

Towards Robust Deep Learning Systems

The Plasticity Hypothesis

A thesis submitted in partial fulfillment of the
requirements for the award of the degree of

MASTER OF SCIENCE

Soumadeep Saha

15MS024

under

Dr. Utpal Garain

and

Dr. Rumi De

to the

Department of Physical Sciences




INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH
KOLKATA

June, 2020

CERTIFICATE

This is to certify that the work contained in the thesis entitled “*Towards Robust Deep Learning Systems*”, submitted by *Soumadeep Saha* (*Department of Physical Sciences, IISER Kolkata*) for the award of the degree of *Master of Science* to the **Indian Institute of Science Education and Research, Kolkata (IISER Kolkata)**, is a record of bonafide research work carried out by him under my direct supervision and guidance. I considered that the thesis has reached the standards, and fulfills the requirements of the rules and regulations relating to the nature of the degree. The contents embodied in the thesis have not been submitted for the award of any other degree or diploma in this or any other university.



Date: **19.06.2020**

Place: **ISI, Kolkata**

Dr. Utpal Garain, Professor.

**CVPR Unit,
Indian Statistical Institute
203, B. T. Road,
Kolkata 700108, India.**

DECLARATION OF THE STUDENT

I hereby declare that this thesis is my own work and to the best of my knowledge, contains no materials previously published or written by any other person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at IISER Kolkata or any other educational institution, except where due acknowledgement is made in the thesis.

Soumadeep Saha

IISER KOLKATA

June, 2020

Contents

Acknowledgements	iii
Dedication	iv
Abstract	v
1 Introduction	1
1.1 The lay of the land	1
1.1.1 History	1
1.1.2 Successes	3
1.1.3 Challenges	4
1.2 What is machine learning?	4
1.2.1 Deep Learning	5
1.3 Adversarial Attacks	9
2 Adversarial Attacks	12
2.1 What are adversarial attacks?	12
2.1.1 Targeted vs Non-Targeted	13
2.1.2 Blackbox vs Whitebox	13
2.2 Popular Paradigms	15
2.2.1 L-BFGS	15
2.2.2 FGSM	16
2.2.3 Basic Iterative Methods	16
2.2.4 Other attack paradigms	17

Contents

- 2.2.5 GANs 18
- 2.3 Defenses 18
 - 2.3.1 Modifying Networks 19
 - 2.3.2 Add-on Techniques 22
 - 2.3.3 Modifying Training or Inputs 23
- 3 Cryptographic Approaches 24**
 - 3.1 Protecting the Network 25
 - 3.2 Protecting User Data 25
 - 3.3 Putting it Together 26
 - 3.4 Issues with cryptographic approaches 27
 - 3.4.1 The Obfuscation Problem 28
 - 3.4.2 Practical Issues 28
- 4 Robustness vs Accuracy 30**
 - 4.1 Motivation 30
 - 4.2 Observations 31
- 5 The Plasticity Hypothesis : Further Results 34**
 - 5.1 Robustness is Harder 34
 - 5.2 Capacity is Limited 36
 - 5.2.1 Universal Approximator Theorem 36
 - 5.2.2 Model Compression 37
 - 5.3 Plasticity Hypothesis 38
 - 5.4 Results 39
 - 5.5 Future Work 40
- 6 Conclusions 43**

Acknowledgements

I would like to express my deepest appreciation to my supervisor Professor Utpal Garain for taking me under his wing and continually instilling a spirit of inquiry. He has been a constant source of support and went out of his way to assist me when required. Without his guidance, encouragement, and inspiration this dissertation would not have been possible.

I would also like to thank my co-supervisor Professor Rumi De for her role in this dissertation. Her feedback and guidance proved to be invaluable throughout my final year at this institute.

In addition I would like to thank my lab mate Akshay Chaturvedi. He is the first person I turned to whenever I required assistance, in both technical and scholarly matters. He has been on the receiving end of a barrage of questions and has doled out immensely valuable feedback throughout my tenure at the lab. I laud him for his patience and express my deep gratitude for his guidance.

Furthermore, I am deeply indebted to my parents for all they have done and continue to do, and for supporting my academic pursuits.

Finally I would like to thank my institute as a whole; not only all the brilliant professors who have taught me all I know, but all the support staff who have made my life at the institute smooth sailing. I also thank Kishore Vigyan Pratoshana Yojna (KVPY) for conferring me with the prestige of their fellowship.

Dedication

Dedicated to the memory of Dr. Pushan Majumdar. Your attitude towards academics will always be inspirational and your brilliance will be thoroughly missed. Your untimely demise has left the world darker.

Abstract

With the advent of deep learning we have pushed the boundaries of what was computationally possible, and have made significant impacts in other fields as wide ranging as high energy physics to medicine. Certain technologies which were once considered a science fiction fever dream are commonplace, and we are certainly far from unlocking its full potential. It is now possible for a computer to analyze an MRI without the aid of a doctor, and futurists predict that cars will drive themselves in the coming decades.

However there are some looming issues which we need to resolve before we can realize our utopic visions. One of these problems is the black box nature of the algorithms. Since, we don't really know what is going on under the hood, we cannot infer causal relationships, which is of critical importance in certain applications.

Another problem that is pervasive, is that of adversarial attacks. They were first discovered by Szegedy et al. [51] in 2014 and have since become much more sophisticated and has been extended to every kind of architecture imaginable. An adversarial attack is when an adversary makes unnoticeable changes to the input, to drastically change the output of the deep learning network, often being able to choose the output in question. The prevalence of these attacks in almost all networks, and oftentimes their transferability suggests an underlying problem with how the system "learns". This learning problem and various ways to address it forms the subject matter of this dissertation.

We will first be looking at cryptographic approach to solving this issue. If we can somehow perform computations on the input and produce the desired result wherein we can mathematically guarantee no one knows the intermediate steps, our adversary will be greatly impeded in their efforts. This also has potential additional benefits such as access

control, safety, etc. However, we have found that, the cryptographic approaches, when possible at all, are limited by the raw computational capacity of our times, rendering them ineffective.

One idea proposed by Szegedy et al. [51] has showed promise in solving the adversarial attack problem, wherein we generate adversarial data and train our networks on adversarial examples in addition to samples from the input space. This "adversarial training" paradigm has been shown to be at odds with the accuracy of deep networks, and some have gone so far as to claim that this is an inevitability [47].

However, upon exploration of several architectures and their behavior under adversarial training we are led to the following hypothesis. A deep network has a finite capacity to learn, and if we wish to adversarially train a network i.e. make it more robust while maintaining accuracy we have to increase its capacity to learn (by adding more parameters, increasing depth. etc). We have found some evidence in support of this hypothesis, and it opens up many possible avenues of inquiry in the future.

1 Introduction

In this chapter we will briefly trace the history of deep learning, give a short introduction on the basics of the topic, and then discuss the problem at hand in greater detail.

1.1 The lay of the land

1.1.1 History

As computational prowess grew back from the days of vacuum tubes we have been finding more complicated problems for computers to solve. In 1949 John von Neumann suggested that the ENIAC be used to calculate the decimal expansion of π [41]. The computer calculated over 2000 digits over a weekend far out-passing anything a human might have hoped to achieve. Very soon everything involving numerical computation was handled by machines, and we started looking for more interesting problem for computers to solve. Our first instinct was to look for problems which we typically associate with the smartest of people. The poster child for exhibiting reasoning prowess became *chess*.

Big names in computation attacked this problem, including the likes of Alan Turing. By the late 1950s algorithms that could play chess were developed, but they were limited by hardware capabilities. By May 1997 Deep Blue had beaten the chess grand master Gary Kasparov. This historic event took place well over two decades ago. So where are the AI supercomputers that were promised?

It turns out that chess is only a hard problem to solve for humans, who are limited by computational and memory capacity. Chess is a rule based Markovian game where the next move can be calculated without knowledge of the history. Soon heuristics were

1 Introduction

developed that could assign scores to states of a chess board. Then it became a problem of tree traversal. Each possible move creates a branch in our decision tree and since the states can be numerically evaluated the problem turned into one of maximizing your score while minimizing your opponents. Today chess engines are so powerful that a grand master is akin to a complete beginner when compared with it.

So chess wasn't as difficult a problem to solve as was originally thought. This is indicative of the nature of progress in AI, whenever a problem initially thought of as 'intelligent' is solved, the goalpost is moved to harder problems. However, difficult problems are a dime a dozen.

Some of these problems appear to be related. We as humans can solve certain problems instinctively without even thinking about it. Like listening to someone speak and knowing what they mean, or looking at a picture and recognizing objects in it. These problems as it turns out are much harder to solve for computers. Some of the reasons are as follows.

- Solutions are context dependent and depend on a vague understanding we have. For example a hot beverage and a hot day both have the same adjective but mean wildly different temperatures.
- Either do not have well defined rules or have so many that it would be impossible to note down in a case by case basis. For example if we ask the question, "Does this picture have a bird in it?", our rule set would have to have enormous size. Penguins, ostriches and sparrows look very different but are all birds. Are figurines of birds, birds?
- Often have high level abstract details we do not fully understand. For example, how do you explain to a machine what sarcasm is?

Rosenblatt [42] is credited with coming up with the perceptron, which when put together with the back propagation algorithm [53] serves as the building blocks of the field of deep learning. However, two other advances would be required before they would prove to be ubiquitously useful like it is today. The first is the increase in computational

proceed through faster processors and numerous cores working in parallel. The second and perhaps more important advancement was our access to datasets of astronomical sizes.

Today we use deep learning to attack a diverse set of problems, including those mentioned above. Deep learning is best understood as methods to find complex non-linear functions to fit our data by optimizing an error function. We hoped a structure like this would work as it was modeled based on our understanding of how human brains work. This is why these techniques are often labelled as Artificial Neural Networks.

1.1.2 Successes

Deep learning has had major successes in recent times. Almost all state of the art computer vision technologies use deep learning today. Major strides have been made in the field of natural language processing using deep learning, and tasks like translation, captioning, etc are routinely automated, and they even outperform humans at certain tasks. Networks have been trained to draw in the style of Picasso [13] or compose like Chopin [39]. A certain medical study found that it could predict whether a person was a smoker or not based on retinal images [59]. Particle accelerators and astronomers heavily rely on deep learning to analyze their data.

Many applications have trickled down to the average consumer. The "face-unlock" feature found on most smartphones today are based on CNN-feature extractors¹. Several companies have launched pilot programmes testing the feasibility of completely autonomous vehicles.

However these problems are still tractable in some sense, where we might not be able to exactly define a solution but we know a solution when we see one. This is the regime of supervised learning wherein we have labeled datasets and we try and discover an underlying relationship which is hopefully indicative of the whole input space. There are certain sets of problems where that is not the case. For instance, we don't know what the best space station design is. This class of problems where we wouldn't know a solution

¹the technology is proprietary in most cases, but this is the best educated guess

even if we stumbled upon it somehow is much more challenging. Deep learning is helping push the boundary in this class of problems too, by modelling the often complicated action \rightarrow reward function [8]. We have recently seen Alpha-Zero beat the best human players at Go without any information save for the rules of the game [49].

1.1.3 Challenges

In spite of all its successes, there are a few major problems that require addressing to push the field further. One of the problems is that causality is near impossible to establish. The models used are of the "black box" kind. So, after we get a prediction from a model we cannot answer the question why the model made the prediction it did.

The other looming issue is that of *adversarial attacks*, which is the focus of this work, and we will be talking about it in much greater detail in the coming sections.

1.2 What is machine learning?

Machine learning is a collection of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

Machine learning is a broad umbrella term, which encompasses several algorithms and techniques, ranging from simple linear regression models, nearest neighbor algorithms, SVM, random forests, and artificial neural networks. Artificial neural networks in conjunction with *big data* (large data sets) broadly speaking constitutes deep learning. We will discuss deep learning briefly before we introduce the *learning problem*.

