regularizer on MNIST. We compare the baseline model, the adversarially trained robust models and two different versions of RP-Regularizer model, namely $\mathcal{R}_{\mathrm{v1}}$ (5.1) and $\mathcal{R}_{\mathrm{v2}}$ (5.2). Adversarial perturbations are produced on the baseline model using Deepfool and Carlini & Wagner attacks.

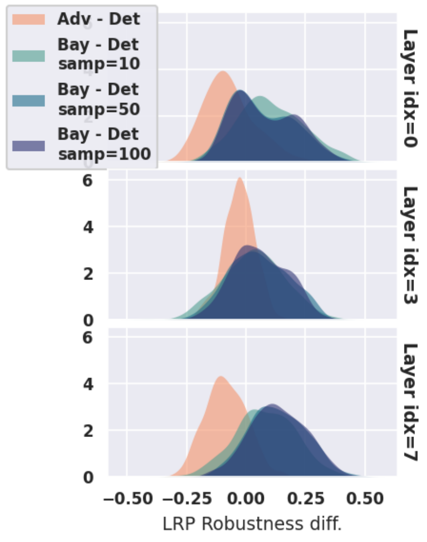
B.2 Bayesian LRP robustness

p-value	**symbol**
$p > 0.05$	n.s.
$p \leq 0.05$	*
$p \leq 0.01$	**
$p \leq 0.001$	***
$p \leq 0.0001$	*****

Table B.5: Asterisk notation for Mann-Whitney test.

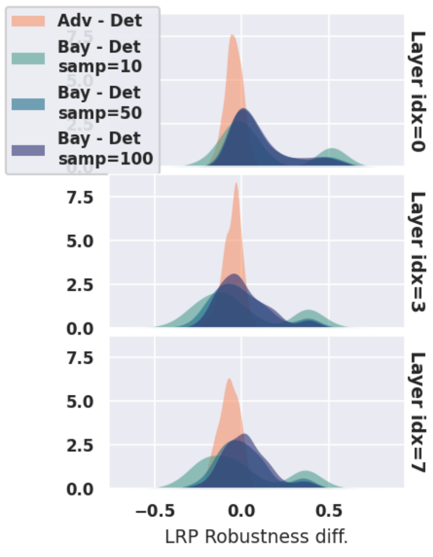


(a)

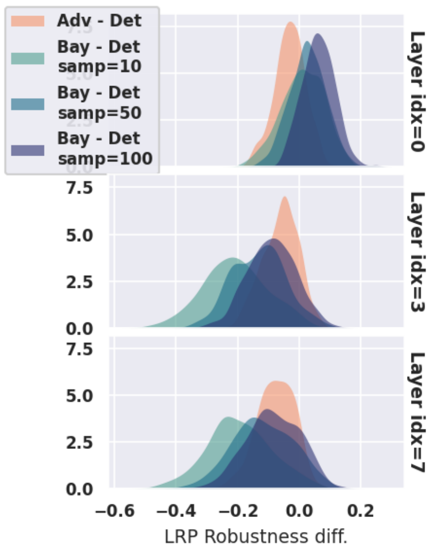


(b)

Figure B.1: LRP robustness differences for FGSM (a) and PGD (b) attacks computed on 500 test points from Fashion MNIST dataset using the Epsilon rule. NNs in (a) have a fully connected architecture and the BNN is trained with HMC. NNs in (b) have a convolutional architecture and the BNN is trained with VI. BNNs are tested using an increasing number of samples $(10, 50, 100)$.

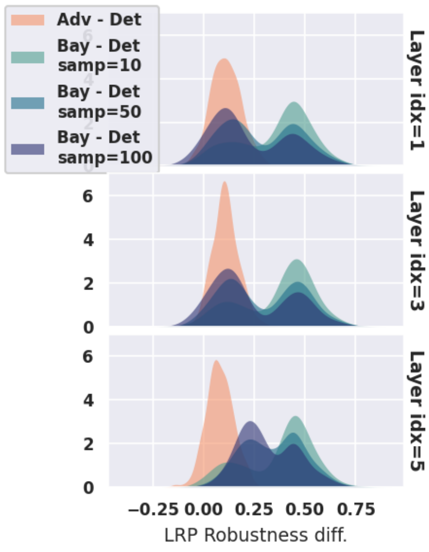


(a)

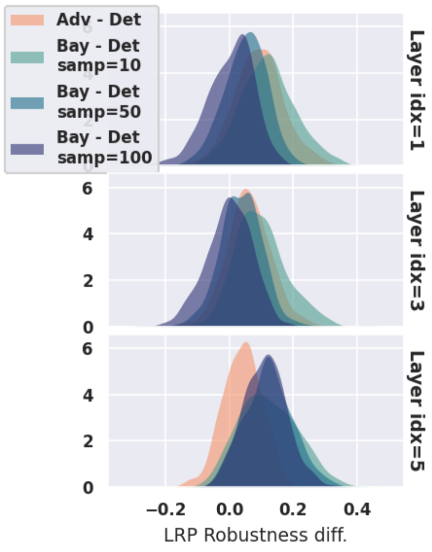


(b)

Figure B.2: LRP robustness differences for FGSM (a) and PGD (b) attacks computed on 500 test points from MNIST dataset using the Epsilon rule. NNs have a convolutional architecture. BNNs are trained with VI and tested using an increasing number of samples $(10, 50, 100)$. Layer indexes refer to the learnable layers in the architectures.

