Effectiveness of adversarial examples on class-imbalanced Convolutional Neural Networks

Rafael Carvalhaes Possas
School of Information Technologies
University of Sydney

A thesis submitted for the degree of
Master of Information Technology

Sydney, 2017

This thesis is dedicated to
someone
for some special reason

Acknowledgements

plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle.

Abstract

plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pattern Recognition and Data Mining is a field of study that focuses on using relatively complex algorithms to discover knowledge from large pools of data. These are usually used to predict the future or to recognise patterns and label data points that are close together. The increase of computational power on the last two decades leveraged the use of techniques such as Neural Networks [4] to tackle more complex problems such as the one of labeling digital images. The work of Lecun (1989) [17] was a one of the stepping stones for all work on image recognition using Neural Networks as he was able to prove the effectiveness of stacking multiple layers of neurons to form what we call a Neural Network. These studies usually refer to how the human brain works to explain the inspiration of such techniques and each neuron was later named as perceptrons.

The efforts on the early days were focused in understanding the main principles behind human learning. For instance, recognizing handwritten digits could be seen as a trivial and effortless job for most people, however, making a computer to be able to perform this same task was not as easy as it seemed. By discovering the pattern behind digit recognition, computers would also be able to start understanding broader classes of images [15]. The use of computer vision techniques along with Machine Learning algorithms are nowadays the state of the art technique to overcome these challenges.

Computer Vision is a field of study that focuses on processing digital images and has Neural Networks as one of the underlying foundations for its algorithms. These are usually focused on learning models that recognise patterns on data with several dimensions (e.g. images with width x height number of pixels). Recent advancements in both Computer Vision and Neural networks have led to the development of a new class of algorithms which is nowadays known as either Deep Learning or Deep Feed Forward Networks.

Extremely deep networks (e.g. one containing stacked perceptrons layers) are classified as Deep Learning algorithms and can be more popularly represented through deep feed forward neural networks [12] and, more recently, recursive neural networks [7]. While the latter focuses on solving problems where the data points are dependent on one another (e.g. Time series) the first is more largely used on image recognition and, therefore, should be the focus of this work.

A more specific approach on feed forward nets is to extract important features from images before trying to classify them as a predefined class. This approach is widely used on a specific Neural Network architecture called Convolutional Neural Networks [19]. The feature extraction process can be summarized as applying specific operations in order to learn the edges of a set of images and feed these on a traditional fully connected network for classification.

Recent work has shown that the generalization learned into those networks is rather sparse [16]. This sparsity opens up an opportunity for methods that are able to go to unexplored data spaces in order to intentionally fool the network into predicting a different class for a given image. This operation is comprised of changing current images by adding just enough intentional noise to each pixel in order to fool an algorithm into thinking that the image has a different label [9][25][16][28]. The resulting images of this process are known as Adversarial Examples and their generation is done through the use of a method called Fast Gradient Sign [9].

This thesis presents an experimental approach on which factors could lead to less or more robustness of the Convolutional Neural Networks to Adversarial attacks. Through the use of the Fast Gradient Sign Method we will try to understand how class imbalance during training time can affect the network resistance to such attacks. Although this research focuses not only on a specific technique but also a special data space (i.e. images), the discoveries could also be generalized to other classification problems as it focuses on understanding the relationship between the algorithm capability of exploring a given data space and its overall accuracy when presented to previously unseen data points.

1.1 Motivation

As the number of dimensions in a dataset grows, the harder it is for one to visualize or explain the feature space. Therefore, exploiting vulnerabilities is one way of explaining where a specific algorithm is not performing well.The scientific understanding of such

behaviors can lead to new breakthroughs on this field as the recent work of Goodfellow et al. (2014) shows on the GANs(Generative Adversarial Networks) architecture [8].

Although the biological inspiration for neural networks comes from understanding human vision, the work from Nguyen et al.(2015) [21] have shown that unrecognizable images can be classified with pretty high confidence by DNNs. This result shows that although the technique is framed to mimic human vision, the feature space of images still needs to be explored further to avoid this kind of outcome.

Transferability in Machine Learning also shows that algorithms are vulnerable to systematic attacks as long as they are trained for the same purpose [25]. With the ever increasing number of systems using the same techniques, one should be careful on the emerging threats being posed to their systems as several tasks are starting to heavily rely on deep learning methods.

The motivation for Adversarial robustness comes largely from being able to shield image recognition systems from behaving unexpectedly. The world is embracing Artificial Intelligence and there are already a large number of solutions that rely on these algorithms to perform tasks that range from plate recognition in car parking to general image recognition APIs such as Amazon Rekognition. In particular, differentiable machine learning algorithms in general could benefit from better understanding of gradient calculations and how they could explore the data space more efficiently.

1.2 Thesis Goals

1.3 Thesis Structure

Chapter 2

Background

This chapter provides a review of the current research on Deep Neural Networks, their optimization techniques and the state of art results for image recognition. Optimization methods will be first discussed followed by current DNN architectures.

2.1 (Shallow) Neural Networks

In the context of machine learning, Neural Networks are a class of differentiable algorithms [4]. One can think of them as a set of perceptrons aligned into several layers having outputs from layer L mapped to inputs of the layers $L+1$. The output of each perceptron is named activation and the set of activations in the final layer gives the desired classification output. For each connection, a neuron has one weight connecting with all neurons of the next/previous layer, and one bias. The goal is to find the values that minimizes a given cost function. For instance a network could have 10 neurons in the last layer that would map to a digit classification problem using the MNIST [18] dataset. The neurons from 1-10 on this layer corresponds to each number, and the one with the highest activation value would be chosen as the classification output.

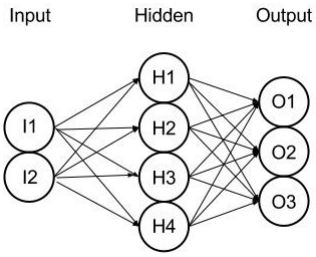

Figure 2.1: Multi-layer Perceptron

2.2 Gradient Methods and Backpropagation

As most machine learning algorithms, Neural Networks models rely on optimizations of weights ω and biases β. In order to calculate these values one should choose a cost function that measures the variance between the actual result and the desired output on each iteration of the training phase. The two methods for running this optimization algorithm on Neural Networks are known as *Backpropagation* and *Gradient Descent* [1]. The main goal of the algorithm is to propagate small changes δ applied to any of the neurons of the network all the way to the final layer.