1.2.1 Deep Learning

Simple models like k Nearest Neighbors, Random forests or Support Vector Machines are reliant on a domain expert designing the data set. For example, we might want to predict the chance of heart disease given someone's age, weight, height and blood cholesterol level (features). We could start from the labelled data set and infer the relationship between those factors and heart disease. But how do we know we ought to concentrate on these factors, and not, for example hair color? These algorithms therefore require that the data be curated by domain experts. Often domain experts are not available. Often the feature set is subjective. Sometimes higher level abstract details are present which the domain experts don't even know to look for (as in the medical study mentioned above)[59].

This is where deep learning shines. When we have a vast quantity of raw (labelled or unlabeled) data and we can train a model on it without first deciding which features of the data is important or should be focused on. We hope that with enough examples the system will learn to discriminate the important from the unimportant and produce the desired result.

Deep learning has three major parts , the artificial neural network, the data, and the training algorithm.

An artificial neural network is a network of stacked "perceptrons". Perceptrons are the parallels of individual neurons. They have an activation represented by a continuous number (sometimes between 0 and 1) which depends on the activations of the perceptrons connected to it. This is denoted as

$$a_i^{(n)} = W_{ik}^{n,n-1} a_k^{(n-1)} + b_i^{(n)}$$

Where $a^{(0)}$ is the input and $a^{(n)}$ is the nth layer activation. $W_{ij}^{(n)}$, $b^{(n)}$ are the weights

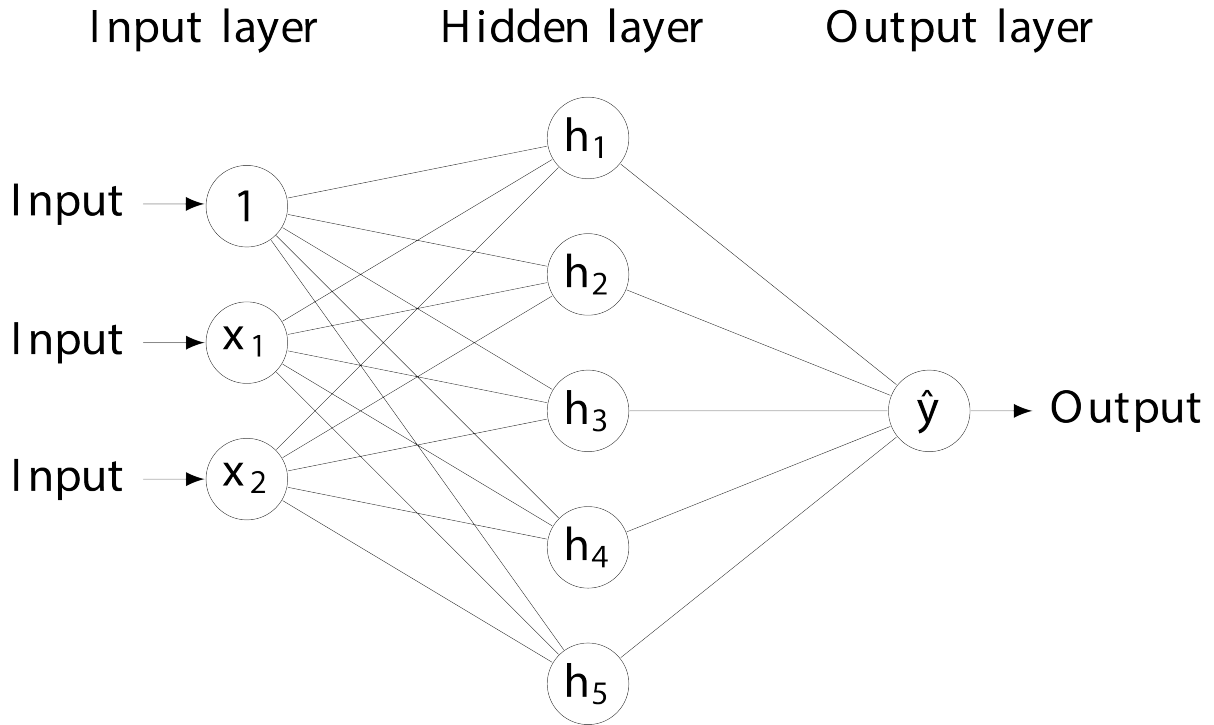


Figure 1.1: Artificial Neural Network [Source : Wikipedia]

and biases which are learned.

Algorithm 1: Back Propagation Algorithm

$(x_i, y_i) \in \mathcal{D}$ is the data ;
 η is the learning rate ;
 f_{θ_j} is the network with parameters θ_j ;
 $\ell \leftarrow \infty$, $j \leftarrow 0$;
while $\ell > \beta$ **do**
 Calculate $f_{\theta_j}(x_i)$;
 $\ell \leftarrow \ell(f_{\theta_j}(x_i), y_i)$;
 $\delta \leftarrow \nabla_{\theta_j} \ell$ # using chain rule ;
 $\theta_{j+1} \leftarrow \theta_j - \eta \cdot \delta$ #update parameters ;
 $j \leftarrow j + 1$;
end

The training protocol is described in Algorithm 1. ℓ is a loss function, it can be as simple as a ℓ_2 -norm of the calculated vs expected output, or could be a probabilistic

measure. After sufficiently many iterations, the loss of the network decreases and it gets better at predicting the output given an input. η is called *learning rate* and is a hyper-parameter describing the how big the step size is in the gradient descent step. β is an arbitrarily chosen stopping point. Generally, the inputs are split into batches so that the gradient step is not sensitive to each input.

CNN feature extractor

CNN stands for Convolutional Neural Networks. This is a special type of neural network specifically suited for images, where in each layer, the neurons of that layer are only locally connected (spatially) to the layers before it. Formally, we perform kernel convolutions with several filters (the specifics of the filter is learnt), each producing a version of the image (feature map) with specific features highlighted. Many such layers are stacked, so that more complicated patterns can be picked up on by the network. In between the Convolutional layers we have pooling layers which down sample the feature maps.

Thus, at the end of the CNN we have a set of hundreds of smaller images which only highlight the key features (see Figure 1.2). These can then be fed through a standard fully connected network, for classification related tasks or the features can be used to train networks for a wide array of tasks involving images, where these CNN features serves as a more useful embedding of the information in the image.

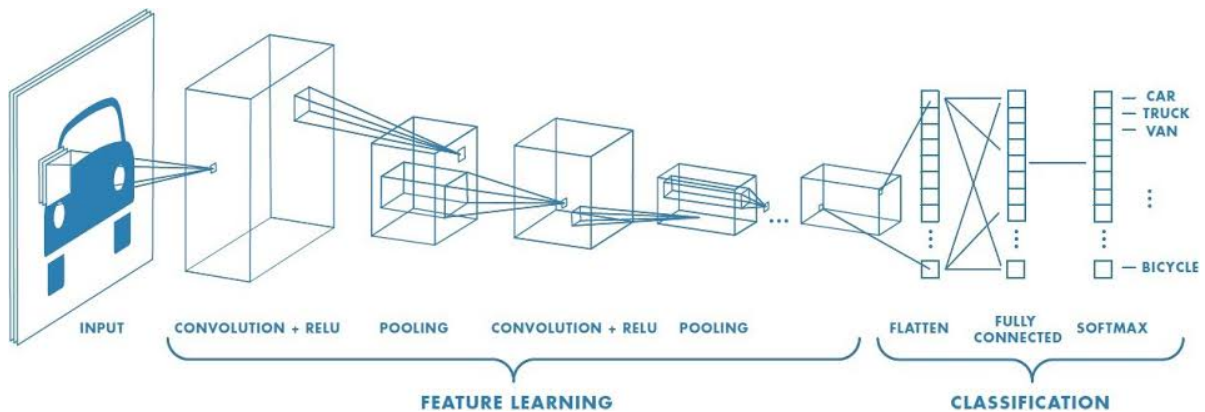


Figure 1.2: CNN [Source : towardsdatascience.com]

LSTM Cell

The basic premise of **Recurrent Neural Networks** is that standard deep neural network architectures are incapable of handling sequence/time series data. RNNs solve this problem by defining a deep neural network which has some notion of memory, which is a function of its previous inputs. Mathematically we have

$$\begin{aligned} h_t &= f_W(h_{t-1}, x_t) \\ y_t &= g_{\tilde{W}}(h_t) \end{aligned} \tag{1.1}$$

Here, x_t, y_t is the input and output at time t respectively. h_t represents the network

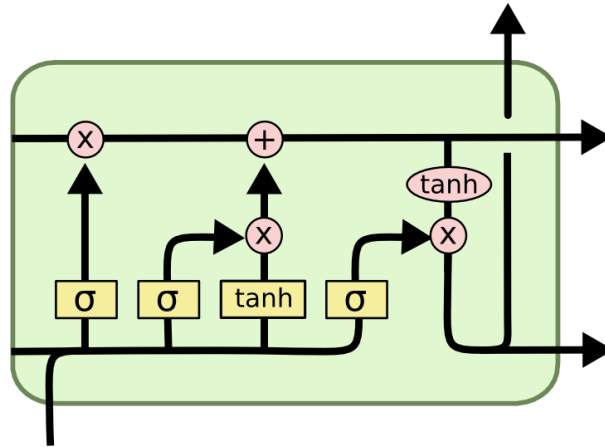


Figure 1.3: LSTM Cell [Source : Blog by Christopher Olah]

memory, and $f_W, g_{\tilde{W}}$ are learned functions with parameter set W, \tilde{W} respectively.

These networks are notoriously hard to train, because they require long sequences of back propagation through time, and normally a finite truncated version of back propagation through time is used.

Additionally, they have an exploding/vanishing gradient issue. LSTM (Long Short Term Memory Networks) were proposed in 1997 [21] in an effort to fix this issue with RNNs. The function f_W is usually a series of matrix multiplications, and while back propagating gradients, multiplying the gradients by the same matrix several times usually

leads to the gradients either growing or shrinking exponentially. To address this issue in particular, a shortcut path is added in LSTM (see Figure 1.3) for the memory, so that they aren't subjected to repeated multiplication.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \times W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \quad (1.2)$$

$$c_t = f \cdot c_{t-1} + i \cdot g$$

$$h_t = o \cdot \tanh(c_t)$$

$$y_t = \tilde{y}_w(h_t)$$

i, f, o, g are vectors called input, forget, output, gate respectively. The input and 'gate' vectors learn to decide how much of the input to take into consideration, the forget gate learns how much of the current memory to forget, and the output gate learns how much of the memory is to be exposed to the output. The control c_t avoids the problem of repeated multiplication in this architecture.

These networks are used to either embed knowledge about sequences into vectors (as in describing the quality of a sentence), turn embedded knowledge vectors into sequences (as in image captioning), or map sequences to sequences (as in translation).

1.3 Adversarial Attacks

It has been shown that imperceptible perturbations can be added to inputs to make the output of the deep learning system drastically different, and these perturbations can often be calculated to produce the adversary's desired output (see Figure 1.4). These attacks have been performed on a huge array of networks and have a near perfect accuracy (see Figure 1.5) [6].