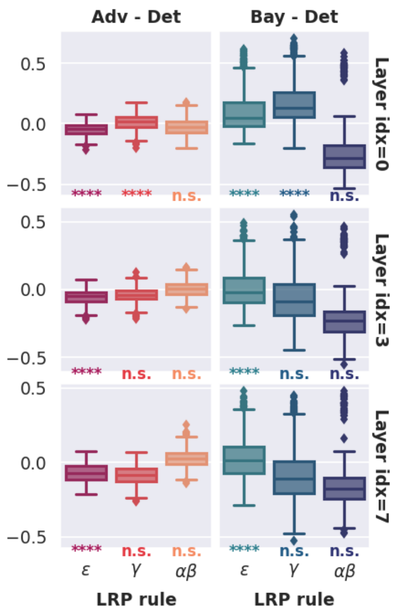


(a)

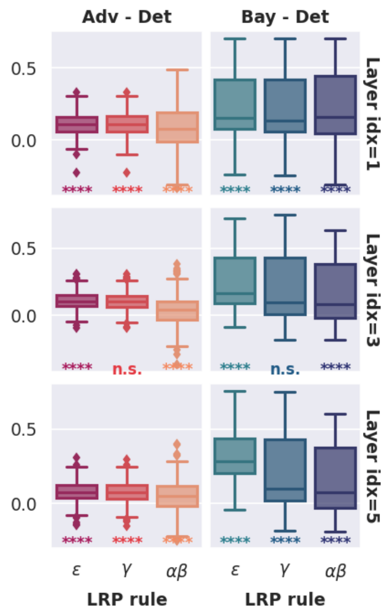


(b)

Figure B.3: LRP robustness differences for FGSM (a) and PGD (b) attacks computed on 500 test points from MNIST dataset using the Epsilon rule. NNs have a fully connected architecture. BNNs are trained with HMC and tested using an increasing number of samples $(10, 50, 100)$.

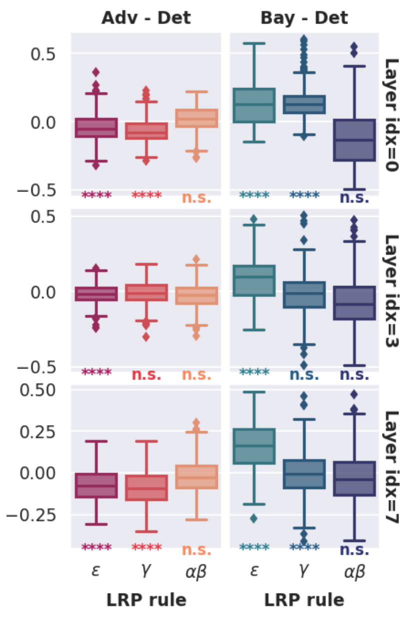


(a)

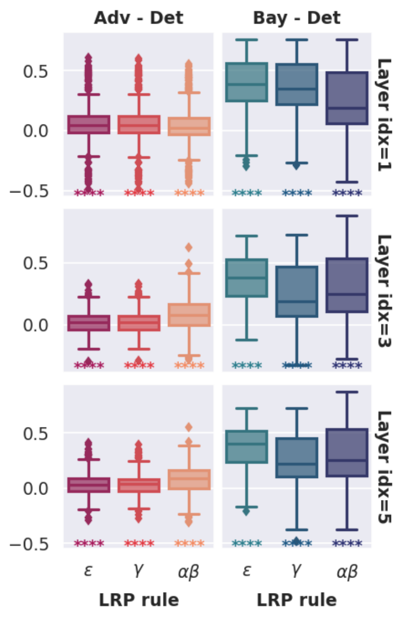


(b)

Figure B.4: LRP robustness differences for FGSM attacks computed on 500 test points from MNIST dataset using Epsilon, Gamma and Alpha-Beta rules on the Top$_{20}$ pixels. NNs in (a) have a convolutional architecture, while NNs in (b) have a fully connected architecture. The BNN in (a) is trained with VI and the BNN in (b) is trained with HMC; Both are tested using 100 posterior samples.

