Comenius University Bratislava

Faculty of Mathematics, Physics and Informatics

Adversarial Attacks and Their Effect on Neural Networks in Classification Tasks

Dissertation thesis

2024 Mgr. Iveta Bečková

Comenius University Bratislava

Faculty of Mathematics, Physics and Informatics

Adversarial Attacks and Their Effect on Neural Networks in Classification Tasks

Dissertation thesis

Study program:	Informatics
Field of study:	2508 Informatics
Department:	Department of Applied Informatics
Supervisor:	prof. Ing. Igor Farkaš, Dr.

Bratislava, 2024 Mgr. Iveta Bečková

47806310

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Mgr. Iveta Bečková
Study programme:	Computer Science (Single degree study, Ph.D. III. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Dissertation thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Adversarial attacks and their effect on neural networks in classification tasks

Annotation: Even though adversarial examples were first discovered in 2013, and have been an object of intense research ever since, a definite solution has not been found yet. As elimination of adversarial examples seems to be out of reach, the focus of research is shifting towards analysing the effect they have on neural networks.

Aim: 1. Compose an overview of existing literature on adversarial examples.
2. Analyse inherent robustness of various neural network classifier architectures.
3. Propose and test a method for analysing the effect of adversarial examples on a trained neural network.

Literature: Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In IEEE Symposium on Security and Privacy (SP), 39–57
Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations.
Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In International Conference on Learning Representations.

Tutor:	prof. Ing. Igor Farkaš, Dr.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	27.01.2020
Approved:	27.01.2020

prof. RNDr. Rastislav Kráľovič, PhD.
Guarantor of Study Programme

....

Student Tutor

47806310

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Mgr. Iveta Bečková
Študijný program:	informatika (Jednoodborové štúdium, doktorandské III. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	dizertačná
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Adversarial attacks and their effect on neural networks in classification tasks
Adverzariálne útoky a ich vplyv na neurónové siete v klasifikačných úlohách

Anotácia: Napriek tomu, že adverzariálne príklady boli prvýkrát objavené v roku 2013 a odvtedy sú predmetom intenzívneho výskumu, definitívne riešenie sa zatiaľ nenašlo. Keďže sa zdá, že elimináciu adverzariálnych príkladov sa nepodarilo zatiaľ vyriešiť, ťažisko výskumu sa presúva smerom k analýze ich vplyvu na neurónové siete.

Cieľ: 1. Urobte prehľad existujúcej literatúry o adverzariálnych príkladoch.
2. Analyzujte inherentnú robustnosť rôznych architektúr neurónových sietí v klasifikačných úlohách.
3. Navrhnite a otestujte metódu analýzy vplyvu adverzariálnych príkladov na natrénovanú neurónovú sieť.

Literatúra: Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In IEEE Symposium on Security and Privacy (SP), 39–57
Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations.
Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In International Conference on Learning Representations.

Školiteľ:	prof. Ing. Igor Farkaš, Dr.
Katedra:	FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry:	doc. RNDr. Tatiana Jajcayová, PhD.
Dátum zadania:	27.01.2020

Dátum schválenia: 27.01.2020 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

....
študent školiteľ

I hereby declare that I wrote this work by myself, only with the help of the referenced literature.

Bratislava, 2024 . . . .

Mgr. Iveta Bečková

I would like to express my gratitude to my thesis supervisor, prof. Ing. Igor Farkaš, Dr., for his patience, guidance, and flexibility. Next, I want to thank Mgr. Štefan Pócoš for supporting me and always being there for me, and to all my awesome colleagues who made my doctoral study a very pleasant experience, especially during the last few months when I needed it the most.

Abstract

In the past few years, it has been shown that most Deep Neural Networks (DNNs) exhibit inherent vulnerability to maliciously crafted inputs, called Adversarial Examples (AEs). These are usually created using inputs from the dataset by adding a small but very specific noise, which causes large errors on the DNN output.

Despite intense research in this area, the problem still pertains and has not yet been satisfactorily solved. Numerous defence mechanisms against AEs, as well as methods for detecting such modified inputs, have been suggested, yet, none of them provides complete robustness against adversarial attacks. Due to that, the focus of research has slowly shifted towards analysing the AEs and searching for the exact reason why and how they cause DNNs to fail.

In our work, we perform numerous experiments within the image classification domain with the aim of better understanding the properties of adversarial examples and various ways in which they cause faulty outputs. More specifically, our main contributions concern two parts: 1) we analyse the inherent robustness of networks with sigmoidal activations, and how it relates to saturating of the individual neurons. Moreover, we examine the robustness of the newly proposed Recurrent Vision Transformer (RecViT) architecture and explore the possibilities of different training strategies to make RecViT more robust; and 2) we assess the manifold disentanglement theorem using multiple methods and follow with the proposal of a novel algorithm for investigating the hidden-layer representations. This algorithm is based on analysing the proximity of certain inputs to the original data manifolds, and in our experiments, we use it to compare the behaviour of multiple distinct types of AEs.

Keywords: adversarial attacks, robustness, deep neural networks, image classification, explainability

Abstrakt

V priebehu ostatných pár rokov sa ukázalo, že väčšina hlbokých neurónových sietí vykazuje citlivosť voči zákerne vygenerovaným vstupom, zvaným adverzariálne. Tieto vstupy sú zvyčajne vytvorené z originálnych dát pridaním malého, no veľmi špecifického šumu, ktorý spôsobuje veľkú chybu na výstupe siete.

Napriek intenzívnemu výskumu v tejto oblasti, problém stále pretrváva a nie je uspokojivo vyriešený. Početné obrany proti adverzariálnym vstupom, ako aj metódy ich detekcie, boli navrhnuté, nanešťastie, žiadna z nich neposkytuje úplnú robustnosť voči adverzariálnym útokom. Z toho dôvodu sa ťažisko výskumu pomaly presúva k analýze adverzariálnych vstupov a skúmaniu dôvodov prečo a ako spôsobujú chybovosť hlbokých neurónových sietí.

V našej práci sa venujeme viacerým experimentom v doméne klasifikácie obrázkov, za cieľom lepšieho pochopenia vlastností adverzariálnych vstupov a mechanizmov, ktorými spôsobujú chybné výstupy. Konkrétnejšie sa náš prínos dá rozdeliť do dvoch hlavných oblastí: 1) analyzujeme inherentnú robustnosť sietí so sigmoidálnou aktiváciou, a ako súvisí so saturáciou jednotlivých neurónov. Navyše vyšetrujeme robustnosť novo navrhnutej architektúry RecViT (Recurrent Vision Transformer) a skúmame rôzne trénovacie stratégie na zvýšenie jej robustnosti; a 2) vyhodnocujeme teóriu rozbaľovania manifoldov (manifold disentanglement theorem) viacerými metódami, následne nadväzujeme návrhom nového algoritmu na analýzu skrytých reprezentácií siete. Tento algoritmus je založený na analýze blízkosti konkrétnych vstupov k manifoldom originálnych dát, a v našich experimentoch ho používame na porovnanie správania viacerých rôznych typov adverzariálnych vstupov.

Kľúčové slová: adverzariálne útoky, robustnosť, hlboké neurónové siete, klasifikácia obrázkov, vysvetliteľnosť

List of Symbols

$\boldsymbol{x}$	input vector
l	label
$\boldsymbol{Z}(\boldsymbol{x})$	logits — pre-softmax output
$y(\boldsymbol{x})_i = \mathrm{softmax}(\boldsymbol{Z}(\boldsymbol{x}))_i$	post-softmax probability of class i
$f(\boldsymbol{x}) = \underset{i}{\mathrm{argmax}}\ y(\boldsymbol{x})_i$	output class
$\boldsymbol{\theta}$	vector of model parameters
$J(\boldsymbol{x}, l; \boldsymbol{\theta})$	loss function
ϵ	maximal allowed magnitude of a perturbation
$\boldsymbol{\delta}$	perturbation
$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\delta}$	perturbed input
t	target class (in targeted attack)

List of Abbreviations

NN	Neural Network
AE	Adversarial Example
DNN	Deep Neural Network
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
MNIST	Modified National Institute of Standards and Technology
FMNIST	Fashion MNIST
SVHN	Street View House Numbers
CIFAR	Canadian Institute For Advanced Research
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
CNN	Convolutional Neural Network
VGG	Visual Geometry Group
ViT	Vision Transformer
RC	Rubbish Class
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno
FGSM	Fast Gradient Sign Method
kNN	k-Nearest Neighbours
BIM	Basic Iterative Method
PGD	Projected Gradient Descent
CW	Carlini & Wagner
EOT	Expectation Over Transformation
UAP	Universal Adversarial Perturbation
ZOO	Zeroth Order Optimization
BA	Boundary Attack

JSMA	Jacobian-based Saliency Map Attack
EAD	Elastic-net Attack to DNN
AT	Adversarial Training
ALP	Adversarial Logit Pairing
BPDA	Backward Pass Differentiable Approximation
PCA	Principal Component Analysis
RecViT	Recurrent Vision Transformer
t-SNE	t-distributed Stochastic Neighbourhood Embedding
KL divergence	Kullback–Leibler divergence
UMAP	Uniform Manifold Approximation and Projection
SNN loss	Soft Nearest Neighbour loss

List of Figures

List of Tables

Contents

Introduction

The vulnerability of Neural Networks (NNs) to Adversarial Examples (AEs) was first identified in the domain of image classification by Szegedy et al. (2014), who showed that it is often possible to find a tiny, in many cases imperceptible, perturbation of a given input, causing misclassification. Since then, the research has advanced greatly. More efficient ways of generating AEs were proposed, from simple one-step methods (Goodfellow et al., 2015) to strong optimisation-based attacks (Carlini and Wagner, 2017c). Alongside the novel attacks, researchers also devised numerous defence and detection strategies that would protect NNs from all known attack methods. However, with the development of stronger attacks, almost all empiric defences were broken (Carlini and Wagner, 2017a; Athalye et al., 2018a; Tramèr et al., 2020b). One of the last promising methods that remain unbroken is adversarial training, as proposed by Madry et al. (2018).

It is a different story for certified defences, that cannot be broken, as they are provably robust. However, the guaranteed level of robustness is not nearly sufficient enough (Gowal et al., 2019; Cohen et al., 2019). Due to these difficulties with defending NNs, multiple researchers turned to analysing the AEs and proposing numerous hypotheses as to why even such a vulnerability of Deep Neural Networks (DNNs) exists (Tsipras et al., 2019; Ilyas et al., 2019). We follow this line of research in two ways: studying the inherent robustness of different NN architectures and analysing the hidden layer activations of AEs with respect to the clean data manifolds.

We begin this work in Chapter 1 with an introduction to deep learning, a short description of a few benchmark datasets, and the most influential state-of-the-art works of the last few years. Then, we continue the theoretical part in Chapter 2, which is dedicated to the research of adversarial examples — their properties, the methods of generating them, as well as a number of proposed defence and detection mechanisms, including several certified defences. We also discuss a few hypotheses trying to explain the existence of adversarial examples.

In Chapter 3, we begin the practical part by describing a number of our experiments, focused on quantifying the inherent robustness of various classifier architectures. In the first part, we analyse saturating logistic sigmoid-based networks and the relationship between neuron saturation and robustness, published in Bečková et al. (2020); in the second part, we present a pilot robustness analysis of a novel NN architecture — recurrent vision transformer, published in Pócoš et al. (2024b,a). Chapter 4 follows with a summary of the experiments from Pócoš et al. (2021); Bečková et al. (2022); Pócoš et al. (2022), aimed at analysing the hidden layer activations of adversarial examples and how they progress throughout the network with respect to clean data manifolds. We especially focus on analysing the differences between AEs produced by attacks constrained in different L_p norms. Both practical chapters provide some ideas regarding possible future work.

Chapter 1

Introduction to deep neural networks

1.1 Notation and training of neural network classifiers

Research of artificial neural networks began with the first model of perceptron, a computational approximation of a neuron, described by Rosenblatt (1958). A single *discrete perceptron* takes a vector of n inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$ and produces output y according to the formula:

$$y = \Theta\Big(\sum_{i=1}^{n} w_i x_i + b\Big), \tag{1.1}$$

where Θ is the Heaviside step function defined as: $\Theta(x) = 1$ if $x \geq 0$, and $\Theta(x) = 0$ otherwise. Trainable parameters w_i are called *weights*, and b is *bias*. In a binary classification task, input vectors with an output equal to 1 are considered as being from one class, and inputs with an output equal to 0 are from the other class. The hyperplane defined by $\boldsymbol{w}$ and b splits the input space into regions (half spaces) corresponding to the two classes. Therefore, it can be interpreted as the *decision boundary.* This idea can be generalised into a *Multilayer Perceptron (MLP)*: let's have a number of perceptrons, each with different weights but taking the same input, then we can consider outputs of these perceptrons as inputs to another "layer" of perceptrons. Moreover, the step function can be replaced with any *activation function* g, resulting in a more general formula:

$$y = \sum_{i=1}^{m}\Big(w_i^{\text{out}} g\Big(\sum_{j=1}^{n}(w_{ij}^{\text{in}} x_j + b_j)\Big)\Big), \tag{1.2}$$

where w_{ij}^{in} are weights connecting inputs $\boldsymbol{x}$ with the first layer of neurons (referred to as hidden, since they are neither inputs nor outputs), b_j are biases, and w_i^{out} are weights connecting the hidden layer with the output neuron. However, multilayer perceptrons were not very popular at first, as it was not clear how to train them. As a computational breakthrough, Rumelhart et al. (1986) proposed the so-called *back-propagation* algorithm, based on the idea of using the chain rule to analytically compute the partial derivatives of complicated nested functions. Another remarkable contribution came from Hornik et al. (1989), who proved that MLPs with a single hidden layer (described by Eq. 1.2) were universal approximators. Given a sufficient number of neurons m and a continuous, monotonically increasing function g, they can approximate any continuous function with an arbitrary precision on a finite set of input data points. Common choices of activation functions are:

- Logistic sigmoid: $\sigma(x) = \dfrac{1}{1+e^{-x}}$
- Hyperbolic tangent: $\tanh(x) = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- Rectified Linear Unit: $\mathrm{ReLU}(x) = \max(0, x)$

A very specific activation function is softmax:

$$\mathrm{softmax}(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}, \tag{1.3}$$

which is commonly used on the output layer of a classifier, where the number of neurons equals the number of classes. Pre-softmax activations $\boldsymbol{Z}(\boldsymbol{x})$ are usually referred to as *logits*. Softmax has a nice property, that it always produces a vector of the same length as input, which sums up to 1, and all its elements are from the interval [0,1]. Softmax of logits can then be interpreted as probabilities of individual classes:

$$\boldsymbol{y}(\boldsymbol{x}) = \mathrm{softmax}(\boldsymbol{Z}(\boldsymbol{x})), \tag{1.4}$$

while the final output class is just the argmax, i.e., the class with the highest output probability:

$$f(\boldsymbol{x}) = \operatorname*{argmax}_i \; y(\boldsymbol{x})_i. \tag{1.5}$$

To train a NN classifier that classifies inputs into k classes, a surrogate, differentiable loss function is used. For that, each label l is transformed into a one-hot vector $\boldsymbol{e}_l$ of length k, which is the standard basis vector with 1 at the l-th position and 0 elsewhere. Then, we can use the mean squared error between one-hot encoded label and output probabilities, or, in classification more common, the cross-entropy loss to measure the network performance. This ensures that standard gradient optimisation methods, such as *Stochastic Gradient Descent (SGD)*, can be used to optimise the NN parameters. Using the back-propagation algorithm, each weight is trained iteratively according to the update rule:

$$w(t+1) = w(t) - \alpha \nabla_w J(\boldsymbol{x}, l; \boldsymbol{\theta}), \tag{1.6}$$

where $J(\boldsymbol{x}, l; \boldsymbol{\theta})$ is the loss function for a given input $\boldsymbol{x}$, its respective class l, and the current network parameters $\boldsymbol{\theta}$ (i.e., the weights and biases). ∇_w denotes the partial derivative with respect to w, and α is the *learning rate*. There has been some effort to modify the optimisation to achieve faster convergence. Various ideas included second-order methods, adaptive learning rate, or using a momentum (adding the "copy" of the previous update to the current update). Currently, one of the most popular methods is Adam (Kingma and Ba, 2014). It uses leaky averaging to keep estimates of the momentum and the second moments of derivatives for each weight:

$$\begin{aligned} v_w(t) &= \beta_1 v_w(t-1) + (1-\beta_1)\nabla_w J(t), \qquad & v'_w(t) &= \frac{v_w(t)}{(1-\beta_1^t)}, \\ s_w(t) &= \beta_2 s_w(t-1) + (1-\beta_2)\nabla_w^2 J(t), \qquad & s'_w(t) &= \frac{s_w(t)}{(1-\beta_2^t)}. \end{aligned} \tag{1.7}$$

Then the weight update is computed according to:

$$w(t+1) = w(t) - \frac{\alpha}{\sqrt{s'_w(t)} + \gamma} v'_w(t), \tag{1.8}$$

to simplify notation, we use $J(t)$ as the loss function in iteration t. The stabilising term γ is some small constant, β_1 and β_2 are adjustable hyperparameters, and their commonly used values are $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

1.2 Popular image classification datasets

In this section, we briefly introduce a couple of the most popular image classification datasets, some of which were also used in our experiments. An illustration of inputs from four of the used datasets is in Fig. 1.1.

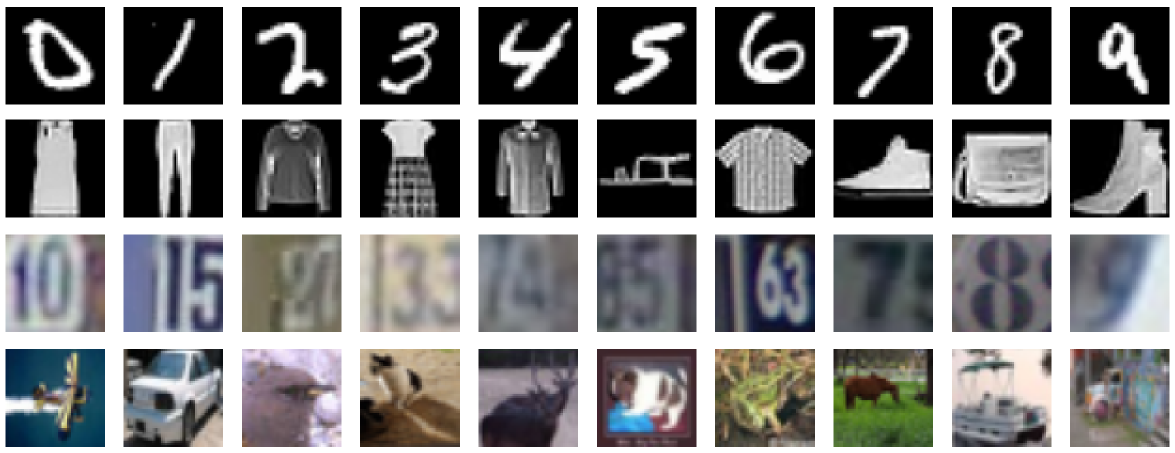


Figure 1.1: Sample input images for each class of the 10-class datasets. The rows contain from top to bottom: MNIST, FMNIST, SVHN, and CIFAR-10 images.

1.2.1 10-class datasets

MNIST (Modified National Institute of Standards and Technology) dataset by LeCun et al. (1998b) is one of the first and most used image classification datasets. It consists of 28×28 px grayscale (single channel) images split into classes 0–9 representing individual digits. Each image contains a single centred handwritten digit. The dataset comes with 60 000 training and 10 000 testing samples.

Being one of the most popular image classification datasets of all times, MNIST inspired a number of other similar datasets. One of them, which we also chose to use in our work, is the FMNIST (Fashion MNIST) dataset (Xiao et al., 2017). Same as MNIST, the dataset contains 28×28 px grayscale images forming ten classes. However, since FMNIST was proposed as a slightly more challenging replacement for MNIST, each image has one centred photo of a type of clothing or accessory. The dataset provides 50 000 training and 10 000 testing images.

Another dataset very similar to MNIST is the SVHN (Street View House Numbers) dataset (Netzer et al., 2011). The task is also to classify digits into classes 0–9, however, the data contains RGB images with a resolution of 32×32, which is slightly higher than the resolution of MNIST. Moreover, as these are real-world images of house numbers taken from Google street view, some may contain more than one digit, in which case the goal is to classify the one in the centre. The dataset contains over 73 000 training and more than 26 000 testing samples.

CIFAR-10 (Canadian Institute For Advanced Research) by Krizhevsky and Hinton (2009) is another popular choice for image classification. The dataset is made of RGB images with the same resolution as for SVHN, i.e., 32×32 px, which are also split into ten classes, representing various animals and vehicles. The dataset has 50 000 training and 10 000 testing images.

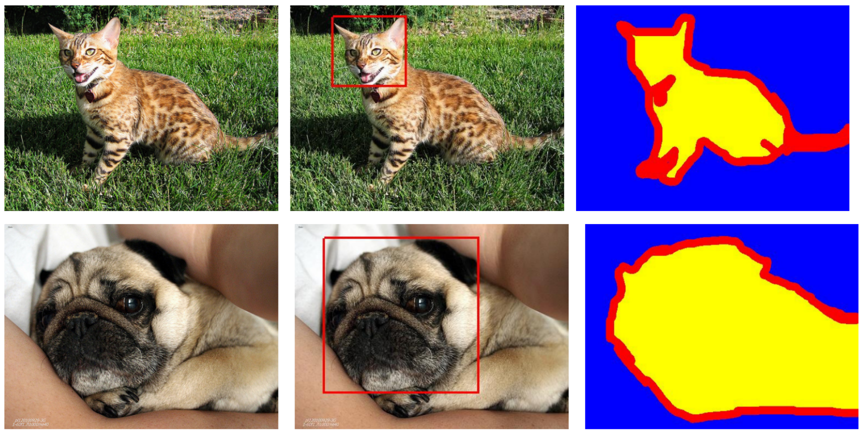


Figure 1.2: Examples of data contained in the Oxford-IIIT Pet dataset. Alongside class labels, head bounding box (middle) and trimap segmentation (right) are provided for each input image (left). Image adapted from Parkhi et al. (2012).

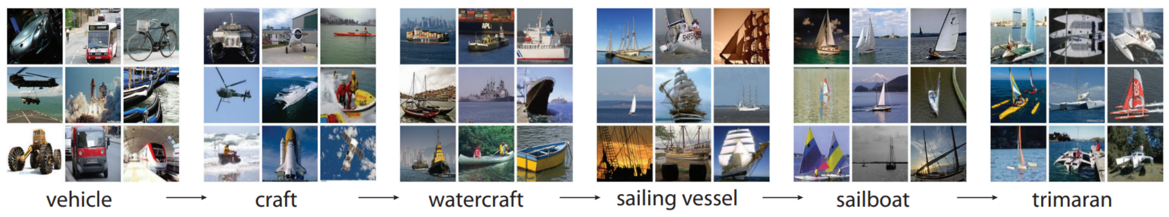


Figure 1.3: A sample branch from ImageNet hierarchy with examples of contained images. Image adapted from Deng et al. (2009).

1.2.2 Larger-scale datasets

A more challenging classification task is offered by the Oxford-IIIT Pet dataset (Parkhi et al., 2012). It contains RGB images of various sizes, partitioned into 37 classes of cat and dog breeds. The dataset has ≈ 200 samples per class. One of the advantages is that it also provides head bounding boxes and trimap segmentations, as seen in Fig. 1.2.

One of the most popular large-scale image classification datasets is the ImageNet (Deng et al., 2009). It is a hierarchical database with over 14 million sample images split into more than 21 thousand classes. An example of a class branch with respective images is shown in Fig. 1.3. ImageNet became popular mostly due to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015), which features a 1 000-class subset of the ImageNet dataset, usually referred to as ImageNet-1k, while the full dataset is denoted ImageNet-21k.

1.3 Development of the state-of-the-art

Research in computer vision has evolved greatly, from using hand-crafted features as inputs for simple classifiers to end-to-end trained DNNs that often surpass even human performance. In this section, we summarise a few of the most influential concepts leading to breakthroughs in image classification.