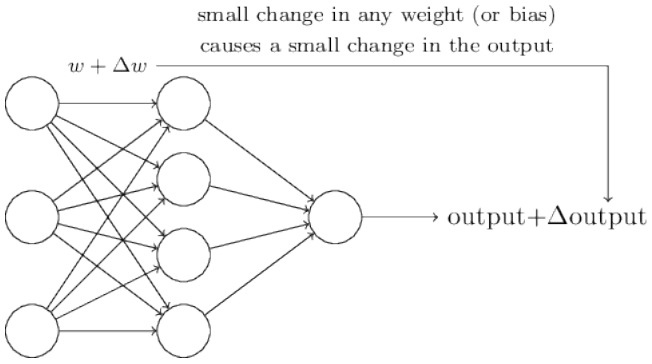

Figure 2.2: Output change with regards to layer(s) weight change [22]

The network learns from data using convex optimization techniques that involves calculating derivatives of linear and polynomial cost functions.The learning algorithm should be able to find weights and biases that best approximates the output *y(x)*. This approximation consists of finding the global minimum of a chosen cost function. Different machine learning algorithms require different cost functions [22]. Ultimately, one should be interested in calculating the change on the cost with respect to the weights and biases of all the neurons. This calculation makes use of partial derivatives with respect to the ω and β in order to find the direction of the minimum for a given function $f(x)$, namely gradient [1].

$$\nabla C \equiv \frac{\partial C}{\partial \omega}, \frac{\partial C}{\partial b}$$

Gradient Descent is one of the methods for finding a global/local minimum of a function. This calculation yields what is a called a gradient vector ∇C which is subtracted from current weight and biases on each iteration in order to move the result towards the global or local minimum. The gradient calculation can be seen as repeatedly computing ∇C, and then moving in the opposite direction "falling down" the slope of the valley.

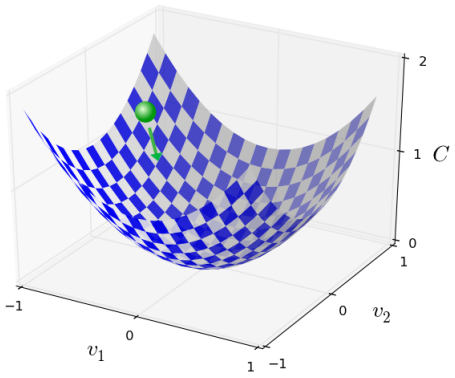

Figure 2.3: Gradient calculation representation [22]

In order to learn the best representation of a given data set a Neural Network should go through the optimization process where each data point is used many times to estimate the network current weights and biases. Each iteration pushes the ball into the direction of the valley by comparing a given y(x_n) with its corresponding real y_n. The process should be stopped when the ball hits the lowest point of the valley, in other words, when the algorithm has converged. Mathematically speaking this is known as reaching a minimum of a function [22]. Some optimization problems are unable to find a global minimum due to their complexity, on this case, convergence is achieved by finding the local minimum.

However, when the data set is too big for running one update after a complete loop over the entire data, one should come with better strategy for updating the network parameters. Such method is that where each iteration runs on a subset of the training input namely *Stochastic Gradient Descent*. One of the biggest advantages is that to speed up learning while the estimation of the gradient ∇C is being computed over a subset of the data instead of all training inputs. The subset is chosen randomly on every iteration and can be referred as mini-batch. These batches are selected every *epoch* and the user provides how many epochs the algorithm should run. Due to the large amount of parameters, neural networks used in Deep Learning make use of this method so training can happen within feasible time frames.

The training is comprised mainly of two different types of calculations: Feedforward and Backpropagation. Feedforward is the starting point of the network weights and biases optimization.The goal is to calculate the activation of each neuron from the first layer to the output layer. It involves evaluating the activation function with

respect to the current weights and biases of the network by forward-propagating x through the entire architecture. This operations yields an error δ, which triggers the backpropagation part. When feedforward reaches the last layer, the error is then calculated and backpropagated to the L-1 layer. For each layer, one should find the rate of change of the cost function for each of the weights and biases with respect to its neurons. This operation is repeated subsequently until the first input layer, where the current "belief" of the network is updated making it ready for another feedforward calculation. The overall process should stop when there is no relevant changes in the output of the cost function.

2.3 Convolutional Neural Networks

Convolutional Networks are a class of deep learning algorithms that can use many layers of nonlinear or linear processing in cascade for feature extraction and transformation. They are still very similar to ordinary Neural Networks (made up of neurons that have learnable weights and biases). However, such algorithms makes the explicit assumption that the inputs are images and, therefore, are based on learning abstract representations of data with spatial properties [1]. For example, an image could be represented by a vector of intensity values from 0-255 for each pixel and after being processed by the first layers of a Convolutional Neural Network those would become more abstract representation such as set of edges, regions of particular shape and etc [13].

A convolutional neural network is different from the traditional neural network. Instead of connecting each pixel of an image, for instance, to all the neurons in the next layer, groups of pixels of fixed sizes known as patches, are connected to different groups of neurons. Each group specializes in learning specific features from the data and are not necessarily connected with all other groups in the current or next layer. The input region or group of neurons connected to a patch in the image is known as local receptive field [22]. The main advantage over shallow architectures is that the latter does not take into account the inherent spatial structure of images, in other words, all pixels are treated equally.

The main architectural difference of ConvNets is that they have added some different layers to the traditional Neural Nets mix. The name "convolutional" originates from the Convolution layer which is responsible for computing the output of neurons that are connected to local regions in the input. This operation is commonly referred in image/signal processing as a convolution between two matrices/vectors of variable

sizes. The smaller regions in which the image is connected is referred as filters and/or kernels, and these would be responsible for detecting abstract features on different regions of a picture for instance. After each sequence of convolutional layers there are also the Pooling Layers. The main goal is to perform downsampling operation along spatial dimensions of an image (i.e width and height). This is important as an image with higher dimensions will be reduced at each layer forcing the network to learn deeper and deeper features at each iteration.

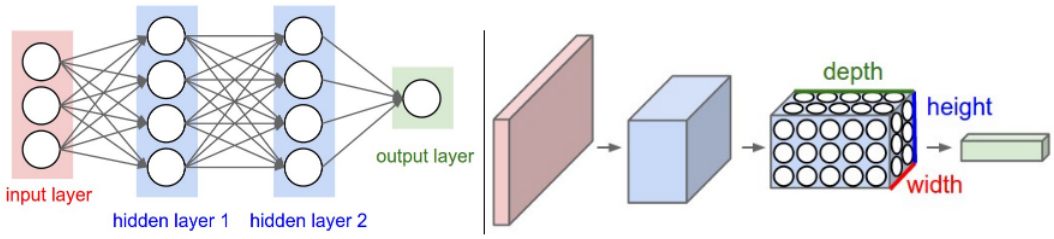

Figure 2.4: Shallow Neural Network vs Convolutional Neural Network [13]

Convolutional Neural Networks are considered the state of the art solution for computer vision problems. The archutecture developed by Krizhevsk et al. (2012) achieved an averaged top-1 and top-5 test set error rates of 37.5% and 17% where the previously record was 45.7% and 25.7% by a different technique. The network was comprised of eight layers, 5 of which were convolutional layers and the remaining were three fully connected layers. The output of the three last layers was fed to a 1000-way softmax to produce 1000 different image classes labels of the ImageNet dataset. Besides, max-pooling layers were also used following the fifth convolutional layer and response-normalization layers. Adding or removing any layers on this architecture yielded worst results. Overfitting was treated by using both Data Augmentation and Dropout [11] techniques since the architecture had over 60 million parameters. These results show that deep convolutional networks are capable of achieving above the average results even when challenging datasets with several classes are used.

2.4 CNN Feature Learning and Visualization

Visualizing the features of a convolutional layer helps one to understand what are the main characteristics of the input that are not only being learned but are also responsible for maximizing neurons activations citezeiler2014visualizing. Until recently there was no clear understanding of why Convolutional Networks perform so well on

image classification tasks. Zeiler et al (2014) have produced a novel visualization technique that gives insights into the outputs of the intermediate feature layers of the network. These visualizations were then used to perform diagnostics of models and, thus, find architectures that can outperform current state of the art algorithms. One can understand that all the convolutional layers are in fact "feature extraction" layers while the fully connected ones are the actual classifier.