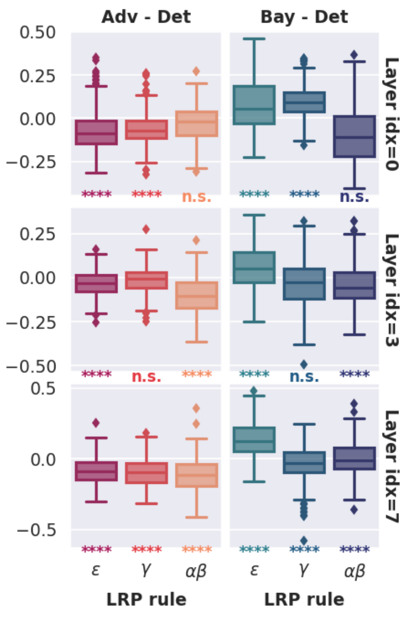


(a)

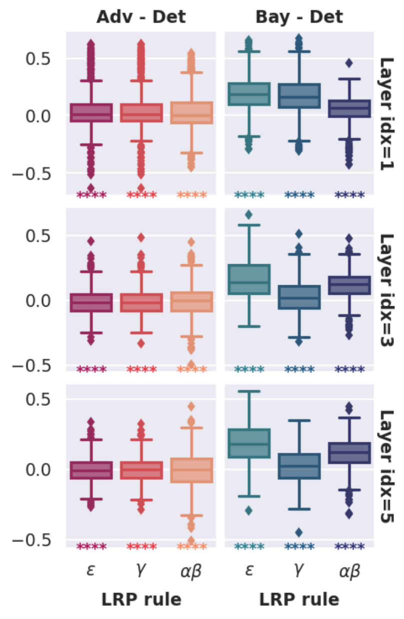


(b)

Figure B.5: LRP robustness differences for FGSM attacks computed on 500 test points from Fashion MNIST dataset using Epsilon, Gamma and Alpha-Beta rules on the Top$_{20}$ pixels. NNs in (a) have a convolutional architecture, while NNs in (b) have a fully connected architecture. The BNN in (a) is trained with VI and the BNN in (b) is trained with HMC; Both are tested using 100 posterior samples.



(a)



(b)

Figure B.6: LRP robustness differences for PGD attacks computed on 500 test points from Fashion MNIST dataset using Epsilon, Gamma and Alpha-Beta rules on the Top$_{20}$ pixels. NNs in (a) have a convolutional architecture, while NNs in (b) have a fully connected architecture. The BNN in (a) is trained with VI and the BNN in (b) is trained with HMC; Both are tested using 100 posterior samples.

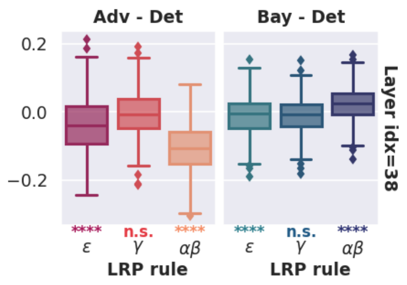
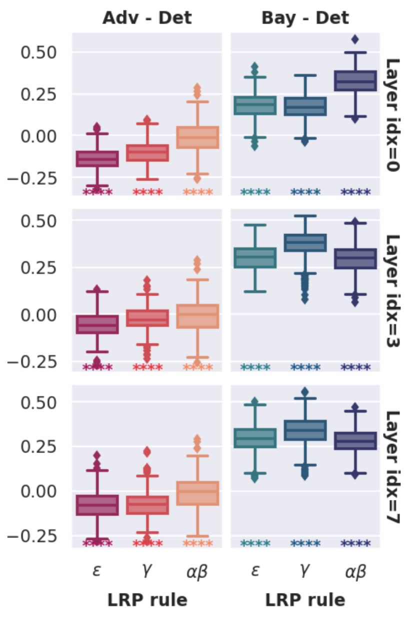
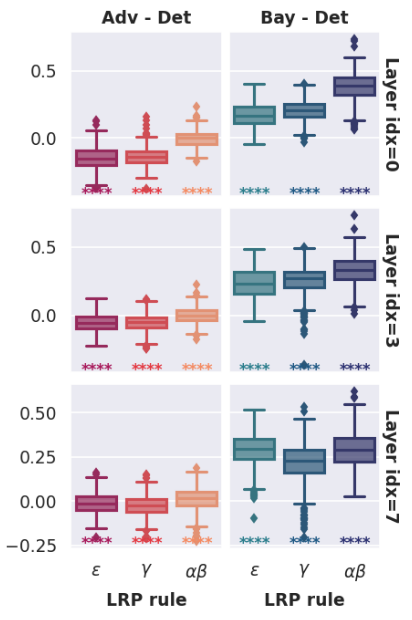


Figure B.7: LRP robustness differences for PGD attacks computed on 500 test points from CIFAR-10 dataset using Epsilon, Gamma and Alpha-Beta rules on the Top$_{20}$ pixels. NNs have a ResNet20 architecture from bayesian_torch library [87]. The BNN is trained with VI and tested using 100 posterior samples.