1.3.1 Convolutional neural networks

The first big breakthrough of DNNs on visual data came with the rise of *Convolutional Neural Networks (CNNs)*. While there were some earlier efforts, the idea was popularised mainly due to the development of LeNet-5 by LeCun et al. (1998a) (architecture depicted in Fig. 1.4), a CNN trained to recognise handwritten characters that outperformed other types of classifiers which were, at that time, considered state-of-the-art. CNNs employ convolutional layers to extract features from an input image, resulting in end-to-end training. This is in contrast with the more "traditional" approach in com-

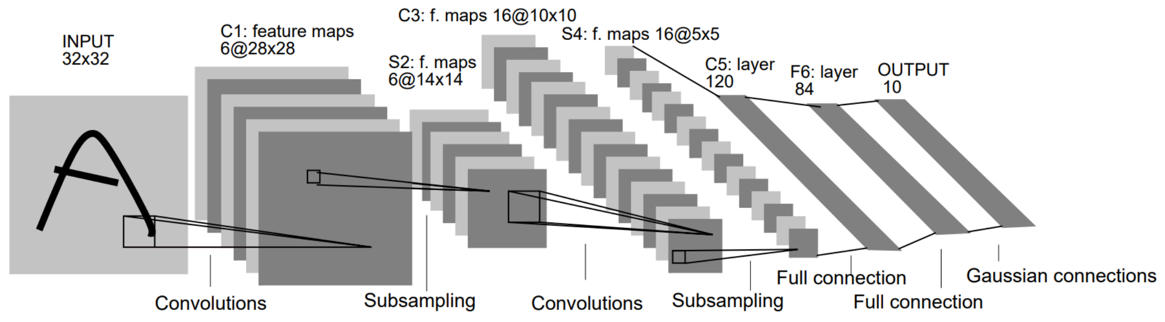


Figure 1.4: The architecture of the LeNet-5 network, one of the first CNNs, successfully applied to the task of recognising handwritten characters. Image adapted from LeCun et al. (1998a).

puter vision, using simple classifiers on outputs from hand-crafted feature extractors such as edge detectors. A single output of the convolutional layer is computed as the dot product between a *convolution kernel* (filter) and a small contiguous part of the input of the same size. The entire layer is comprised of outputs from such dot products with convolution kernels centred in different parts of the input (in a sliding window manner), arranged to maintain the spatial information (i.e., outputs from neighbouring input patches are also adjacent). Besides preserving information about the input structure, convolutional layers have more advantageous properties:

- **Translation invariance**: thanks to the dot product with the convolution kernel being computed over different parts of the input, the same feature can be detected regardless of its exact position.

- **Sparse connectivity**: as each output is connected to only a small part of the input (its *receptive field*), convolutional layers have fewer trainable parameters than fully connected layers (where each input is connected with each output).

- **Shared weights**: the fact that the same convolution kernel is used to compute dot products with different parts of the image further reduces the number of trainable parameters of convolutional layers. Therefore, CNNs can be trained more efficiently, using fewer samples.

In practice, multiple convolution kernels are used simultaneously to detect the presence of different features. These result in multiple output *channels*, forming an additional third dimension in the neuron layer.

To further reduce the dimensionality of hidden layers, sub-sampling (also called *pooling*) is usually used. Pooling works in the same way as convolution, except that instead of the dot product of input parts and a convolution kernel, the output is computed as either the average or the maximum of the features. Also, the receptive fields of individual output neurons are usually non-overlapping, while in convolutional layers, they mostly are. While LeNet-5 used weighted averages, nowadays, max pooling is more popular. It can be interpreted as a logical OR, i.e., detecting whether a feature is present somewhere in the receptive field. Pooling layers are usually followed by some non-linear activation function.

A typical CNN architecture consists of a number of repetitions of convolutional layers, pooling layers, and non-linearities, which are then followed by a couple of fully connected layers. In that case, the convolutional part is usually considered a trained feature extractor, which is then followed by a simple MLP performing classification on the extracted features.

1.3.2 CNNs on ImageNet

Since LeNet-5, employment of CNNs slowly grew, with a great leap in popularity in 2012, when AlexNet (Krizhevsky et al., 2012) won the ILSVRC with a huge margin (15.3% error rate, while the second best competitor achieved 26.2%). AlexNet is a CNN trained on ImageNet, though much larger than the LeNet-5. The authors state that the successful training of such a big model was only possible thanks to a couple of smart design choices. One of them is the use of ReLU. Commonly used activation functions logistic sigmoid and hyperbolic tangent have derivatives with zero limits in $\pm\infty$. Therefore, if a neuron activation gets too big (in absolute value), the derivative in the backward pass gets close to zero. Longer training of networks with these activation functions leads to saturating neurons and slow weight adjustment. This problem is commonly known as *vanishing gradients*. ReLU activation function prevents such problems by having a constant derivative (for all inputs greater than 0), so even very long training (which is required for more complicated tasks) does not lead to saturation. Another crucial component is *dropout* (Srivastava et al., 2014). It is a regularisation technique based on randomly zeroing neurons in the forward and backward pass. During training, each neuron is zeroed with probability p. In testing, the neuron activations are kept but scaled by p. This promotes networks learning more distributed knowledge instead of strongly relying on some specific neurons.

Another improvement and a possibility of even deeper networks were brought by Simonyan and Zisserman (2015). They proposed a number of models collectively referred to as VGG (Visual Geometry Group). Thanks to using smaller convolution kernels, the networks have manageable numbers of trainable parameters despite reaching depths up to 16 convolutional layers. Moreover, the experiments clearly showed the potential of improving the network performance by increasing its depth. However, training even deeper networks remained a great challenge up till ResNet (He et al., 2016) was proposed. It is based on the observation that if a network is too deep, its performance may

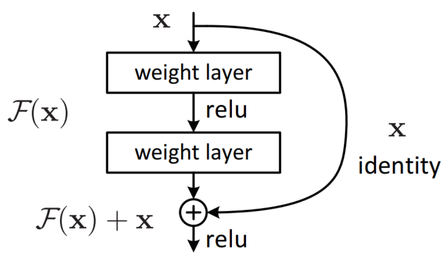

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 1.5: An illustration of a residual connection, which connects a layer of neurons with another, not directly the next one, therefore skipping a couple of layers. Image taken from He et al. (2016).

be lower than that of a more shallow network due to difficult optimisation. Theoretically, it should be sufficient to set the additional layers to perform identity to maintain accuracy. The observed drop in performance of deep networks hints that learning the identity mapping might be difficult for NNs. The authors decided to provide help in the form of the so-called "skip" or *residual connection*, which adds a copy of activations from a few layers earlier, as illustrated in Fig. 1.5. In that case, to perform identity, the learned weights just need to be zero. As shown in the results, this is a much easier task since, using this approach, it is possible to train an exceptionally well-performing (the first place in ILSVRC 2015) network with as many as 152 layers.

1.3.3 Vision transformer

The latest revolution in deep learning came with the transformer networks (Vaswani et al., 2017), originally proposed as a replacement for recurrent neural networks in natural language processing tasks. Only a couple of years later, they were successfully adapted to the computer vision domain in the form of *Vision Transformer (ViT)* (Dosovitskiy et al., 2021). The transformer architecture, as well as the ViT architecture, is based mainly on the self-attention mechanism, which takes a number of input vectors, linearly projects them to queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$, and then computes the output according to:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{1.9}$$

where d_k is the dimensionality of the keys and queries, and it is a tunable hyperparameter. In transformers, inputs are usually word embeddings. For ViT, it works as follows (also illustrated in Fig. 1.6): first, the input image is sliced into smaller patches that are flattened, projected, and added to their respective position embeddings (encoding from which part of the image the patches come). These are the inputs in the self-attention. However, one extra vector is used — the *class token*. Its initial value

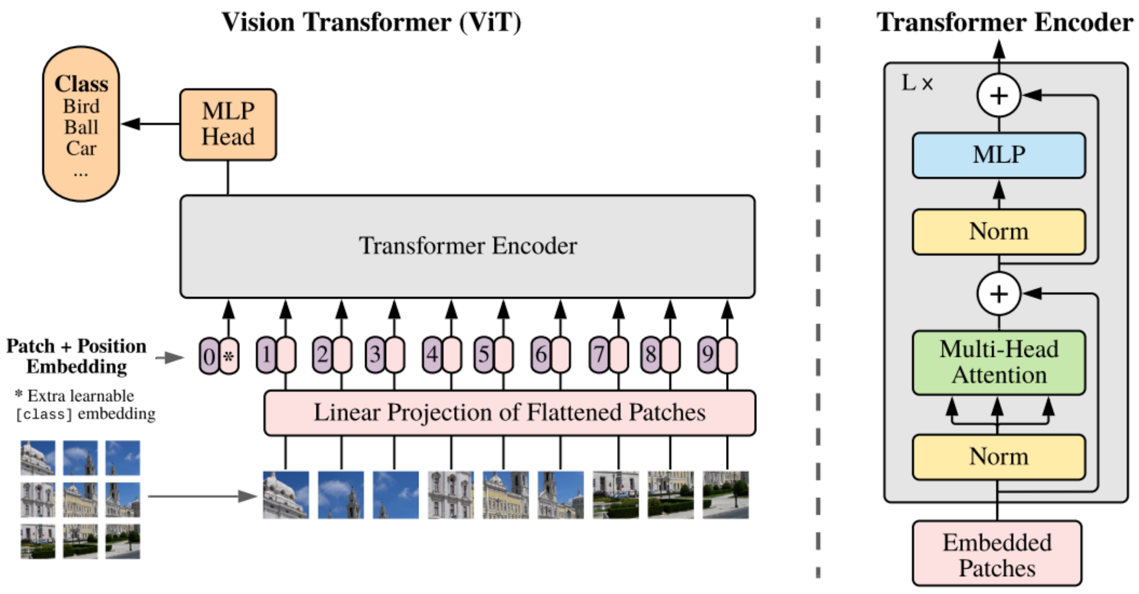


Figure 1.6: Visualisation of the vision transformer architecture (left) along with its most crucial component, the transformer encoder (right). Image reproduced from Dosovitskiy et al. (2021).

is learned, while its output from the transformer encoder is used in a simple MLP to produce the final classification.

Similar to transformers, ViTs also heavily depend on extensive pre-training (training on a larger, more universal dataset), and subsequent fine-tuning to perform a given, specific task. Without pre-training, the performance of ViT on ImageNet-1k is comparable to that of ResNet. But with pre-training on a large dataset (such as ImageNet-21k), ViT becomes superior and sets the new state-of-the-art.

Chapter 2

Adversarial examples

In the classification task, *adversarial examples* are inputs that are very similar to correctly classified samples from the input distribution, yet are misclassified by a trained model. Such inputs exist even for deep neural networks which have high testing accuracy (and are therefore expected to generalise well). However, AEs are hard to find just by randomly sampling the vicinity of the clean data. For that reason, their existence is quite counter-intuitive and has only first been revealed by Szegedy et al. (2014).

It should be noted that the idea of modifying inputs with the intention of causing misclassification was already suggested by other authors, for example, Biggio et al. (2013). However, they focused on simple machine learning models such as support vector machines or linear classifiers and their resulting images cannot really be considered AEs as they diverged too far from the original inputs.

2.1 Fooling images

Around the same time, another class of problematic inputs — *fooling images*, also referred to as *Rubbish Class (RC)* examples, was described by Nguyen et al. (2015). RC examples are meaningless inputs that are not recognisable by humans, but are classified with very high confidence ($\geq 99\%$) as belonging to one of the classes by a deep learning model. By definition, these are not AEs, because they are not similar to the original data. However, they are often studied alongside the AEs or in the same context.

The original paper described multiple ways of generating RC examples in a *targeted* way, i.e., picking a certain *target class* and searching for an input that is classified as such. Methods based on genetic algorithms found regular images with symmetries or repeating patterns. Some of the images were initially not recognisable by humans, but after seeing the output class of the network, they were able to find some "correct" features in them. A few samples are shown in Fig. 2.1. Notice, for example, the red

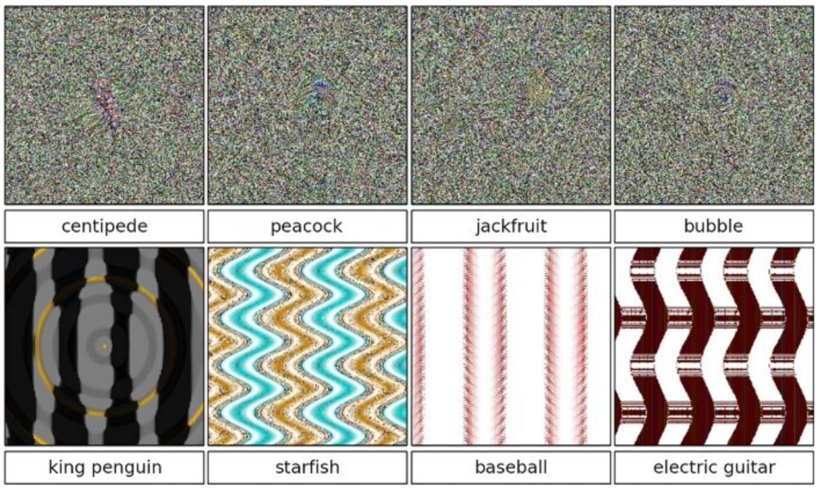


Figure 2.1: Image taken from Nguyen et al. (2015). The top row shows fooling images (along with the classes outputted by the used model) generated by gradient ascent. The bottom row depicts images generated by genetic algorithm.

lines resembling stitches in a baseball.

Another set of inputs was produced by gradient ascent in the input space, maximising the output of the target class, starting from a randomly perturbed mean of the ImageNet-1k dataset inputs. The resulting images were completely unrecognisable. The gradient approach was more extensively studied by Goodfellow et al. (2015) with the slight modification that the starting point for the gradient ascent was random noise. Therefore, the resulting images also resembled just noise. The authors further noted that using genetic algorithms to generate these images is an "overkill", as the gradient method produces images classified as a desired class with high success, the exact percentage dependent on the target class.

There was some effort to tackle the problem by adding an extra output neuron representing the rubbish class and fine-tuning the network by training on these examples. However, both Nguyen et al. (2015) and Goodfellow et al. (2015) showed this procedure to be highly ineffective and unable to "fix" the model.

2.2 Properties of adversarial examples

Since the first mention of AEs in 2013, there has been much effort in the deep learning community to explore and understand this phenomenon. Despite the immense amount of the work done in this field, the current state-of-the-art does not seem to be much closer to solving the problem of AEs. Quite the opposite, they were found to have certain unpleasant properties, revealing fundamental flaws of deep learning. This section summarises the most important findings yet.

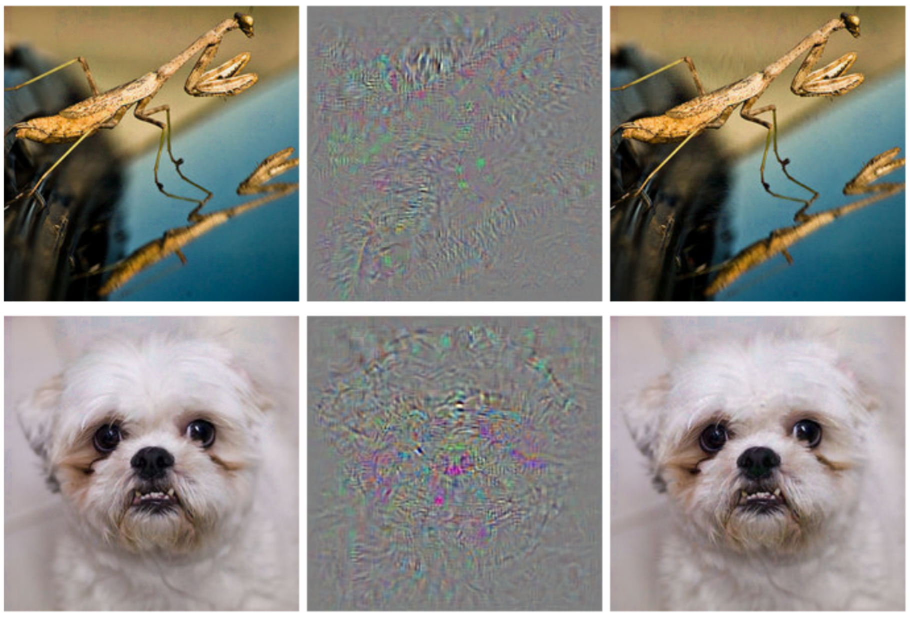


Figure 2.2: A sample of AEs by Szegedy et al. (2014). The left column contains original images, the middle shows their respective adversarial perturbations (scaled for better visibility). The resulting AEs are in the right column. Both AEs are classified as ostrich by AlexNet.

2.2.1 Computational complexity of adversarial attacks

Szegedy et al. (2014) formalised the problem of AEs in the domain of image classification as a constrained optimisation. Following the notation from Sec. 1.1, let's denote $f\colon \mathbb{R}^n \to \{1, \ldots, k\}$ a classifier classifying n-dimensional vectors into k classes, and $\boldsymbol{x}$ a given input image, then the goal of an adversary (or an *adversarial attack*) is to find an *adversarial perturbation* $\boldsymbol{\delta}$, such that $\|\boldsymbol{\delta}\|_2$ is minimal and $f(\boldsymbol{x}+\boldsymbol{\delta}) = t$, where t is the target output class (different from the correct label l from the dataset) and $\boldsymbol{x}+\boldsymbol{\delta} \in [0,1]^n$ to ensure that $\boldsymbol{x}+\boldsymbol{\delta}$ is a valid image. This means the goal is to find the closest image to $\boldsymbol{x}$, which is classified by f as belonging to class t. In practice, the authors used the box-constrained L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) optimisation (Liu and Nocedal, 1989). Their results (some of them depicted in Fig. 2.2) included images almost indistinguishable from the original correctly classified ones, that were misclassified by the network, often with surprisingly high confidence. However, the optimisation procedure was computationally costly, and for some time, it was believed that the generation of AEs may not be an easy task. The authors even hypothesised that the AEs might be located only in extremely rare, hard to find "pockets" in the input space.

Shortly after, this assumption was proven wrong by Goodfellow et al. (2015), who proposed a simple, computationally cheap algorithm for generating AEs — the *Fast Gradient Sign Method (FGSM)*. This attack computes the gradient of the loss function with respect to the input to find the direction in which to shift the given input so as to maximise the locally linearised cost (and, therefore, lower the output probability of the correct class). Note that in contrast to Szegedy et al. (2014), FGSM was originally proposed as an *untargeted* attack. Instead of modifying the input to be classified as belonging to the target class, the target is not specified, and the only goal is misclassification. Their approach differs from Szegedy et al. (2014) also in the norm used to measure the magnitude of the perturbation. FGSM uses the max-norm L_∞, while Szegedy et al. (2014) opted for the Euclidean L_2 norm to constrain the attacker. Another thing to note is that FGSM does not actually try to minimise the adversarial perturbation, just to keep it within a specified distance from the original input. The resulting optimal perturbation is

$$\boldsymbol{\delta} = \epsilon \, \mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, l; \boldsymbol{\theta})), \tag{2.1}$$

where $\boldsymbol{\theta}$ are the network parameters, l is the correct class, ϵ is the maximal allowed perturbation magnitude, and $J(\boldsymbol{x}, l; \boldsymbol{\theta})$ is the cost function.

The reasoning behind FGSM is the hypothesis that AEs are actually caused by the linearity in the models. The authors showed that considering the L_∞ norm, the vulnerability of a linear network (the possible magnitude of error that the attacker can cause) grows with the dimensionality of the input as follows: given a dot product $\boldsymbol{w}^T\boldsymbol{x}'$ of a modified input vector $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\delta}$ and a weight vector $\boldsymbol{w}$, using a perturbation $\boldsymbol{\delta} = \epsilon$ sign$(\boldsymbol{w})$ results in a change of magnitude ϵmn, where n is the dimensionality of $\boldsymbol{x}$ and m is the average magnitude of elements of $\boldsymbol{w}$. As deep neural networks are a cascade of multiple linear and non-linear transformations, they are probably just as vulnerable, if not more.

2.2.2 Transferability

Another important finding of Szegedy et al. (2014) was that AEs exhibit a non-negligible level of *transferability* across different models. In their experiments, some AEs generated on a certain model were misclassified also by other models (with different hyperparameters and architecture) trained on the same dataset. Moreover, after splitting the training data into two halves and training two different models, each on one of the halves, a small percentage of generated AEs still transferred across the models. This suggests that AEs are neither model-dependent, nor data-dependent.

To make matters worse, Papernot et al. (2017) showed that it is possible to find adversarial examples for a target model even without any knowledge of its architecture or the data it was trained on. The only information needed is the target model output class for any given input (therefore, the target model is referred to as an *oracle*). To craft the AEs, the authors used a substitute model and a substitute synthetic dataset. The entire attack works in the following steps: the substitute dataset is created iteratively, starting with a small initial set of inputs representing the task performed by the oracle (for example, images of digits in the case of digit classification). Then, the substitute model is trained on these inputs to output the same class as the attacked model, while the dataset is continually augmented during training with new inputs chosen based on the Jacobian of the substitute model in some of the inputs from the previous iterations. After the substitute model is sufficiently trained, a gradient-based attack is employed to craft AEs fooling it. These are then used to attack the oracle. The authors showed that using this approach, it is possible to craft AEs even for non-differentiable classifiers, such as decision trees or k-Nearest Neighbours (kNN). Attacks exploiting the transferability property (using a different model to craft the AEs than the one that is actually being attacked) are referred to as *transfer attacks*. Transfer attacks are the most simple type of *black-box* attacks — attacks that treat the target network as a black-box without assuming any knowledge of it. The opposite is *white-box* attacks, which use some knowledge about the attacked model to craft AEs, for example, its gradients, architecture, etc.

Shortly after the introduction of transfer attacks, Liu et al. (2017) performed an extensive study of transferability, especially in the context of the targeted attacks. Their results showed that only a very small percentage of AEs constructed with targeted attacks are still misclassified as the target class when evaluated on different models. Many of them were still misclassified, though (just not as the target class). As a solution, the authors suggested the idea of using an *ensemble* of models to craft AEs. Their hypothesis was that if an AE can successfully fool multiple models, then it is more likely to also transfer to another model. Such AEs can be crafted by computing adversarial perturbations for a given image over multiple models and aggregating them into a single perturbation. It turns out that AEs created in this fashion have much higher transferability rates than standard AEs (evaluated on a model that was not used to craft them). This is most prominent in targeted attacks, where it significantly increases the fraction of AEs that transfer along with the target class. Therefore, the authors showed that it was possible to craft targeted AEs using transfer attacks.

2.2.3 Optimising AEs with stronger attacks

The first iterative approach was proposed by Moosavi-Dezfooli et al. (2016). Their attack, called *DeepFool*, was the first computationally effective method that tried to find the minimal adversarial perturbation (instead of just keeping it within a fixed ϵ-ball around the original input). DeepFool is based on splitting the input space according to the learned decision boundaries. In the case of a linear model, these decision boundaries are hyperplanes, and they define polyhedral regions that are classified as a certain class. Then, considering a given input image that is classified correctly, the minimal adversarial (untargeted) perturbation can be computed analytically as an orthogonal projection onto the complement of the polyhedron classified as the correct class. By locally linearising the decision boundaries (i.e., computing partial derivatives of the output probabilities $\boldsymbol{y}$ for individual classes in the given input), this method can be used to approximate the optimal perturbation even for general, non-linear classifiers. However, in that case, the computed perturbation may not actually cause misclassification. Therefore, the linearisation and projection are repeated iteratively until the attack succeeds. The authors reported that usually, in practice, not more than three iterations are needed. This results in an attack that is much faster to compute than the previously used L-BFGS while producing much smaller perturbations than the FGSM.

Kurakin et al. (2018) formulated a different iterative attack for generating AEs, the *Basic Iterative Method (BIM)*. It works like FGSM but runs in iterations. In each iteration, the image is modified just by a fraction of the admissible perturbation magnitude ϵ, and if the total magnitude exceeds the threshold, it is clipped to stay within ϵ distance from the original input. Using the same value for ϵ, BIM produces AEs that are perceived as more similar to the original images than FGSM.

Next, Madry et al. (2018) proposed the *Projected Gradient Descent (PGD)* attack, an extension of BIM. It works exactly the same, but instead of starting in the clean input $\boldsymbol{x}$, it starts from a random point within the ϵ-ball around the original image. Such randomisation of the starting point makes the attack more general. By running the attack multiple times and choosing the AE which causes misclassification with the highest output confidence, PGD can find better AEs than BIM.

A quite different, very strong method of generating AEs, referred to as the *Carlini & Wagner (CW)* attack, was proposed by Carlini and Wagner (2017c). The attack works by employing the Adam optimiser to solve the following problem:

$$\begin{gathered}\underset{\boldsymbol{w}}{\operatorname{argmin}}\Big(\|\frac{1}{2}(\tanh(\boldsymbol{w})+1)-\boldsymbol{x}\|_2^2+c\cdot F(\frac{1}{2}(\tanh(\boldsymbol{w})+1),t)\Big), \text{ where}\\ F(\boldsymbol{x}',t)=\max(\max\{Z(\boldsymbol{x}')_i\colon i\neq t\}-Z(\boldsymbol{x}')_t,-\kappa).\end{gathered} \tag{2.2}$$

Here, t is the target class, and $\boldsymbol{Z}(\boldsymbol{x})$ are the network logits. Binary search is used to find the optimal value of the constant c. The resulting AE is then given by $\boldsymbol{x}'=\frac{1}{2}(\tanh(\boldsymbol{w})+1)$. This smart reparametrisation trick is used to ensure that the AE satisfies the box-constraint $\boldsymbol{x}'\in[0,1]^n$, while not constraining the optimisation itself. Parameter $\kappa\geq 0$ can be varied, with greater values resulting in AEs classified with higher confidence, though also greater magnitudes of adversarial perturbations. These "high confidence" AEs were shown to have better transferability, and thus, higher values of κ should be considered when CW is used for a transfer attack. The authors also proposed simple modifications to this attack so that it can be used to find AEs constrained in the L_0 norm or the L_∞ norm. All versions of CW attack were shown to produce AEs better than or equal to (in terms of the perturbation magnitude) the previous state-of-the-art attacks.