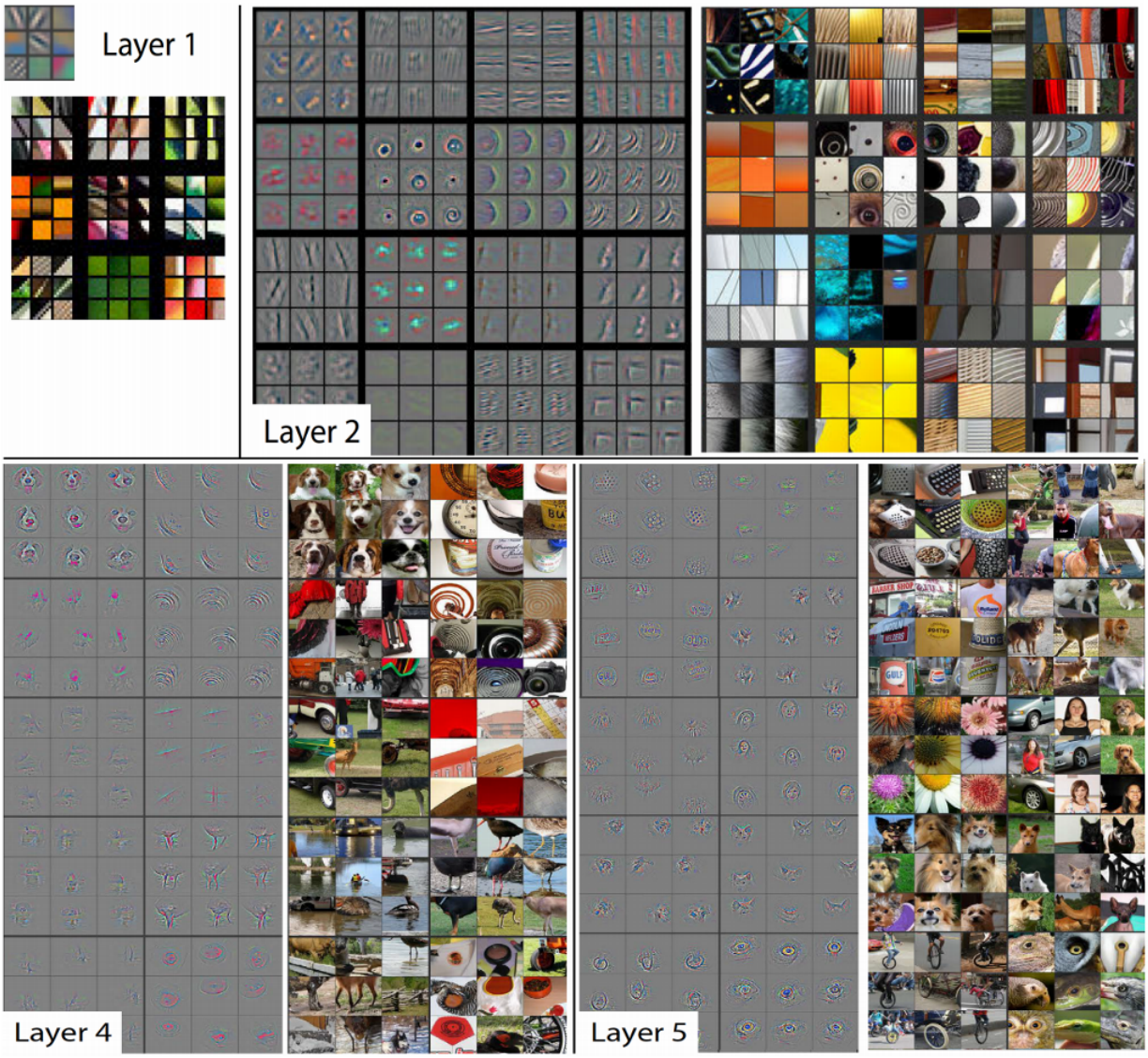

Figure 2.5: CNN layer visualization
[30]

Features are learned from more abstract on initial layers to more specific and concrete on deeper layers. Figure 2.5 shows that the first neurons are learning different types of edges while later neurons are capable of learning full representations

of important regions of an image. In general these features are applicable to many datasets and tasks. As features transition from general to specific, transferability becomes negatively affected [29]. For instance two similar datasets as the ImageNet and Cifar10 would share many common features and, therefore, a network trained on one, could be used as a pre training or weight initialization process for another, increasing the generalization and accuracy of the model.

2.5 CNN Architectures

The use of Deep Neural Networks (DNNs) as a prominent technique was mainly possible due to the standardization of the dataset used for such tasks. ImageNet has become the most used classification challenge for those wanting to find better CNN architectures. The main goal of such challenges are to find models with better accuracies without focusing too much on resource utilisation. One one hand, the most important metric of a good machine learning model is its accuracy, in other words, how well it can generalize on different datasets. On the other hand, several other important metrics should be taken into consideration when using an specific DNN system in real life such as: memory footprint, parameters, operations count and etc [5].

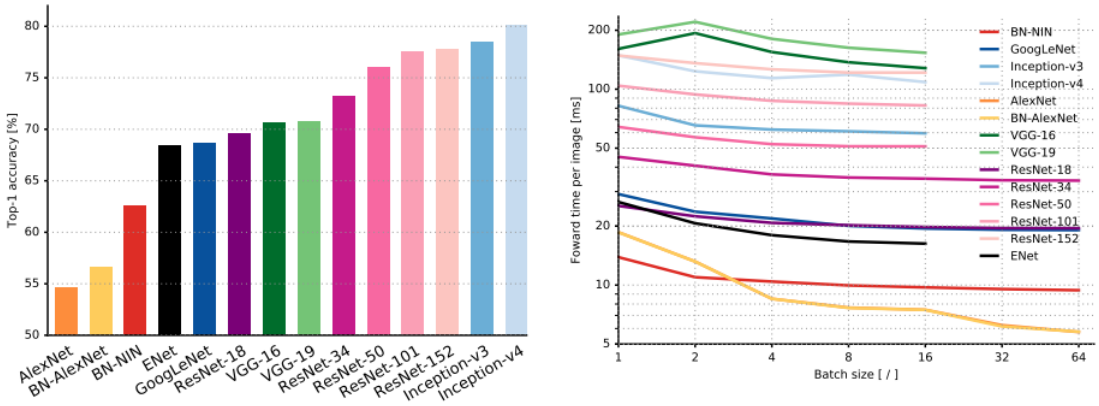

Figure 2.6: (Left) Accuracy of Different CNN models (Right) Inference time for different batch sizes

[5]

As new methods develop, one should always pick the one that is capable of not only learning the features of the target domain but also would fit the resource capacity of the system. From one of the first CNN architectures (AlexNet) to the latest models

(Inception V4), the accuracy has improved from 55% to almost 80% on the ImageNet dataset (comprised of 1.000 different class labels).