(a)



(b)

Figure B.8: LRP robustness differences for top-k (a) and target region (b) attacks [63] (Sec. 3.5.3) computed on 500 test points from MNIST (a) and Fashion MNIST (b) datasets using Epsilon, Gamma and Alpha-Beta rules on the Top$_{20}$ pixels. NNs in (a) have a convolutional architecture, while NNs in (b) have a fully connected architecture. BNNs are trained with VI and tested using 100 posterior samples.

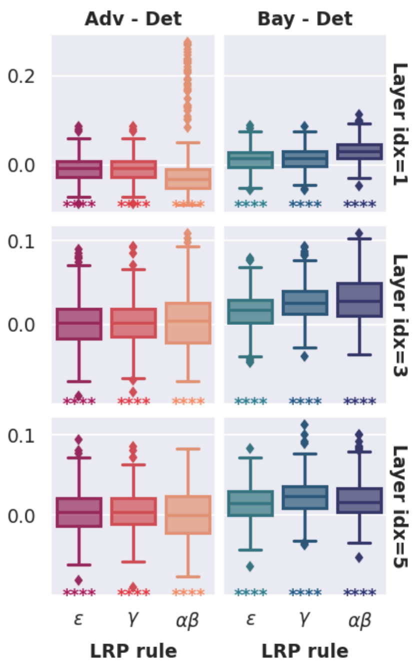


Figure B.9: LRP robustness differences for target beta attacks [46] (Sec. 3.5.3) computed on 500 test points from MNIST dataset using Epsilon, Gamma and Alpha-Beta rules on the Top$_{60}$ pixels. NNs have a fully connected architecture (Tab. A.13). The BNN is trained with HMC and tested using 100 posterior samples.

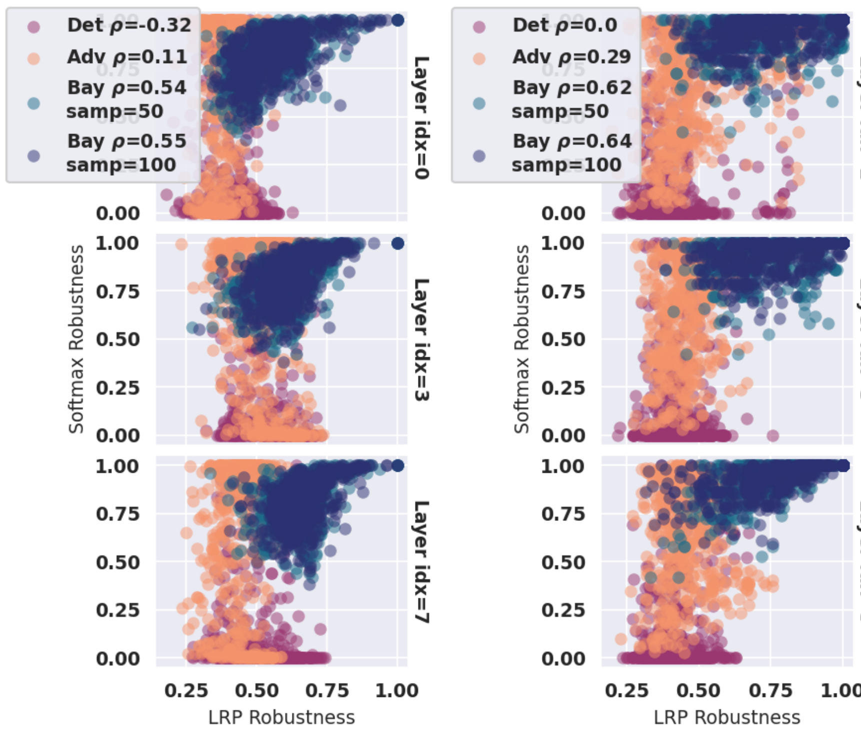


Figure B.10: LRP vs softmax robustness of deterministic, adversarially trained and Bayesian NNs trained on Fashion MNIST dataset and tested against FGSM attacks. ρ denotes the correlation coefficient. LRP Robustness is computed with the Epsilon rule on the 20% most relevant pixels. BNNs are trained with VI (a) and HMC (b) and are tested using an increasing number of samples $(50, 100)$.