2.2.4 Physical adversarial examples

Deep learning is often used in computer vision, but in most cases, it seems unlikely that an attacker could directly modify the captured image. However, they might be able to modify the object that is being captured. With this idea in mind, Kurakin et al. (2018) studied the possibility of crafting adversarial images in the physical world. In their experiments, they created AEs in a standard manner and then printed them on paper. These were then captured by a camera. A non-trivial number of AEs were still misclassified. The authors also assessed the effect of standard data augmentation transformations (such as changes of contrast or blurring) on the AEs. Even though these transformations significantly reduced the misclassification rate, none of them was able to reduce it to zero. Comparing FGSM and BIM attacks, despite the fact that BIM is considered a stronger attack (due to being iterative), the misclassification rate of the printed BIM AEs was lower than that of the printed FGSM AEs.

Athalye et al. (2018b) took AEs to an even higher level. They proposed the *Expectation Over Transformation (EOT)* attack, which optimises the adversarial example $\boldsymbol{x}'$ for a given input $\boldsymbol{x}$ over a distribution T of admissible transformations $\tau \in T$, resulting in a Lagrangian-form maximisation:

$$\underset{\boldsymbol{x}'}{\operatorname{argmax}}\Big(\mathbb{E}_{\tau\sim T}[\log y(\tau(\boldsymbol{x}'))_t] - \lambda\,\mathbb{E}_{\tau\sim T}[d(\tau(\boldsymbol{x}'),\tau(\boldsymbol{x}))]\Big), \tag{2.3}$$

where $\boldsymbol{y}$ is the vector of output probabilities, t is the target class, and d is a chosen metric. The optimisation is solved by the projected gradient ascent, randomly sampling a transformation τ in each iteration and clipping the result into a valid image range $[0,1]^n$. Then, the authors used EOT with a specific set of transformations consisting of rendering transformations (from texture to 3D object), different camera angles, various lighting conditions, printing inaccuracies, and more, to find adversarial textures for multiple 3D objects. To prove the validity of this approach, they used a 3D printer to print two objects with adversarial textures, one of them being the turtle depicted in Fig. 2.3.

Also in pursuit of the physical AEs, Eykholt et al. (2018) proposed an attack specifically tailored to the problem of modifying real-world objects by applying posters or stickers to them. This is achieved by using a pixel-wise mask when computing the attack, to only consider those parts of the image that belong to the target object. The optimisation problem is modified by adding a special loss term accounting for possible printing errors, and, similarly to Athalye et al. (2018b), considering multiple transformations of the target object. After the perturbation is computed, it is printed

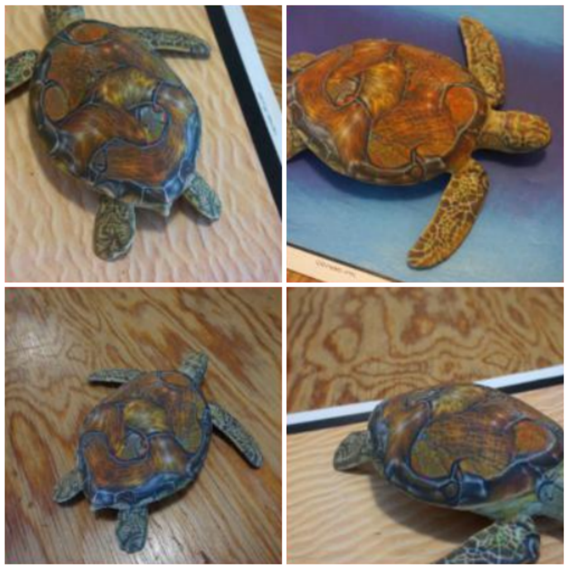


Figure 2.3: 3D-printed turtle with texture computed using EOT by Athalye et al. (2018b). The turtle is classified as a rifle from 82% of evaluated viewpoints.

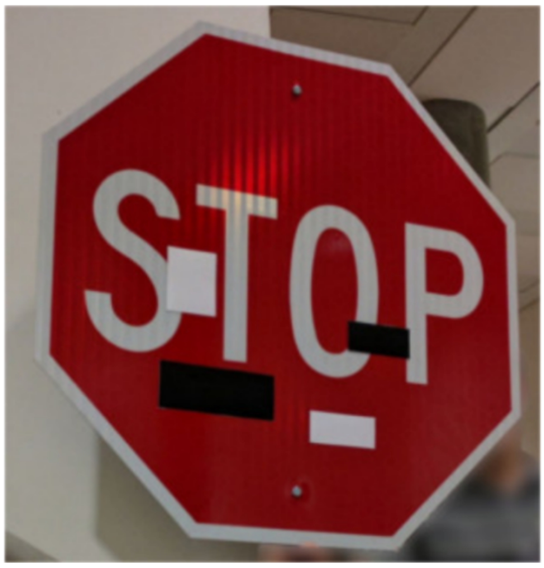


Figure 2.4: Adversarial stop sign by Eykholt et al. (2018). Viewed from different distances and angles, it is consistently misclassified as a speed limit 45 sign.

and applied to the physical object. As a proof of concept, the authors modified a couple of objects in this way, for example, the stop sign, which can be seen in Fig. 2.4. By using a few black and white stickers on the sign, they managed to make it consistently misclassified when viewed from various distances and angles.

2.2.5 Universal adversarial perturbations

As if the problem of AEs was not hard enough, Moosavi-Dezfooli et al. (2017) found that it is possible to create an adversarial perturbation, which is *universal* in the sense of being input-agnostic. There is a high probability that such a perturbation will cause misclassification when added to any of the original inputs. The authors proposed an algorithm to craft these untargeted *Universal Adversarial Perturbations (UAPs)*. Considering $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$ the set of inputs, the initial perturbation $\boldsymbol{v}(1)$ is computed using DeepFool on input $\boldsymbol{x}_1$. In subsequent iterations, $\boldsymbol{v}(i) = \boldsymbol{v}(i-1) + \boldsymbol{\delta}_i$, where $\boldsymbol{\delta}_i$ is the perturbation found by DeepFool on input $\boldsymbol{x}_i + \boldsymbol{v}(i-1)$. It is also possible to constrain the maximal perturbation norm to a chosen ϵ by projecting the $\boldsymbol{v}(i)$ in each iteration onto an L_2-ball with radius ϵ centred around zero. The algorithm stops if the perturbation reaches the desired success rate, i.e., a sufficient fraction of the set X is misclassified after adding this perturbation.

Empirical evaluation showed that the set X of considered inputs need not be very large to achieve high rates of misclassification on the entire dataset. Moreover, as the algorithm is stochastic, by permuting inputs in X, it is possible to find multiple diverse

UAPs. Similar to the standard (input-specific) adversarial examples, images modified by UAPs also transfer across different models.

In their subsequent study, Moosavi-Dezfooli et al. (2018) tried to explain the UAPs geometrically by assuming the existence of a subspace $\mathcal{S}$ in the input space, containing directions along which the curvature of the decision boundary is positive. They empirically showed this assumption to be true. They also proposed an efficient algorithm for generating UAPs using this knowledge, based on approximating S by estimating the average hessian of the decision boundary and taking the first few of its eigenvectors as the base of S. UAPs can then be found by randomly sampling directions in S.

2.2.6 AEs outside the scope of this work

In this work, we are only interested in AEs in image classification, but there is also a significant amount of work on AEs in other domains. For example, Carlini et al. (2016) created imperceptible voice commands (even though they were not AEs in the sense of being similar to clean examples, but more like the audio analogy of fooling images). Even more disturbingly, it is possible to craft AEs for deepfake-image detectors (Carlini and Farid, 2020) and malware detection systems (Grosse et al., 2017b). Especially interesting are AEs in deep reinforcement learning (Huang et al., 2017), where time can be considered as another variable to optimise (i.e., minimising the attacking frequency to keep the attacks hard to detect).

All attacks considered in this work are performed at the test-time (i.e., attacking a classifier that is already fully trained). These are usually referred to as *evasion* attacks. The opposite class of attacks that deals with imperceptibly modifying data during training is called *data poisoning* (Chen et al., 2017b).

2.3 Other noteworthy attacks

In this section, we present other attacks that we think are worth mentioning. They are either black-box, and therefore practical and easy to use, or use some non-standard norms to measure the magnitude of the adversarial perturbation, which results in unique and qualitatively different AEs.

2.3.1 Non-transfer black-box attacks

Chen et al. (2017a) proposed an attack that, instead of training a substitute model, tries to directly approximate the target model gradients. They called it the *Zeroth Order Optimization (ZOO)* method. It assumes knowledge of the target model output probabilities for any given input, which is a slightly stronger assumption than in Papernot et al. (2017), but the attack is usually still considered black-box. The authors took inspiration from the CW attack and, taking a clean input $\boldsymbol{x}$ and the target class t, they used the Adam optimiser to solve a slightly modified optimisation given in Eq. 2.2, with $F(\boldsymbol{x}', t)$ defined as

$$F(\boldsymbol{x}', t) = \max\{\max_{i \neq t} \log y(\boldsymbol{x}')_i - \log y(\boldsymbol{x}')_t, -\kappa\}, \tag{2.4}$$

where $y(\boldsymbol{x}')_i$ is the probability of input $\boldsymbol{x}'$ belonging to class i. Therefore, ZOO replaces the difference of logits used in the CW attack with the difference of logarithms of probabilities. To ensure the box constraint $\boldsymbol{x}' \in [0,1]^n$, ZOO uses the same reparametrisation trick as is in Eq. 2.2. Next, to approximate the gradient of $\boldsymbol{y}$ needed for optimisation, ZOO computes finite differences:

$$\frac{\partial F(\boldsymbol{x}, t)}{\partial x_j} \approx \frac{F(\boldsymbol{x} + \boldsymbol{e}_j h, t) - F(\boldsymbol{x} - \boldsymbol{e}_j h, t)}{2h}, \tag{2.5}$$

where h is a small constant and $\boldsymbol{e}_j$ is the standard basis (one hot) vector with 1 at the j-th place and 0 elsewhere. This means that to approximate a single partial derivative, two forward passes need to be computed. This is an issue for tasks with high input dimensionalities. For that reason, the authors used mini-batch coordinate descent, updating only a mini-batch of coordinates in each iteration of optimisation (therefore reducing the number of needed forward passes). To ensure faster convergence, they used importance sampling to choose the coordinates, assigning probabilities based on the magnitude of the adversarial perturbation in the current iteration (pixels with a higher magnitude having higher importance). Lastly, the authors proposed a hierarchical attack. In the first couple of iterations, the attack searches for adversarial perturbations in a lower-dimensional space, upsampling the perturbation to fit the input dimension. When the value of the optimised objective does not change much any more, the search space is refined, and the optimisation continues with the same perturbation. The refinement and optimisation might be repeated a couple of times. Especially in tasks with high input dimensionalities, the hierarchical approach helps to speed up the attack.

The same idea of directly estimating the attacked model gradient from output probabilities was explored by Ilyas et al. (2018). In contrast to the ZOO method, their *query-limited attack* works with the L_∞ norm and, instead of finite differences, uses a variant of NES (Salimans et al., 2017) to efficiently estimate the gradient. Moreover, it provides a specific attacking strategy in case the number of queries to the target model is strictly limited. Testing the attack on ImageNet-1k, the authors showed that in most cases, less than 50 000 queries are required to craft a targeted AE.

A purely decision-based non-transfer attack, assuming no more knowledge than the output class for any given input, was proposed by Brendel et al. (2018). They called it the *Boundary Attack (BA)*, as it tries to move along the model decision boundary.

The AE generation process is inverted, it starts from a point $\boldsymbol{x}'(0)$ in the input space that is classified as desired (either the target class in case of targeted attack or any incorrect class in case of untargeted). $\boldsymbol{x}'(0)$ is usually very far from the original image, thus, it is iteratively modified to get gradually closer to the original image $\boldsymbol{x}$. Each iteration consists of two steps. In the first step, a random perturbation of magnitude β is added to $\boldsymbol{x}'(i)$, which is then projected onto an L_2-ball around $\boldsymbol{x}$ with radius $\|\boldsymbol{x}'(i) - \boldsymbol{x}\|_2$. This step is orthogonal to the direction from $\boldsymbol{x}$ to $\boldsymbol{x}'(i)$, keeping the distance between the original and the adversarial image constant. The second step is along the direction toward the original input, scaled by γ. If any of these steps results in an undesirable classification, a different random step is sampled (basically performing rejection sampling). The orthogonal step size β is adaptively modified so that $\approx 50\%$ of orthogonal (random direction) steps are accepted. If the overall success rate of both steps added together is too low, γ is decreased. If it gets close enough to 0, the attack stops. Magnitudes of the resulting adversarial perturbations are comparable to those produced by white-box attacks (FGSM, CW, and DeepFool), but the required number of computations is much higher, sacrificing time in favour of making the attack black-box.

2.3.2 Attacks using non-standard L_p norms

Adversarial attacks usually use L_p norms to measure the magnitude of the perturbation. The general formula is

$$\|\boldsymbol{\delta}\|_p = (\sum_{i=1}^{n} \delta_i^p)^{\frac{1}{p}}, \tag{2.6}$$

where n is the dimensionality of $\boldsymbol{\delta}$. However, all of the previously mentioned attacks use specifically either the L_∞ norm, defined as $\|\boldsymbol{\delta}\|_\infty = \max_i(\delta_i)$, or the L_2 norm, which is the standard Euclidean norm $\|\boldsymbol{\delta}\|_2 = \sqrt{\sum_1^n \delta_i^2}$. But this is not a necessity.

Papernot et al. (2016a) were the first to minimise the number of perturbed features, i.e., pixels, during the computation of the attack. This corresponds to the minimisation of the L_0 norm. In their attack, usually referred to as the *Jacobian-based Saliency Map Attack (JSMA)*, they computed saliency maps to decide which pixels to modify. The saliency of each pixel is based on the Jacobian of the classifier output probabilities $\boldsymbol{y}$. If modifying a certain pixel results in a decrease of y_t or an increase in any y_i with $i \neq t$, where t is the target class of the attack, then the saliency of that pixel is 0. Otherwise, it is computed as the product of the amount of probability increase in the correct class and the sum of probability decreases in all other classes. The attack then picks the top pixel and changes its value by some predefined constant c. It works iteratively, in each iteration re-computing the Jacobian and modifying a single pixel until the input is classified as the target class or the maximum number of iterations (which is equal to

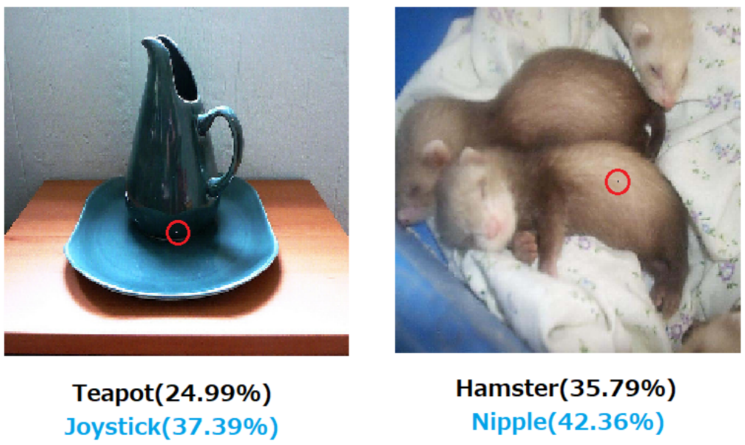


Figure 2.5: AEs created by one pixel attack along with their output class and probability (in blue) and the output class and probability of the original image. The single modified pixel is highlighted by red circle (Su et al., 2019).

the maximum allowed distortion) is reached.

This attack can be easily extended to a more efficient version, modifying multiple pixels in each iteration. The saliency is then computed for each combination of pixels analogically, but the effect on output probabilities is summed across considered pixels.

Another method considering the L_0 norm is the *one pixel attack* by Su et al. (2019). As the name suggests, it searches the space of perturbations, that modify only a single pixel of a given image. The method is black-box, requiring only output probabilities of individual classes for any given input, and can be made both targeted and untargeted.

The adversarial perturbation is found by differential evolution. Evolutionary algorithms, in general, follow the same scheme of running in iterations, each iteration consisting of a set of individuals (population). These produce another set of individuals (children). All of them are evaluated according to a fitness function, and the best of them are kept for the next iteration.

In this specific case, each individual represents a perturbation encoded as a 5-dimensional vector consisting of the x and y coordinates of the perturbed pixel, and the RGB values of the perturbation. The initial population is generated randomly, x and y coordinates are from a uniform distribution, RGB values are from Gaussian distribution. Each population has 400 individuals $\{a_1, \ldots, a_{400}\}$ and for each individual a_i, one child is produced according to the formula

$$c_i(g) = a_{r1}(g) + 0.5(a_{r2}(g) - a_{r3}(g)), \tag{2.7}$$

where $r1 \neq r2 \neq r3$ are random indices, and g is the index of the current generation (iteration). Each parent–child pair is evaluated against each other, and the one with a higher fitness survives and proceeds to the next generation:

$$a_i(g+1) = \operatorname{argmax}\{F(c_i(g)), F(a_i(g))\}. \tag{2.8}$$

In the targeted attack, the fitness function F is defined as the output probability of the target class for the image modified by the given perturbation. In the untargeted attack, fitness is the negative output probability of the correct class.

This attack can also be extended to consider changing more pixels, the length of individuals is five times the number of modified pixels. The authors included results for a 3-pixel attack and a 5-pixel attack. Though these were more successful in fooling the attacked models, in some cases, it was indeed sufficient to only modify a single pixel. For example, the success of the targeted one pixel attack on ImageNet-1k was slightly above 16%. These results are in contrast with the hypothesis of Goodfellow et al. (2015), who explained misclassification in AEs as a result of multiple tiny changes across input dimensions. Apparently, one (greater) change to a single input dimension might also be enough.

Chen et al. (2018) proposed a generalisation of the CW attack, the *Elastic-net Attack to DNN (EAD)*. It uses elastic-net regularisation to find the optimal perturbation, which combines L_1 and L_2 norms in the optimisation

$$\underset{\boldsymbol{x}'}{\operatorname{argmin}}\Big(\|\boldsymbol{x}' - \boldsymbol{x}\|_2^2 + \beta\|\boldsymbol{x}' - \boldsymbol{x}\|_1 + c \cdot F(\boldsymbol{x}', t)\Big), \tag{2.9}$$

such that $\boldsymbol{x}' \in [0,1]^n$ and $F(\boldsymbol{x}', t)$ is defined as in Eq. 2.2, and therefore, CW is a special case of EAD for $\beta = 0$. Moreover, β controls the relative importance of the L_1 and L_2 norm. However, the authors found that using the same reparametrisation trick as CW to ensure the box-constraint does not work well, the optimisation is insensitive to β and produces results similar to those when considering only the L_2 norm, so instead, EAD

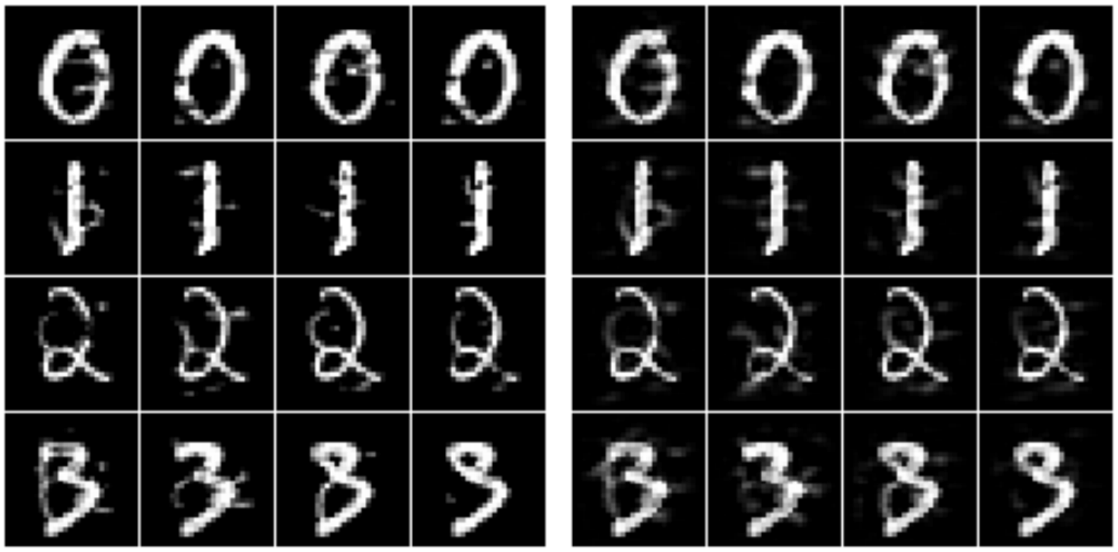


Figure 2.6: Comparison of AEs produced using EAD (left) and CW (right) from the same original MNIST inputs. The L_1-term in EAD promotes sparsity and results in visually sharper perturbations (Chen et al., 2018).

uses the iterative shrinkage-thresholding algorithm (Beck and Teboulle, 2009) to solve the optimisation. Results showed that this is a viable approach, and with increasing β, EAD produces perturbations with low L_1 norms but slightly higher L_2 and L_∞ norms, revealing a trade-off. AEs produced by EAD are, therefore, qualitatively different from those produced by CW, a comparison depicted in Fig. 2.6.

2.4 Mitigating adversarial attacks

Naturally, researchers have been trying to solve the problem of AEs ever since they were discovered. Thus, many different defence methods have been proposed. Some of them aim to modify network architectures to make them inherently more robust, others alter the training process. Also, numerous detection methods, which try to detect and eliminate any adversarially modified inputs, were designed. However, as we will discuss further, none of these have been completely successful.

2.4.1 Adversarial training

Adversarial Training (AT) has been the most promising and most studied defence mechanism so far. It is based on a simple idea of generating AEs during training and using them as part of the training data. Of course, new AEs must be generated continually to replace those that the network can already classify correctly with the new ones, resulting in a very computationally costly procedure. AT was already proposed by Szegedy et al. (2014). However, it was not feasible at that time due to a very inefficient AE generation procedure.

The idea of adversarial training was revisited by Goodfellow et al. (2015). However, instead of adding AEs to the training data, they defined an adversarial loss (based on replacing the clean sample $\boldsymbol{x}$ with an AE computed by the FGSM attack) as

$$J(\boldsymbol{x} + \epsilon\,\mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, l; \boldsymbol{\theta})), l; \boldsymbol{\theta}) \tag{2.10}$$

and then trained a classifier with a linear combination of standard and adversarial loss. Therefore, at each pass considering the accuracy on both clean and FGSM-generated inputs. Using this approach, the authors were able to train a model achieving (at the time) state-of-the-art classification accuracy on the MNIST dataset while also reducing the success of the FGSM attack on this model by more than 70%.

Moosavi-Dezfooli et al. (2016) performed adversarial training with their method DeepFool while comparing it to the FGSM, although they opted for the standard adversarial training (augmenting the data with AEs) instead of using the adversarial loss. The adversarially trained networks were then attacked with DeepFool. As DeepFool

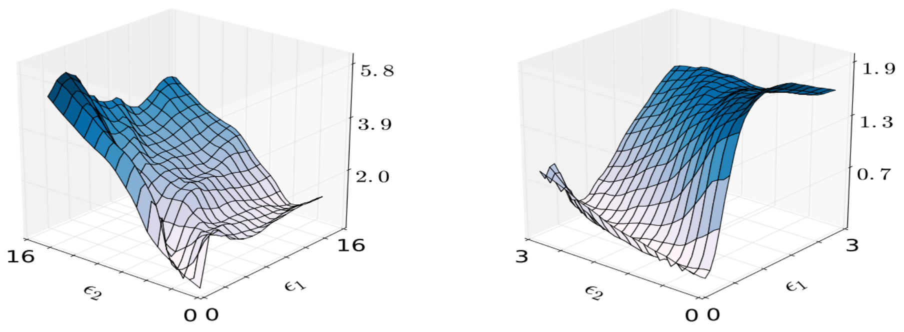


Figure 2.7: Two examples of the loss function of a model adversarially trained with the white-box FGSM attack. Loss is visualised for inputs $\boldsymbol{x}^* = \boldsymbol{x} + \epsilon_1 \boldsymbol{g}_1 + \epsilon_2 \boldsymbol{g}_2$, where $\boldsymbol{x}$ is an input image taken from the training data distribution, $\boldsymbol{g}_1$ is the gradient of the model in $\boldsymbol{x}$, and $\boldsymbol{g}_2$ is an orthogonal adversarial direction (Tramèr et al., 2018).

aims to produce (an approximation of) the minimal adversarial perturbation, its magnitude can then be used as a measure of robustness. The results showed that while AT with DeepFool-generated AEs increases network robustness, AT with FGSM-generated AEs decreases it. Authors hypothesised that this was due to the fact that FGSM, being a method optimised mainly for quick computation instead of minimising the perturbation, produces overly perturbed AEs.