2.6 Deep Neural Networks Properties

Deep Neural networks can be considered models with high expressiveness that can achieve extremely good performance on computer vision tasks. At the same time that being highly expressive helps them to succeed, it also drives them to learn solutions that are not easily understandable [28]. Usually there is a discontinuity in the input-output mappings of neural networks which can lead to miss-classification of images when the network prediction error is maximized [10].The learning process of these networks through the use of backpropagation is rather complex and sometimes difficult to understand.

In order to visualize the semantic meaning of individual units, studies are currently focusing on understanding the factors leading to the activation of network neurons. It has been argued that deep neural networks should be stable enough to provide robustness to small perturbation of its inputs [28]. However, it has been found by mainly Goodfellow et al. (2014) and Szegedy et al. (2013) that minimal local perturbations can indeed affect the network's predictions bringing down the assumption that DNN have very good local generalization. Methods for exploiting this vulnerability were created and proven to be effective by having a very high confidence classification of adversarial examples [16].

Generalization is usually achieved by making non-local assumptions of the training inputs. Deep stacks of non-linear layers are one of the ways to have the model encoding a non-local generalization prior over the input space [10]. Therefore, output units should be able to assign low probabilities to regions of the input space where no training examples are found within its vicinity. The representation of low-probability "pockets" of space on images can lead to the creation of Adversarial examples. These are created by adding small localized perturbations on the input space which can ultimately lead to the wrong classifier outcome.

The following sections are going to focus on Adversarial Examples and how the exploitation of the aforementioned network properties can be used to craft such examples. Three methods will be presented along with results found by different studies. Finally, methods for using this adversarial information for regularizing networks will also be shown as a possible solution for making deep neural networks less vulnerable to these kind of attacks.

Chapter 3

Adversarial Examples Taxonomy

This chapter focuses on Adversarial Crafting and how these examples can be used to exploit some of the Deep Neural Networks caveats seen in the last chapter. Firstly the Fast Gradient Sign method will be explained along with some results. Secondly, a brief discussion on the empty pockets of space created by neural networks allow such technique to create adversarial. Finally a discussion of the potential threats these pose to systems relying on machine learning algorithms to perform their tasks.

3.1 Foundations

Understanding of why adversarial samples can exist requires exploration of how learning models are built. The training data is a corpus of samples taken from a expected input distribution and are labeled accordingly to their desired class. For instance, this sample data would be a large number of emails or a huge data set of images. The labels are then taken as ground truth when constructing models to be used at runtime.

The integrity of deep learning systems usually is that of that measures how accurate the system is when performing a classification task. This metric is of paramount importance and, therefore, should be a common target for techniques trying to exploit such algorithms vulnerabilities. Specifically, an adversary of a deep learning system seeks to provide an input X' that results in incorrect output classification. The incorrectness of the prediction can be represented into different natures and can impact the classifier output in different ways.

Adversary drivers could be explained into four goals as discussed by Papernot(2016) [23]. Confidence reduction is the adversary potential to introduce class ambiguity by reducing classification confidence. Misclassification happens when a label of the model being previously correct is changed to an incorrect output label. On the same

way, one could use a Targeted misclassification to produce inputs that forces outputs into a specific label. Finally, the source/target misclassification forces the output classification of a specific input to be a specific target class.

3.2 Domain Shift

Regardless of the technique, a machine learning model represents an approximation of the phenomena being modeled. In most cases the training data is unable to represent all possible input feature vectors and, therefore, can not full capture a complete understanding of the target domain. A problem arises when input examples are able to exploit the system by providing samples that are not within the aforementioned input domain. They usually use information about the system to find where the model is inaccurate owing to missing items of the given training set.

Classification accuracy should be carefully measured when training a model. For instance, the value of the training set is usually higher than the one on the test set. This happens when the samples of the training can not cover the entire data distribution space and therefore the domain covered differs from the one on the test set. A poor performance on the test set means that the divergence of both distributions (training and test domains) is high.

What adversaries do is to force the domain shift in a way that the model is unable to generalize well on test data. Since data in almost circumstances can not cover the entire feature space, the real decision boundary of a classification model generally becomes more complex as the phenomenon becomes more nuanced and the feature and dimension space becomes larger. This complexity is exploited by adversaries through the use of the model error as a guideline for perturbing a sample.

3.3 Fast Gradient Sign Method - FGSM

The Fast Gradient Signed method developed by Goodfellow et al. (2014) has been used as the foundation of many of the experiments in adversarial crafting. The results have led to the hypothesis that DNNs can possibly have linear behavior when in very high dimensional space. Most inputs were miss-classified not only on Goodfellow et. al [9] experiments but many other. This shows that adversarial examples are not hard to find. The method consists on using gradient information to generate image noises that changes classification outputs.

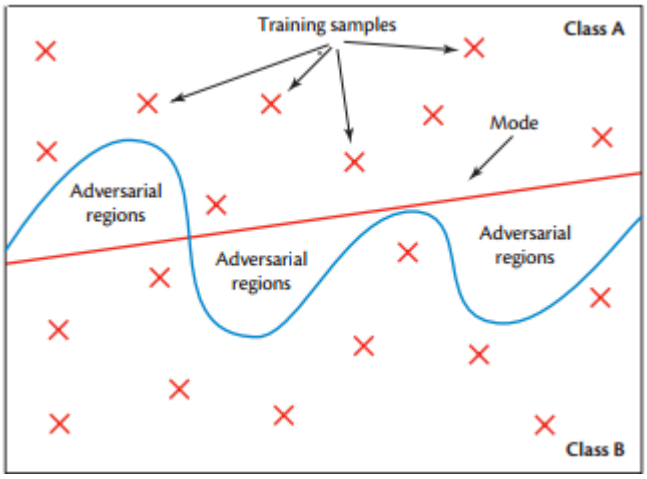

Figure 3.1: Two Dimensional Representation of unexplored adversarial regions [24]

$$C(x + \delta) \approx C(x) + \delta * \nabla C$$

The equation aims into adding noise that emphasizes the pixels in the image with the highest importance, so the resulting perturbation can likely lead to a misclassified result. By using the *(*sign) function of the gradient, it is assured that the value will grow linearly with the number of the dimensions [9]. The result of many small pixel changes is likely to generate an image with a wrong label in the network output.

$$C(x + \delta) \approx C(x) + \delta * sign(\nabla C)$$

The Gradient Sign is an exploitation of the loss function in an optimization process so one can maximize any class score for a given input image. Since everything in a ConvNet is differentiable it is relatively straight forward to ask for the gradient information of any specific class of the input domain. The process can be done by doing a forward pass on a network with the desired class output being set to 1 in the final layer, then the backpropagation retrieves the necessary changes on gradients that would make any image looks like the desired class.

Billovits et al (2016) categorized four different categories of adversarials generated by FGSM. True Adversarial are those given a completely different label after being perturbed. Re-Focused adversarial is the method that changes the focus of an image by giving a classification of an object that used to have lower significance while keeping the original object presence. Conaturally Adversarial are those where the new output has some close relation to the miss-classified result (e.g. Dog and Cat). Finally, Benign adversarial happens when neural networks misses the top prediction

of the original image but the adversarial example gets classified correctly with high confidence [3].