B.3 Structural change induced by adversarial mutations

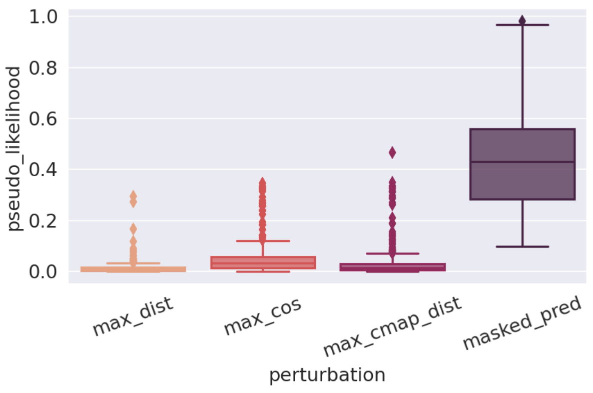
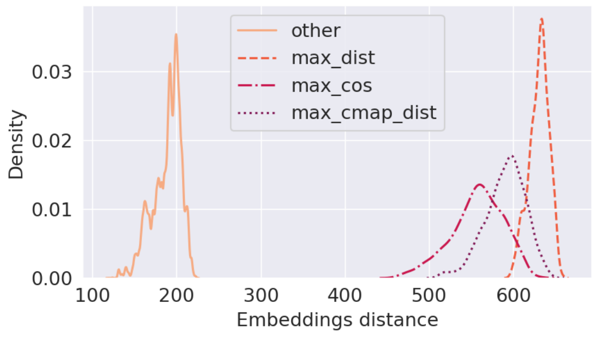
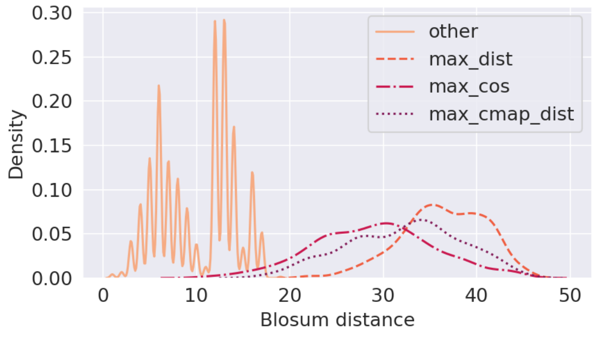


Figure B.11: Pseudo-likelihood of adversarial (columns 1-3) and masked prediction (column 4) mutations at target token indexes. Values refer to 3 sites mutations obtained from ESM-1b on 100 sequences from domain PF00533.

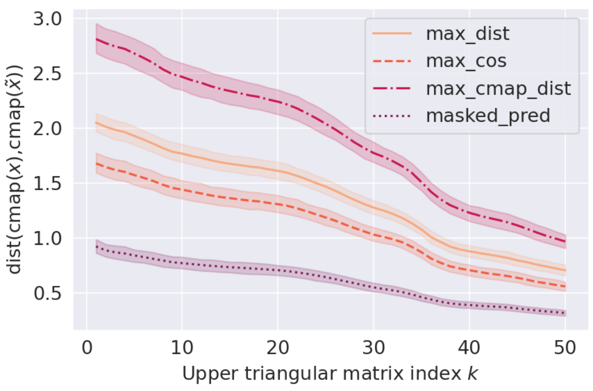


(a)

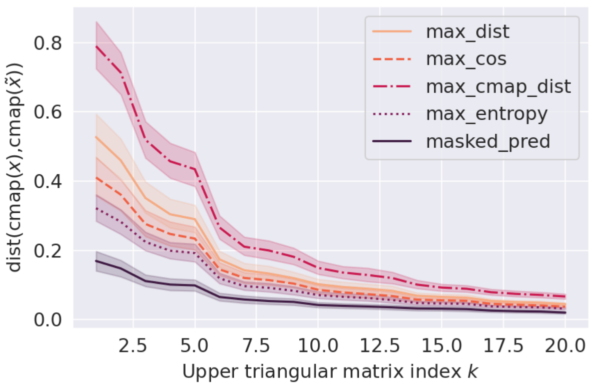


(b)

Figure B.12: L1 distances $||\mathbf{z} - \tilde{\mathbf{z}}||_1$ between the embeddings of original and perturbed sequences in the first embedding space (a) and Blosum distances $\text{BLOSUM}(\mathbf{x}, \tilde{\mathbf{x}})$ between original and perturbed sequences. Adversarial mutations on 3 sites are computed using ESM-1b model on domain PF00533. "Other" refers to adversarial perturbations obtained from all the discarded plausible token substitutions at the chosen target positions.

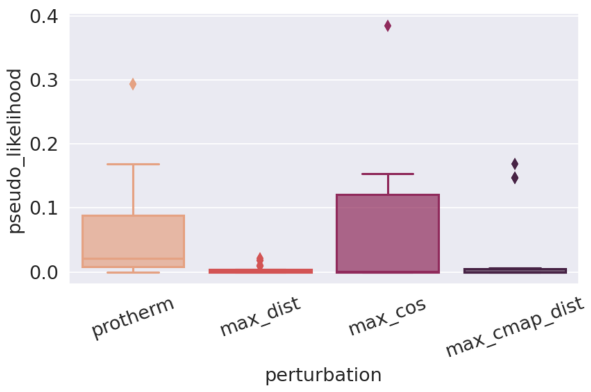


(a)