Similar findings were provided by Kurakin et al. (2017), who, using FGSM-generated AEs, managed to adversarially train a network on ImageNet-1k. However, the resulting model was only robust to white-box single-step attacks. Unfortunately, as shown by Tramèr et al. (2018), the model also remained vulnerable to transferred single-step attacks. The reason is that AT with FGSM results in models with a locally more curved loss function, with gradients failing to capture the global behaviour of the loss in the vicinity of the input data points. Visualisation of this phenomenon is shown in Fig. 2.7. Therefore, the linear approximation computed by FGSM on these models does not find the correct "adversarial direction," and the resulting AEs have lower attack success even on unsecured models (trained only with clean, unperturbed data). On the other hand, AEs computed with FGSM on an unsecured model do transfer to adversarially trained ones, as the linearisation in FGSM approximates the loss on these models well. Motivated by these findings, the authors proposed *ensemble adversarial training*. It works by training an ensemble of unsecured models, which then serve as source networks for generating AEs. Another, different model, is adversarially trained on these AEs, along with white-box AEs (generated using gradients from the adversarially trained model). This training strategy greatly improves robustness to black-box transfer attacks.

Madry et al. (2018) formalised the goal of training a robust network as a min-max optimisation, basically optimising the network for its worst-case performance over all allowable perturbations:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\Big[\max_{\delta\in\mathcal{S}} J(\boldsymbol{x}+\boldsymbol{\delta}, l; \boldsymbol{\theta})\Big], \tag{2.11}$$

where $\mathcal{D}$ is the training data distribution and $\mathcal{S}$ is the space of considered perturbations (most commonly an L_∞-ball with radius ϵ). In this formalisation, an adversarial attack serves as an approximation of the inner maximisation. The stronger the attack used for training, the better the approximation. The authors proposed to use their PGD attack (described in Sec. 2.2.3). Recall that the starting point for optimisation is randomised in each computation of the AE, which helps prevent such local deformations of loss function around the original data, as observed by Tramèr et al. (2018) on networks adversarially trained with FGSM. Moreover, the PGD attack might also be run multiple times when producing a single AE, each time starting at a different point and then choosing an AE that maximises the loss function. This results in particularly strong AEs.

However, even with iterative attacks such as PGD, the optimisation is still not solved exactly. Carlini et al. (2017) tried to mitigate this problem by using Reluplex (Katz et al., 2017), a verification tool which, for a given network, input $\boldsymbol{x}$ and maximal allowable perturbation magnitude ϵ, either finds an AE within this constraint, or proves that such an input does not exist. Using binary search to find minimal ϵ for which an AE does exist, this method can be used to measure the robustness (i.e., the average distance to the closest AE) almost exactly. Reluplex has some limitations, though. It can only be used on small (order of hundreds of neurons), piecewise-linear networks. This limits the activation functions to ReLU and max-pooling. Moreover, it can also only work with piecewise-linear constraints, i.e., L_∞ and L_1 norm. Despite these limitations, the authors managed to show that using AT as proposed by Madry et al. (2018) with small networks trained on the MNIST dataset provably increases the minimal ϵ needed to craft an AE on these networks 4.2 times on average.

There were some attempts to modify the AT procedure to make it more efficient, though not all of them were successful. For example, *Adversarial Logit Pairing (ALP)* proposed by Kannan et al. (2018) originally seemed to increase robustness to white-box attacks while not losing any black-box robustness. The idea is to add another loss term during (adversarial) training of the network, which penalises it for the distance between logits of the clean image and logits of its adversarial counterpart. This can be interpreted as teaching the network that a clean image and its respective AE should not only be of the same class, but should be close in the output (logit) space as well. Unfortunately, Engstrom et al. (2018) showed that by simply increasing the number of iterations in the PGD attack (against which ALP was evaluated), the robustness of ALP-trained models drops to almost zero. Moreover, visualising the loss function (depicted in Fig. 2.8) reveals similar artefacts as in the case of FGSM-based AT.

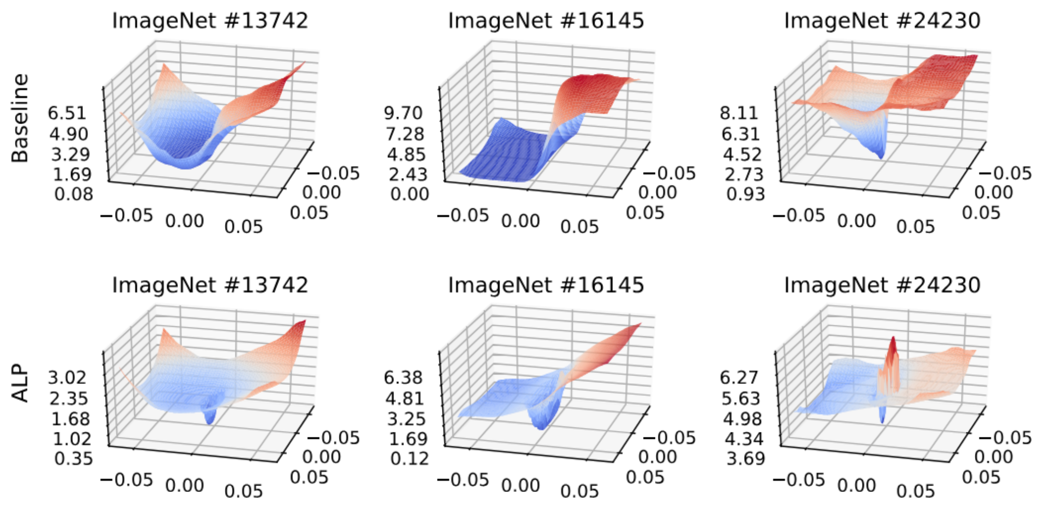


Figure 2.8: Comparison of loss landscapes of the networks trained on clean data (top row) and with ALP (bottom row) for inputs in form $\boldsymbol{x}^* = \boldsymbol{x} + \epsilon_1 \boldsymbol{g}_1 + \epsilon_2 \boldsymbol{g}_2$, where $\boldsymbol{x}$ is a clean image from the test-set, $\boldsymbol{g}_1$ is the gradient of the model in this image, and $\boldsymbol{g}_2$ is a random Rademacher vector. Coefficients ϵ_1 and ϵ_2 correspond to the x and y axes, the z axis is the value of the loss. We can see that ALP results in sharp local minima around the original inputs (Engstrom et al., 2018).

One of the more successful variants of adversarial training is the *feature denoising* by Xie et al. (2019). In this approach, a convolutional neural network is modified by adding denoising blocks after convolutions. Each denoising block consists of a denoising operation (the authors propose four different options), 1×1 convolution, and a skip connection. The modified network is then trained using standard AT. Results on ImageNet-1k showed that feature denoising outperforms other defences by a huge margin, and in contrast to ALP, it retains its success even against the strong 2 000-iteration PGD attack. The issue is that the increased robustness comes with the cost of a non-negligible drop ($\approx 14\%$) in clean accuracy.

Ultimately, even adversarially trained networks with provably increased robustness fail on AEs crafted using norm constraints different from those used in training. Tramèr and Boneh (2019) showed that the model by Madry et al. (2018), although robust to L_∞-bounded perturbations, is surprisingly fragile to attacks constrained in other norms. Moreover, they argued that there is a trade-off between robustness to perturbations of different types. They suggested *multi-perturbation adversarial training*, which considers multiple different spaces of allowable perturbations $\mathcal{S}_1, \ldots, \mathcal{S}_k$ in the Eq. 2.11 and computes the inner maximisation as either the maximum across these spaces or average. The spaces $\mathcal{S}_i$ can be balls in different L_p norms, but also, for example, a space of small affine transformations, such as rotations or translations, as these were also shown to be capable of producing AEs (Engstrom et al., 2019). Then, the authors compared their training strategy with multiple networks, each trained using a standard AT with a different perturbation type. The network trained with multi-perturbation AT attained lower robustness to a specific perturbation type than a network trained to be robust only to that perturbation type, but much higher robustness to other perturbation types, empirically confirming the aforementioned trade-off.

2.4.2 Other defences

Over the years, many other defences not based on AT have been proposed. Unfortunately, most of them were proven unable to provide actual robustness. Among the first broken defences, *defensive distillation* by Papernot et al. (2016b) stands out. It combines two concepts — network distillation (Hinton et al., 2015) and manipulation of temperature in the output softmax activation function. Distillation was originally proposed for transferring knowledge between different models. First, a classifier is trained in a standard way to perform a certain task. Second, another (usually a much smaller) model is trained, while instead of using the one-hot labels from the original dataset, the model is trained on output probability vectors predicted by the first model. The idea is that output probabilities of the trained model contain more information than the one-hot vectors from a dataset. Using this extra information, it is possible to train a smaller model to perform the task, while not losing accuracy. The second element of the defensive distillation is based on the generalisation of the softmax activation function from Eq. 1.3 by adding a temperature parameter T as follows:

$$\mathrm{softmax}(\boldsymbol{x}, T)_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}. \tag{2.12}$$

A high value of T is used when training the first model. This model is then distilled, i.e., a second model is trained to output the same probabilities as the first one. The second model has the same architecture and is trained with the same value of T as the first one. After the training is finished, its temperature is set to $T = 1$. This change of temperature is equivalent to multiplying the network logits by T and as partial derivatives of softmax grow smaller with input values getting larger (in absolute value), the gradients of defensively distilled network are very close to zero. Thus, gradient-based attacks fail to find any AEs. As the authors did not evaluate their defence against any black-box attack, it was initially not clear whether this strategy actually makes the network more robust (in the sense that there do not exist any AEs fooling it), or it just makes the tested attacks unable to find them.

Soon enough, the answer was provided by Carlini and Wagner (2017c) who showed that all three versions of their CW attack (described in Sec. 2.2.3) completely break defensively distilled networks, reducing their accuracy on CW-generated AEs to zero, while an average magnitude of the produced AEs increases (in comparison to undistilled networks) only marginally.

The effect that defensive distillation has on the loss function was called *gradient masking* (Papernot et al., 2017). Both FGSM-based AT and ALP, which were discussed earlier, also exhibit some signs of masked gradients. Athalye et al. (2018a) noted that some defences seem to mask the gradients on purpose. They referred to it as *gradient obfuscation*, while identifying three main categories: gradient shattering, caused by non-differentiable elements or numerical instability; stochastic gradients, as a result of randomness during inference; and vanishing/exploding gradients, which are a well-known problem in very deep computations. For each type of obfuscated gradients, the authors proposed a specific attacking strategy. For shattered gradients, they used the *Backward Pass Differentiable Approximation (BPDA)* method, which, in backward pass, i.e., computation of gradients, replaces the non-differentiable part of the defence by a differentiable approximation. Stochastic gradients can be mitigated using the EOT attack described by Eq. 2.3, while instead of considering a distribution of transformations, one computes the expectation over the stochastic element in the given defence. Exploding/vanishing gradients are alleviated with a re-parametrisation. Using these strategies, the authors devised specific attacks, which managed to break 7 additional defence methods.

However, these attacking strategies, designed to overcome the obfuscated gradients, are not perfect either. For example, Gao et al. (2022) showed that some randomised defences with insufficient randomness, formerly broken with EOT attack, can also be broken with the much simpler PGD attack, if used with smaller step size and more iterations. Therefore, the use of EOT in evaluating those defences is unnecessary. On the other hand, BPDA fails on defences containing elements that are hard to approximate (Sitawarin et al., 2022), and overestimates their robustness. In general, the best practise when evaluating a certain defence is to evaluate it against an *adaptive attack* (Carlini and Wagner, 2017a), i.e., an attack specifically devised against that particular defence. The problem is, that it is not always clear, what the optimal adaptive attack is, or how to construct it. Tramèr et al. (2020b) analysed thirteen different defences, all of which were evaluated against an adaptive attack in their respective original papers, while breaking all of them with different, stronger, adaptive attacks. In conclusion, designing an ultimate method for properly evaluating defences remains an open problem.

2.4.3 Detecting adversarial examples

A different way to tackle the problem of AEs is provided by detection methods. These are applied on a fully trained model during inference to mark each input as being benign or possibly adversarially modified. Among the first proposed detection methods was the one by Metzen et al. (2017). This detection is performed by a simple neural network classifier, which takes activations on a hidden layer of the original network as inputs. The classifier is trained using standard back-propagation on the original dataset extended with AEs generated by an adversarial attack, one AE for each benign input, resulting in a balanced training dataset. In this way, the authors analysed three different attacks: FGSM, BIM, and DeepFool. Overall, they discovered a trade-off (dependent on the magnitude of the adversarial perturbation) between the misclassification rate and the detectability. The AEs with greater perturbations were more successful in causing misclassification but easier to detect.

A very similar detection approach was proposed by Gong et al. (2017), with the distinction that their detector takes the images themselves as inputs, instead of the hidden layer activations. However, the resulting detection accuracy was by far not as good, mainly due to very high false positive rate (benign images classified as AEs).

A different possibility for detection algorithms is using statistics. For example, Li and Li (2017) successfully used statistics (Principal Component Analysis (PCA) coefficients, extremal values, 25-th 50-th and 75-th percentiles) of convolutional layer activations. To have a distribution for each input image and network layer, individual pixels/neurons are considered c-dimensional samples, where c is the number of channels on a given layer, drawn from the distribution. These statistics are then used as inputs to a cascade classifier (Viola and Jones, 2004) to detect AEs crafted by L-BFGS. Moreover, the authors also analysed RC examples, crafted as proposed by Nguyen et al. (2015) and showed that they are very easy to detect, as their statistics diverge greatly from those of the clean data.

Another method inspired by statistics was proposed by Grosse et al. (2017a), who showed that statistical tests could be used to detect AEs (using FGSM and JSMA attacks in their experiments). However, the statistical tests need samples of size around 50 or more to provide confident results, so they cannot be used to classify a single image as being adversarial or benign. Therefore, the authors once again opted for machine learning to solve the problem, and chose to train a classifier with an additional $(k+1)$-st class corresponding to adversarial examples. The procedure works as follows: first, a classifier with $(k+1)$ output neurons is trained on the original dataset, then a batch of AEs is crafted. Second, the classifier is fine-tuned on these AEs to classify them as belonging to the $(k+1)$-st class. The results showed that a classifier trained in this way with JSMA-crafted AEs is later able to detect AEs crafted with FGSM. However, it does not work the other way around and models trained to classify FGSM-crafted AEs fail to detect JSMA-generated AEs.

Feinman et al. (2017) designed two different statistical approaches of detecting AEs. The first of them is based on approximating original data class manifolds on the last

DNN hidden layer by kernel density estimation and assuming that AEs are off this manifold. The second one uses dropout (the training regime version) to introduce randomness into the classifier inference. By running the inference multiple times and taking variance of outputs as an estimate of prediction uncertainty, most of the AEs can be identified as inputs with high output uncertainty.

Similarly to defence methods, most detections were also broken shortly after being proposed. For example, Carlini and Wagner (2017a) managed to break all previously mentioned detection methods (and a few more), either by using the strong CW attack, or by designing a specific loss function to be optimised during the attack. The only detection method that was broken only partially (in the sense that it significantly increased the required AEs perturbation magnitude) was the one based on estimating uncertainty using dropout. However, the most recent research (Lucas et al., 2023) suggests that randomness in general may not be a helpful aspect in defending against AEs.

Moreover, He et al. (2017) showed multiple detections based on ensembling to be ineffective against adaptive attackers. Among others, also the popular *feature squeezing* (Xu et al., 2018), which uses multiple "squeezing" transformations on the inputs: bit-depth reduction, median filtering, and non-local blur. If the output probabilities of any squeezed version of the image differ significantly from the output probabilities of the unsqueezed input, it is marked as an AE. Besides breaking detections specifically designed to work as ensembles, He et al. (2017) also combined training a DNN detector by Gong et al. (2017), training a detector that takes hidden activations as input by Metzen et al. (2017), and kernel density estimation by Feinman et al. (2017) in a single model, and designed an attack that can break all three of these detections at once, showing that combining multiple (weak) defence methods is not strong.

Even though research on robustifying DNNs has been mostly independent from research on detecting AEs, and detection has been considered a much easier task, Tramèr (2022) proved a link between robustness of a standard classifier and a detector — classifier with the option to reject inputs (i.e., mark them as AEs). The exact result is that the existence of a detector with an accuracy of α within ϵ distance from clean inputs implies the existence of a classifier with the same accuracy α within $\epsilon/2$ distance from clean inputs, and vice versa. It was shown that applying this theorem to multiple detection methods would imply the existence of classifiers with robustness greatly above the current state-of-the-art, therefore questioning the validity of the detections evaluations.

2.4.4 Certified defences

A completely different method against AEs is provided by *certified defences*. They are based on the concept of having a classifier, which, alongside classification, can also generate a *robustness certificate* for a given input. This means, that it is possible to prove that within an ϵ distance from this input (usually measured in some L_p norm), the output of the classifier remains the same. Of course, it may not be possible to obtain a certificate for each input. The problem is then split into two sub-problems: devising ways of formally proving robustness for given inputs, and training classifiers, which are able to issue the robustness certificates with a high success rate.

The advantage is that these classifiers remain robust even after the emergence of new, stronger attacks. The downside is that the certified robustness is much lower than the empirically evaluated robustness of uncertified defences such as AT. Moreover, the early certified defences were restricted to some specific class of NNs. For example, Raghunathan et al. (2018) proposed a training method for certified robustness, which is only applicable to NNs with a single hidden layer. With this approach, the authors trained a network with a provably robust test error below 35% on MNIST, considering the perturbation constraint $\epsilon < 0.1$ in the L_∞ norm.

Wong and Kolter (2018) used a different method to provide certificates, but their method is similarly limited. It can only be applied to NNs using exclusively the ReLU activation function. The results on the MNIST dataset seemed quite promising, with the same constraint of $\epsilon < 0.1$ in the L_∞ norm, they managed to train a network with provable test error $< 5.8\%$. However, the results on slightly more challenging datasets (FMNIST and SVHN) were not as pleasing. For example, the achieved provable upper bound on robust test error for SVHN was 40.67% for $\epsilon < 0.01$.

Later, a couple of methods not limited to any specific NN architectures were proposed. For example, *interval bound propagation* by Gowal et al. (2019) can be used to train relatively large certifiably robust networks. With this method, the authors were able to lower the certified error on MNIST under $\epsilon < 0.1$ in L_∞ norm to 2.23%, but also train the first certifiably robust network on ImageNet-1k (though using inputs downscaled to 64×64 px). However, they also observed that the training with this method is highly unstable and requires careful parameter tuning and scheduling.

Another popular method applicable to any NN is called *randomised smoothing* (Cohen et al., 2019). It works with the L_2 norm and uses Gaussian noise to create a "smoothed" version of a given classifier f. The smooth output f' is defined as the most probable output $f(\boldsymbol{x}')$ for $\boldsymbol{x}' = \boldsymbol{x} + \mathcal{N}(0, \sigma^2 I)$, i.e., $\boldsymbol{x}$ perturbed by adding random isotropic Gaussian noise. Based on the probabilities of the top class and the second most prevalent class, it is possible to analytically compute the radius ϵ of the ball centred in $\boldsymbol{x}$, for which the output is certifiably robust. One obvious disadvantage is that the probabilities cannot be computed exactly. In practice, they are estimated using Monte Carlo sampling. Then, the certificate does not mean that the output is provably robust, but the probability that the input was adversarially modified, is provably smaller than some fixed constant γ. The success rate with which the certificates are generated depends on the choice of γ and the number of used samples, but also on the output probabilities of the individual classes. Therefore, the authors proposed to train the classifier not only on the clean data, but also on data perturbed by the aforementioned Gaussian noise, to "teach" the classifier to predict the same (correct) class over such a distribution with high probability. The choice of σ controls robustness–accuracy trade-off, more closely studied by Gao et al. (2022). Greater σ improves (certified) robustness for greater radii ϵ while degrading accuracy for smaller ϵ.

Eventually, even if it was possible to train perfectly (certifiably) robust image classifiers, one problem still remains — the commonly used L_p metrics do not capture human visual perception exactly. This is reflected in two problems: 1) the existence of perceptually very similar inputs belonging to the same class that are far in an L_p metric and classified differently, and 2) the existence of images that are close in an L_p metric and classified as the same class, but perceptually dissimilar, and labelled differently by human labellers.

The first mentioned problem was studied by Ghiasi et al. (2020). They devised an attacking method called *shadow attack*, which applies a large (in some L_p norm) but almost imperceptible perturbation to an input $\boldsymbol{x}$, resulting in a wrongly classified input $\boldsymbol{x}'$, such that an entire neighbourhood of $\boldsymbol{x}'$ (given some norm) is classified the same. Therefore, certifiably robust NN both misclassifies this sample and generates a certificate stating that the classification is robust. Such an input is not an AE in the standard definition constraining it in an L_p norm, but it is an AE in the sense that it is an adversarially modified input, perceptually similar to a clean one, created with the intention of causing misclassification. An example of an image produced by the shadow attack is in Fig. 2.9.

The second problem was identified by Tramèr et al. (2020a), who studied a unique set of AEs created from original data by adding a small perturbation, such that the true label (according to human labellers) of the input changes, while the classifier output remains the same. The misclassification, in this case, arises from the fact that even though the perturbation is small, it changes the correct output class. Classifiers that are too invariant (insensitive to input changes) produce the same output for both clean and the perturbed image, thus, at least one of them must be incorrect. Therefore, these AEs are called *invariance-based*. Two examples crafted using different L_p norms are in Fig. 2.10.

The main problem with invariance-based AEs is that when considering defences

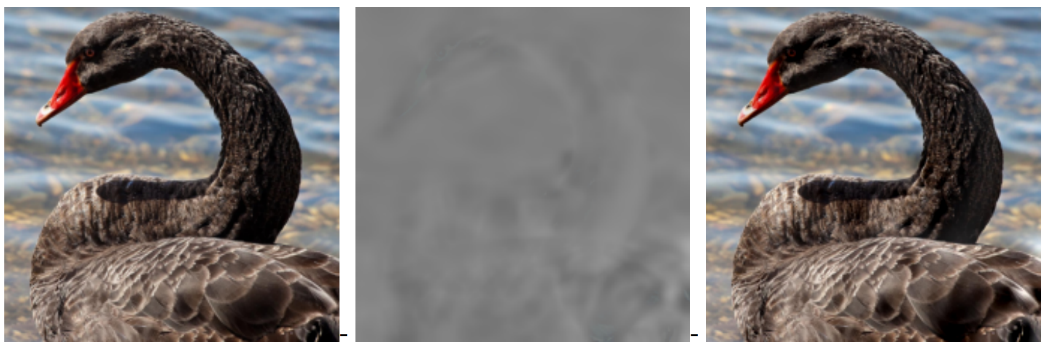


Figure 2.9: An image from the ImageNet dataset (left), its adversarial perturbation computed by the shadow attack (middle), and the resulting AE (right). Image from Ghiasi et al. (2020).

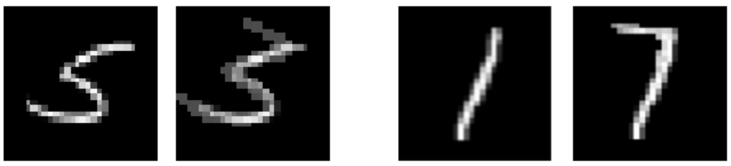


Figure 2.10: Pairs of a clean image and its invariance-based AE constrained in the L_∞ metric (left) and the L_0 metric (right). Output labels for the AEs were assigned by human labellers. Image adapted from Tramèr et al. (2020a).

against the "standard" AEs, the goal is usually to train networks that are more invariant and, consequently, more prone towards invariance-based AEs. Specifically, in the case of certifiably robust networks, obtaining a robustness certificate for a given input image and a certain radius ϵ means that any invariance-based AE produced from this image with a perturbation of magnitude smaller than ϵ is provably misclassified by this network. Therefore, if we resort to considering only the L_p norms, we encounter a trade-off between the robustness against invariance-based AEs and the robustness against standard AEs (Tramèr et al., 2020a).