Just as we can adjust the values of the parameters based on the derivatives of the loss

1 Introduction

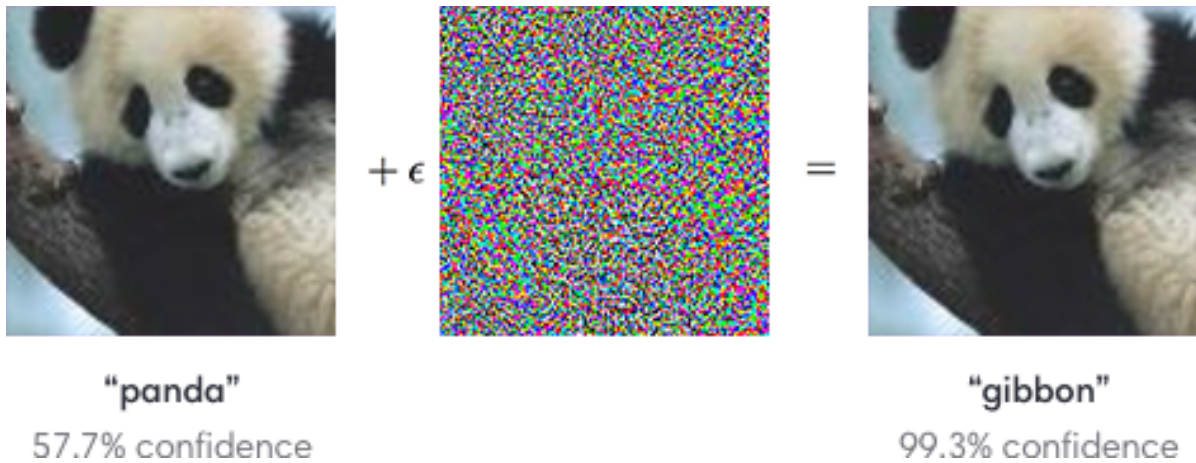


Figure 1.4: An adversarial input, overlaid on a typical image, can cause a classifier to miscategorize a panda as a gibbon. [Source : Goodfellow et al. [14]]

function we can rather fix the parameters and compute the derivative of the loss with respect to the input. If we intentionally change the input in this manner to get a desired erroneous output, we can often get the desired output with imperceptible tweaks to the input. (see Figure 1.4). This is the idea behind most of the popular attack algorithms in use today [14].

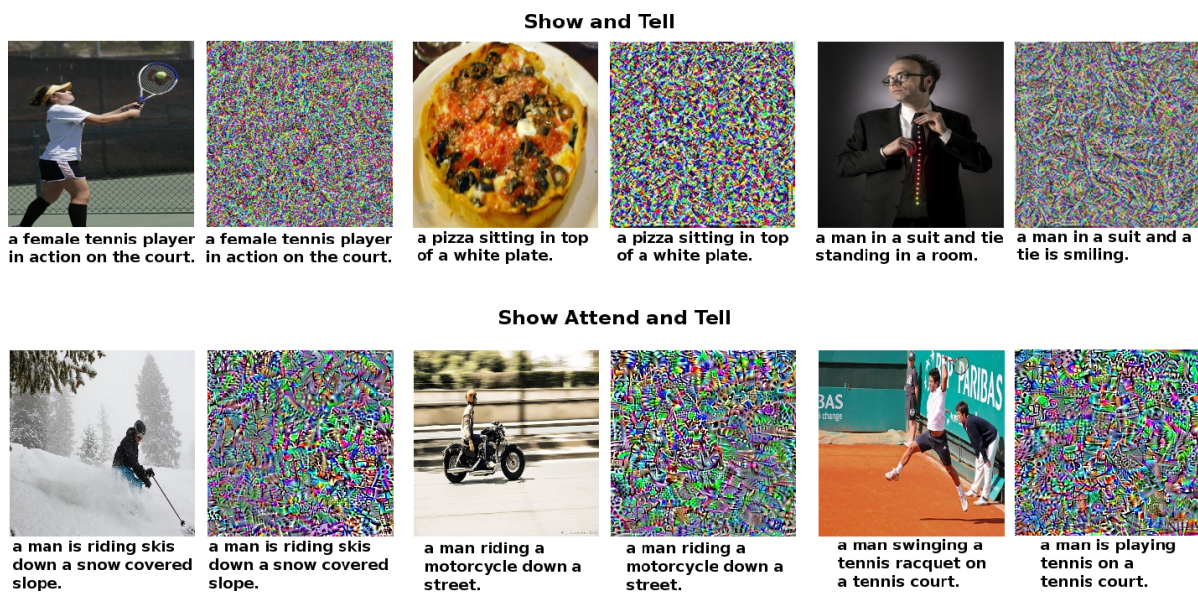


Figure 1.5: Adversarial attacks on common systems [Source : Chaturvedi et al. [6]]

This points to the existence of some fundamental underlying problems with the way

we do deep learning. It does not learn in the same sense as humans do.

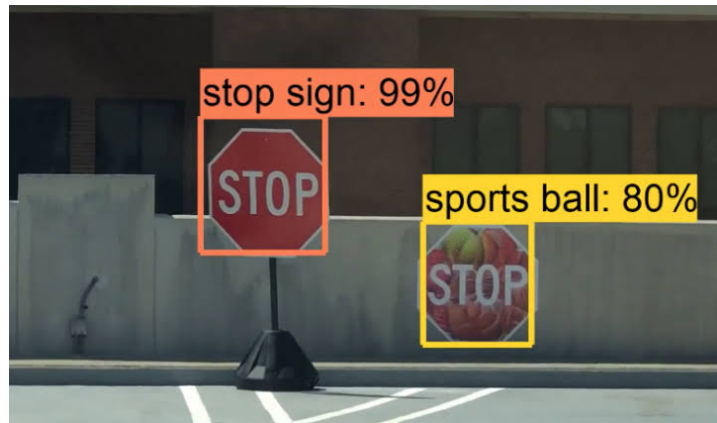


Figure 1.6: An adversarially tampered stop sign being misclassified. [Source : Eykholt et al. 2018 [12]]

This is a big problem when it comes to the widespread adopt-ability of deep learning techniques. What if an adversary could inject perturbations in a presidential speech, such that when translated by a deep network, would lead to tumult? What if someone could place a sticker on a stop sign, thus making it effectively invisible to the cameras of an autonomous vehicle (see figure 1.6)? This problem might have several unforeseen, and far reaching deleterious consequences.

2 Adversarial Attacks

In this section we will go over the basics of adversarial attacks (in the context of deep learning networks), look at specific strategies that show promise and finally look at some of the solutions to this problem that have been proposed in the literature.

2.1 What are adversarial attacks?

Let us consider a deep learning network as a function f that takes in input $x_i \in X$ and produces corresponding output $y_i \in Y$. It has several internal parameters like weights, biases, etc denoted together with the set $\theta \in \Theta$ (Θ is the space of all possible parameter values). We are optimizing on a loss function ℓ which tells us how accurate our prediction is at the current time.

$$\begin{aligned} f_\theta : X &\rightarrow Y \\ x_i &\mapsto y_i \end{aligned} \tag{2.1}$$

In this context the goal of the training procedure is

$$\operatorname{argmin}_{\theta \in \Theta} \ell(f_\theta(x_i), y_i) \quad \forall i \tag{2.2}$$

This is in general achieved by the following protocol

$$\theta_n = \theta_{n-1} - \eta \cdot \nabla_{\theta_{n-1}} \ell(f_\theta(x_i), y_i) \tag{2.3}$$

Where η is a hyper-parameter, called the learning rate. The gradients are usually computed analytically with an algorithm known as backpropagation.

The goal of an adversarial attack then is to find a perturbation δ such that

$$f_{\theta}(x_i + \delta) \neq y_i \tag{2.4}$$

There are several ways to go about this and we will discuss them in the coming sections.

2.1.1 Targeted vs Non-Targeted

In a **targeted** attack the adversary chooses the desired incorrect output, i.e. a $y'_i \neq y_i$ is chosen and the aim of the adversary is to

$$\operatorname{argmin}_{\delta} \ell(f_{\theta}(x_i + \delta), y'_i) \tag{2.5}$$

Additionally the secondary goal is to minimize the adversarial perturbation δ . The attack is said to be successful when $f(x_i + \delta) = y'_i \neq y_i$. Typically this type of attack is harder than the non-targeted variant.

In a **non-targeted** attack the adversary just seeks to change the output from the correct behavior, i.e.

$$\operatorname{argmax}_{\delta} \ell(f_{\theta}(x_i + \delta), y_i) \tag{2.6}$$

The attack is said to be successful when $f(x_i + \delta) \neq y_i$. Contrast this to equation 2.2. This min-max game is central to some of the defense strategies.

2.1.2 Blackbox vs Whitebox

In a **blackbox** type attack the adversary has no information about the internal network in question save for an oracle like access to the network. The adversary may have access to the dataset upon which the network it was trained or it may not.

In a **whitebox** attack the adversary has full information about the internal network,

including the network topology, the parameter values, training data, training procedure, etc. This type of attack is easier, and we will go over popular strategies used in section 2.2.

However even in the blackbox case all hope is not lost. Generally an effective strategy is to first approximate the behavior of the network being attacked, i.e. the adversary finds f', θ' such that

$$\operatorname{argmin}_{\theta', f'} \ell(f_{\theta}(x_i), f'_{\theta'}(x_i)) \quad \forall x_i \in X \quad (2.7)$$

After having trained a proxy network to closely approximate the behavior of the original network the problem reduces to the whitebox kind. There is a small caveat however, in this scenario we expect that when a suitable adversarial perturbation δ is found for $f'_{\theta'}, x_i$ it will also be a valid adversarial perturbation for f_{θ}, x_i (Rozsa et al. [44]). There is no known reason as to why this is true, however Goodfellow et al. noted this might be due to the high dimensionality of the spaces of adversarial examples [14].

In the previous case we assumed no knowledge about the network f save for an oracle like access. However we assumed full knowledge about the training dataset $x_i \in X$. What if that is not the case? In this situation where even the training data is not known, significant headway can be made.

The first step is to make a discriminator network D . The job of the discriminator network is to look at the output from the target network f for a candidate dataset \tilde{X} and determine whether it was from the original dataset X or not, i.e.

$$D(f(\tilde{x}_i)) = \begin{cases} 1 & \text{if } \tilde{x}_i \in X \\ 0 & \text{otherwise} \end{cases} \quad (2.8)$$

This is usually possible with techniques from unsupervised learning. Following this step we can make a proxy dataset $X' = \{\tilde{x}_i | D(f(\tilde{x}_i)) = 1\}$. Once we have the proxy dataset we can train a proxy network f' on X' and exploit the transferability of attacks to find adversarial examples for f .

So, fundamentally in blackbox attacks we use the oracle like access to the network to

approximate all the information about the network in question. After the approximation step is completed, attack can proceed exactly as in the whitebox case. Keeping this in mind, we will only talk about whitebox attacks in what proceeds.¹

2.2 Popular Paradigms

We have talked about the goal our adversary has in the preceding section. However, we have said nothing about how an adversary might go about achieving these goals. In this section we will lay out the fundamental attack strategies.

Ironically, the same tool which makes deep learning possible also makes it possible to generate adversarial examples. In the training phase, we use the gradients of the loss with respect to the parameters to take a step towards a local minima of the loss landscape. Instead, we could fix the parameters and take the gradient of an adversarial loss with respect to the input, and thus find out how to nudge the input so as to produce an adversarial input.

2.2.1 L-BFGS

The earliest proposed method for adversarial attack, suggested by Szegedy et al. hinged on the *Limited memory BFGS* algorithm²[51]. It is a quasi Newtonian iterative method for solving non-linear optimization problems. As we have discussed earlier, the adversary seeks to find r such that,

$$f(x_i + r) = l \neq y_i \quad (2.9)$$

It is easy to see how a root finding algorithm can be used to tackle this problem. However, this method is rather computationally inefficient and isn't widely used in practice.

¹There are other forms of attacks which tamper with the training dataset in question to engineer desired erroneous behavior as in [50]. However, we will not discuss them here as they are not relevant to our work. We will assume the datasets used in training are outside the control of the adversary.

²BFGS is an abbreviation for Broyden-Fletcher-Goldfarb-Shanno the creators of the algorithm.