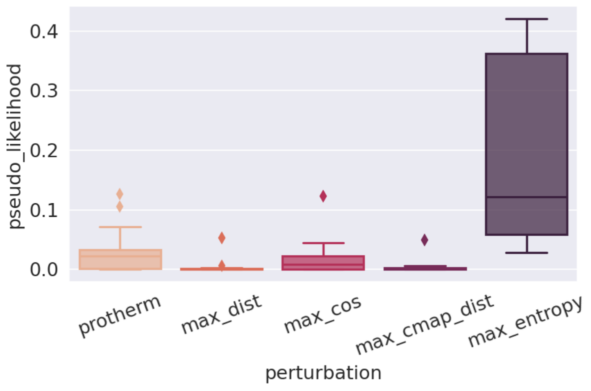


(b)

Figure B.13: Distances between upper submatrices of contact maps of original and perturbed sequences, as the index k of upper triangular submatrices increases. Adversarial perturbations at 3 sites are computed by ESM-1b model 100 sequences from domain PF00533 in (a) and by MSA Transformer model on 100 sequences from domain PF00627 in (b).

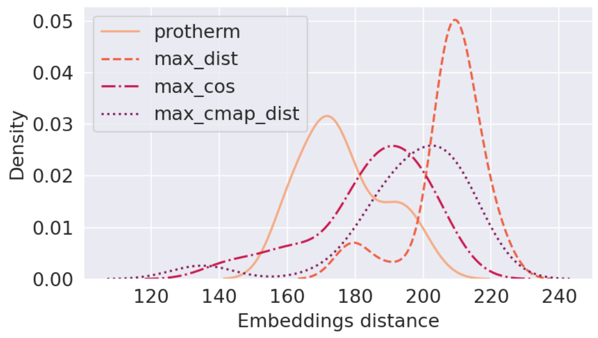


(a)

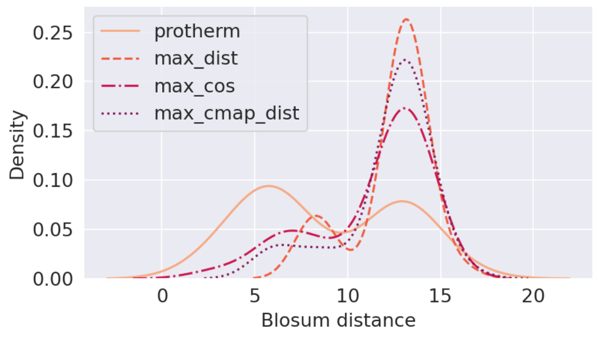


(b)

Figure B.14: Pseudo-likelihood of ProTherm (first column) and adversarial (other columns) mutations at target token indexes. Values refer to single residue mutations obtained on ProTherm database using ESM-1b model in (a) and MSA Transformer model in (b).



(a)



(b)

Figure B.15: L1 distances $||\mathbf{z} - \tilde{\mathbf{z}}||_1$ between the embeddings of original and perturbed sequences in the first embedding space (a) and Blosum distances BLOSUM$(\mathbf{x}, \tilde{\mathbf{x}})$ between original and perturbed sequences. Adversarial mutations on 3 sites are computed using ESM-1b model on ProTherm database.

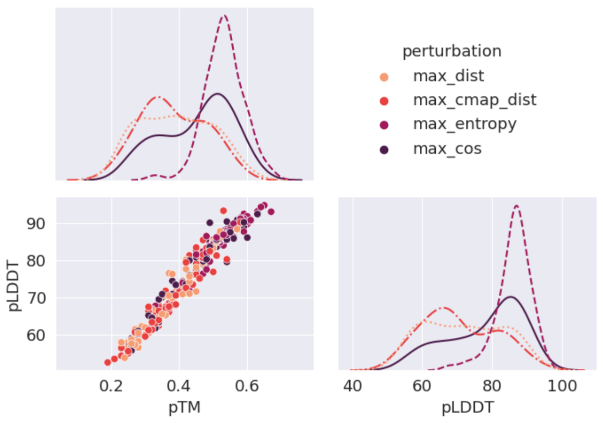


Figure B.16: Confidence scores for structure prediction performed in ColabFold on adversarial sequences from PF00627 such that pLDDT > 80 on the original structures.

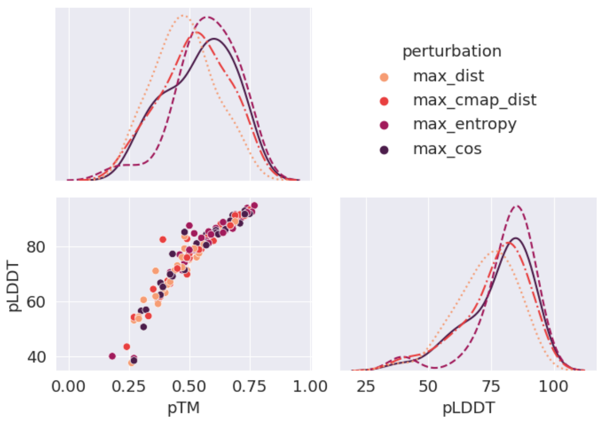


Figure B.17: Confidence scores for structure prediction performed in ColabFold on adversarial sequences from PF00533 such that pLDDT > 80 on the original structures.