2.5 Analysing and explaining adversarial examples

Some fraction of the papers about AEs were not dedicated to proposing any new attacks or defences, but to analysing the existing ones. Most importantly, researchers have been trying to answer the question as to why AEs even exist if they are so hard to find by chance. Another important question is where exactly the AEs lie with respect to the clean data. Some authors argue that AEs are actually out-of-distribution, they do not lie on the clean data manifolds and, therefore, it is only natural that they are misclassified. Other authors think the opposite is true and try to prove the existence of on-manifold AEs. In this section, we summarise a couple of the most influential works.

2.5.1 Explaining the existence of AEs

Already the very first paper on AEs (Szegedy et al., 2014) speculated about the reason for their existence. The authors compared AEs to rational numbers in the space of real numbers. They have an extremely low probability, therefore, they are not found in the testing data, yet they are dense and can be found using optimisation.

Goodfellow et al. (2015) challenged this hypothesis and proposed a different explanation. The authors suggested linearity of NN in combination with higher dimensionality of the input data as the possible cause and supported their statement by the fact that their linearity-based FGSM attack can successfully find AEs for almost every given input.

Gilmer et al. (2019) also blamed the input data dimensionality as the source of some non-intuitive properties. Using isotropic Gaussian noise to augment the data, and average distance to decision boundary as the measure of robustness, they provided a formula to capture the relationship between test error on noisy data and robustness on clean data, considering a linear classifier. Even very low non-zero noisy test error implies extremely low robustness for classifiers with high input dimensionalities. To support this theory also for non-linear classifiers, they used the average distance to PGD-generated AEs to approximate the robustness and showed that the relationship holds almost the same. Next, they compared AT with training on noisy images and found some similarities in the resulting networks. Therefore, they suggested evaluation on noise-corrupted inputs as the first sanity check of defences against AEs.

Other authors argued that vulnerability to adversarial attacks might be just a consequence of training the networks to achieve high classification accuracy on clean data. For example, Tsipras et al. (2019) noticed that AT with large numbers of training samples leads to reduced accuracy on the original data. Therefore, they hypothesised the existence of a trade-off between clean and robust accuracy. To support their claims, they constructed a synthetic dataset for which it is impossible to achieve high classification accuracy without losing robustness. The dataset was created to contain *robust features* that cannot easily be manipulated by small perturbations but that are not sufficient to reach accuracy above some threshold. Besides, it also contains *non-robust features* that can be easily manipulated by an adversary, but that are necessary to increase the classification accuracy.

In their follow-up paper (Ilyas et al., 2019), the authors further promoted the idea of non-robust features in the data by marking them as the main cause of vulnerability to AEs. To show that natural datasets are also comprised of both robust and non-robust features, they proposed a way of modifying the CIFAR-10 dataset to contain only the robust features. To define which features are robust, they used the adversarially trained model from Madry et al. (2018), more specifically, the part of it up to the layer before the logit layer (let's denote it $\boldsymbol{g}$), which we can think of as a feature extractor for the last layer (which is then just a linear classifier). Then, for each image $\boldsymbol{x}$ from the original dataset, a robust image $\boldsymbol{x}_r$ was created by projected gradient descent optimisation, starting from a random different image from the dataset, optimised so that $\|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}_r)\|_2$ was minimal. The motivation behind this optimisation is that as the adversarially trained model is more robust, it relies more on the robust features. So, by optimising an input to result in similar activations, the robust features are introduced into the input. The authors used the resulting robust dataset to train a classifier and showed that even though standard training was used, the classifier attained non-trivial levels of robustness.

In their second experiment, the authors demonstrated the existence of non-robust features by creating a non-robust version of the dataset. For each input image $\boldsymbol{x}$, its non-robust version was created by picking an incorrect class t (either randomly or according to some fixed permutation of classes) and performing a targetted PGD attack with target t. The resulting AE $\boldsymbol{x}'$ was assigned t as its label in the non-robust dataset (so to humans, this label seems wrong because the AE still looks very similar to the original image). After training a classifier on this non-robust dataset, it achieved fairly high classification accuracy on the original CIFAR-10 test-set.

In conclusion, the authors stated that the existence of AEs is not a property of the trained models but of the data itself. This hypothesis would also explain the transferability phenomenon, as two different models trained on the same data are likely to learn the same non-robust features and, therefore, be vulnerable to perturbations of the same type.

2.5.2 AEs and clean data manifolds

The problem of robustness–accuracy trade-off that we mentioned earlier is closely connected to the question of whether improving generalisation of the trained models harms or improves robustness. In other words, are AEs drawn from the original data distribution or not? Or, from the geometrical point of view, are they included in the original data manifold?

These questions are not easily answered. The fact that many proposed detection methods based on statistics were initially successful against some types of attacks hints that at least some AEs are statistically different from the original data and, therefore, out-of-distribution. However, these detections were later broken by different attacks. Does that mean that there are also some in-distribution AEs, or are the detection methods just not accurate enough?

Considering the geometrical interpretation with data manifolds, it is the same story. Numerous defence methods were based on the hypothesis that AEs lie off the data manifold, only to be later broken. For example, MagNet (Meng and Chen, 2017) and APE-GAN (Shen et al., 2017) were broken by Carlini and Wagner (2017b). PixelDefend (Song et al., 2018) and Defense-GAN (Samangouei et al., 2018) were broken by Athalye et al. (2018a).

To actually prove the existence of on-manifold AEs, Gilmer et al. (2018) constructed a very simple dataset consisting of two concentric spheres, each belonging to a different class. To craft *on-manifold* AEs, the attack just needs to be constrained to the surface of the original sphere, which, in practice, is achieved by a simple projection after each step of the PGD attack. The authors trained a NN classifier on this dataset using 50 million unique training points and evaluated it over an additional 20 million points while observing 0% test error. However, they were still able to craft on-manifold AEs for this network.

Stutz et al. (2019) followed the idea of on-manifold AEs with more realistic datasets. They created one MNIST-inspired synthetic dataset, called FONTS, containing white letters on a black background. The crafting process was randomised by considering different fonts for the letters, but also various spatial transformations. As the used transform distribution was known exactly, so was the data manifold. Crafting on-manifold AEs was then performed by optimisation in the space of transform parameters. By also enforcing the similarity constraint in the transform space, the authors achieved higher perceptual similarity of the crafted AEs than by constraining them in the image space. Moreover, they used a couple of additional datasets for which the exact data manifolds were not known. Therefore, they were approximated by training generative NNs to model the data distribution. Similarly to FONTS, while crafting on-manifold AEs, the optimisation was performed in the hidden space of the generative NNs. Then, the authors compared the crafted on-manifold AEs with those crafted using the standard

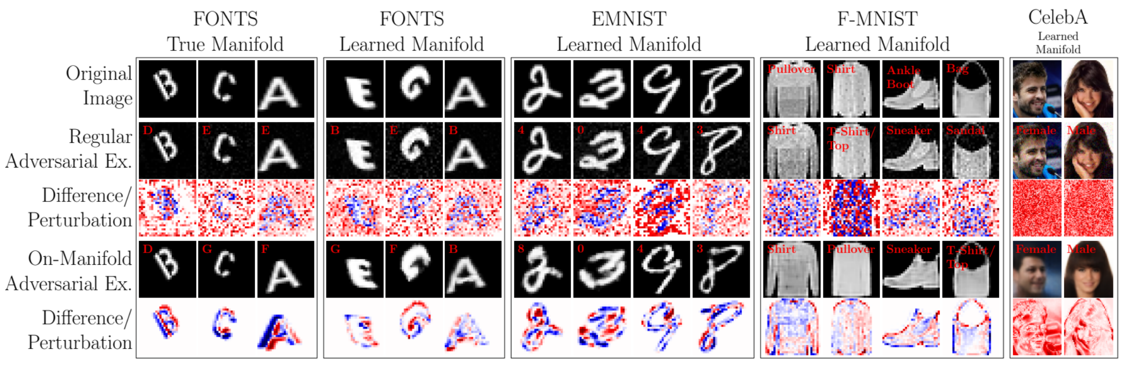


Figure 2.11: Comparison of on-manifold and off-manifold (regular) AEs crafted by Stutz et al. (2019).

PGD attack. They showed that PGD-crafted AEs indeed do leave the data manifolds. However, as they proved by crafting them, on-manifold AEs exist too. Moreover, contrary to the previous beliefs, by training on on-manifold AEs, the generalisation of the models increases. However, the observed increase in testing accuracy was most prominent on the dataset for which the data manifold was known exactly. Using this approach with other datasets clearly suffered from the fact that the trained generative NNs did not manage to fit the data distribution exactly, as can be seen in Fig. 2.11, for example, in the "FONTS with learned manifold" column, on the slightly distorted letter 'A.' Unfortunately, AT with on-manifold AEs did not increase robustness against regular, off-manifold AEs.

Chapter 3

Inherent robustness of neural networks

As none of the proposed defence methods provides perfect robustness, even small advancements may be beneficial. We believe that the study of the inherent robustness of various neural network architectures is an important research direction. If we found NNs that are inherently more robust and identified their aspects that provide increase in robustness, then this knowledge could be used to design novel architectures that further promote the aforementioned aspects and, thus, gain robustness.

3.1 Robustness of networks with logistic sigmoid

In our very first experiments, published in Bečková et al. (2020), we aimed at monitoring the progress of robustness during the training of a network. Therefore, we opted to work only with architectures that are rather simple, allowing for better separation of effects caused by adversarial attacks. To gain better insight into the processes taking place during training, we deliberately analysed networks with distinct training profiles, namely, networks with logistic sigmoid activation function with varying values of temperature T:

$$\sigma(x, T) = \frac{1}{1 + e^{-x/T}}. \tag{3.1}$$

Based on the value of T, neurons with this activation function saturate at different rates. We also hoped to take a closer look at the linearity hypothesis by Goodfellow et al. (2015). With decreasing temperature, logistic sigmoid approaches the (piece-wise linear) step function. With increasing temperature, the slope of the sigmoid gets less steep. Therefore, based on the linearity assumption, we expected networks with low temperatures to be less robust than networks with high temperatures.

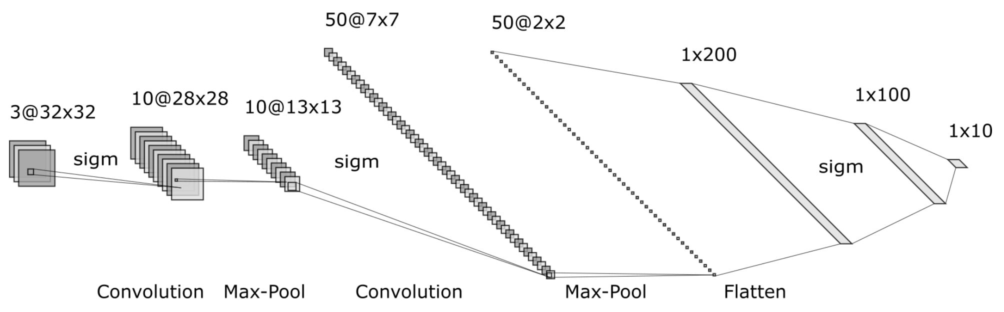


Figure 3.1: CNN architecture used for SVHN and CIFAR-10. Image plotted using NN-SVG tool (LeNail, 2019).

3.1.1 Models and data

In our study, we used four different datasets. On MNIST and FMNIST, we trained MLPs having a single hidden layer of 256 neurons with logistic sigmoid as the activation function, considering $T = \{1/64, 1/32, \ldots, 4, 16\}$. For SVHN and CIFAR-10, we used a simple CNN depicted in Fig. 3.1. All sigmoids in this network were set to have the same temperature, using values $T = \{1/8, 1/4, \ldots, 4, 8\}$.

To assess the levels of neuron saturation, we analysed the *net* values, i.e., neuron values before the activation function. For each of the analysed sigmoids, we split the space of real numbers into three disjoint regions based on their value in the respective sigmoid:

- **Linear region**: the linear region is defined according to the tangent line of the sigmoid graph computed at $x = 0$. The region is comprised of all numbers x, for which the point $(x, \sigma(x, T))$ is closer to the tangent line than a threshold d_1.
- **Non-linear region**: the non-linear region contains those values x that are not included in the linear region and for which the first derivative of $\sigma(x, T)$ is larger than a given threshold d_2.
- **Saturated region**: the saturated region is made of all the remaining values, i.e., all x, for which the derivative of $\sigma(x, T)$ is smaller than d_2.

These regions are visualised in Fig. 3.2. In our experiments, we used threshold values $d_1 = 0.05$ and $d_2 = 0.0005$. For simplicity, in the case of SVHN and CIFAR-10, we only analyse the neurons from the last hidden layer.

To analyse the network robustness, we needed to generate AEs. Since one of our main aims was to evaluate the robustness at each epoch during training, we chose two computationally efficient methods. The first of them is the FGSM attack constrained in the L_∞ norm with $\epsilon = 0.1$. The second one is constrained in the L_0 norm, and it works as follows: first, the gradient of the loss function is computed (the same gradient is needed for the FGSM attack, so this step is basically for free). Then, from the

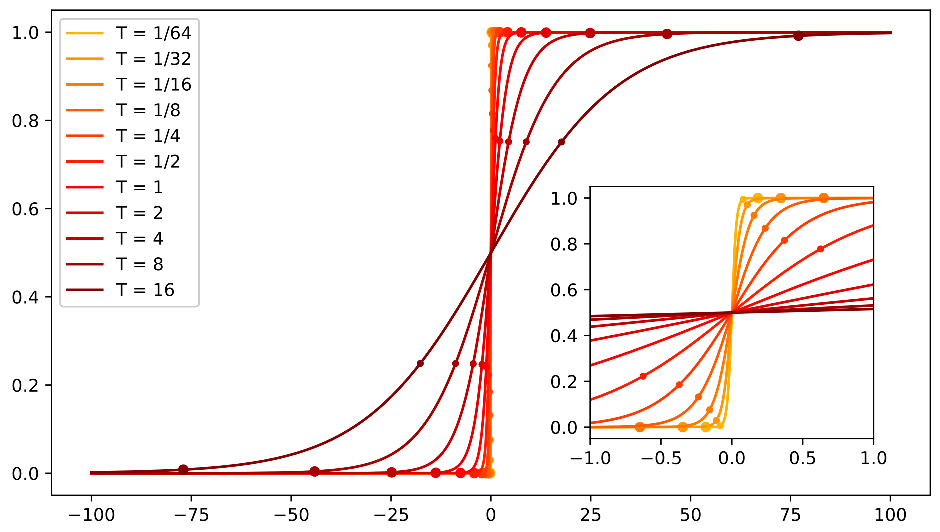


Figure 3.2: Graph of the logistic sigmoid for various values of the temperature parameter. The smaller dots denote the border points between linear and non-linear regions, the larger dots divide non-linear and saturated regions.

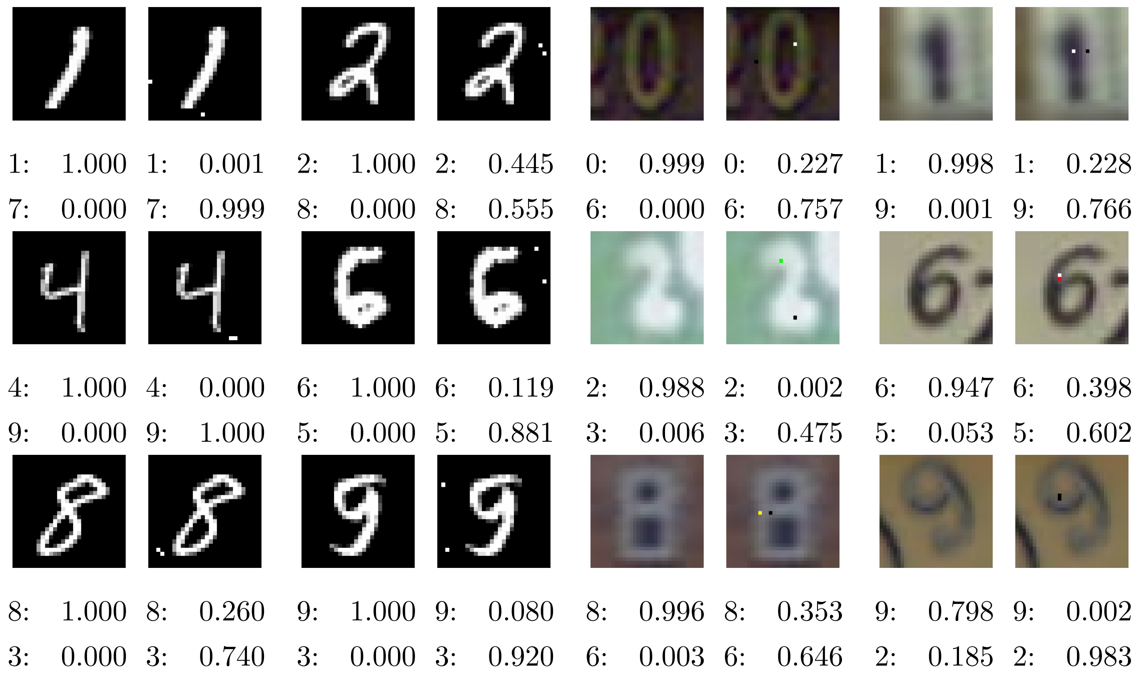


Figure 3.3: Examples of original–adversarial pairs from the MNIST (left) and SVHN (right) datasets generated by our L_0-constrained method. Below each image, the output confidence for the original class and the incorrectly predicted class (of the corresponding AE) is included.

gradient, we identify those k pixels in the input image, for which the absolute value of the partial derivative of the loss is the highest. These pixels will be modified. In our experiments, we consider changing 1, 2 or 3 pixels. Their new value is found using grid-search. For simplicity, for each pixel, we consider only the extremal values 0/1 (which, in the case of RGB inputs, corresponds to 8 possible colours, as each channel is modified independently). To the best of our knowledge, we are the first to use an attack defined in this way. It is much computationally faster than the one pixel attack by Su et al. (2019) and conceptually simpler than JSMA. Some examples of the resulting AEs with 2 modified pixels can be seen in Fig. 3.3.

3.1.2 Results

Visualisations of the saturation development for each of the tested networks can be found in Fig. 3.4. The "starting point" differed greatly across the various temperature values, but in all cases, we noticed a gradual shift from linear and non-linear activations towards saturated ones. If we compare these values with the development of robustness (measured as the ratio of failed adversarial attack attempts) visualised in Fig. 3.5, we

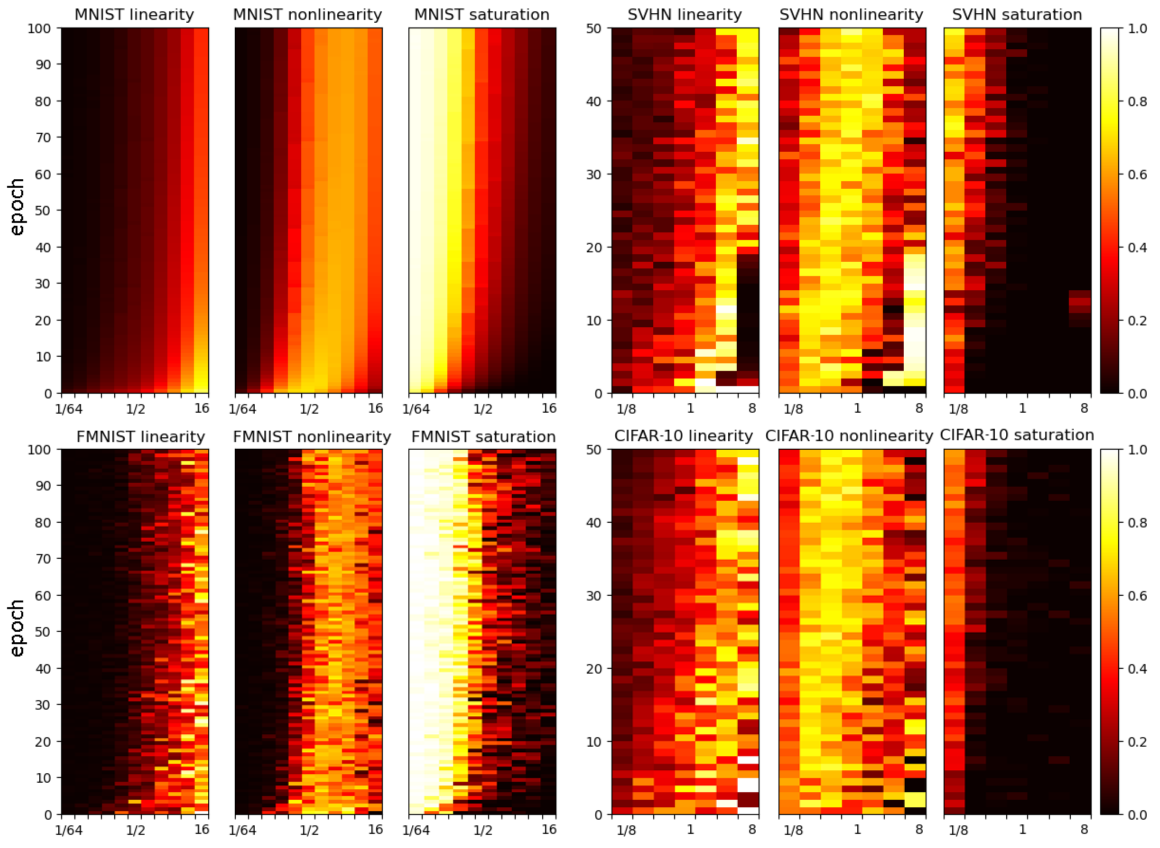


Figure 3.4: Graphs of the development of neuron saturations for all tested networks across all training epochs. Colour denotes the fraction of activations belonging to a given region.

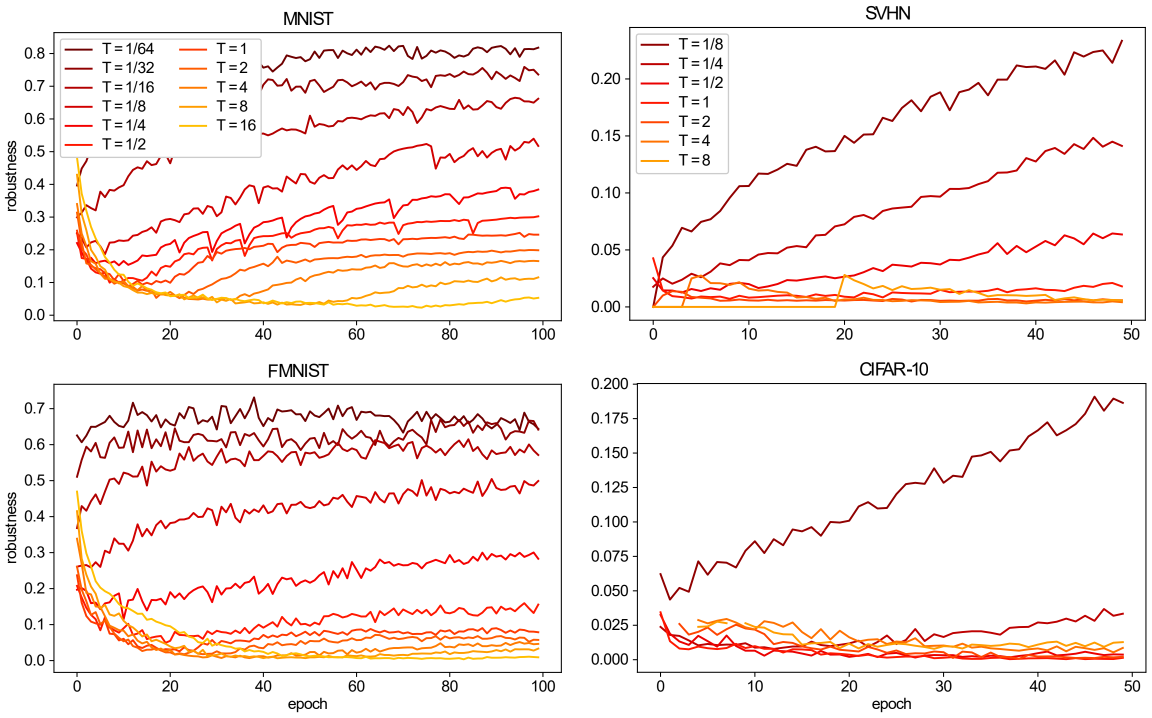


Figure 3.5: Development of network robustness across all training epochs, measured as the resilience against the FGSM attack. For all four datasets, networks with low temperatures are harder to attack.

can notice a clear correlation between the network saturation and robustness. Moreover, while for SVHN and CIFAR-10 the robustness mostly increased during training, for MNIST and FMNIST this was not the case for networks with higher temperatures.