2.2.2 FGSM

FGSM stands for *Fast Gradient Sign Method* and was suggested by Goodfellow et. al. in 2014 [14]. This method directly relies on back propagating the gradients of the loss with respect to the input, and taking one large step in that direction to increase the loss.

$$x_i^{\text{adv}} = x_i + \epsilon \cdot \text{sign}(\nabla_{x_i} \ell(f_\theta(x_i), y_i)) \quad \text{in the non-targetted case.} \quad (2.10)$$

$$x_i^{\text{adv}} = x_i - \epsilon \cdot \text{sign}(\nabla_{x_i} \ell(f_\theta(x_i), y'_i)) \quad y'_i \neq y_i \quad \text{in the targetted case.} \quad (2.11)$$

The ϵ serves to keep the adversarial perturbation defined by $\delta = \text{sign}(\nabla_{x_i} \ell(f_\theta(x_i), y_i))$ bounded in a L_∞ norm. We could instead take a gradient step and project it onto a L_p ball to get the projected fast-gradient sign method.

$$\delta = \epsilon \cdot \frac{\nabla_{x_i} \ell(f_\theta(x_i), y_i)}{\|\nabla_{x_i} \ell(f_\theta(x_i), y_i)\|_p} \quad (2.12)$$

2.2.3 Basic Iterative Methods

The FGSM algorithm outlined above is a one step attack. This can be iterated to increase the loss even further and increase the chances for a successful adversarial attack. This is represented as :

$$x_i^{n+1} = \mathbf{Clip}_\epsilon(x_i^n + \alpha \cdot \nabla_{x_i} \ell(f_\theta(x_i), y_i)); \quad x_i^0 = x_i \quad (2.13)$$

And, as before, the targeted version of this, known as the **Iterative Least likely Class Method (ILCM)** and was suggested by Kurakin et al. [26].

$$x_i^{n+1} = \mathbf{Clip}_\epsilon(x_i^n - \alpha \cdot \nabla_{x_i} \ell(f_\theta(x_i), y'_i)); \quad x_i^0 = x_i \quad (2.14)$$

The step-size α is a hyper-parameter and the **Clip** function keeps the adversarial perturbation inside a ϵ -ball under L_∞ norm.

Just as in the one step attack case, we can have a L_p bounded adversary. This attack method is called the **Projected Gradient Descent Algorithm (PGD)** and

is a common algorithm in convex optimization. The fact that BIM was the L_∞ version of PGD was pointed out by Madry et al. [33].

$$x_i^{n+1} = \Pi_\epsilon(x_i^n + \alpha \cdot \nabla_{x_i} \ell(f_\theta(x_i), y_i)); \quad x_i^0 = x_i \quad (2.15)$$

Here Π is the projection map onto a ϵ -ball under L_p norm.

2.2.4 Other attack paradigms

Several other attack paradigms have been suggested in literature. One interesting set of attacks are L_0 bounded attack. In this type of attack only a few dimensions of the input is changed. Papernot et al. [38] came up with **Jacobian-based Saliency Map Attack (JSMA)**. In this attack one pixel of the input was changed at a time and a saliency map was created using the gradients of the outputs of the network layers. Following this only those pixels were changed which would result in a maximal change of the output. This process is iterated until the allowed number of pixels have been changed or the attack was successful.

Su et al. extended this work further by creating **one pixel attacks** which were up to 70.97% successful in fooling three different network models, by altering just one pixel in the input. They used evolutionary algorithms to generate the adversarial input.

Other attacks suggested by **Carlini and Wagner** [4] came up with algorithms that bound the perturbations under L_∞, L_2 , and L_0 norms simultaneously making them nearly imperceptible. These attacks are widely successful even in cases where precautions³ have been taken to prevent adversarial attacks.

Several other attack paradigms have been proposed in literature, spanning a diverse set of techniques, like using adversarial nets, optimizing over specially manufactured loss functions, etc. We shall not go into detail on all such techniques, and we direct the reader to [1] for a summary.

³Defensive distillation

2.2.5 GANs

Generative Adversarial Network (GAN) is one area in which techniques from adversarial networks have seen practical use in a non-adversarial context. Goodfellow et al. [15] suggested using two networks G (the generator) and D (the discriminator) be trained simultaneously in an adversarial fashion.

The job of the discriminator is to take in either the output from the generator or from the underlying dataset and output a single number in $[0, 1]$ denoting the probability that the input is from the dataset and not the generator. The generator on the other hand takes in noise as input and spits out a result, which tries to emulate the underlying dataset. It is trained to fool the discriminator.

Thus the two networks are in a two player min-max game. Training is completed when the discriminator gives a output of $\frac{1}{2}$ for all outputs of the generator. The two player value function is given as

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (2.16)$$

After training is completed the discriminator is thrown out, and the generator is used to create data that resembles the dataset in question.

These techniques are often extended for use in generating adversarial examples. One such technique is the UPSET network proposed by Sarkar et al. [46]

2.3 Defenses

There are three broad strategies [1] which are used to defend against adversarial attacks.

- Modified training procedures, or modifying training inputs.
- Modifying networks to suppress attacks.
- Using add on components to detect and suppress attacks.

We will go over each of these paradigms briefly in what follows.

2.3.1 Modifying Networks

If one were to make an educated first guess in order to combat adversarial perturbations, one would think to include some sort of a De-noising operation to the input to suppress adversarial artifacts. However, if the adversary is aware of this, it will almost invariably fail to have any effect. Most approaches that do work in practice also involve using additional train-able components in the network.

Gu and Rigazio introduced **Deep Contractive Networks** [16] to address the adversarial attack problem by adding an auto-encoder to suppress attack artifacts. An auto-encoder is a system of two networks, one designed to compress the input data into a lower dimensional latent space, and the other to recreate the original input from the compressed data. Upon training we hope that the network learns some of the underlying relationships or structures that helps it store the information in a lower dimensional space and recreate it losslessly, as illustrated in Figure 2.1.

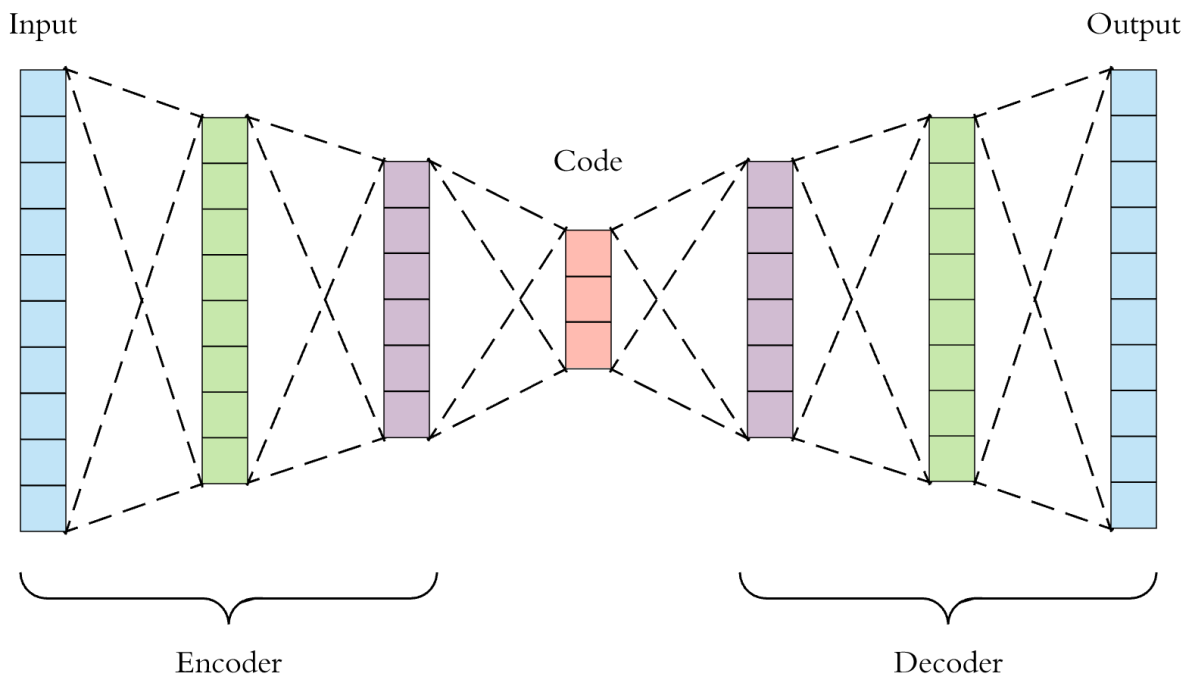


Figure 2.1: Auto-Encoder [Source : towardsdatascience.com]

However it was soon discovered that if the adversary uses the auto-encoder information to inform his attack, that is attack the entire model end-to-end, it actually makes it even more vulnerable to adversarial perturbations.

Since adversarial attacks are reliant on the gradients with respect to the inputs, many defense strategies make attempts to suppress artifacts in the gradients. This is sometimes done using **gradient regularisation** wherein large gradients with respect to the input is penalized as part of the training procedure (as in Ross and Doshi-Velez [43]). It was shown that this method when combined with adversarial training is reasonably robust. Similar attempts by Nguyen and Sinha [36] introduced a masking based defense against C & W attacks by adding noise to the logit output. Other similar approaches include [48] [32].

Defensive distillation was suggested by Papernot et al. [37] where they trained a preliminary model using (input,label) pairs, following which they trained a second network on the class probabilities output by the first network. This process of distillation is usually done to transfer the properties of large networks to smaller networks which can be used on smaller resource restricted networks, such as smartphones.

Papernot et al. showed its efficacy in deflecting small adversarial perturbations [37]. They essentially exploited the knowledge extracted from the network to make itself more robust. However, they have been shown to be vulnerable to C & W attacks [4].

There are several other approaches that follow this general procedure, each with limited effectiveness or limited in types of attacks they can repel.

One interesting approach we will talk about in greater detail is **feature denoising**. This is a technique usually applied to image processing networks, in particular the ResNet152 Denoise model [55] (see figure 2.2 for an illustration of the architecture of ResNets [19]) introduced by Xie et al.

ResNets are different from other deep convolutional neural networks, as they employ residual connections between layers. Normally, very deep neural networks when being trained through back-propagation gives vanishingly small gradients for layers near the input. Having these residual connections gives a shortcut path for gradients to flow to

the early layers, thus allowing ResNets to be much deeper than other architectures. They also have shallow fully connected layers close to the outputs opting instead to rely on feature maps, and pooling. This helps keep the number of parameters smaller, as most of the parameters in deep CNNs are dedicated to the fully connected layers.