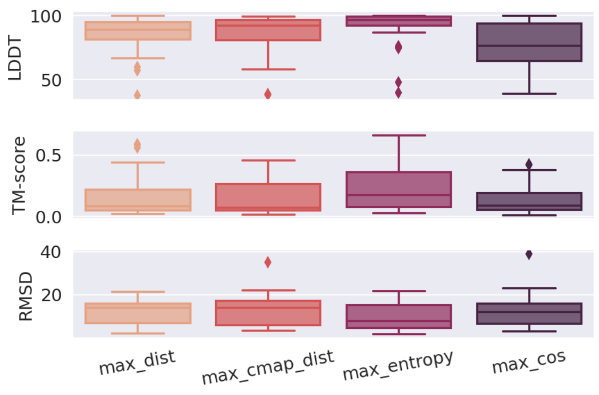


Figure B.18: LDDT, TM and RMSD scores between original and perturbed structures for 3 sites mutations on 100 sequences from domain PF00533. Adversarial perturbations are computed by MSA Transformer model, while structure predictions are performed in ColabFold.

Index

Notation

Acronyms

NN	Neural Network
DNN	Deep Neural Network
BNN	Bayesian Neural Network
SGD	Stochastic Gradient Descent
VI	Variational Inference
HMC	Hamiltonian Monte Carlo
FGSM	Fast Gradient Sign Method
PGD	Projected Gradient Descent
C&W	Carlini & Wagner
ZOO	Zeroth Order Optimization
TV	Total Variation
RP	Random Projections
LRP	Layer-wise Relevance Propagation
LM	Language Model
PDB	Protein Data Bank
Pfam	Protein family database
BLOSUM	BLOcks SUbstitutions Matrix
MSA	Multiple Sequence Alignment
ESM	Evolutionary Scale Modeling
ProTherm	Thermodynamic Database for Proteins and Mutants
Cmap	Contact map
LDDT	Local Distance Difference Test
TM-score	Template Modeling score
RMSD	Root-Mean-Square Deviation
CPU	Central Processing Unit
RAM	Random Access Memory

Abbreviations

e.g.	Exempli gratia (“for the sake of example”)
i.e.	Id est (“that is”)
s.t.	Such that
w.r.t.	With respect to
w.l.o.g.	Without loss of generality
a.e.	Almost everywhere
a.s.	Almost surely
i.i.d.	Independent and identically distributed

Neural network architecture and training

$\mathcal{S} \subset \mathbb{R}^d$	Ambient space
$\mathcal{M} \subset \mathcal{S}$	Smooth closed data manifold
$\mathbf{x} \in \mathcal{M}$	Point on the data manifold
$\|\| \cdot \|\|_p$	p-norm in a real vector space
$f^{true} : \mathcal{M} \to \mathbb{R}^K$	Target function
$f : \mathcal{S} \times \mathbb{R}^{n_\mathbf{w}} \to \mathbb{R}^K$	Neural network with $L+1$ layers and n_m neurons in each layer
$\mathbf{w} \in \mathbb{R}^{n_\mathbf{w}}$	Vector of weights and biases in the neural network, where $n_\mathbf{w} = n_0 + \ldots + n_L$ is the total number of weights across the $L+1$ layers
$f^\infty : \mathcal{M} \to \mathbb{R}^K$	Infinitely wide architecture
$D_N = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,N}$	Set of N training points $\mathbf{x} \in \mathcal{M}$ and with labels y
$\mathcal{L}(\mathbf{x}, \mathbf{w}) = \ell(f(\mathbf{x}, \mathbf{w}), f^{true}(\mathbf{x}))$	Training loss function

Bayesian neural networks and Bayesian inference

$p(\mathbf{w})$	Prior distribution over $\mathbf{w}$
$p(D)$	Data distribution on $\mathcal{M}$
$p(D\vert\mathbf{w})$	Likelihood of D given $\mathbf{w}$
$p\left(\mathbf{w}\vert D\right) \propto p\left(D\vert\mathbf{w}\right) p\left(\mathbf{w}\right)$	Posterior distribution of the weights $\mathbf{w}$
$f : \mathcal{M} \times \Omega \to \mathbb{R}^K$	Bayesian Neural Network with weights $\mathbf{w} \in \Omega \subset \mathbb{R}^{n_\mathbf{w}}$
$p(f(\cdot, \mathbf{w})\vert D) = \mathbb{E}_{p(\mathbf{w}\vert D)}[f(\cdot, \mathbf{w})]$	Posterior predictive distribution for a BNN f trained on a dataset D, where $\mathbb{E}_{p(\mathbf{w}\vert D)}[\cdot]$ denotes the expectation under the posterior distribution
$T_\mathbf{x}\mathcal{M}$	Tangent space to the data manifold $\mathcal{M}$ at $\mathbf{x} \in \mathcal{M}$
$T_\mathbf{x}\mathcal{M}^\perp$	Orthogonal complement of $T_\mathbf{x}\mathcal{M}$ in $\mathcal{S}$
$\nabla_\mathbf{x}\mathcal{L}(\mathbf{x}, \mathbf{w})$	Gradient of the loss function $\mathcal{L}$ w.r.t. $\mathbf{x}$