We decided to examine this phenomenon more closely, using additional values of ϵ to constrain the perturbation magnitude. Fig. 3.6 shows the development of robustness during training for four different values of ϵ and three different temperatures. Regardless of the used ϵ, the robustness exhibits the same profile. It is almost constant for the network with low temperature, but it decreases for networks with higher temperatures.

Based on these results, we hypothesised that the networks with low temperatures were, unfortunately, only seemingly robust, and, similar to defensive distillation, the

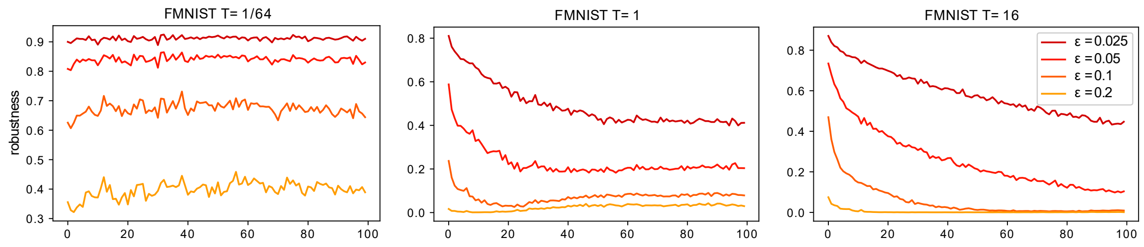


Figure 3.6: Development of robustness against the FGSM attack during training of a network trained on the FMNIST dataset, evaluated for different values of ϵ.

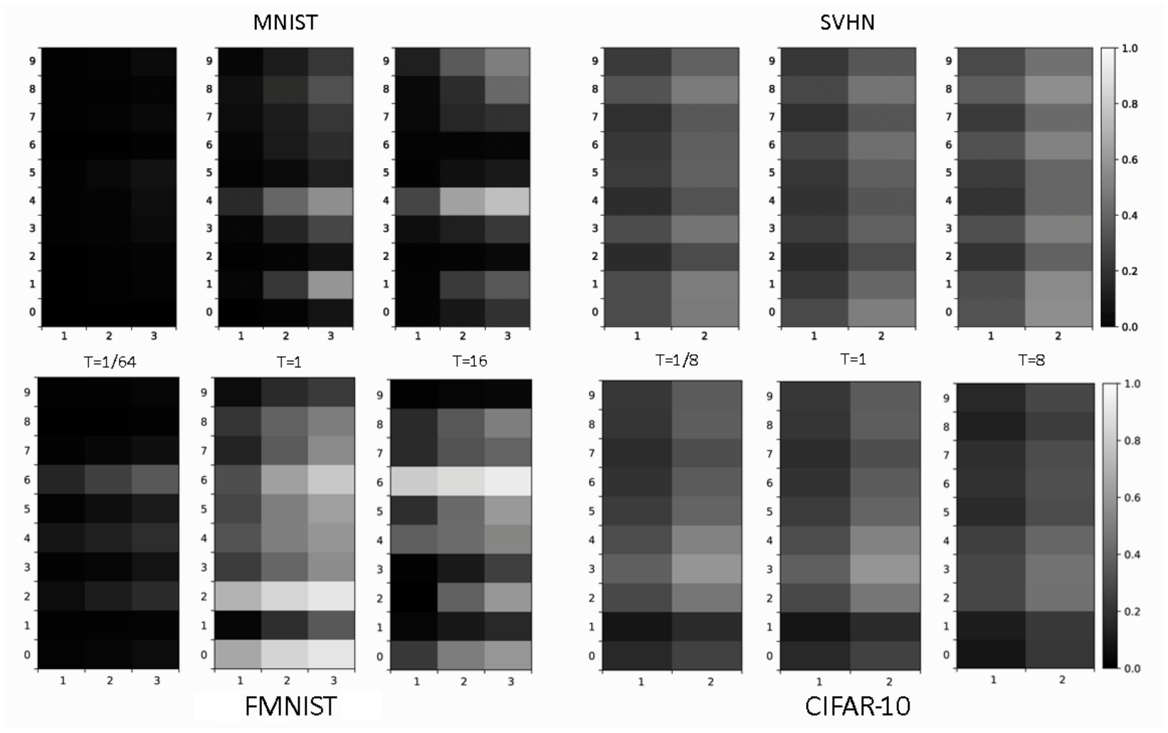


Figure 3.7: Success of our L_0-constrained attack across all classes and all four datasets for different T. Columns correspond to the number of perturbed pixels and rows to individual classes.

saturation of neuron activations provided some sort of gradient masking. Networks with low temperatures were highly saturated from the beginning, therefore, their gradients were not "useful" in finding an adversarial perturbation. This did not change during training, hence, the robustness remained almost constant. Using higher temperatures, the gradients in the first few epochs were not useful for the attack, because the network was not yet trained properly. However, during training, they became more meaningful, so the robustness dropped.

Evaluation of robustness using our L_0-constrained attack also supported the gradient masking theory. As can be seen in Fig. 3.7, the results were slightly different. Low temperatures were still the most robust for the MNIST and FMNIST datasets. However, for SVHN and CIFAR-10, the robustness did not seem to depend on the choice of temperature. We explain this by the fact that our attack is not as much gradient-based (thanks to the grid search across values) as FGSM, therefore, it suffers less from the gradient masking.

In our last experiment, we examined the gradients of loss computed with respect to the input. We defined $G(\boldsymbol{x})$ as the average absolute value of partial derivatives of loss $J(\boldsymbol{x}, l; \boldsymbol{\theta})$ with respect to input components x_i:

$$G(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\partial J(\boldsymbol{x}, l; \boldsymbol{\theta})}{\partial x_i} \right|, \tag{3.2}$$

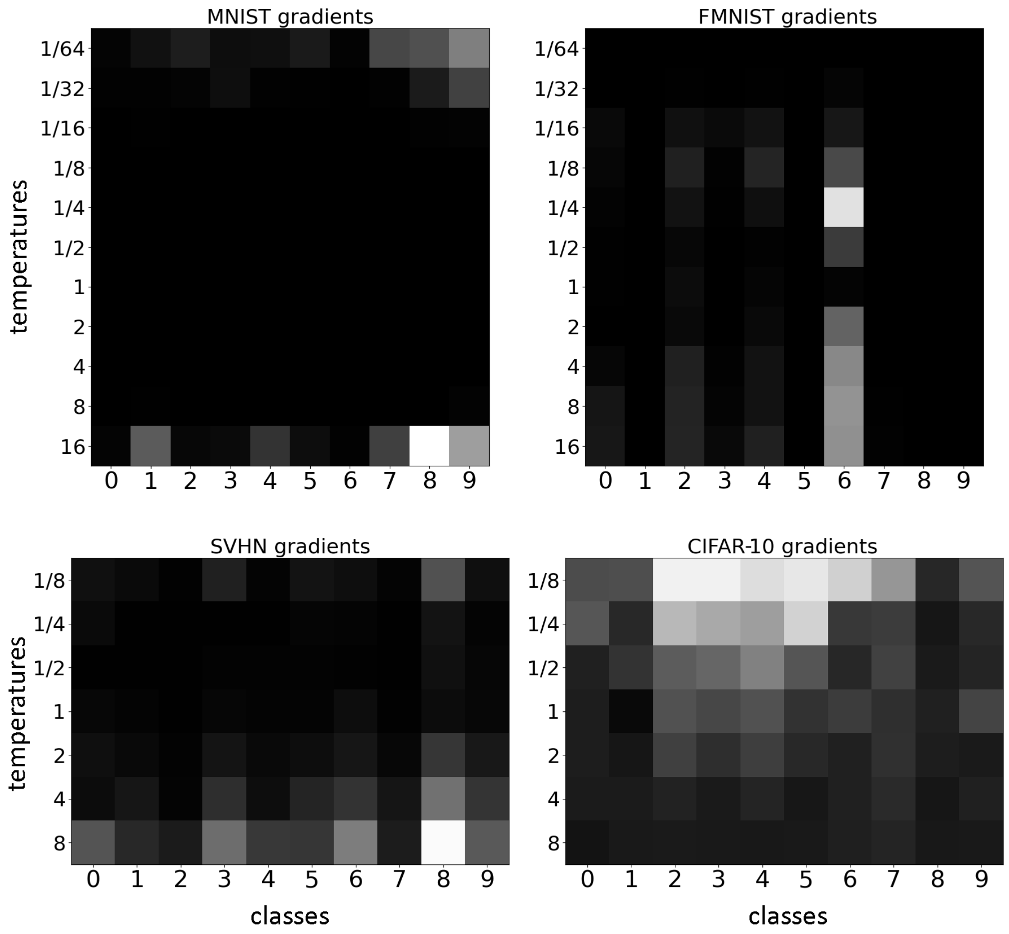
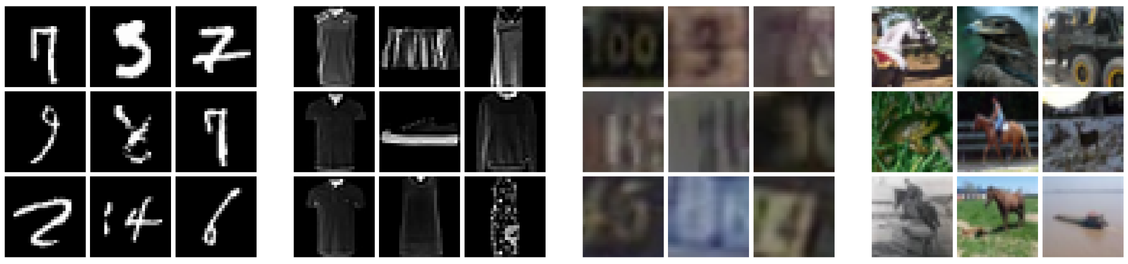


Figure 3.8: A visualisation of medians of the values of $G(\boldsymbol{x})$ per each class and network temperature. Graph values are in range $[0, 0.0002]$ for MNIST, $[0, 0.2]$ for FMNIST, $[0, 0.04]$ for SVHN and $[0, 0.15]$ for SVHN, all ranges scaled from black to white.

where n is the dimensionality of the input. We evaluated $G(\boldsymbol{x})$ for each class individually, as we observed some differences between classes, especially in the case of FMNIST. We also noticed some outliers, a few of them visualised in Fig. 3.9. To our surprise, some of the inputs with abnormally large $G(\boldsymbol{x})$ are "hard", for example the incorrectly cropped '4' in MNIST or the images of shirts in FMNIST that are probably too dark. Therefore, instead of computing the average across all inputs from a certain class, we computed the median. The results are illustrated in Fig. 3.8. Contrary to our expectations, the median values of $G(\boldsymbol{x})$ did not correlate with the success of adversarial attacks. They were higher for very low temperatures in the case of MNIST and CIFAR-10, and very high temperatures in the case of MNIST and SVHN. The median $G(\boldsymbol{x})$ of networks trained on the FMNIST dataset did not seem to depend on the temperature as much as the class. This suggests that even though the networks with low temperatures achieved high levels of saturation, their gradients did not vanish, they just became completely unusable for the attacks.



(a) MNIST (b) FMNIST (c) SVHN (d) CIFAR-10

Figure 3.9: Examples of images with a large value of $G(x_i)$, all from models with $T = 1$.

3.2 Robustness of RecViT

In this section, we present a pilot adversarial robustness study of the newly proposed NN architecture *RecViT (Recurrent Vision Transformer)* developed by our lab member Štefan Pócoš. Preliminary results of these experiments were published in Pócoš et al. (2024b,a).

3.2.1 RecViT architecture

The main motivation behind the RecViT architecture was provided by the work of Stollenga et al. (2014), who enhanced a CNN by adding top-down feedback in the form of adaptive weighing of convolutional kernels in an iterative way. Building upon this idea, the RecViT adds a recurrent connection to standard ViT to facilitate the top-down information flow.

The inference in RecViT is very similar to ViT. First, the input image is split into patches, which are linearly projected, and combined with their corresponding positional embeddings. An extra trainable vector is prepended, the class token. All these vectors are then processed by multiple transformer encoder blocks. While the resulting representations of image patches are not used anymore, the representation of the class token is processed further. In the standard ViT, it is directly used as the input to a simple MLP that performs the classification. However, in RecViT, the representation of the class token is copied as the initial value of the class token in the next iteration (even though it might also be fed into the MLP in the non-final iterations to track the intermediate predictions). Representations of the image patches may be computed in each iteration from the same input image, but, as we will explain later, it can be beneficial to introduce some meaningful input transformations. After a fixed number of iterations k, the class token is sent as the input into the MLP to obtain the final classification. An illustration of the process is visualised in Fig. 3.10.

Our hypothesis is that RecViT could be more robust than the standard ViT. The reasoning behind this hypothesis is that by adding the top-down connection, we provide

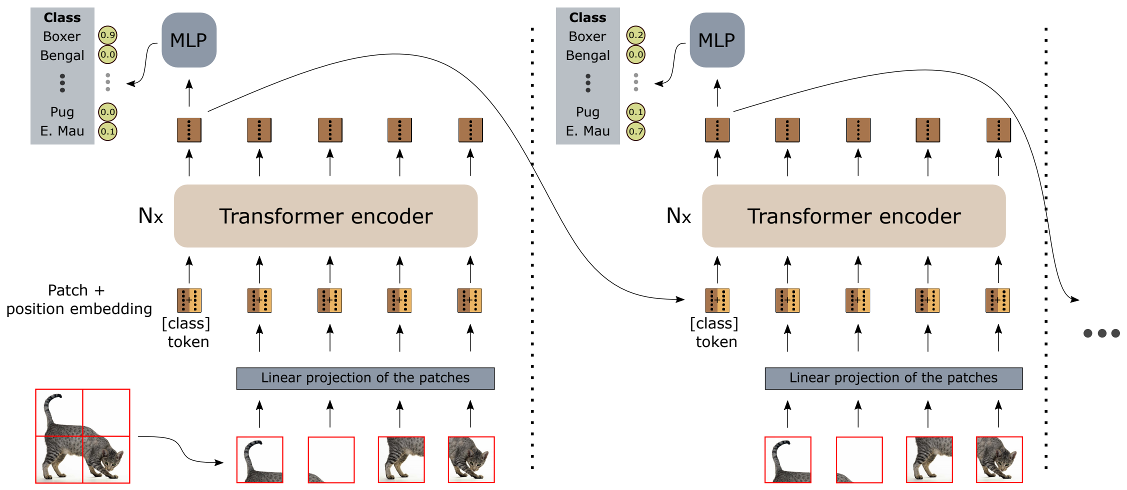


Figure 3.10: The first and the second iteration from the RecViT classification process. After an iteration is computed, the resulting class token representation is used in the next iteration, facilitating the top-down information flow.

some sort of expectation about the output class. Then, the network gains an opportunity to "reconsider its decision" by seeing the input image a couple more times, possibly focusing on more class-specific details of the input, similar to the enhanced CNN by Stollenga et al. (2014).

3.2.2 RecViT variants and data

We train RecViT using the method commonly used for recurrent NNs, which is back-propagation through time. It is basically standard back-propagation, but through the entire unrolled computational graph. Depending on outputs from which iterations we considered "important" (i.e., from which iterations we back-propagated the errors), we distinguish two training methods:

- **Method 1 (M1)**: this method computes the loss using only the classification in the last iteration, which emphasises the idea of "reconsidering" the output. It does not matter if the classifications in the non-final iterations are incorrect, the network is not penalised for that.

- **Method 2 (M2)**: in this method, we feed the class token representation to the MLP in each iteration and collect the outputs. The loss is computed as the average of losses from predictions across all iterations. This approach focuses more on the fact that, to provide some sort of "expectation" in the top-down connection, even the preliminary outputs should be somewhat correct.

We also experiment with applying transformations to inputs through iterations. There is a biological motivation behind this decision. As shown in Elsayed et al. (2018), humans are also susceptible to AEs if they have only limited time to see them. So, "seeing" the input image multiple times under some subtle transformations is more realistic (more similar to how humans process visual inputs when they process them correctly) and might help reduce the error rate. Therefore, in our experiments, we compare four input transformation strategies:

- **Vanilla**: the Vanilla strategy does not apply any transformations to the inputs. The same image is used in every iteration. This serves as the baseline for comparison with other strategies.

- **Random Transform (RandT)**: this strategy simulates the dynamic conditions of perceiving a moving object, or a stationary object perceived by a moving observer. For a given input, we randomly sample parameters for translation, rotation, and scaling, which are composed together to form a single transformation. The input in the first iteration is the original image, inputs for subsequent iterations are created by applying the combined transformation to the input from the previous iteration. Thus, the input image becomes progressively more modified.

- **Blur**: this strategy is similar to the previous one, but instead of applying a transformation composed of translation, rotation, and scaling, the inputs across iterations become increasingly blurred by Gaussian blur, which simulates focusing on different levels of details in the input image.

- **Inverse Blur (InvBlur)**: this strategy uses the same inputs as the Blur version, however, their order is inverted. The most blurred one is used in the first iteration, while the original image is in the last. Thus, this reversed order gives the greatest importance to the unmodified image.

As a basis for our implementation, we use the ViT-tiny (Steiner et al., 2022) implemented in the timm library (Wightman, 2019), which was pre-trained on the ImageNet-21k dataset and then fine-tuned on ImageNet-1k. In our experiments, we further fine-tune the models on the CIFAR-10 and Oxford-IIIT Pet (PET) datasets.

To analyse the robustness of the various RecViT variants, along with the robustness of the standard ViT, we opt to use AEs generated using transfer attack[1]. We do not use standard gradient-based white-box attacks on RecViT, a recurrent NN, because they could suffer from the vanishing/exploding gradients problem. Moreover, we do not use ViT models as the source for transfer attack, as in that case, the source would be much more similar to ViT than RecViT. It was shown before that AEs tend to transfer better between similar architectures (Mahmood et al., 2021). Therefore, using

[1]AEs were generated using the Adversarial Robustness Toolbox Python library (Nicolae et al., 2018).

ViT as the source would likely skew the results in favour of RecViT. Eventually, we chose to use three different CNN architectures — AlexNet, VGG-11, and ResNet-18 — as the source networks to generate AEs. Untargeted PGD was used on each of them while setting $\epsilon \in \{0.01, 0.02, ..., 0.2\}$ for CIFAR-10 and $\epsilon \in \{0.007, 0.022, ..., 0.202\}$ for PET to obtain more varied results. When evaluating the robustness on these AEs, we specifically focus on those which successfully fooled all three source networks. We denote them as cross-validated (C-V). It was hypothesised by Liu et al. (2017) that AEs capable of fooling multiple different models are more likely to transfer to another one. Therefore, this subset of our AEs is expected to be the most problematic for the tested models.

3.2.3 Results

In the first stage of our experiment, we aimed to compare the two training methods and four input transformation strategies, both in terms of clean accuracy and robustness. For each combination of training method, input transformation, and number of iterations $k \in \{2, 3, 4\}$, we trained 5 independent runs to obtain more statistically robust results. As this created 24 combinations in total, in this stage, we only used the PET dataset. We also trained 10 runs of standard ViT-tiny for comparison.

The results were the most interesting mainly from the point of view of different input transformations. While Vanilla and RandT achieved practically the same clean accuracy as the baseline ViT-tiny, their robustness (accuracy on AEs) was slightly elevated. Blur and InvBlur were noticeably less accurate on the clean data (2–3% drop), but their robustness was significantly higher. The highest robustness was by far achieved by the InvBlur strategy ($\approx$ 8% gain in comparison with ViT-tiny). Regarding the different values of k and different training methods, the results were pretty inconclusive, though M2 seemed to be slightly more robust. Therefore, in the next stage of our experiment, we only focused on the InvBlur strategy, while still analysing all $k \in \{2, 3, 4\}$ and both training methods.

In the second stage of our experiment, we trained 5 runs for each combination of $k \in \{2, 3, 4\}$ and two training methods on the CIFAR-10 dataset. On this dataset, the superiority of M2 was way more pronounced. For all values of k, the robustness of M1 was on par with the robustness of ViT-tiny. However, the robustness of M2 was higher by more than 20% while losing only 2–3% in the clean accuracy.

In this stage, we also performed an additional comparison to assess the effect of blurring alone. For that, we trained ViT-tiny on inputs blurred in the same way as in the InvBlur strategy. As the standard ViT cannot take multiple inputs in a single forward pass, during training, the inputs were sampled randomly from among all considered levels of blurring. In testing, the output class was determined by averaging

Table 3.1: Comparison of accuracy on the test-set and C-V AEs of RecViT InvBlur models with ViT-tiny and ViT Blur models trained on the PET dataset.

	Test-set	C-V AEs
RecViT variant	Mean±Std	Mean±Std
ViT-tiny	88.80±0.57	10.66±1.42
InvBlur M1, k=2	86.99±0.98	18.93±6.63
InvBlur M1, k=3	86.32±1.25	14.83±5.48
InvBlur M1, k=4	85.33±2.74	10.58±2.16
InvBlur M2, k=2	85.27±1.28	20.54±2.15
InvBlur M2, k=3	78.09±7.14	21.83±4.91
InvBlur M2, k=4	74.31±5.54	22.36±2.26
ViT Blur k=2	86.21±0.83	18.20±1.25
ViT Blur k=3	79.19±1.50	15.33±2.03
ViT Blur k=4	68.45±3.37	12.71±2.21

Table 3.2: Average clean accuracy and robustness (in %) of the top 3 runs (sorted according to the test-set accuracy) from InvBlur RecViT models trained on the PET dataset.

	Test-set	C-V AEs
M1, k=2	88.13	28.44
M1, k=3	87.68	21.11
M1, k=4	88.15	12.47
M2, k=2	87.14	24.02
M2, k=3	85.59	24.28
M2, k=4	84.01	26.53

the logits from forward passes across all blurring levels. We refer to networks trained and evaluated in this way as ViT Blur.

The results for networks trained on PET are in Tab. 3.1. For each value of k, InvBlur achieved comparable or higher accuracy than ViT Blur on both test-set examples and C-V AEs. However, after closer inspection, we noticed quite high values of standard deviation in the accuracies of InvBlur models. Therefore, in the next stage, we analysed the models on a run-by-run basis.

In the third stage of the experiment, we were interested in analysing the correlation between clean accuracy and robustness. If it were strong and negative, it would hint at a robustness–accuracy trade-off. On the other hand, positive correlation could mean that the training was unstable and some runs failed to achieve optimal performance. Sorting the runs according to the clean accuracy and averaging the results for the top three runs revealed that it is probably the second case. As we can see in Tab. 3.2, runs with higher clean accuracy achieved also higher robustness. To gain even better insight, we trained additional 10 runs of the InvBlur M2 $k = 3$ model, as its accuracy had the highest standard deviation. The results are visualised in Fig. 3.11, along with ViT Blur models, for a better comparison. We can clearly see a high positive correlation (0.81 Pearson correlation coefficient) between clean accuracy and robustness.

Given these results, we are optimistic about the performance of InvBlur RecViT models. Our future research should focus on mitigating the training instabilities, possibly by performing an extensive hyperparameter search. Eventually, even if the search

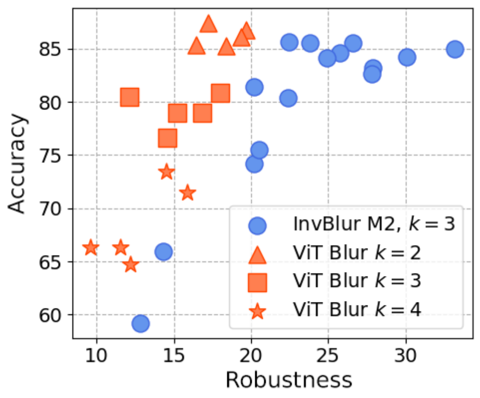


Figure 3.11: Visualisation of the robustness vs accuracy of all individual InvBlur RecViT M2 k=3 runs, compared to ViT Blur models with various k.

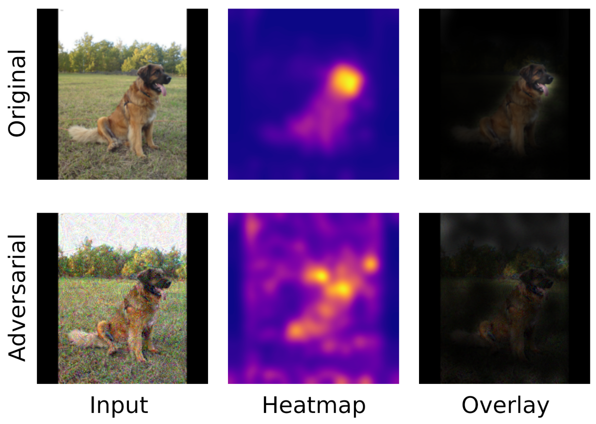


Figure 3.12: Comparison of attention maps and their overlap with the input for a clean image and its respective AE on the 8th layer of a Vanilla RecViT, trained with M1 and k=3.

failed at finding some more stable combinations of hyperparameters, the problem has a simple, though time-consuming, workaround by training more independent runs and using those that have the best test-set accuracy.