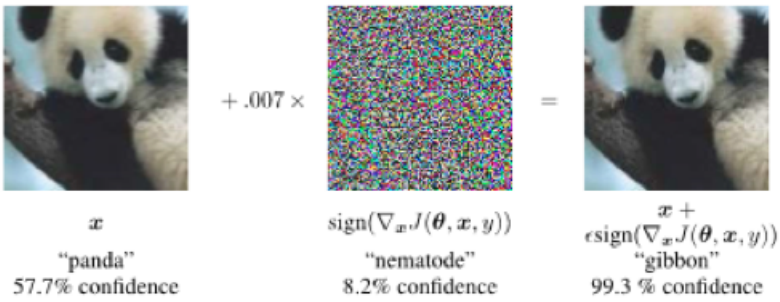

Figure 3.2: Adversarial example crafting with fast gradient sign [9].

3.4 Iterative Gradient Sign Method - IGSM

The method from the previous section shows that small perturbations can intentionally change classifiers class labels. However, this results strongly depends on how "confident" the network is regarding a specific class domain. For instance, one could have trained a network with better samples of a specific class and, therefore, would have led to a better feature space exploration of that specific class. Sometimes, more than one iteration of the Gradient Sign method is needed in order to have an image being incorrectly classified. By progressively apply small perturbations to an image one could achieve miss-classification of the desired sample that was not affected by one single iteration [16].

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\Big\{\boldsymbol{X}_N^{adv} + \alpha\,\text{sign}\big(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{true})\big)\Big\}$$

Iteratively repeat

Figure 3.3: Iterative FGSM Method [16].

The extension of the fast method is reached by applying the gradient sign multiple times and clipping pixels that are not in the boundaries of the original image. Figure 3.4 shows the nature of perturbations of both methods. Since the FGSM applies a single batch of perturbation to the image, it needs a higher amount of noise at a time to be able to successfully generate an adversarial.

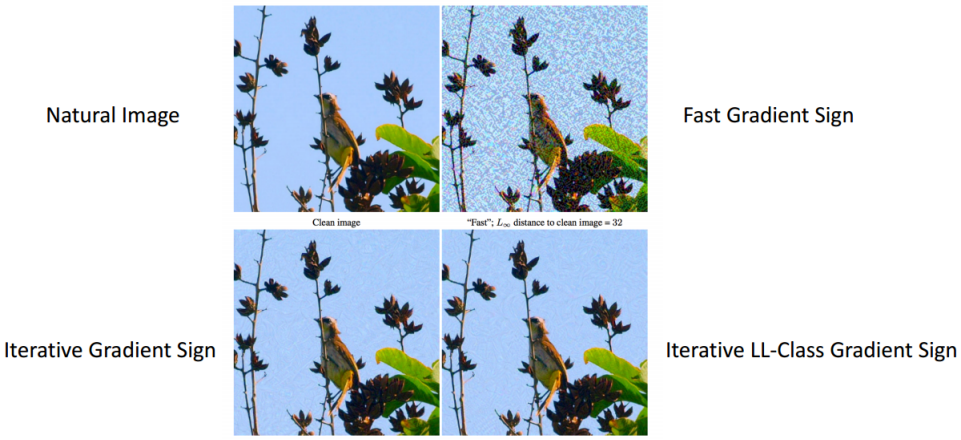

Figure 3.4: Visual Comparison of Gradients-based Methods [16].

3.5 Ascent and Descent Perturbations

Adding or subtracting noise from images would have different generated adversaries. By adding noise to an image, one would be making the gradient to go "uphill" and therefore moving away from the desired class, this results in an increase in value of the loss function and thus should be referred as the Ascent Method. On the other hand, when moving the opposite direction (down), one would be doing a process similar to the optimization of a loss function where we approach the minimum of a function and, thus, we get closer to the desired class, this approach is hereby known as the Descent Method. These two methods can be coupled with iterative methods or fast to take progressive/single steps towards a class directions. Some images/classes would be more robust to these perturbations and would, therefore, require more iterations.

$$C(x+\delta) \approx C(x) + \delta * \nabla C - Ascent$$

$$C(x+\delta) \approx C(x) - \delta * \nabla C - Descent$$

Chapter 4

Attacking Machine Learning Systems

In this chapter, we present approaches for attacking machine learning algorithms with adversarial techniques presented in the previous chapter. We discuss that the knowledge of the architecture and weight parameters is sufficient to derive adversarial samples against DNNs. Further discussion goes into black box attacks where the attack has minimal information about the underlying system. The discussion is then closed with how model's knowledge can be transferred between different algorithms/techniques.

4.1 Transferability

Papernot (2015) presented that many adversarial examples crafted to fool one specific model are also likely to affect another different model. As long as the models were trained to perform the same task, knowledge can be transferred when querying the victim model, namely oracle, to label a synthetic training set for the substitute.

The machine learning transferability property constitutes a threat vector for many state of the art methods, thus, one should be able to quantify most vulnerable classes of models by generating accurate comparison of the attack surface of each class. Attacks can, however, depend on some specific information of the target as shown in Table 4.1. Techniques can mainly be split into both Intra-Technique and Cross-Technique. These are discussed in more details on the following sections.

Model Parameters (Weights and Biases)	Have full access to model wieghts
Model Architecture / Type	Know what the model looks like
Training data	Understand the data domain
Oracle/Black box	Query model with input X, get label Y

Table 4.1: How Much Information is needed to fool a model?

4.2 Intra-technique transferability

The Intra-technique transferability is done by reproducing the learning process between two identical algorithms [25]. Even though, the algorithms can differ in terms of architecture, they are still based on the same fundamental learning concept. For example, algorithms could be categorized into three different classes: differentiable algorithms like DNN and Logistic regressions, lazy learners like KNN and non-differentiable models like SVM and Decision Trees. Therefore, this technique consists of keeping the same learning method while differing the hyperparameters/architecture and using queried subset of the training data to train the local model.

In order to make a comparison between these techniques, Papernot N. et. al (2015) [25] created five different dataset models of the MNIST to train the algorithms and compare how they perform when using different and same models of training data. All models had non-negligible vulnerability to this kind of approach. While DNN and LR were highly vulnerable to these attacks, SVM, DT and KNN were more robust achieving better overall resillience. The results have led to the hypothesis that non-differentiable techniques are more robust to black-box attacks using locally generated adversarial sample in between two algorithms of the same type [26]. Figure 4.1 shows classification performance when using intra-technique methods.

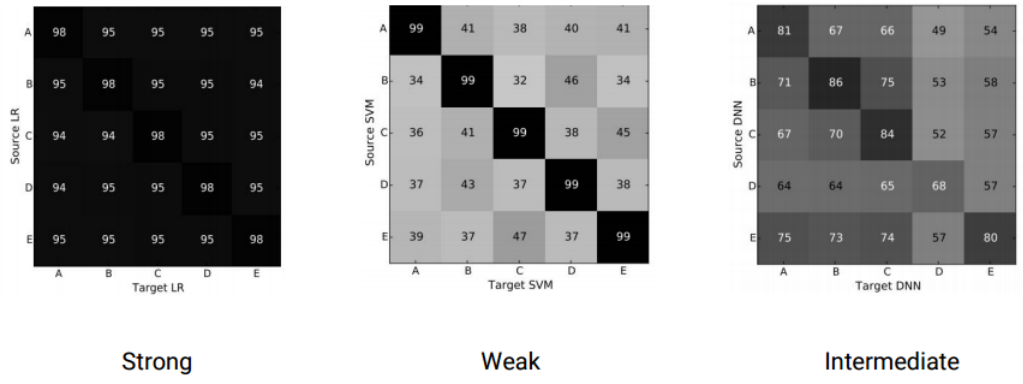

Figure 4.1: Knowledge transfer level using Intra-Technique Transferability [25]

4.3 Cross-technique transferability

Cross-Technique transferability was referred as the knowledge transfer between two different machine learning techniques. This problem has a higher degree of difficulty than the method shown on section **??** as it involves models using possibly very different techniques like DNNs and Decision trees. Yet, this can be seen as quite strong phenomenon to which techniques like Logistic Regression, Support Vector Machines and Decision Trees along with Ensemble based models are extremely vulnerable [25].