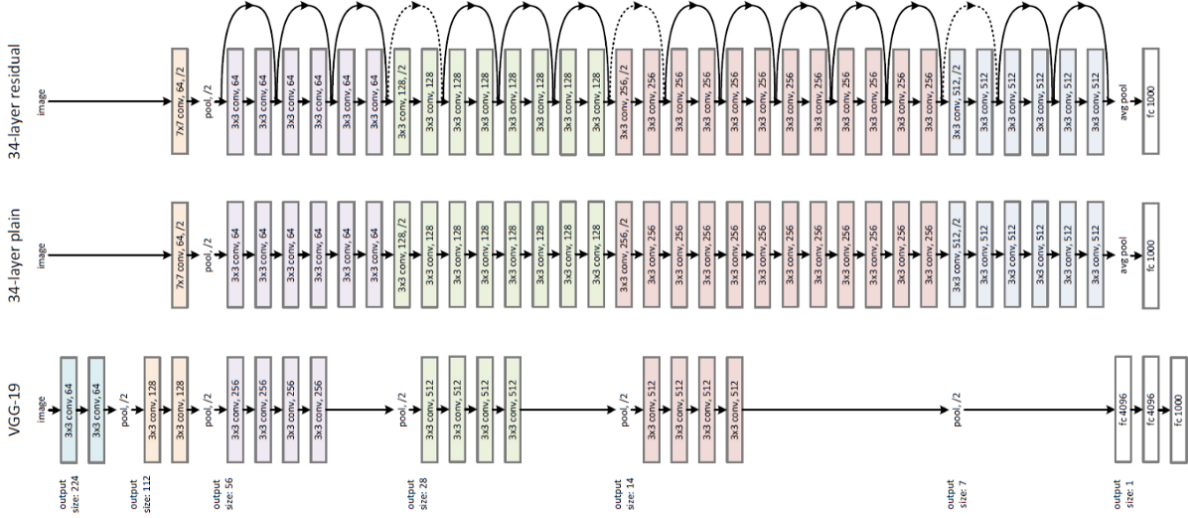


Figure 2.2: Design of ResNet a deep convolutional neural network with residual connections compared to other state of the art deep learning networks. [Source : towardsdatascience.com]

The guiding principle is that adversarial noise added to the input results in noisy features in the intermediate layers. Thus the authors introduced intermediate layers (see figure 2.3) which denoise these intermediate features with non-local means or other filters. Non local means denoise a feature map by taking a weighted mean of features in all spatial location.

$$y_i = \frac{1}{C(x)} \sum_{\forall j \in \mathcal{L}} f(x_i, x_j) \cdot x_j \quad (2.17)$$

Here $C(x)$ is a normalizing factor and $f(x_i, x_j)$ is a feature dependent weighting function. The authors noted that the best performing weighting function was a Gaussian, defined as

$$f(x_i, x_j) = \exp\left(\frac{1}{\sqrt{d}} \cdot \theta(x_i)^T \phi(x_j)\right) \quad (2.18)$$

Here d is the number of channels, and θ, ϕ are two embeddings learned by the network

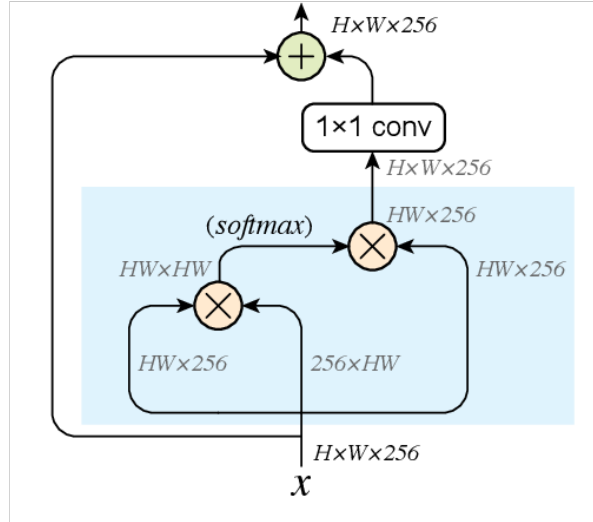


Figure 2.3: De-noising blocks. [Source : Xie et al. [55]]

during training.

This approach, in combination with adversarial training, (see section 2.3.3) demonstrated good robustness. We will discuss these results further in chapter 4.

2.3.2 Add-on Techniques

Techniques in this category are reliant on training additional networks whose only task is to deal with adversarial examples. Some methods are **detection only**, they only raise an alarm if an adversarial example is encountered, other methods train networks to suppress adversarial perturbations. Their outputs are then fed to the main network to repel attacks. Some use GAN like techniques as proposed by Lee et al. in [29]. They proposed training a generator whose job is to generate adversarial perturbations alongside the network.

Another technique proposed by Xu et al. [57] called **feature squeezing** compares the result of a manipulated (smoothed, etc) input with the actual input and flags it when there is a discrepancy. Others (Meng et al. [34]) proposed training additional networks whose job is to discriminate between adversarial and real inputs. Most of these networks however depend on additional deep learning networks which are themselves vulnerable to

adversarial attacks. Detection only methods are somewhat safe in this regard, but their utility is limited. We would like the networks to work despite the presence of adversarial perturbations, not refuse to work altogether when faced with an adversary.

2.3.3 Modifying Training or Inputs

Several methods of defense has been proposed in the literature following this paradigm. Some of these include data compression (as in [11] Dziugaite et al.) who suggested using JPEG compression. Other compression, low-pass filter, etc methods have been suggested, but they are not resilient against sophisticated attacks.

Another novel approach was a **foveation** based defense suggested by Luo et al.[31]. They exploited the scale and translation symmetries to come up with an attack detection strategy. Random resizing of the input (Xie et al [54]) was also shown to have some success.

Adversarial Training

This is a defense strategy that has arguably seen the most success ([33] [14] [51]), and there is consensus in the literature regarding its efficacy. The strategy is simple, after an initial preliminary phase of training, we attack the network and generate adversarial examples. We then use these adversarial examples to retrain the network. We cyclically keep generating and training on adversarial examples until a satisfactory level of robustness has been achieved.

These methods are computationally expensive and require thousands of iterations of strong attacks. However they are very versatile, and can be extended to most types of attacks. Additionally, this can be used in combination with other methods mentioned earlier to bolster their effectiveness [55]. These methods are by no means completely resistant to attacks, but the attacks normally require significantly higher amounts of noise and have to be more computationally expensive. However, there is a caveat; it was observed in literature that there is a trade-off between accuracy and robustness in deep learning networks, which is the subject of Chapter 4.

3 Cryptographic Approaches

We have seen the pervasiveness of adversarial attacks in the previous chapter. Our first approach at addressing this issue involved tools from cryptography. The basic idea is to encrypt the network, so that an adversary cannot use straightforward whitebox adversarial attacks.

This does not solve the underlying learning problem though, because if an adversary somehow does manage to get their hands on the network then they are free to generate adversarial examples. However this approach comes with some benefits. One of the huge hurdles to applying machine learning techniques to the medical field is the confidentiality of medical records. If we could somehow deploy machine learning algorithms on encrypted data this would open up several new possibilities in the field of medicine. This approach however has many limitations, as we will see in the coming sections.

We can not use straightforward encryption to pass the necessary data of the weights, biases, topology and other aspects of the network to our intended recipient as that requires decryption before the network can be used. Once decrypted the network is vulnerable and this approach puts the burden of security on the users. If the intended user can generate inferences without having to decrypt the network first, then even if an adversary gets hold of the network, white-box attacks become impossible. Often deep learning networks are deployed in autonomous systems which can be tampered with by an adversary. If the network could be deployed in an encrypted state, the tampering would be fruitless.

The design of full solution is broken into three parts, encrypting the network, encrypting the data and finally putting it together.

3.1 Protecting the Network

We want a way to encrypt the data in a neural network such that it can be used without decrypting it first.

This has some potential side benefits. Training a neural network requires valuable resources, including computer time and huge labeled data sets. Thus limiting access to certain users can be beneficial. Often it is dangerous to distribute a network, as was demonstrated in a recent case, where a neural network that generates fake news articles which are almost indistinguishable from real ones[60], was not made available during publication. Since the potential for misuse is huge, the distribution to reliable users without encryption is precarious at best.

$$F_{W_{1ij}, W_{2ij}, \dots, W_{nij}, B_{1i}, B_{2i}, \dots, B_{ki}} : \mathbb{R}^m \rightarrow \mathbb{R}^n \quad (3.1)$$

$$v \mapsto F(v)$$

Here F refers to the network in question, and $W_{1ij}, W_{2ij}, \dots, W_{nij}, B_{1i}, B_{2i}, \dots, B_{ni}$ are the weights and biases. We would want an algorithm or function that can encrypt F while still being able to produce the desired result without first decrypting it.

i.e we want E_s such that:

$$E_s(F_{W_{1ij}, W_{2ij}, \dots, W_{nij}, B_{1i}, B_{2i}, \dots, B_{ki}}) = F_s \quad (3.2)$$

$$F_s(v) = F(v) \quad \forall v \in V \subset \mathbb{R}^m$$

3.2 Protecting User Data

Encrypting the user data, although not strictly necessary, has lots of potential benefit. There are certain data sets which are of sensitive nature, especially in medical applications where patient data is bound by confidentiality clause. In these cases training neural networks will be tricky without giving out confidential information. In cases such as

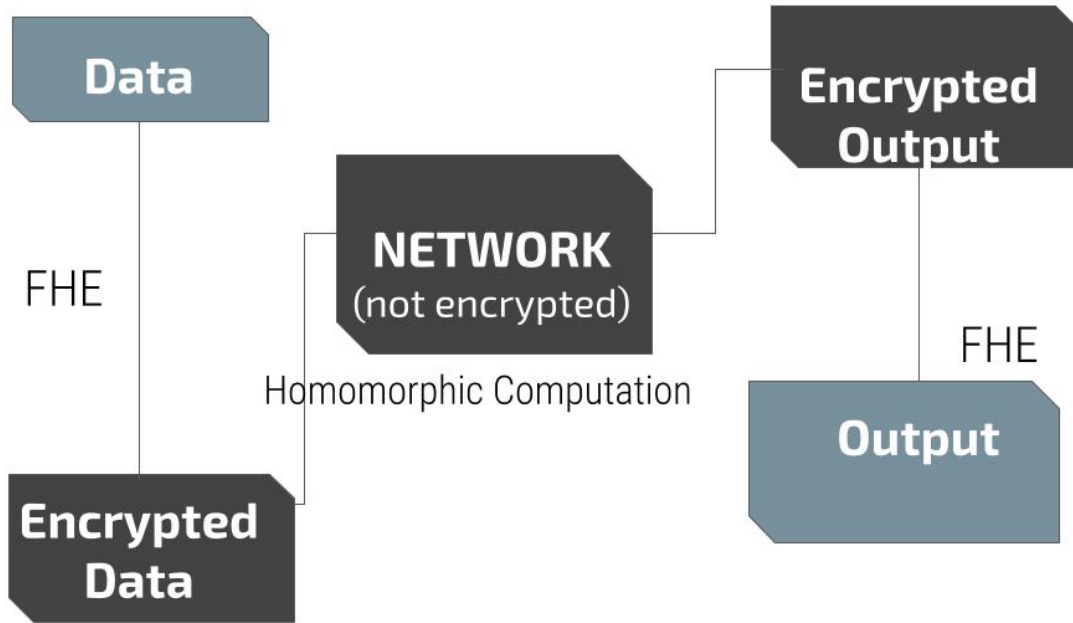


Figure 3.1: Design of privacy preserving machine learning system.

these, where we want to know something about the data without giving away the data itself, the problem can be stated as follows.

$$\begin{aligned}
 &\text{We want to compute } F(p) \text{ given } F, p \\
 &E_s(p) \rightarrow c \text{ is an encryption scheme} \\
 &\text{The encryption scheme should be such that,} \\
 &F(p) = E_s^{-1}(F(c))
 \end{aligned}
 \tag{3.3}$$

A **fully homomorphic encryption** scheme [3] solves this problem effectively. This is schematically represented in Figure 3.1.