Regarding the robustness of RecViT, our future work should cover a more comprehensive analysis including adaptive white-box attacks and a comparison with other blurring-based defences, such as feature denoising. We are also planning to evaluate the robustness of RecViT trained with a modified adversarial training, where multiple AEs would be presented to the network across iterations.

In the last stage of our experiment, we took advantage of the fact that the self-attention mechanism used in ViT (and, subsequently, RecViT) provides us with an inherent explainability of the models. The attention scores computed in Eq. 1.9 can be visualised on a patch-wise basis. These can be upscaled and blurred to obtain more visually pleasing per-pixel attention maps.

When analysing the attention maps of our models, we noticed some differences between the clean and adversarial inputs, as illustrated in Fig. 3.12. It is a well known fact that AEs generated with standard attacks have severe impact also on saliency maps generated by post-hoc explanation methods (Kotyan and Vargas, 2021). While there were some successful efforts to develop adversarial attacks that change the NN prediction while keeping the saliency maps of post-hoc methods intact (Zhang et al., 2020), we are not aware of any similar attack devised for inherent explanations, such as attention maps. Therefore, we consider it an interesting open question for possible future research.

Moreover, as it is possible to compute the attention maps for each iteration in the RecViT, each transformer encoder in every iteration, and each attention head in every transformer encoder block, this results in thousands of attention maps that need to be analysed. Therefore, a rigorous analysis requires an effective numerical way of evaluating the quality of attention scores. We hope, that this problem will be possible to solve on the PET dataset, thanks to the provided trimap segmentations, which can be used as examples of the “ideal” attention maps. We published some preliminary results using the cosine similarity in Pócoš et al. (2024b), while a different method was presented in Pócoš et al. (2024a). However, none of them provided satisfactory results, therefore, this is another open problem for our future work.

Chapter 4

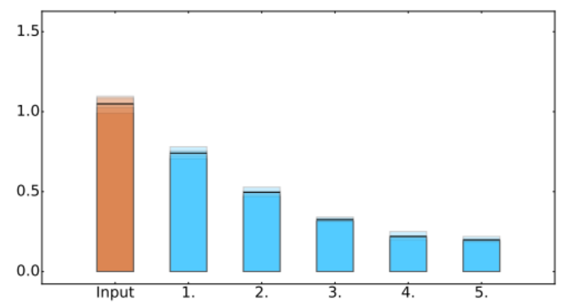
Examining proximity of adversarial examples to data manifolds

As we mentioned earlier, the question of whether AEs lie on the original data manifolds is one of the most important. However, so far, the research has focused mostly on the manifolds in the input image space. As most NNs can be viewed as a sequence of transformations between adjacent network layers, another interesting approach is the analysis of the data layer-by-layer. What happens with an AE at the beginning of a DNN? What at the end? At which layer does the AE activation diverge so far from the original data activations that it eventually becomes misclassified?

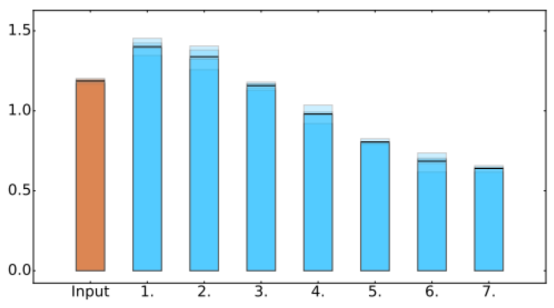
4.1 Manifold disentanglement hypothesis

Analysing such layer-by-layer development of activations is not completely off-topic even without taking AEs into account. Considering only the original data, one of the interesting theories is the *manifold disentanglement hypothesis* (Brahma et al., 2015). It states that the input data belonging to some class (such as, for example, all possible images of cats) form a low-dimensional manifold in the input space. Manifolds of individual classes are highly entangled and hard to separate. However, as they are gradually transformed by individual NN layers more and more, the manifolds disentangle and become lower-dimensional, up to the point that at the penultimate layer, which is followed only by the last linear transformation and softmax, they are (almost perfectly) linearly separable.

Our research group first inspected this hypothesis in Kuzma and Farkaš (2019). The authors assessed the degree of entanglement by computing the *embedding complexity* — the error rate with which it is possible to project the data activations into lower-dimensional space. Multiple projection methods were tried, and t-SNE (t-distributed Stochastic Neighbourhood Embedding) (Van der Maaten and Hinton, 2008) was found



(a) Embedding complexity through layers of 5-hidden-layer MLP trained on MNIST.



(b) Embedding complexity through layers of 7-hidden-layer MLP trained on SVHN.

Figure 4.1: Development of embedding complexity throughout individual layers of MLPs trained on the MNIST and SVHN datasets. Images adapted from Kuzma and Farkaš (2019).

to work best. This method uses the Kullback–Leibler (KL) divergence to measure the quality of the projection. The same measure was also used to quantify the embedding complexity. To also provide the option of visualising the results, the projections considered were always into 2D space.

As the paper also analysed the effect of NN architecture on the manifold unfolding, multiple different networks were examined. All of them were simple MLPs with 1–7 hidden layers, each consisting of 100 neurons. Every layer except the last one employed the same activation function, one of logistic sigmoid, hyperbolic tangent, ReLU, or softsign, which is defined as $\text{softsign}(x) = x/(1 + |x|)$. This resulted in 28 different combinations in total. These architectures were trained on MNIST and SVHN. The results supported the manifold disentanglement hypothesis, as the embedding complexity consistently decreased throughout the trained networks, regardless of the depth of the network or the used activation function. The greatest differences were observed between the two used datasets, as can also be seen in Fig. 4.1. Networks trained on the MNIST dataset exhibited a monotonous decrease of embedding complexity through network layers, whereas in the case of SVHN, the complexity increased in the first 2–3 layers and then decreased steadily.

Due to the observed differences between the two used datasets, in Pócoš et al. (2021), we extended the previous research by analysing two more datasets — FMNIST and CIFAR-10. Networks trained on FMNIST used the same architecture as described above (MLPs with 1–7 hidden layers, 100 neurons each), while for CIFAR-10, to achieve acceptable performance, we used a convolutional neural network comprised of 1–4 VGG-type blocks (convolution, convolution, pooling), followed by a single 100-neuron fully connected layer. As the results in Kuzma and Farkaš (2019) did not seem to depend on the choice of activation function, in our work, we chose to only use ReLU.

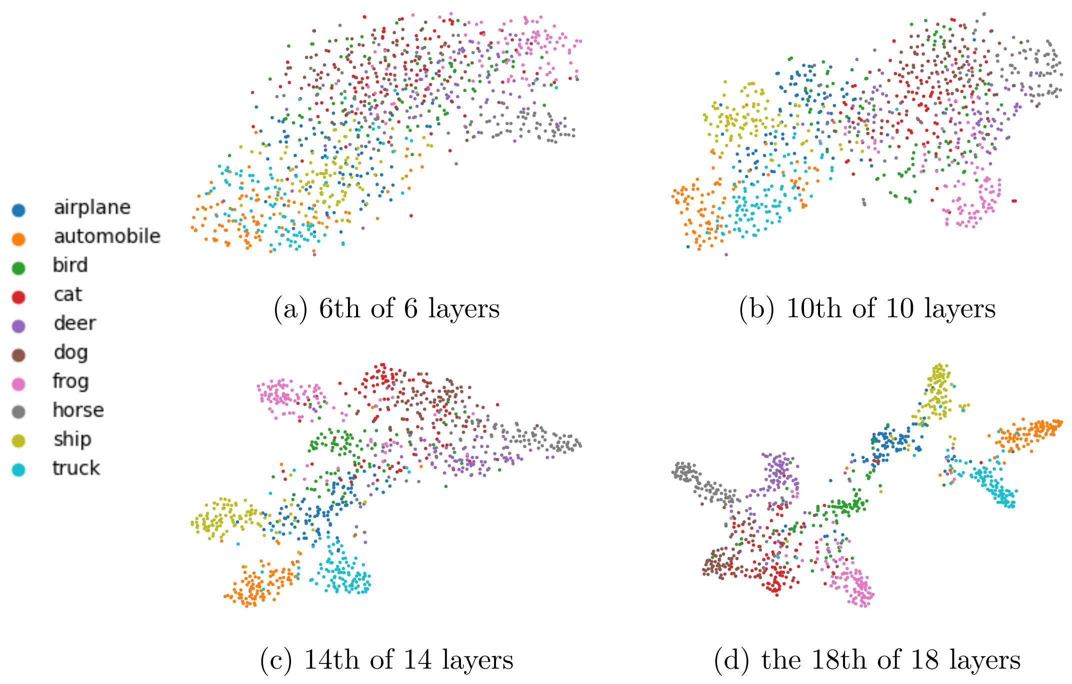


Figure 4.2: Visualisations of t-SNE projections computed on the last hidden layers of networks containing 1–4 VGG-type blocks[1]. More blocks are beneficial for better unfolding of the classes, which correlates with higher classification accuracy.

Instead, to provide a broader analysis, we opted to employ two additional methods of measuring the entanglement.

Additional t-SNE projections First, we analysed the two additional datasets using t-SNE. While Kuzma and Farkaš (2019) only evaluated the KL divergence of projections into 2D, we also explored the possibility of projecting the activations into 3D space, and confirmed that the embedding complexity still decreases through the network layers. Moreover, analysis of networks trained on CIFAR-10 showed a clear correlation between the network performance and the quality of the learned representations on the last hidden layer. Deeper networks attained higher accuracy while also learning representations that led to better class separability. This can be seen in t-SNE projections visualised in Fig. 4.2.

Eigenvalues analysis Because t-SNE is based on stochastic optimisation, and the resulting projection heavily depends on the choice of parameters, we also decided to verify the manifold unfolding using two additional, deterministic methods. The first of them performs PCA and counts how many eigenvalues are needed to explain 95% of the variance in the data. Since we were interested in the complexity of manifolds

[1]When adding one VGG-type block, the number of layers increases by 4, as in the implementation, dropout (included in each block) works as a stand-alone layer.

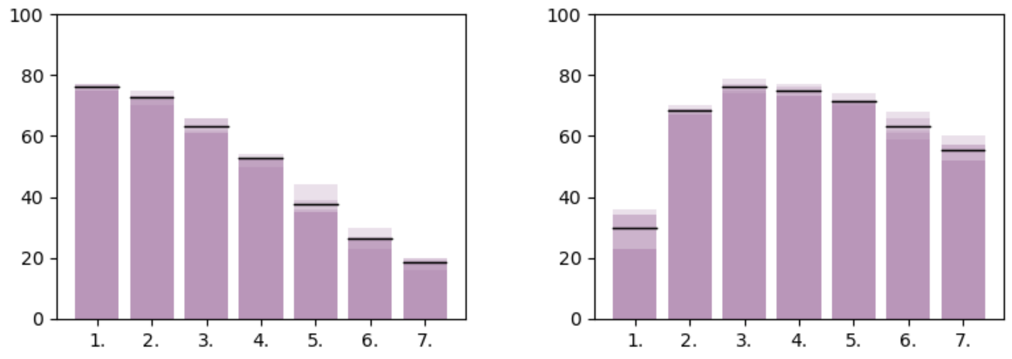


Figure 4.3: Development of the number of eigenvalues explaining 95 % of the variance in activations of data from MNIST class 5 (left) and SVHN class 7 (right). In each case, a network with 7 hidden layers was used, trained on the respective dataset.

of individual classes, PCA was also computed on a per-class basis. Once again, the results depended mainly on the dataset used. In the case of MNIST, the number of eigenvalues decreased consistently, while for SVHN, it increased at first and only decreased afterwards. This can be seen in Fig. 4.3. The results for FMNIST were quite class-dependent. Easily separable classes that were classified with higher accuracy exhibited a similar trend as MNIST, while more difficult classes behaved more similar to SVHN. For a more in-depth analysis of the individual FMNIST classes, see the original paper (Pócoš et al., 2021). We did not use this method to analyse the networks trained on CIFAR-10, as due to the use of convolutions and pooling, the dimensionality of hidden layers varies, which also affects the results of PCA and forbids any meaningful layer-by-layer comparison.

Inter-class vs intra-class distances The second additional method is based on computing the ratio of mean inter-class and intra-class Euclidean distances of class-specific activations of the network layers. For a given layer, let's denote the vector of

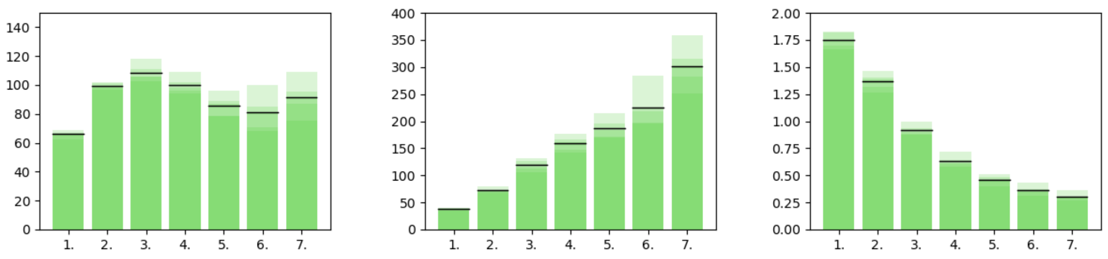


Figure 4.4: Development of the average intra-class distance (left), inter-class distance (middle), and their ratio (right) computed from the activation vectors of a model with 7 hidden layers trained on the F-MNIST dataset.

activations of the j-th sample from class c as $\boldsymbol{x}_j^c$, and let N^c be the number of samples belonging to class c. Let $\boldsymbol{m}^c$ be the element-wise mean of activation vectors of the c-th class and $\boldsymbol{M}$ the element-wise arithmetic mean of all samples. We define the mean inter-class distance as the mean squared distance of class means from $\boldsymbol{M}$ and the mean intra-class distance as the average of the squared distances of all class samples from the corresponding class mean, i.e.,

$$Dist_{\text{Inter}} = \frac{1}{k}\sum_{c=1}^{k} \|\boldsymbol{m}^c - \boldsymbol{M}\|_2^2, \qquad Dist_{\text{Intra}} = \frac{1}{k}\sum_{c=1}^{k} \frac{1}{N^c}\sum_{j=1}^{N^c} \|\boldsymbol{x}_j^c - \boldsymbol{m}^c\|_2^2, \tag{4.1}$$

where k is the number of classes. As the comparison of Euclidean distances in spaces with non-matching dimensionalities is inconsistent, we again could not analyse the networks trained on CIFAR-10. Out of the remaining three datasets, results from FMNIST have been the most interesting. Therefore, for brevity, we chose to employ this method only on the networks trained on FMNIST. The results showed that while the intra-class distances did not exhibit any clear trend, the inter-class distances tended to grow larger towards the final layers of the network. Therefore, their ratio steadily decreased, as can be seen in Fig. 4.4.

4.2 Relationship between data manifolds and AEs

After empirically confirming the manifold disentanglement theorem using three different methods, we moved towards analysing the relative position of AEs with respect to the clean data manifolds. The experiment and results presented in this section were published in Bečková et al. (2022); Pócoš et al. (2022).

4.2.1 Experimental setup

For our experiments, we used three different NN architectures:

- A fully connected network with two hidden layers per 128 neurons with ReLU trained on the MNIST dataset (further referred to as MNIST FC),
- a CNN with two convolutional layers with 16 channels, pooling and ReLU trained on the MNIST dataset (MNIST Conv.),
- and a CNN with three VGG-type blocks and a 256-neuron fully connected layer with ReLU trained on CIFAR-10.

This way, we could compare a fully connected to a convolutional network on the same dataset, as well as compare results on a simple to a slightly more challenging dataset.

As Stutz et al. (2019) suggested that robustness against on-manifold AEs is basically generalisation, we opted to analyse the more challenging, off-manifold AEs. To provide a thorough analysis, we investigated four different types of AEs, each crafted considering a different L_p norm[2], along with two types of rubbish class examples:

- **L_0 norm**: To generate AEs while minimising the L_0 norm, i.e., the number of modified pixels, we slightly refined our approach from Bečková et al. (2020). The attack works iteratively, always modifying the pixel for which the partial derivative of loss is the greatest. Then, grid search is used to determine its final value. The attack stops as soon as misclassification occurs.

- **L_1 norm**: AEs with perturbation of minimal L_1 norm were created with the untargeted EAD attack.

- **L_2 norm**: To craft AEs with perturbations optimised in the L_2 norm, we used the untargeted version of the CW attack.

- **L_∞ norm**: AEs with perturbations measured in L_∞ were created with the untargeted PGD attack. However, as this attack works by constraining the magnitude instead of minimising it, to get more diverse AEs, we set the constraint ϵ to multiple values: $\epsilon \in \{0.01, 0.02, \ldots, 0.15\}$ for MNIST and $\epsilon \in \{0.01, 0.02, \ldots, 0.05\}$ for CIFAR-10.

- **RC random uniform**: The first type of RC examples, from now on referred to as RC$_{rnd}$, was generated by running targeted PGD, starting from a random isotropic noise (i.e., value of each pixel was chosen uniformly from all admissible values). For greater variability, we chose to craft targeted RC examples, the same number of them for each class. The reason was, that most noisy images (before applying PGD to them) were classified as the same class.

- **RC from distribution**: RC examples of the second type, denoted as RC$_{distrib}$, were created using the targeted PGD in the same way. However, in this case, the starting image was created differently. The value of each pixel was picked at random, from all values appearing in the dataset at the position of that pixel. This resulted in RC examples that are still rubbish (not recognisable as any class), but "harder" in the sense of being more similar to the original data distribution.

As explained earlier, optimising the perturbation magnitude in different norms produces qualitatively different AEs. Tab. 4.1 shows average magnitudes of the resulting perturbations measured in different norms. One can notice that each norm is indeed minimised by the attack specifically devised for it. A visual comparison of all 6 different types of crafted malicious data is in Fig. 4.5. One of the main goals of this experiment was to compare the different types of AEs and their effect on the trained NNs.

[2]All AEs except those optimised in the L_0 norm were generated using the Adversarial Robustness Toolbox Python library (Nicolae et al., 2018).

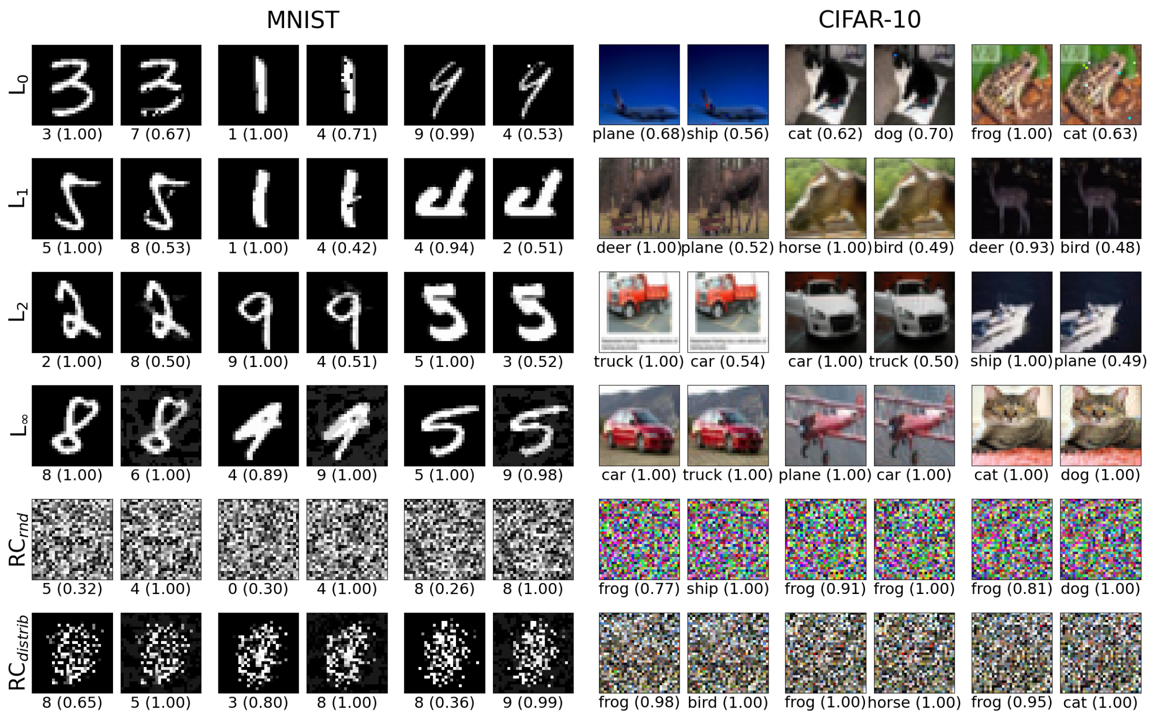

MNIST
CIFAR-10
L0
L1
L2
L∞
RCrnd
RCdistrib
3 (1.00) 7 (0.67) 1 (1.00) 4 (0.71) 9 (0.99) 4 (0.53) plane (0.68) ship (0.56) cat (0.62) dog (0.70) frog (1.00) cat (0.63)
5 (1.00) 8 (0.53) 1 (1.00) 4 (0.42) 4 (0.94) 2 (0.51) deer (1.00) plane (0.52) horse (1.00) bird (0.49) deer (0.93) bird (0.48)
2 (1.00) 8 (0.50) 9 (1.00) 4 (0.51) 5 (1.00) 3 (0.52) truck (1.00) car (0.54) car (1.00) truck (0.50) ship (1.00) plane (0.49)
8 (1.00) 6 (1.00) 4 (0.89) 9 (1.00) 5 (1.00) 9 (0.98) car (1.00) truck (1.00) plane (1.00) car (1.00) cat (1.00) dog (1.00)
5 (0.32) 4 (1.00) 0 (0.30) 4 (1.00) 8 (0.26) 8 (1.00) frog (0.77) ship (1.00) frog (0.91) frog (1.00) frog (0.81) dog (1.00)
8 (0.65) 5 (1.00) 3 (0.80) 8 (1.00) 8 (0.36) 9 (0.99) frog (0.98) bird (1.00) frog (1.00) horse (1.00) frog (0.95) cat (1.00)

Figure 4.5: An illustration of all 6 types of adversarial data created for the convolutional networks trained on MNIST and CIFAR-10 respectively (columns). For each attack (rows), three pairs of original (left) vs adversarial (right) images are shown, along with the corresponding predicted classes and output probabilities.

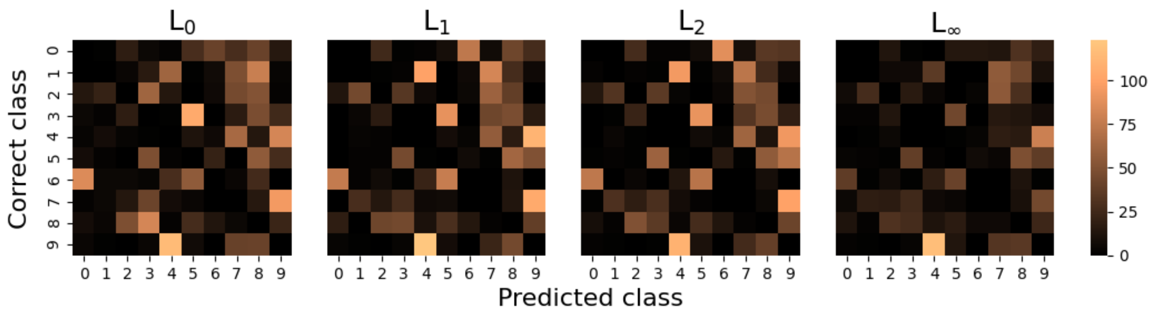


Figure 4.6: Visualisation of the number of generated AEs per each original–predicted class pair for all attack types on the MNIST Conv. network. We can notice that the counts are similar across all used attacks.

Table 4.1: Average magnitudes of adversarial perturbations crafted with different attacks (rows), measured in different norms (sub-columns), for all three tested networks (columns).

	MNIST FC				**MNIST Conv.**				**CIFAR-10**			
	L_0	L_1	L_2	L_∞	L_0	L_1	L_2	L_∞	L_0	L_1	L_2	L_∞
Our (L_0)	**10.42**	8.50	2.69	1.00	**13.14**	9.91	2.86	1.00	**9.78**	10.01	2.19	0.81
EAD(L_1)	51.88	**8.28**	1.68	0.73	42.14	**7.82**	1.83	0.84	66.34	**1.97**	0.29	0.11
CW(L_2)	286.52	13.21	**1.13**	0.26	180.21	11.83	**1.38**	0.43	428.98	3.85	**0.21**	0.04
PGD(L_∞)	493.27	47.06	2.21	**0.11**	516.68	49.28	2.35	**0.12**	1022.81	76.83	1.46	**0.03**

4.2.2 Manifold projection method

The most simple way to study the proximity of AEs to data manifolds would be to compute the average distances between AEs and the clean data. However, as already noted before, this approach is not viable for networks with varying hidden-layer dimensionalities, because the standard Euclidean distance is dimensionality-dependent. To provide a more consistent comparison regardless of the used NN architecture, we proposed a novel algorithm based on projecting the AEs onto the class-specific manifolds.