Adversarial attacks

$\tilde{\mathbf{x}}$	Generic adversarial attack on $\mathbf{x}$ against f
$\tilde{\mathbf{x}} = \mathbf{x} + \epsilon \cdot \text{sgn } \nabla_\mathbf{x}\mathcal{L}(\mathbf{x}, \mathbf{w})$	FGSM adversarial attack against a deterministic NN f with training loss $\mathcal{L}$ and attack strength ϵ
$\tilde{\mathbf{x}} = \mathbf{x} + \epsilon \cdot \text{sgn } \mathbb{E}_{p(\mathbf{w}\vert D)}\left[\nabla_\mathbf{x}\mathcal{L}(\mathbf{x}, \mathbf{w})\right]$	FGSM adversarial attack against a BNN f with training loss $\mathcal{L}$ and attack strength ϵ

Random projections

$X \in \mathbb{R}^{N\times d}$	N training points from the data manifold $\mathcal{M} \subset \mathbb{R}^d$, represented in matrix form
$Q_j \in \mathbb{R}^{k_j\times d}$	Random projection matrix from $\mathbb{R}^d$ to $\mathbb{R}^{k_j}$, whose weights are independently drawn from $\mathcal{N}(0, 1/k_j)$
$\mathcal{P}_j(X) = XQ_j^T \in \mathbb{R}^{N\times k_j}$	k_j-dimensional projection of the data matrix X
$\psi_j : \mathbb{R}^{k_j} \times \mathbb{R}^{n_{\mathbf{w}_j}} \to \mathbb{R}^K$	Classifier trained on projections $\mathcal{P}_j(X)$, where $n_{\mathbf{w}_j}$ is the size of the j-th weight space
$Q_j^\dagger \in \mathbb{R}^{d\times k_j}$	Moore-Penrose pseudo-inverse of Q_j
$\mathcal{R}_{\text{v1}}, \mathcal{R}_{\text{v2}}$	RP-Regularizers

Saliency explanations

$R(\mathbf{x}, \mathbf{w})$	Relevance heatmap of an image $\mathbf{x}$ w.r.t. the weights $\mathbf{w}$
$\mathrm{Top}_k(R) = \mathrm{Top}_k(R(\mathbf{x}, \mathbf{w}))$	Top k percent most relevant pixel indexes in the absolute value of a heatmap R
$k\text{-LRP}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{w})$	k-LRP robustness of the relevance heatmap $R(\mathbf{x}, \mathbf{w})$ w.r.t an adversarial attack $\tilde{\mathbf{x}}$
$k\text{-LRP}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{w}, l)$	k-LRP robustness of $\mathbf{x}$ at a learnable layer l
$\mathbb{E}_{p(\mathbf{w}\mid D)}[k\text{-LRP}(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{w}, l)]$	Bayesian k-LRP robustness of $\mathbf{x}$ at a learnable layer l

Adversarial mutations

$\mathbf{x} = (x_1, \dots, x_D) \in \mathcal{A}^D$	Protein sequence of length D on the 25-character alphabet $\mathcal{A}$ of amino acids
$\mathbf{z}$	Continuous embedding of a sequence $\mathbf{x}$ in the first embedding space
$h \in \{1, \dots, H\}$	Attention head index in a Transformer architecture
$l \in \{1, \dots, L\}$	Layer index in a Transformer architecture
$A_{h,l}(\mathbf{x})$	Attention scores of $\mathbf{x}$ at layer l and head h
$R_{h,l}(\mathbf{x})$	Row attention scores of $\mathbf{x}$ at layer l and head h
$C_{h,l}(\mathbf{x})$	Column attention scores of $\mathbf{x}$ at layer l and head h
$\hat{\mathbf{x}}_i$	Sequence masked at position $i \in \{1, \dots, D\}$
$p(\mathbf{x}\|\hat{\mathbf{x}}_i)$	Pseudo-likelihood of $\mathbf{x}$ w.r.t. masked prediction at position $i \in \{1, \dots, D\}$
$\mathrm{cmap}(\mathbf{x})$	Contact map of $\mathbf{x}$
$\Delta\Delta G$	Change in free energy