Papernot N. et. al [25] have shown a strong but heterogeneous phenomenon. While DNN's ended up as being the most robust of the methods with misclassification rates varying between 0.82% and 38.27%, Decision Trees were the most vulnerable with rates from 47.20% to 89.29%. Interesting enough, ensemble methods – focused on measuring the output of all the "experts" in the group – have shown quite vulnerable to the experiment. The hypothesis is that the technique explores the individual vulnerability within each of the ensemble methods.

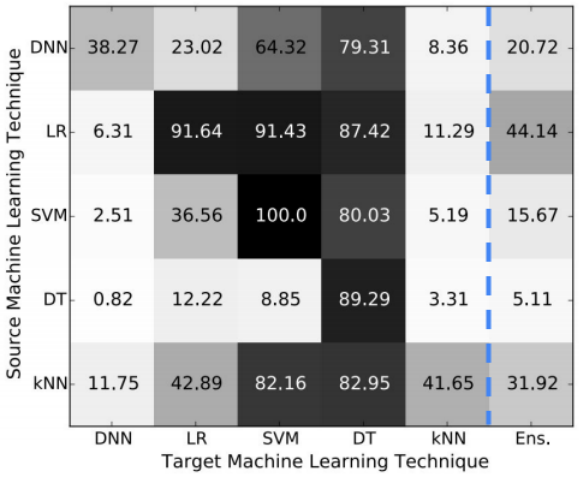

Figure 4.2: Knowledge transfer using Cross-Technique Transferability and Ensemble Methods [25]

4.4 Black-box Attacks

Black-Box attack to machine learning systems alleviates the dependence on knowing both victims training data and model information. This method solely depends on accessing the label assigned by the target for any chosen input. The strategy consists

of learning a substitute for the target model using a synthetic dataset generated by the adversary and labeled by the observed victim, namely here, the Oracle [26].

Training the substitute model that approximates the Oracle poses some challenges. Selecting an architecture for the substitute ends up in being an arbitrary process, as one should try different models and evaluate the one with the best result. Generating the synthetic dataset needs to limit the number of queries sent to the oracle so the approach is tractable.

Experiments from Papernot et al. (2016) were performed against real-world remote systems in order to validate the effectiveness of such attacks. The results have shown that systems using DNNs are usually more robust and require more queries to have the substitute being able to generate samples that are misclassified by the oracle.

Remote Platform	ML technique	Number of queries	Adversarial examples misclassified (after querying)
MetaMind	Deep Learning	6,400	84.24%
amazon web services	Logistic Regression	800	96.19%
Google Cloud Platform	Unknown	2,000	97.72%

All remote classifiers are trained on the MNIST dataset (10 classes, 60,000 training samples)

Table 4.2: Black-Box attacks results against real world systems

4.5 Unrecognizable Images

Adversaries, on the other hand, are not only comprised of small perturbations on known images. Nguyen et al (2015) presented a method for producing images that are unrecognizable to humans, but are nonetheless labeled as recognizable objects by DNNs [21]. For instance, a DNN would classify a noise-filled image crafted using their technique with high confidence. These images were named *foolingimages* since they do not have a source class but are crafted solely to perform a targeted misclassification attack.

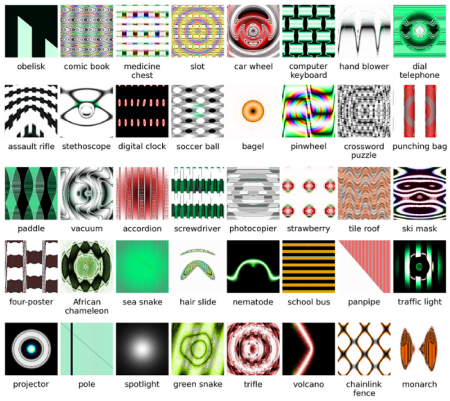

Figure 4.3: Examples of noisy images classified with high confidence [21].

4.6 Adversarials in the Physical World

All the aforementioned techniques were based into feeding information directly into machine learning systems. Such model only takes into consideration situations in which attacks take place entirely within the computer. For instance, these techniques could be used by attackers to avoid spam filters or malware detectors. Even though the study is utterly relevant, a recent study conducted by [16] have shown that it is possible to craft adversarial samples in order to perform attacks on machine learning systems which are operating in the physical world.

Although the Fast Gradient Sign method have been successful on crafting adversarial examples, there are some extensions of the method that can be used in order to create perturbations that are more likely to work in the physical world. Firstly, [16] introduced a variation named *Basic Iterative Method*. This technique consists of applying the fast method multiple times with small step sizes and making sure that all pixels are within a ϵ-neighbourhood of the original image. The number of iterations was chosen heuristically with goals of being sufficient enough to have the adversarial example reaching the edge of the ϵ max-norm.

In order to perform experiments, clean photos of adversarial examples created using the three methods were taken and fed into a machine learning system using Deep Convolutional Neural Networks (Inception V3). Adversarial Images created using the "fast" method were more robust when compared to the iterative methods. The hypothesis behind the result is that iterative methods create more subtle perturbations that can be easily be destroyed by the photo transformation process (Photo Printing as described above). Overall, it could be expected that about 2/3 of the images would be top-1 misclassified and about 1/3 top-5 misclassified by the fast method using an

ϵ of 16.

Adversarial examples is not only feasible on digital attacks but also on physical world scenarios. By using the correct perturbation algorithm with optimized hyper-parameters one can use printed digital photos to fool day-to-day machine learning systems. As more and more machine learning is becoming part of our environment, techniques for avoiding such attacks need to be developed so these systems can become less vulnerable to any kind of attack.

4.7 Defending Against Adversarial Attacks

Since adversarials are exploiting intrinsic network properties, these could also be used when training a network in order to develop robustness to possible examples crafted using the same methods. By using the worst case perturbation of a point x instead of x itself it is possible to derive an equation that includes the perturbation within its objective function. This form of training was able to reduce the error rate of adversarial examples from 0.94% to 0.84% [9]. Adversarial training can be seen as a way of teaching the model how an adversarial looks like and that it should be able to generalize not only normal images but also perturbed ones. Another way of creating robustness was developed by using bayesian non-parametric methods. Estimating the confidence that an input is natural during the training phase can lead the network to generate priors that take into account adversarial perturbation of points [3].

$$C(\omega, x, y) = \alpha C(\omega, x, y) + (1 - \alpha)C(\omega, x + \epsilon sign(\nabla_x C(\omega, x, y))$$

Most adversarial construction techniques use the gradient of the model to make an attack. In other words, they look at a picture of an airplane, they test which direction in picture space makes the probability of the "cat" class increase, and then they give a little push in that direction. These are hard to defend against because it is hard to construct a theoretical model of the crafting process. These examples are solutions to an optimization problem that is non-linear and non-convex for many ML models, including neural networks. Since there is no good theoretical tools for explaining the solutions of these complicated problems, it makes it very hard to make any kind of theoretical argument that a defense can improve an algorithm from a set of adversarial examples.