Taking inspiration from Stutz et al. (2019), we used the convex hull of the k nearest neighbours as an approximation of the data manifold. To study the activations of AEs, we project them onto manifolds approximated in this way. In general, such projection is computed as follows: considering a vector $\boldsymbol{a}$ and its nearest neighbours $\boldsymbol{a}_{i_1}, \dots, \boldsymbol{a}_{i_k}$, the projection coefficients $\gamma_1, \dots, \gamma_k$ are given by the solution of the following convex optimisation:

$$\underset{\gamma_1,\dots,\gamma_k}{\operatorname{argmin}} \Big\| \sum_{j=1}^{k} \gamma_j \boldsymbol{a}_{i_j} - \boldsymbol{a} \Big\|_2, \qquad \text{such that } \sum_{j=1}^{k} \gamma_j = 1 \text{ and } \gamma_j \geq 0. \tag{4.2}$$

The resulting projection $\tilde{\boldsymbol{a}}$ is then computed as $\tilde{\boldsymbol{a}} = \sum_{j=1}^{k} \gamma_j \boldsymbol{a}_{i_j}$. A simple function *Project*, which computes the indices of the kNN, projection coefficients and the projection itself, is given in Alg. 1.

Algorithm 1 Project

```
Require: $\boldsymbol{a}$                                          ▷ Get the input to project
Require: $\boldsymbol{A} = \{\boldsymbol{a}_1, \dots, \boldsymbol{a}_n\}$      ▷ Get data to approximate manifold
  $i_1, \dots, i_k \leftarrow$ indices of kNN of $\boldsymbol{a}$ from $\boldsymbol{A}$        ▷ Compute indices of the kNN
  $\gamma_1, \dots \gamma_k \leftarrow$ solve according to Eq. 4.2          ▷ Compute projection coefficients
  $\tilde{\boldsymbol{a}} \leftarrow \sum_{j=1}^{k} \gamma_j \boldsymbol{a}_{i_j}$                           ▷ Compute the projection
  return $(i_1, \dots, i_k)$, $(\gamma_1, \dots \gamma_k)$, $\tilde{\boldsymbol{a}}$          ▷ Return all relevant data
```

As we wanted to focus on the process of how an AE diverges from the manifold of the correct, original class c_o to the manifold of the incorrectly predicted class c_p, we

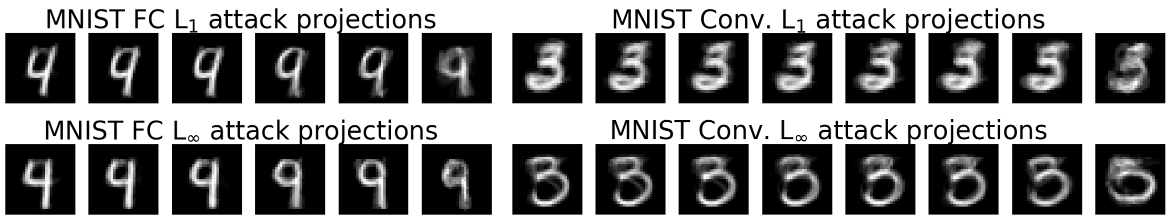

MNIST FC L_1 attack projections
MNIST Conv. L_1 attack projections
MNIST FC L_∞ attack projections
MNIST Conv. L_∞ attack projections

Figure 4.7: Projection reconstructions of activations across network layers. Visualised are the results of AEs from $Adv_{4\to9}$ computed on the MNIST FC network (left) and AEs from $Adv_{3\to5}$ computed on the MNIST Conv. network (right). The top row shows projection reconstructions of AEs computed using the EAD attack, and the bottom row shows reconstructions of PGD-generated AEs.

split all AEs based on these classes and always focused on the subset of AEs with fixed c_o and c_p, which we denote as $Adv_{c_o\to c_p}$.

To compute the proximity of an AE to the class manifolds (on a certain hidden layer), we need to collect activations $\boldsymbol{A}$ of all training inputs, and specifically, the activation $\boldsymbol{a}'$ of the AE. First, a projection of $\boldsymbol{a}'$ in the hidden space is computed, using all data regardless of their class to represent the manifold. The indices of the kNN and projection coefficients are kept to reconstruct the respective combination $\tilde{\boldsymbol{x}}$ in the input space. A visualisation of such combinations computed across all NN layers for the same AE can be seen in Fig. 4.7. Then, $\tilde{\boldsymbol{x}}$ is projected again, onto class-specific manifolds of classes c_o and c_p. Distances to these projections are then returned as the proximity scores. The entire procedure is summarised in Alg. 2.

Algorithm 2 Compute proximity

```
Require: $\boldsymbol{a}'$, $c_o$, $c_p$          ▷ Get an AE activation and its original and predicted class
Require: $\boldsymbol{X}$, $\boldsymbol{A}$                ▷ Get all training inputs and their activations
Require: $\boldsymbol{X}_{c_o}$, $\boldsymbol{X}_{c_p}$             ▷ Get all inputs from class $c_o$ and class $c_p$
  $(i_1,\dots,i_k)$, $(\gamma_1,\dots\gamma_k)$, _ ← Project($\boldsymbol{a}'$, $\boldsymbol{A}$)      ▷ Project in the activation space
  $\tilde{\boldsymbol{x}} \leftarrow \sum_{j=1}^{k} \gamma_{n_i}\boldsymbol{x}_i$          ▷ Reconstruct the combination in the input space
  _, _, $\tilde{\boldsymbol{x}}_{c_o}$ ← Project($\tilde{\boldsymbol{x}}$, $\boldsymbol{X}_{c_o}$)         ▷ Project onto original class manifold
  _, _, $\tilde{\boldsymbol{x}}_{c_p}$ ← Project($\tilde{\boldsymbol{x}}$, $\boldsymbol{X}_{c_p}$)        ▷ Project onto predicted class manifold
  return $\|\tilde{\boldsymbol{x}}_{c_o} - \tilde{\boldsymbol{x}}\|_2$, $\|\tilde{\boldsymbol{x}}_{c_p} - \tilde{\boldsymbol{x}}\|_2$        ▷ Return distances to projections
```

As in our algorithm the computation of the Euclidean distance is always performed in the input space, which has a fixed dimensionality, we could consistently compute and compare the proximity scores across all network layers. In our experiments, we picked a fixed set of AEs $Adv_{c_o\to c_p}$ and averaged the proximity scores on all layers across all AEs from this set.

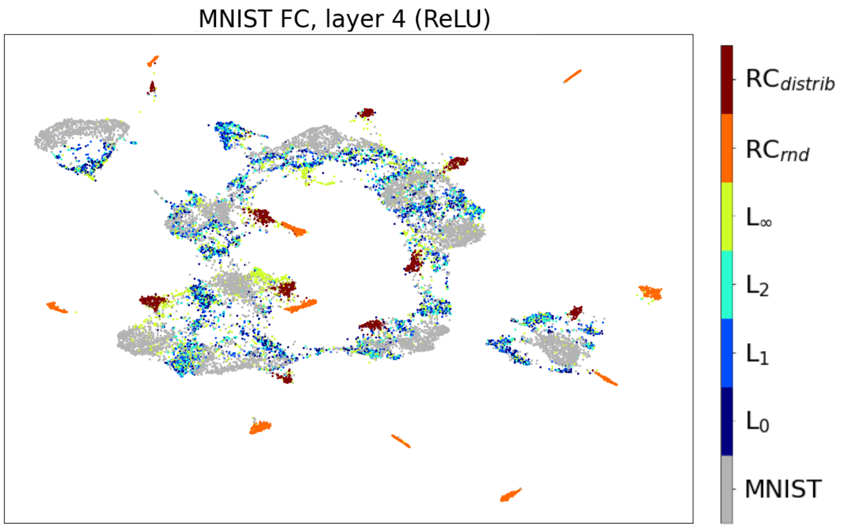


Figure 4.8: UMAP visualisation of activations of four types of AEs, two types of RC examples, and the test-set on the fourth layer of the MNIST FC network.

4.2.3 Assessment of data entanglement

We also decided to inspect the data visually, similar to Pócoš et al. (2021). Instead of using t-SNE, we chose to project the data into 2D using UMAP (Uniform Manifold Approximation and Projection) (McInnes et al., 2018) while, of course, also including the activations of AEs. An example is in Fig. 4.8.

In the UMAP visualisations, we noticed two things. Activations of RC examples were, as expected, quite disconnected from the original data activations, while those of RC$_{distrib}$ seemed to be closer to the train-set activations than RC$_{rnd}$ ones. Activations of AEs minimised in the L_∞ norm seemed quite far from the original data activations, the remaining activations of AEs were closer. Some AEs even seemed to occupy the same regions as the clean data activations. To analyse this phenomenon more closely, we performed a numerical estimation of data entanglement using the Soft Nearest Neighbour (SNN) loss (Frosst et al., 2019), which is defined as:

$$l_{\text{SNN}}(\mathbf{X}, \mathbf{y}, T) = -\frac{1}{n}\sum_{i=1}^{n} \log \left(\frac{\sum\limits_{\substack{j \in \{1,\dots,n\} \setminus i \\ y_i = y_j}} e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{T}}}{\sum\limits_{k \in \{1,\dots,n\} \setminus i} e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_k\|^2}{T}}} \right), \tag{4.3}$$

where $\boldsymbol{X}$ are inputs, $\boldsymbol{y}$ are their respective labels, n is the number of inputs and T is temperature parameter. The common approach is to adjust T to minimise the loss. This was done using the standard SGD. Due to numerical instabilities encountered when optimising the SNN loss on the network trained on CIFAR-10, we only include results for the networks trained on MNIST.

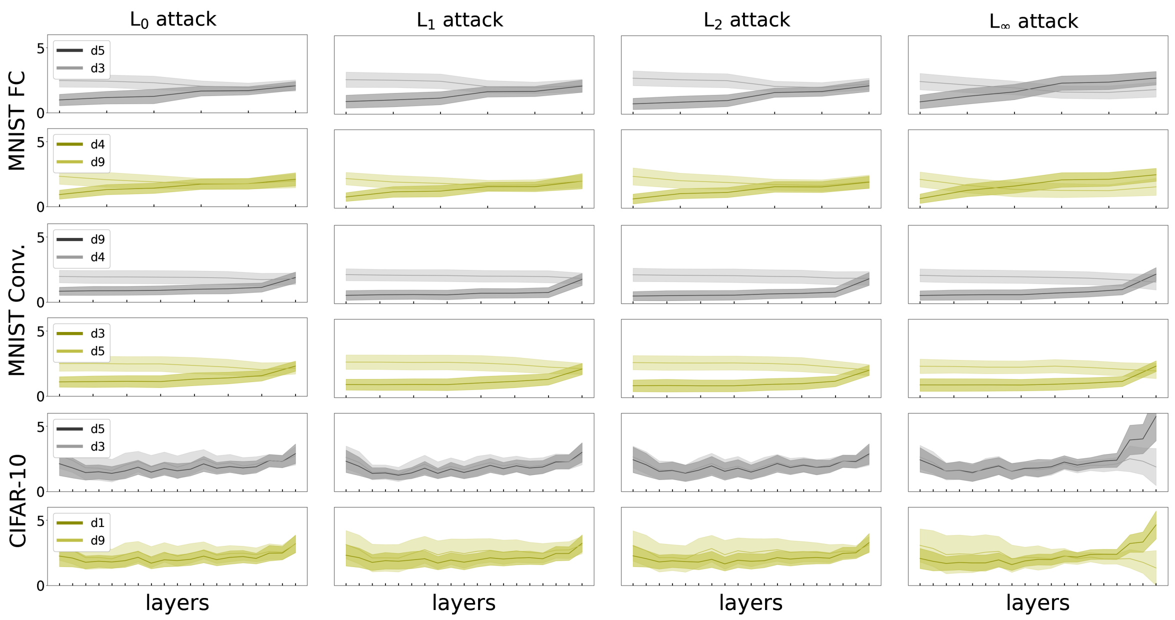


Figure 4.9: Development of proximity scores (y-axis) of the original class (dark shade) and the predicted class (light shade) across the individual network layers (x-axis). In general, we observe a trend of increase of the original class scores and decrease of the predicted class scores.

4.2.4 Results

The results of analysing AEs with our manifold projection method are visualised in Fig. 4.9. As there are 100 possible original–predicted class pairs, we only include results for two pairs for each tested architecture. The pairs were chosen according to the AE counts (for the MNIST Conv. network visualised in Fig. 4.6). To have more statistically robust results, we picked pairs with high counts.

We can notice that the shift in proximity scores from original to predicted class was different across the networks. It was slow and gradual in the MNIST FC network, however, in both convolutional networks, it happened mostly at the last few layers. Comparing different attacks, we can see that the PGD (L_∞-minimising) attack had the greatest impact on the proximity scores. Another interesting fact to note is that some AEs remained closer to the original than the predicted class even at the penultimate layer, but were still misclassified. We currently do not have an explanation for this phenomenon and find it an interesting direction for future work.

Our co-authors also proposed a second method to assess the proximity of AEs to data manifolds. Comparing both methods on the same set of AEs, they provided consistent proximity scores. For full results, we refer the reader to the published paper (Pócoš et al., 2022).

Regarding the SNN loss analysis, the results were consistent with those presented earlier. The entanglement of clean and adversarial data on the MNIST FC decreased

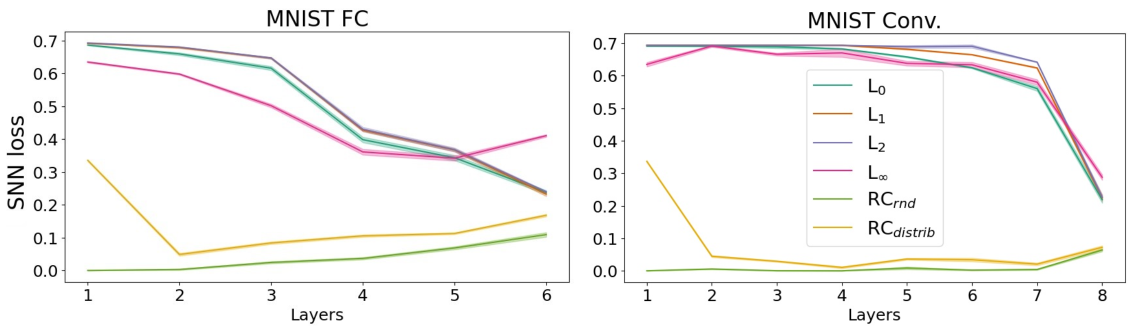


Figure 4.10: Development of the SNN loss scores between malicious inputs (AEs and RC examples) and the test-set through the layers of networks trained on MNIST.

gradually, while on the MNIST Conv. network, the drop was most visible on the last layer. RC examples were quite disentangled, with a clear difference between RC$_{rnd}$ and RC$_{distrib}$, RC$_{rnd}$ being consistently more entangled with the clean data. The development of the SNN loss across network layers is visualised in Fig. 4.10.

The most prominent result is that AEs with perturbations minimised in different L_p norms not only look different but also evoke different effects in the trained NNs. Therefore, networks robust against one type of AEs may not be robust against other types, as the cause of misclassification may differ. This is consistent with the results presented earlier, that even the adversarially trained network by Madry et al. (2018), which achieved remarkable robustness against L_∞-minimised AEs, was surprisingly weak against attacks minimised in other norms.

Due to these findings, future work in this direction should include an analysis of additional adversarial inputs, such as on-manifold AEs. Moreover, to better isolate the effect of individual attack types on the trained NNs, the analysis should also be performed with networks (possibly adversarially trained ones) that are more robust against a certain type of AEs.

Bibliography

Athalye, A., Carlini, N., and Wagner, D. (2018a). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 274–283.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2018b). Synthesizing robust adversarial examples. In *International Conference on Machine Learning*, pages 284–293.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Bečková, I., Pócoš, Š., and Farkaš, I. (2020). Computational analysis of robustness in neural network classifiers. In *29th International Conference on Artificial Neural Networks*, pages 65–76. Springer International Publishing.

Bečková, I., Pócoš, Š., and Farkaš, I. (2022). Skúmanie vzdialeností adverzariálnych vstupov k jednotlivým triedam v hlbokých neurónových sieťach. In *Kognice a umělý život 20*, pages 160–161. České vysoké učení technické v Praze.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases: European Conference*, pages 387–402. Springer.

Brahma, P. P., Wu, D., and She, Y. (2015). Why deep learning works: A manifold disentanglement perspective. *IEEE Transactions on Neural Networks and Learning Systems*, 27(10):1997–2008.

Brendel, W., Rauber, J., and Bethge, M. (2018). Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*.

Carlini, N. and Farid, H. (2020). Evading deepfake-image detectors with white-and black-box attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 658–659.

Carlini, N., Katz, G., Barrett, C., and Dill, D. L. (2017). Provably minimally-distorted adversarial examples. *arXiv preprint arXiv:1709.10207*.

Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D., and Zhou, W. (2016). Hidden voice commands. In *25th USENIX Security Symposium*, pages 513–530.

Carlini, N. and Wagner, D. (2017a). 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.

Carlini, N. and Wagner, D. (2017b). MagNet and "Efficient Defenses Against Adversarial Attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*.

Carlini, N. and Wagner, D. (2017c). Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 39–57.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. (2018). EAD: elastic-net attacks to deep neural networks via adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017a). 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.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. (2017b). Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Cohen, J., Rosenfeld, E., and Kolter, Z. (2019). Certified adversarial robustness via randomized smoothing. In *international Conference on Machine Learning*, pages 1310–1320.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Elsayed, G., Shankar, S., Cheung, B., Papernot, N., Kurakin, A., Goodfellow, I., and Sohl-Dickstein, J. (2018). Adversarial examples that fool both computer vision and time-limited humans. *Advances in Neural Information Processing Systems*, 31.

Engstrom, L., Ilyas, A., and Athalye, A. (2018). Evaluating and understanding the robustness of adversarial logit pairing. *arXiv preprint arXiv:1807.10272*.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. (2019). Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. (2017). Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Frosst, N., Papernot, N., and Hinton, G. (2019). Analyzing and improving representations with the soft nearest neighbor loss. In *International Conference on Machine Learning*, volume 97, pages 2012–2020.

Gao, Y., Shumailov, I., Fawaz, K., and Papernot, N. (2022). On the limitations of stochastic pre-processing defenses. *Advances in Neural Information Processing Systems*, 35:24280–24294.

Ghiasi, A., Shafahi, A., and Goldstein, T. (2020). Breaking certified defenses: Semantic adversarial examples with spoofed robustness certificates. In *International Conference on Learning Representations*.

Gilmer, J., Ford, N., Carlini, N., and Cubuk, E. (2019). Adversarial examples are a natural consequence of test error in noise. In *International Conference on Machine Learning*, pages 2280–2289.

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., and Goodfellow, I. (2018). Adversarial spheres. *arXiv preprint arXiv:1801.02774*.

Gong, Z., Wang, W., and Ku, W.-S. (2017). Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Gowal, S., Dvijotham, K. D., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. (2019). Scalable verified training for provably robust image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4842–4851.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. (2017a). On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*.

Grosse, K., Papernot, N., Manoharan, P., Backes, M., and McDaniel, P. (2017b). Adversarial examples for malware detection. In *22nd European Symposium on Research in Computer Security*, pages 62–79. Springer.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

He, W., Wei, J., Chen, X., Carlini, N., and Song, D. (2017). Adversarial example defense: Ensembles of weak defenses are not strong. In *11th USENIX Workshop on Offensive Technologies*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. (2017). Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. (2018). Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. *Advances in Neural Information Processing Systems*, 32.

Kannan, H., Kurakin, A., and Goodfellow, I. (2018). Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference*, pages 97–117. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kotyan, S. and Vargas, D. V. (2021). Deep neural network loses attention to adversarial images. *arXiv preprint arXiv:2106.05657*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25.

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2017). Adversarial machine learning at scale. In *International Conference on Learning Representations*.

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2018). Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC.

Kuzma, T. and Farkaš, I. (2019). Embedding complexity of learned representations in neural networks. In *28th International Conference on Artificial Neural Networks*, pages 518–528. Springer.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Cortes, C., and Burges, C. (1998b). The MNIST database of handwritten digits.

LeNail, A. (2019). NN-SVG: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747.

Li, X. and Li, F. (2017). Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5764–5772.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528.

Liu, Y., Chen, X., Liu, C., and Song, D. (2017). Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*.

Lucas, K., Jagielski, M., Tramèr, F., Bauer, L., and Carlini, N. (2023). Randomness in ML defenses helps persistent attackers and hinders evaluators. *arXiv preprint arXiv:2302.13464*.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Mahmood, K., Mahmood, R., and Van Dijk, M. (2021). On the robustness of vision transformers to adversarial examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7838–7847.

McInnes, L., Healy, J., Saul, N., and Großberger, L. (2018). UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Meng, D. and Chen, H. (2017). MagNet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. (2017). On detecting adversarial perturbations. In *International Conference on Learning Representations*.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1765–1773.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P., and Soatto, S. (2018). Robustness of classifiers to universal perturbations: A geometric perspective. In *International Conference on Learning Representations*.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). 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.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., et al. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, page 7. Granada, Spain.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436.

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I., and Edwards, B. (2018). Adversarial robustness toolbox v1.2.0. https://arxiv.org/pdf/1807.01069.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *Asia Conference on Computer and Communications Security*, page 506–519.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016a). The limitations of deep learning in adversarial settings. In *European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016b). Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy*, pages 582–597.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. (2012). Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Pócoš, Š., Bečková, I., and Farkaš, I. (2022). Examining the proximity of adversarial examples to class manifolds in deep networks. In *Artificial Neural Networks and Machine Learning*, pages 645–656. Springer Nature Switzerland.

Pócoš, Š., Bečková, I., and Farkaš, I. (2024a). Explainability of vision transformer with top-down connection. In *Cognition and Artificial Life 2024*, pages 28–29. Flow.

Pócoš, Š., Bečková, I., and Farkaš, I. (2024b). RecViT: Enhancing vision transformer with top-down information flow. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3*, pages 749–756. INSTICC, SciTePress.

Pócoš, Š., Bečková, I., Kuzma, T., and Farkaš, I. (2021). Assessment of manifold unfolding in trained deep neural network classifiers. In *Trustworthy AI – Integrating Learning, Optimization and Reasoning: First International Workshop, TAILOR 2020, Virtual Event*, pages 93–103. Springer.

Raghunathan, A., Steinhardt, J., and Liang, P. (2018). Certified defenses against adversarial examples. In *International Conference on Learning Representations*.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. *Biometrika*, 71:599–607.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Samangouei, P., Kabkab, M., and Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*.

Shen, S., Jin, G., Gao, K., and Zhang, Y. (2017). APE-GAN: Adversarial perturbation elimination with GAN. *arXiv preprint arXiv:1707.05474*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, pages 1–14.

Sitawarin, C., Golan-Strieb, Z. J., and Wagner, D. (2022). Demystifying the adversarial robustness of random transformation defenses. In *International Conference on Machine Learning*, pages 20232–20252.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. (2018). PixelDefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Steiner, A. P., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., and Beyer, L. (2022). How to train your ViT? Data, augmentation, and regularization in vision transformers. *Transactions on Machine Learning Research*.

Stollenga, M. F., Masci, J., Gomez, F., and Schmidhuber, J. (2014). Deep networks with internal selective attention through feedback connections. *Advances in Neural Information Processing Systems*, 27.

Stutz, D., Hein, M., and Schiele, B. (2019). Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987.

Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Tramèr, F. (2022). Detecting adversarial examples is (nearly) as hard as classifying them. In *International Conference on Machine Learning*, pages 21692–21702.

Tramèr, F., Behrmann, J., Carlini, N., Papernot, N., and Jacobsen, J.-H. (2020a). Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In *International Conference on Machine Learning*, pages 9561–9571.

Tramèr, F. and Boneh, D. (2019). Adversarial training and robustness for multiple perturbations. *Advances in Neural Information Processing Systems*, 32.

Tramèr, F., Carlini, N., Brendel, W., and Madry, A. (2020b). On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2018). Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness may be at odds with accuracy. In *International Conference on Learning Representations*.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154.

Wightman, R. (2019). Pytorch image models. `https://github.com/rwightman/pytorch-image-models`.

Wong, E. and Kolter, Z. (2018). Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, C., Wu, Y., Maaten, L. v. d., Yuille, A. L., and He, K. (2019). Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509.

Xu, W., Evans, D., and Qi, Y. (2018). Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society.

Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X., and Wang, T. (2020). Interpretable deep learning under fire. In *29th USENIX Security Symposium*.