3.3 Putting it Together

The best case scenario, is where we can do both of the previous steps simultaneously, with authentication for the user data built into the system. This is illustrated schematically

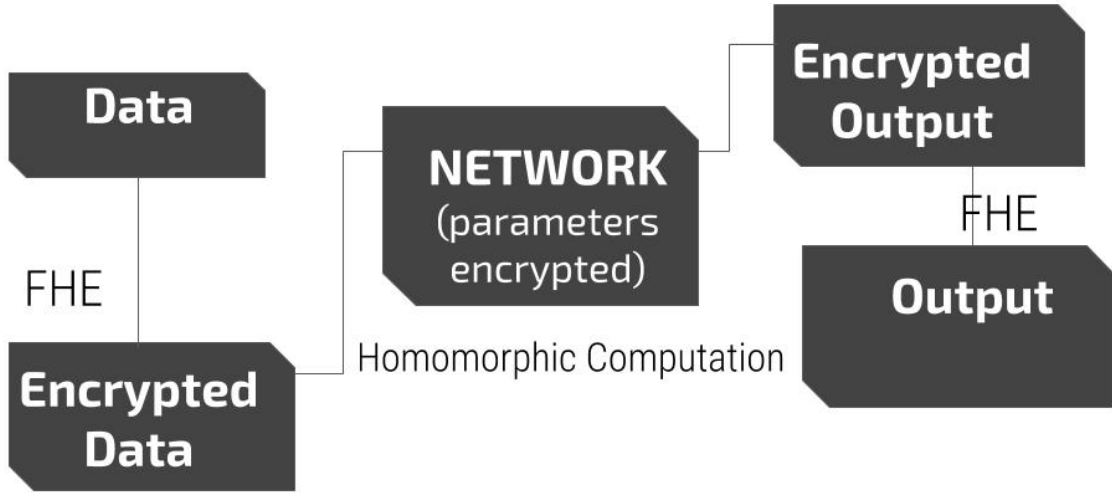


Figure 3.2: Design of encrypted deep learning system.

in Figure 3.2

We want to compute $F_{W_{1ij}, W_{2ij}, \dots, W_{nij}, B_{1i}, B_{2i}, \dots, B_{ki}}(v)$ given $F, v \in V \subset \mathbb{R}^m$

$E_{s_1}(v) = c_{s_1}$ is an encryption scheme with secret key $s_1 \in S_{permitted} \subset \mathbb{S}$

$Enc_{s_2}(F) = F_{s_2}$ Such that,

$$F_{s_2}(c_{s_1}) \begin{cases} = F(v) & s_1 \in S_{permitted} \\ \neq F(v) & \text{otherwise} \end{cases} \quad (3.4)$$

And, computing $F_{s_2}(c_{s_1})$ does not require knowledge of s_2

3.4 Issues with cryptographic approaches

We ran into several issues when it came to the implementation of the system described in Section 3.3. One of which was the fact that most existing deep learning libraries work assuming that the weights and biases are real numbers. However the suitable

crypto-systems work over a subset of the integers. This led us to making our own deep learning library that works over a finite subset of the integers. Also any function that is not a polynomial cannot be implemented in fully homomorphic encryption schemes. This was circumvented with the help of lookup tables.

However, this is trivial when we look at the main problem in implementing in a fully encrypted deep learning system.

3.4.1 The Obfuscation Problem

Suppose there is an algorithm which when given a program P generates a program EP , with the following conditions.

$$\begin{aligned}
 P &\xrightarrow{\text{algorithm}} EP \\
 P(x) &= EP(x) \quad \forall x \\
 EP &\text{ has no information about } P
 \end{aligned}
 \tag{3.5}$$

Such an algorithm has been theoretically shown to be impossible [2].

Our solution unfortunately falls into this category. Our program P is the step by step multiplication with weight matrices and bias vectors. Our encrypted network is then EP . So this automatically excludes the possibility of existence of such an encryption scheme.

3.4.2 Practical Issues

The full solution has been shown to be impossible, but the sub parts are still useful. In particular if we can use neural networks on encrypted data we can still stand to benefit. This would open up several possibilities especially in the field of medicine at the very least. It would bring many sensitive data sets into the fold of machine learning.

This has been implemented by Dowlin et al. [10] which is only designed to draw inferences on encrypted data and by Xu et al. [56] which can train on encrypted data. However with the known encryption schemes these are severely limited by time constrains as shown in Table 3.1.

3 Cryptographic Approaches

	Not Encrypted (time)	Encrypted (time)
Inference on encrypted data	0.1 ms	300s
Training on encrypted data (Functional)	30s	10 minutes
Training on encrypted data (FHE)	30s	10 years

Table 3.1: Time comparison for various encrypted NN schemes.

4 Robustness vs Accuracy

4.1 Motivation

Since adversarial attacks were discovered, it started a cat and mouse game between defense strategies and even more sophisticated attack algorithms. However, a few standards have emerged over the years. Adversarial training has broadly been shown to make a model more robust to attacks. Additionally PGD attacks as discussed in section 2.2.3 has been shown to be "*universal*" in the sense that robustness against PGD, yields robustness against all first order adversaries (Madry et al. [33]).

However, adversarial training as discussed in section 2.3.3 comes with a few caveats. In almost all cases it has been shown to hurt standard testing accuracy. For instance Ragunathan et al. [40] found that adversarial training improved robust accuracy (accuracy under adversarial samples) from 3.5% to 45.8% but, it reduced standard accuracy from 95.2% to 87.3% on a suitable trained network on the CIFAR 10 [25] dataset.

Tsipras et al. [52] tried to explain how robustness and accuracy are at odds from a theoretical standpoint. They demonstrated that if a only a very small subset of the input is highly correlated with the label, and the rest of the input is weakly correlated, a robust and accurate classifier is impossible. However, this feature is not present in a lot of datasets we see in practice. In fact we have good reason to believe robust, and accurate classifiers exist since humans are robust and accurate classifiers.

Some older results seem to contradict this observation however, for instance Goodfellow et al. [14] noted that adversarial training using FGSM seemed to act as a sort of regularisation of the network, resulting in better generalizing results. However, this

generally only holds for small datasets where over-fitting is a primary concern (as was noted by Goodfellow, Kurakin et al. [27]). This also fails when more sophisticated attacks are used.

4.2 Observations

Xie et al. [55] proposed a combination of feature denoising and adversarial training for a robust classifier on the ImageNet dataset. The ImageNet classification dataset [45] has ~ 1.28 million images in 1000 classes. They demonstrated very strong results (42.8% accuracy against a 2000 iteration PGD attack). We also found that an unbounded adversary when attacked with ILCM, **generated an image belonging to the target class!** Also, this being a variant of the existing and popular, ResNet [19] architecture leads us to believe, that this model would help us extend this model's robustness to other learning tasks.

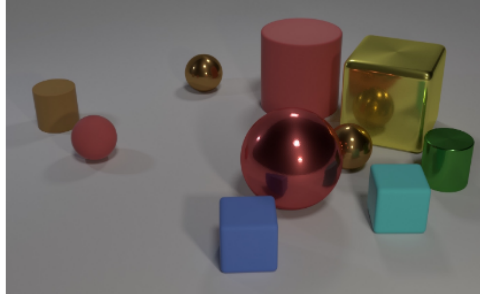
This approach, where we take the knowledge gained from one problem and apply it to another problem is known as transfer learning, and has been a popular paradigm in machine learning. The problem we chose for this task was visual question answering on the CLEVR dataset [23]. This task consists of natural language questions based on an input image, and requires the network to reason about the question (see figure 4.1).

A rather successful model in this regard has been Compositional Attention Networks by Hudson and Manning [22]. Their model consists of a more sophisticated versions of Long Short Term Memory Networks (LSTM) (see section 1.2.1) [21] (with attention), named "MAC cell" (see figure 4.2). The network is composed with chained together MAC cells with additional fully connected layers for output. Additionally they use features extracted from ResNet101 [19] trained on the ImageNet [45] dataset for a useful and feature rich embedding of the input image.

With this approach Hudson and Manning showed that their network far outperformed other proposed solutions and had an accuracy of 98.9%. This network even outperforms humans who scored 92.6% (as shown by Johnson et al. in 2017 [24]).

4 Robustness vs Accuracy

Questions in CLEVR test various aspects of visual reasoning including **attribute identification**, **counting**, **comparison**, **spatial relationships**, and **logical operations**.



- Q: Are there an **equal number** of **large things** and **metal spheres**?
- Q: **What size** is the **cylinder that is left of the brown metal thing that is left of the big sphere**?
- Q: There is a **sphere** with the **same size as** the **metal cube**; is it **made of the same material as** the **small red sphere**?
- Q: **How many** objects are **either small cylinders or red things**?

Figure 4.1: Sample questions from the CLEVR dataset. [Source : CLEVR [23]]

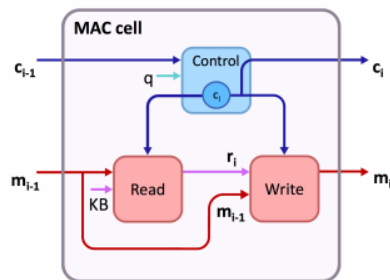


Figure 4.2: Design of the 'MAC Cell' proposed by Hudson and Manning for VQA tasks. [22]

However, as was shown by Chaturvedi et al, these networks are highly vulnerable to adversarial perturbations. Chaturvedi et al. [5] even demonstrated that an attacker who only adds noise to the background (non-focal parts of the image) can reliably fool the network.

So, armed with a robust image classifier (ResNet152 Denoise) [55] and a vulnerable network which uses an accurate but non-robust image feature extractor ResNet101, we sought to improve the robustness of MAC Network with the adversarially robust feature extractor. We trained the MAC network on image features extracted using several variants of the ResNet model.

It was found by Rozsa et al. [44] that models that are more accurate, are generally also more robust. So, it was no surprise to us that this approach of using ResNet152 Denoise in conjunction with MAC Network was very vulnerable, as accuracy was greatly impacted with the robust feature extractor. The results are summarized in Table 4.1.

Image Feature Extractor	Validation Accuracy
ResNet101 [22]	98.9%
ResNet152 [22]	99.0%
ResNet152 Baseline (adversarial training only)	47.24%
ResNet152 Denoise (adversarial training and denoise layers)	90.72%

Table 4.1: Comparing the accuracy on the validation set of different variants of ResNet in conjunction with MAC Network.

Features of size $1024 \times 14 \times 14$ was extracted from Convolution Group 2 of ResNet¹. The most striking result is seen in the difference between row 2 and 3 in Table 4.1. Those data points belongs to two models which are identical, except that one has been adversarially trained. Fine-tuning the whole network in an end to end fashion might have helped, but was not undertaken, as the original model received high accuracy without it, and would give a better baseline to compare against. This observation seems to be in line with the common consensus in literature **robustness and accuracy are at odds!**

¹other layers gave worse performance. Drawing features from later layers of size $2048 \times 7 \times 7$ gives 85% accuracy on the test set.

5 The Plasticity Hypothesis :

Further Results

Having observed the duality between accuracy and robustness in Chapter 4 we turn our attention to answering the question **Why?** No trivial answer to this question exists in the literature and there have been several proposals with varying degrees of success. We will briefly look at the proposals in the literature before we can be a step closer to attempting to answer this question.

5.1 Robustness is Harder

Robustness is a harder problem to solve than vanilla (without adversarial robustness) learning. Goodfellow et al. [14] noted that adversarial examples are not scattered throughout the input space in tiny pockets but rather form large subsets of high dimensional sub-spaces around the distribution samples. Thus if a model has to arrive at the correct decision boundaries in input space, its decision boundaries have to be a lot more precise. This makes the problem reasonably harder than learning the underlying representations.

Madry et al. noted that to solve the robustness problem, we need to simultaneously solve a non-convex outer minimization problem and a non-concave inner maximization problem. Note, that vanilla learning requires us to approach the problem in Equation 5.1.

$$\operatorname{argmin}_{\theta} \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}}[\ell(f_{\theta}(x_i), y_i)] \quad (5.1)$$

Here \mathcal{D} is an underlying distribution from which input, label pairs (x_i, y_i) has been

drawn. f_θ is our network with parameter set θ and ℓ is a suitably chosen loss function. $\mathbb{E}_{(x_i, y_i) \sim \mathcal{D}}[\ell(f_\theta(x_i), y_i)]$ is called the population risk.

Whereas the adversarial robustness problem requires us to solve the problem in Equation 5.2.

$$\operatorname{argmin}_\theta \rho(\theta) \quad \text{where,} \quad \rho(\theta) = \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}} \left[\max_{\delta \in \mathcal{S}} \ell(f_\theta(x_i + \delta), y_i) \right] \quad (5.2)$$

Here, \mathcal{S} defines the set of allowable perturbations that we want our model to be robust to. In particular, a ϵ -ball under L_∞ norm, is a good candidate for \mathcal{S} , as such perturbations are visibly imperceptible.