Chapter 5

Experiment Setup

In this chapter, we present approaches for attacking machine learning algorithms with adversarial techniques presented in the previous chapter. We discuss that the knowledge of the architecture and weight parameters is sufficient to derive adversarial samples against DNNs. Further discussion goes into black box attacks where the attack has minimal information about the underlying system. The discussion is then closed with how model's knowledge can be transferred between different algorithms/techniques.

5.1 CIFAR and ImageNet Datasets

Improvements on Computer Vision were possible due to two main reasons: increased computational power and standardized datasets for benchmarking. The ever increasing amount of image data on the Internet fostered more sophisticated and robust algorithms to work on images and multimedia data. ImageNet is a large-scale ontology of images with 1.000 different classes and with 3.2 million 224x224 high resolution images in total [6]. CIFAR, on the other hand, is a more compact dataset with 32x32 colour images in 10 or 100 classes. The dataset consists of 60.000 images with equal amount of samples per class [14].

Even though both datasets differ on image sizes and number of class labels, they have similar data domain. CIFAR can be seen as a subset of ImageNet dataset, therefore, convolutional filters from the latter could be used to improve performance of the former as shown by Yosinski et al.(2014).

Our experiment uses an imbalanced CIFAR10 on a VGG16 architecture initialized with ImageNet weights. This drastically improves the accuracy of the model when compared to one that was only trained on the CIFAR dataset. Having a better overall accuracy implies into a better feature space exploration, and, therefore a more

robust model. The main reason for this approach is that we want to test adversarial robustness on models that perform similarly to current state of the art deep learning systems.

5.2 The VGG Architecture

As described on Chapter 2, a ConvNet is a sequence of layers, and every layer transforms one volume of activations to another through a differentiable function. Three main types of layers are used to build these networks: Convolutional Layer, Pooling Layer and Fully-Connected Layer.A Convolutional Layer would compute the output of neurons that are connected to local regions in the input, an Activation layer (RELU, Sigmoid etc) will apply elementwise functions, the Pooling layer performs downsampling operations along the spatial dimensions of the input and, finally the Fully-connected layer computes the class scores of the classifier. The chosen architecture should have enough layers for learning good features from the training set domain. For instance, a ConvNet Architecutre for CIFAR-10 could have the following configuration: [INPUT - CONV - RELU - POOL - FC].

Network architectures with higher accuracies have better generalization over the overall data input domain, however, adversarials are efficient into extrapolating the given domain by going to previously unseen regions of space. For this work we had to make a choice of which network architecture would not only provide reasonable accuracy for our selected data domain (CIFAR-10) but also would be reasonable on the resource consumption. This way we can reproduce the requirements of a real world system where not only accuracy is taken into account when selecting a deep learning model.

From all networks tested on Canziani et al (2016), the VGG16-19 from Simonyan et al. (2014) seemed to have the best trade off between accuracy and performance (inference time). The VGG architecture won the first and second places on the ILSVRC-2014 submission on the localisation and classification task.

The main contribution of the VGG network was showing that the depth of the network is a critical component for good classification performance. The model can be assembled with 16 or 19 Conv/FC layers and it features an extremely homogenous architecture that only performs 3x3 convolutions and 2x2 pooling from beginning to end.

The design of the VGG16 had in mind the ImageNet dataset, where there are 1.000 different classes. The two last fully connected layers of the state of the art model

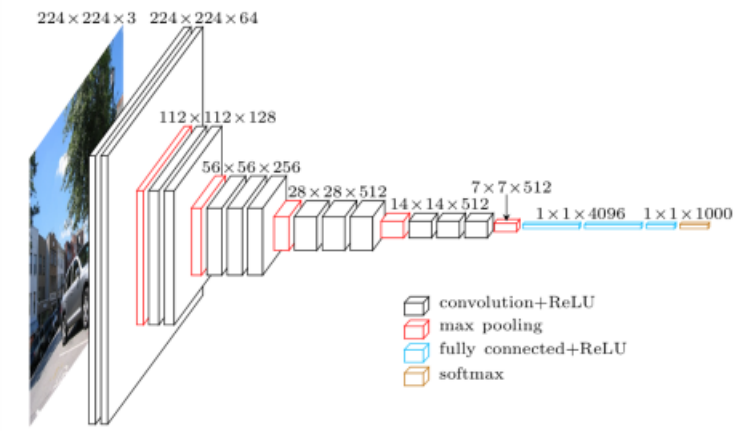

Figure 5.1: VGG16 on ImageNet
[27]

are comprised of 4096 neurons each, leading to much higher parameters complexity. As the dataset used in this work has only 10 classes, the two FC-4096 layers were replaced by one single layer with 512 neurons and RELU activations. This change helped to reduce overfitting when training the network. In addition, the total number of convolutions blocks and pooling was reduced to 3, with the first layer having 2 stacked convolution layers followed by a max pooling of stride 2x2 and the last two layers with 3 stacked convolutions also followed by a max pooling of stride 2x2. The max pooling layers are responsible for reducing the image size by 50% every time an image passes through it. Since CIFAR-10 images are only 32x32, the original VGG16 architecture would end up having an output of shape of only 1x1 pixels at the last layer. In order to avoid this problem, the number of layers were reduced so to fit our dataset domain. The resulting shape fed into the fully connected layers is 4x4x128 (Width x Height x Channels) as it can be seen on table 5.1.

5.3 Training Process

Neural Networks progress was stuck for so many years due to the lack of computational power and good optimisation methods required to extract good performance from these methods. As networks get deeper, the number of resources required to train increases considerably. Even that the Convolutional Neural networks used in the work can share parameters within its convolution layers, we still need to use efficient gradient methods to be able to converge in a reasonable time. All the models comprised in this work were implemented using Python programming language along

Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 32, 32, 32)	896
block1_conv2 (Conv2D)	(None, 32, 32, 32)	9248
block1_pool (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
block2_conv1 (Conv2D)	(None, 16, 16, 64)	18496
block2_conv2 (Conv2D)	(None, 16, 16, 64)	36928
block2_conv3 (Conv2D)	(None, 16, 16, 64)	36928
activation_1 (Activation)	(None, 16, 16, 64)	0
block2_pool (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_2 (Dropout)	(None, 8, 8, 64)	0
block3_conv1 (Conv2D)	(None, 8, 8, 128)	73856
block3_conv2 (Conv2D)	(None, 8, 8, 128)	147584
block3_conv3 (Conv2D)	(None, 8, 8, 128)	147584
activation_2 (Activation)	(None, 8, 8, 128)	0
block3_pool (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_3 (Dropout)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
fc1 (Dense)	(None, 512)	1049088
activation_3 (Activation)	(None, 512)	0
dropout_4 (Dropout)	(None, 512)	0
predictions (Dense)	(None, 10)	5130

Total params: 1,525,738.0
Trainable params: 1,525,738.0
Non-trainable params: 0.0

Table 5.1: Full Model Description

with Tensorflow and Keras frameworks. The first is a high performance calculation engine that uses GPUs to accelerate its matrix/vector calculations. The latter is a Neural Network library that helps on the implementation of any deep learning model. Keras is mainly a wrapper on top of tensorflow that hides some abstraction from the developer, making it one of the best frameworks for DNNs currently.