In this framework, attacks seek to solve the inner maximization problem, and our goal with solving the robust optimization is recognizing that we need to minimize the population risk for the adversarial loss under the action of every (or, every necessary) adversary.

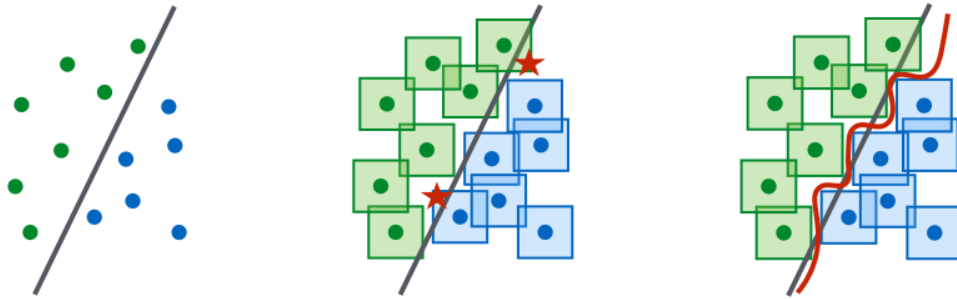


Figure 5.1: Conceptual illustration of standard vs. adversarial decision boundaries [Madry et al. [33]]

An intuitive understanding of this phenomena is offered by Figure 5.1. In the case where we are training the network without an adversary the decision boundary is simple, but in the presence of a L_∞ bounded attacker the decision boundary is significantly more complicated.

5.2 Capacity is Limited

There is consensus in the literature that network capacity often limits robustness. Goodfellow et al. [14] noted that networks with higher capacities perform better under adversarial training. Capacity here roughly equates to the number of parameters in the network.

Nakkiran [35] went so far as to posit that adversarial robustness is at odds with "simplicity". In particular they asked - *Why do current techniques fail to learn classifier with low adversarial loss, while they suffice to learn classifiers with low standard loss and low noise-robust loss?*

Madry et al. [33] noted that higher capacity networks perform better under adversarial training. They went so far as to claim that a higher capacity network alone can mitigate some of the losses associated with adversarial training.

However, all prescriptive claims seem to conclude that wider networks are better equipped to be adversarially trained without loss of accuracy. We can infer the reasoning behind the claims that wider networks tend to be more robust, and we believe it has to do with the **Universal Approximator Theorem**.

5.2.1 Universal Approximator Theorem

Theorem 1 *Let $\phi: \mathbb{R} \rightarrow \mathbb{R}$ be a continuous function¹ (activation function). Let I_m denote the m -dimensional unit hypercube $[0, 1]^m$. The space of real valued functions on I_m is denoted by $C(I_m)$. Then given any $\epsilon > 0$ and any function $f \in C(I_m)$ there exists an integer N , real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}$ for $i = 1, \dots, N$, such that we may define:*

$$F(x) = \sum_{i=1}^N v_i \phi(w_i^T x + b_i) \quad (5.3)$$

as an approximate realization of the function f , i.e.

$$|F(x) - f(x)| < \epsilon \quad \forall x \in I_m. \quad (5.4)$$

¹Usually refers to the sigmoid function $S(x) = \frac{1}{1+e^{-x}}$

This theorem due to Cybenko [9], simply put, posits that a neural network with a single hidden layer, can approximate any function from its input space to \mathbb{R} , if it is sufficiently wide.

Given this result it is not surprising that most recommendations imply that wider networks have higher "capacity". However, we believe this result is too simplistic. It has been shown, that most of the connections in a neural network have little to no impact on the output.

5.2.2 Model Compression

Le Cun et al. [28] showed that a saliency based approach (using Hessians of the target function) to removing neurons do not affect accuracy. They proposed removing the parameters which had the smallest effect on the output and re-training the network. This process can be done iteratively to reduce the number of parameters in the network. They demonstrated that up to 30% of the parameters can be removed from a network without significant losses in accuracy.

Han et al. [18] proposed that we learn the network connections in addition to the parameters. In this paradigm, the network is first trained to learn important connections and then retrained to learn the parameter values. They showed **$9 \times -13 \times$ compression over the model parameters**, i.e. a model with only $\sim 10\%$ of the parameters, have effectively the same accuracy as the original model.

Hinton et al. [20] generalized the notion of distillation to deep learning, where instead of labels, a model is trained on the probabilities of classes given by the output of a larger network. The softmax function defined in Equation 5.5, gives the probabilities from the model logits z ².

$$\mathcal{S}(z)_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}} \quad (5.5)$$

They have demonstrated that the knowledge of larger networks can be transferred into smaller networks. Other proposed methods including **quantization** [17] (to reduce the

² T is a temperature parameter, higher temperatures result in smoother probability distributions

precision of each parameter), **decomposition of parameters**, etc have been immensely successful in compressing models significantly without any loss in accuracy.

So we ask ourselves the following question: *If neural networks can learn relationships so efficiently, why is capacity a bottleneck for robustness?*

5.3 Plasticity Hypothesis

The key take-away from Section 5.1 and 5.2 is that

- Robustness is a significantly harder problem, and requires higher "capacity" of networks.
- Neural networks have significantly higher ability to learn, than simple measures of capacity such as width of the network, or number of parameters leads us to believe.

We also note an interesting observation by Cheng et al. [7] which says that deeper networks have worse performance under pruning than wider networks, further bolstering our idea that width is a bad measure of capacity to learn. Taking into account all the evidence we found and in the literature we are lead to the following.

Hypothesis 1: (*weak version*) *When adversarially training a network, it must be made deeper to increase robustness while maintaining accuracy.*

Hypothesis 2: (*strong version*) *There exists a function $\Phi : \{f, \theta\} \rightarrow \mathbb{R}$ which measures the network's capacity to learn (plasticity).*

Here f is the network (including its topology, training procedure, etc) and θ its parameter set. We do not know the exact nature of the function, but we note some general trends supported by the behavior we see in networks in the literature.

For instance it is well known that deep networks perform better on higher dimensional datasets (successful networks on ImageNet are hundreds of layers deep). It is quite evident from the literature, that depth is strongly correlated with Φ . Madry et al. noted that width is correlated with Φ but only to a certain point.

Pruning and compression can also be understood in this framework. When we **regularize** a network to prevent over-fitting by either a L_p penalty, dropouts, etc we significantly reduce the Φ value of a network. This means a much smaller, or less complicated network can reproduce the same results as they have similar values of Φ .

Of note is the fact that networks with a bounded width, but unbounded depths are also universal approximators [30]. However, the universal approximator theorems, do not give us an algorithm to find the values of parameters which would result in the required approximation, thus not being valuable indicators towards the correlation between depth or width and Φ . But given the algorithms in our toolbox, we find that deeper networks perform better both in terms of accuracy and robustness.

We would like to note that Hypothesis 2 is not very useful in its current state, but we hope to bolster this further with investigation in the future, and hopefully arrive at a good bounds on Φ given any network.

5.4 Results

Experiments

In our experiments we found evidence in support of Hypothesis 1. In particular, we trained a simple convolutional neural network on the MNIST dataset [58] (consisting of 60,000 handwritten digits for training and additional 10,000 for testing). The simple model consisted of two convolutional layers, a max pooling layer and two fully connected layers. The deeper model had three convolutional layers and everything else remained unchanged.

Initially we trained the network for 12 epochs, followed by 20 epochs of training on adversarial data generated by a 200 iteration PGD attacker ($\epsilon = 8$). The robustness of the model increased (from approximately, 56% to 12% success for the adversary), however its accuracy on clean data was adversely affected. Then we repeated the exact same experiment on the deeper model. We found that accuracy on clean images was improved. The results are summarized in Table 5.1 and in Figure 5.2.

Condition	Shallow Model	Deep Model
Train Accuracy	99.50	98.70
Test Accuracy	98.80	98.36
Train Accuracy (Post Adversarial Training)	88.67	94.05
Test Accuracy (Post Adversarial Training)	94.91	97.33

Table 5.1: Performance comparison for the shallow and deep model.

This is the behavior predicted by Hypothesis 1, and is along the lines of the observation in Table 4.1 in Section 4.2.

5.5 Future Work

We believe that the plasticity hypothesis is very promising, however we could not gather nearly enough evidence in support of the hypothesis. Several experiments needs to be carried out in order to put the hypothesis on a more stable footing.

The first step would be to find hard limits on the memory capacity of the network. We tend to always try and avoid the scenario when the network exactly learns the training dataset mapping. However, knowing the bounds on the plasticity for differently structured networks would give us clues towards how the Φ function behaves.

Additionally, the results we found needs to be extended to the CIFAR-10 [25] and CIFAR-100 [25] datasets, which are 10, 100 class classification problems respectively. This would certainly lend more credence to the results we found.

The extension of these results onto the ImageNet dataset[45] would be of even more interest. In a scenario where the examples in each classes are sparse the learning problem is in itself rather challenging. Finding the adversarial decision boundaries might require specific architectures which are several-fold more complex than in the other datasets.

The stronger version of the hypothesis would require us to investigate adversarial training on several different model types. In particular we would have to find the nature of correlation between width and plasticity, and depth and plasticity. If they are indeed

5 The Plasticity Hypothesis : Further Results

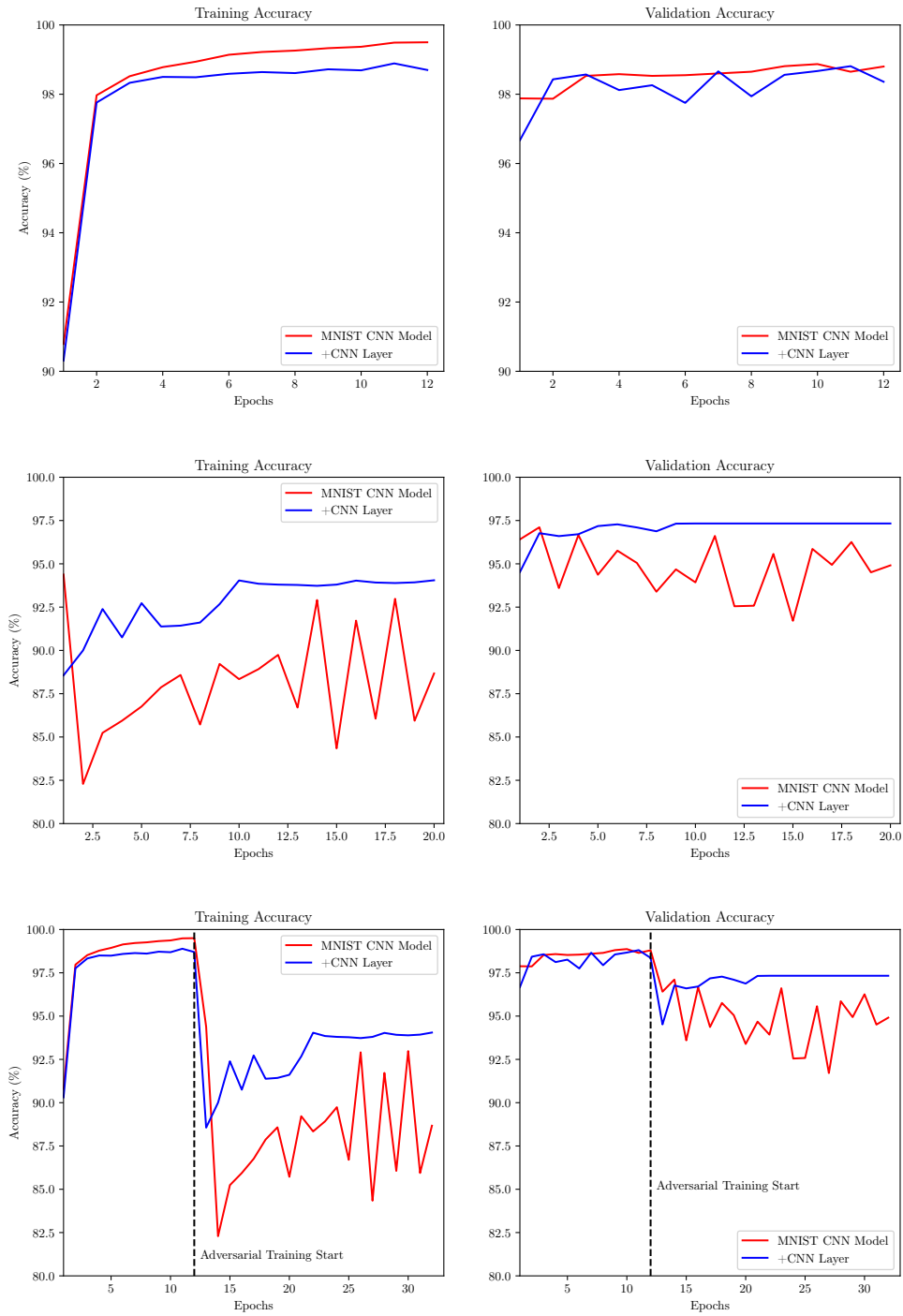


Figure 5.2: Behavior of the two networks under adversarial training. (top) Statistics during training, (middle) Statistics during adversarial attacks, (bottom) Combined. The blue line represents the deeper model, it has improved accuracy on clean images compared to the shallow model.

separable in this sense we can hope to write down plasticity as a function of width and depth of the network.

We would also need to study the effect of regularization, quantization, sparsity, loss function and even the training algorithm on the capacity of a network. Intuition tells us that all these measures strive to reduce the plasticity of the network in order to avoid over-fitting. Studying the effect each technique has on plasticity will be elucidating. This will also lead to us being able to choose hyper-parameters in a more concrete fashion during vanilla training.

The ultimate end goal would be to be able to write down a closed form expression or even an algorithm to determine the plasticity of a network. However, this might be impossible, and even if possible, is a distant goal.

6 Conclusions

After a brief introduction in machine learning we approached the *learning problem* and went over the proposals in literature, to solve the same. Following this we attempted a solution to the learning problem through cryptographic means, and showed it was a dead end, owing to current day limits on computational power, and in some cases, its outright theoretical impossibility.

Our foray into using adversarial training to solve the robustness problems yielded results which were in tune with the consensus in literature. This led us to turn our attention to the question - *Why robustness and accuracy are at odds?*. To that end, we came up with concrete recommendations to train more robust and accurate networks. We also hypothesized the existence of a function Φ which measures the plasticity (or capacity to learn) of a network and noticed the general trends such a function should have.

Our recommendations regarding the robustness, accuracy duality are contrary to recommendations in the literature, however more work is needed to refine the hypothesis and arrive at a more concrete form for the function Φ .

With the increasing prevalence of automation, the learning problem is of critical importance. Before deep learning can be reliably deployed in applications requiring high performance and security, we need to have robust and accurate networks. Otherwise, the day is not far, when we receive a traffic ticket owing to a carefully placed sticker, fooling our self driving car, **and that may just be the least of our worries.**

Bibliography

- [1] Naveed Akhtar and Ajmal S. Mian. “Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey”. In: *IEEE Access* 6 (2018), pp. 14410–14430.
- [2] Boaz Barak et al. “On the (Im)possibility of Obfuscating Programs”. In: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1. URL: https://doi.org/10.1007/3-540-44647-8_1.
- [3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. *Fully Homomorphic Encryption without Bootstrapping*. Cryptology ePrint Archive, Report 2011/277. <https://eprint.iacr.org/2011/277>. 2011.
- [4] Nicholas Carlini and David A. Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *2017 IEEE Symposium on Security and Privacy (SP)* (2017), pp. 39–57.
- [5] A. Chaturvedi and U. Garain. “Attacking VQA Systems via Adversarial Background Noise”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* (2020), pp. 1–10.
- [6] Akshay Chaturvedi and Utpal Garain. “Mimic and Fool: A Task Agnostic Adversarial Attack”. In: (2019).
- [7] Yu Cheng et al. “A Survey of Model Compression and Acceleration for Deep Neural Networks”. In: *ArXiv abs/1710.09282* (2017).

Bibliography

- [8] Paul F. Christiano et al. “Deep Reinforcement Learning from Human Preferences”. In: *NIPS*. 2017.
- [9] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274. URL: <https://doi.org/10.1007/BF02551274>.
- [10] Nathan Dowlin et al. *CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy*. Tech. rep. MSR-TR-2016-3. 2016. URL: <https://www.microsoft.com/en-us/research/publication/cryptonets-applying-neural-networks-to-encrypted-data-with-high-throughput-and-accuracy/>.
- [11] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M. Roy. “A study of the effect of JPG compression on adversarial images”. In: *ArXiv abs/1608.00853* (2016).
- [12] Ivan Evtimov et al. “Robust Physical-World Attacks on Deep Learning Models”. In: *arXiv: Cryptography and Security* (2017).
- [13] L. A. Gatys, A. S. Ecker, and M. Bethge. “Image Style Transfer Using Convolutional Neural Networks”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2414–2423.
- [14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: *CoRR abs/1412.6572* (2015).
- [15] Ian J. Goodfellow et al. “Generative Adversarial Networks”. In: *ArXiv abs/1406.2661* (2014).
- [16] Shixiang Gu and Luca Rigazio. “Towards Deep Neural Network Architectures Robust to Adversarial Examples”. In: *CoRR abs/1412.5068* (2015).
- [17] Song Han, Huizi Mao, and William J. Dally. “Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding”. In: *CoRR abs/1510.00149* (2016).

Bibliography

- [18] Song Han et al. “Learning both Weights and Connections for Efficient Neural Network”. In: *ArXiv* abs/1506.02626 (2015).
- [19] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.
- [20] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. “Distilling the Knowledge in a Neural Network”. In: *ArXiv* abs/1503.02531 (2015).
- [21] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (1997), 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [22] Drew A Hudson and Christopher D Manning. “Compositional Attention Networks for Machine Reasoning”. In: *International Conference on Learning Representations (ICLR)*. 2018.
- [23] Johanna E. Johnson et al. “CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 1988–1997.
- [24] Justin Johnson et al. “Inferring and Executing Programs for Visual Reasoning”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 3008–3017.
- [25] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: 2009.
- [26] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. “Adversarial examples in the physical world”. In: *ArXiv* abs/1607.02533 (2017).
- [27] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. “Adversarial Machine Learning at Scale”. In: *ArXiv* abs/1611.01236 (2017).

Bibliography

- [28] Yann LeCun, John S. Denker, and Sara A. Solla. “Optimal Brain Damage”. In: *Advances in Neural Information Processing Systems 2*. Ed. by D. S. Touretzky. Morgan-Kaufmann, 1990, pp. 598–605. URL: <http://papers.nips.cc/paper/250-optimal-brain-damage.pdf>.
- [29] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. “Generative Adversarial Trainer: Defense to Adversarial Perturbations with GAN”. In: *ArXiv* abs/1705.03387 (2017).
- [30] Zhou Lu et al. “The Expressive Power of Neural Networks: A View from the Width”. In: *ArXiv* abs/1709.02540 (2017).
- [31] Yan Luo et al. “Foveation-based Mechanisms Alleviate Adversarial Examples”. In: *ArXiv* abs/1511.06292 (2015).
- [32] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. “A Unified Gradient Regularization Family for Adversarial Examples”. In: *2015 IEEE International Conference on Data Mining* (2015), pp. 301–309.
- [33] Aleksander Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *ArXiv* abs/1706.06083 (2017).
- [34] Dongyu Meng and Hao Chen. “MagNet: A Two-Pronged Defense against Adversarial Examples”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017).
- [35] Preetum Nakkiran. “Adversarial Robustness May Be at Odds With Simplicity”. In: *ArXiv* abs/1901.00532 (2019).
- [36] Linh Nguyen and Arunesh Sinha. “A Learning Approach to Secure Learning”. In: *ArXiv* abs/1709.04447 (2017).
- [37] Nicolas Papernot et al. “Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks”. In: *2016 IEEE Symposium on Security and Privacy (SP)* (2016), pp. 582–597.

Bibliography

- [38] Nicolas Papernot et al. “The Limitations of Deep Learning in Adversarial Settings”. In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)* (2016), pp. 372–387.
- [39] Christine Payne. *MuseNet*. 2019. URL: openai.com/blog/musenet.
- [40] Aditi Raghunathan et al. “Adversarial Training Can Hurt Generalization”. In: *ArXiv* abs/1906.06032 (2019).
- [41] George W. Reitwiesner. “An ENIAC Determination of π and e to more than 2000 Decimal Places”. In: *Pi: A Source Book*. New York, NY: Springer New York, 2000, pp. 277–281. DOI: 10.1007/978-1-4757-3240-5_34.
- [42] F. Rosenblatt. *The Perceptron: A Probabilistic Model For Information Storage And Organization*. 1958. DOI: 10.1.1.335.3398.
- [43] Andrew Slavin Ross and Finale Doshi-Velez. “Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients”. In: *AAAI*. 2018.
- [44] Andras Rozsa, Manuel Günther, and Terrance E. Boult. “Are Accuracy and Robustness Correlated”. In: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2016), pp. 227–232.
- [45] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.
- [46] Sayantan Sarkar et al. “UPSET and ANCRI : Breaking High Performance Image Classifiers”. In: *ArXiv* abs/1707.01159 (2017).
- [47] Ali Shafahi et al. “Are adversarial examples inevitable?” In: *ArXiv* abs/1809.02104 (2019).
- [48] Uri Shaham, Yutaro Yamada, and Sahand N. Negahban. “Understanding adversarial training: Increasing local stability of supervised models through robust optimization”. In: *Neurocomputing* 307 (2018), pp. 195–204.

Bibliography

- [49] Simonyan et al. Silver Schrittwieser. “Mastering the game of Go without human knowledge”. In: *Nature* 550 (2017), pp. 354–359. DOI: 10.1038/nature24270.
- [50] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. “Certified Defenses for Data Poisoning Attacks”. In: *NIPS*. 2017.
- [51] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *CoRR* abs/1312.6199 (2014).
- [52] Dimitris Tsipras et al. “Robustness May Be at Odds with Accuracy”. In: *arXiv: Machine Learning* (2018).
- [53] Paul J. Webros. “BACKPROPAGATION THROUGH TIME: WHAT IT DOES AND HOW TO DO IT”. In: 1990.
- [54] Cihang Xie et al. “Adversarial Examples for Semantic Segmentation and Object Detection”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 1378–1387.
- [55] Cihang Xie et al. “Feature Denoising for Improving Adversarial Robustness”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [56] Runhua Xu, James B. D. Joshi, and C. C. Li. “CryptoNN: Training Neural Networks over Encrypted Data”. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)* (2019), pp. 1199–1209.
- [57] Weilin Xu, David Evans, and Yanjun Qi. “Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks”. In: *ArXiv* abs/1704.01155 (2018).
- [58] LeCun Y. et al. “Gradient-based learning applied to document recognition.” In: *Proceedings of the IEEE* 86 (1998), pp. 2278 –2324. DOI: 10.1109/5.726791.
- [59] Vaghefi Yang and Hill. “Detection of smoking status from retinal images, a Convolutional Neural Network study.” In: *Sci Rep* 9.7180 (2019). DOI: 10.1038/s41598-019-43670-0.
- [60] Rowan Zellers et al. “Defending Against Neural Fake News”. In: *NeurIPS*. 2019.