As discussed on Chapter 2, it is not feasible to only update the gradients of the network after one full iteration over the entire dataset. Stochastic Gradient methods were one of the first methods developed to overcome this problem and are still being further developed nowadays. The SGD based optimisation technique used in this work was developed by Bengio (2015) [2], namely RMSProp. This method is an adaptive learning rate scheme that can take the absolute values of the Hessian's eigenvalues and, therefore, approximate the equilibration preconditioner. As shown on Bengio's work [2], the method outperforms current SGD methods by achieving convergence faster.The learning rate for the method was set at 10^{-4} and the decay 10^{-5}.

In order to achieve good performance, every algorithm should be trained until it converges. Avoiding overfitting and underfitting is highly important when training DNNs. Our architecture was trained until no more reasonable changes were detected in the validation loss so we could dismiss unnecessary training steps and consequently any kind of overfitting. This was achieved by using the Early Stopping technique as described on [13]. Fundamentally, this consists of a functional callback that runs at the end of every epoch and compares the previous loss with the current one and interrupts training if the difference was below a user provided δ for a specific number of steps in a row. The value of our δ was set at 10^{-4} and the number of steps to 10. For instance, training would be stopped if no improvement over the specified δ was seen for 10 steps in a row. Also, we did put a hard limit of 200 on the total number of epochs.

The confusion matrix helps understanding the individual score for each class and also which targets class were being missclassified into. Figure 5.2 shows that the Cat and Dog classes are often interchangeably missclassified as they have a set of similar features. The network have reached an overall accuracy of 83.45% and a total validation loss of 0.5033.

5.4 Class Imbalanced Training

Classification models are usually required to have similar number of samples of each class so it could equally learn feature representations for each label. As shown on

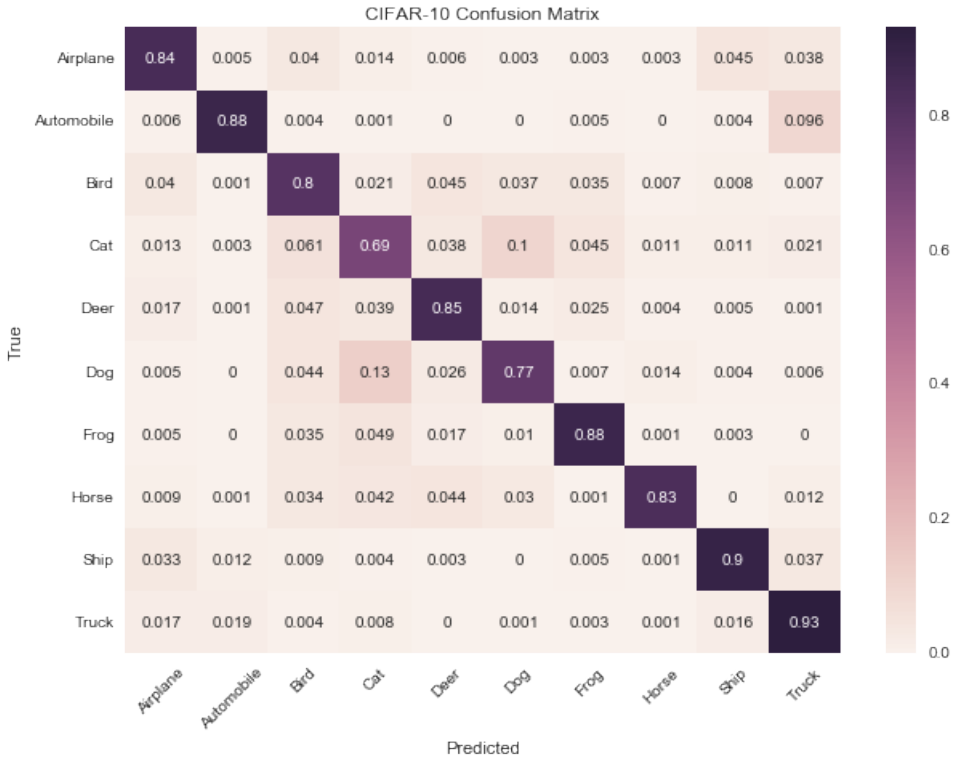

Figure 5.2: Results on the full dataset

Murphey and Guo (2004) [20], Neural Networks have lower generalization capabilities and are biased towards specific classes when trained on datasets with big number of samples differences between classes. Since our dataset of choice is not an imbalanced case, we randomly removed 4000 samples from each class at a time, creating 10 different datasets.

For each of the 10 different class imbalance dataset configuration, a network was trained until convergence using the same parameters as it was aforementioned. Each model was evaluated against a test set of 1000 equally distributed samples and the results are shown on table 5.2. These models will be the starting point for evaluating adversarial robustness later. The initial performance evaluation of each dataset was based on the Overall Accuracy, validation loss and class specific accuracy. Results are shown on table 5.2.

Class Label	Overall Accuracy	Validation Loss	Class-Specific Accuracy	Delta (%)
Airplane	80.26%	0.5731	69%	15%
Automobile	81.85%	0.5424	77%	11%
Bird	80.75%	0.5797	48%	32%
Cat	79.31%	0.5966	26%	43%
Deer	81.36%	0.5497	70%	15%
Dog	81.96%	0.5525	58%	17%
Frog	83.38%	0.5071	83%	5%
Horse	82.39%	0.531	69%	14%
Ship	83.33%	0.5133	80%	10%
Truck	83.48%	0.5215	80%	13%

Table 5.2: CNN Imbalanced Dataset Performance. The delta is calculated from the balanced network results on Figure 5.2

Chapter 6

Results

In this chapter, we present approaches for attacking machine learning algorithms with adversarial techniques presented in the previous chapter. We discuss that the knowledge of the architecture and weight parameters is sufficient to derive adversarial samples against DNNs. Further discussion goes into black box attacks where the attack has minimal information about the underlying system. The discussion is then closed with how model's knowledge can be transferred between different algorithms/techniques.

Chapter 7

Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Appendix A

Sample Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Appendix B

Sample Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bibliography

[1] Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[2] Yoshua Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization.

[3] Chris Billovits, Mihail Eric, and Nipun Agarwala. Hitting depth: Investigating robustness to adversarial examples in deep convolutional neural networks.

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

[5] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[7] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE, 1996.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[10] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[12] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[13] Andrej Karpathy. Convolutional neural networks for visual recognition. 2016.

[14] Alex Krizhevsky. Cifar-10 and cifar-100 datasets. 2009.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. Technical report, arXiv, 2016.

[17] Yann Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[18] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits, 1998.

[19] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5):555–559, 2003.

[20] Guo Hong Murphey, YI L. Neural learning from unbalanced data. 2004.

[21] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. IEEE, 2015.

[22] Michael Nielsen. Neural networks and deep learning. 2016.

[23] Nicolas Papernot. On the integrity of deep learning systems in adversarial settings, 2016.

[24] Nicolas Papernot. Machine learning in adversarial settings. 2017.

[25] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[26] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[29] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[30] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.