Effectiveness of adversarial examples on 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

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 [3] to tackle more complex problems such as the one of labeling digital images. The work of Lecun (1989) [13] 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 [11]. 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 [9] and, more recently, recursive neural networks [4]. 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 [15]. 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 [12]. 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 [6][20][12][21]. 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 [6].

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 Definitions

1.2 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 [5].

Although the biological inspiration for neural networks comes from understanding human vision, the work from Nguyen et al.(2015) [16] 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 black box attacks as long as they are trained for the same purpose [20]. 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.

1.3 Thesis Goal

1.4 Contributions

1.5 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 [3]. 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 [14] 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.

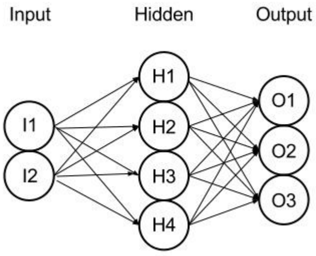


Figure 2.1: Multi-layer Perceptron

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.

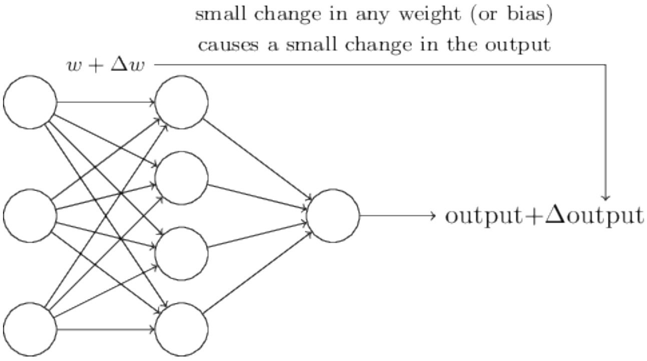

small change in any weight (or bias)
causes a small change in the output
$w + \Delta w$
output+Δoutput

Figure 2.2: Output change with regards to layer(s) weight change [17]

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 [17]. 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.

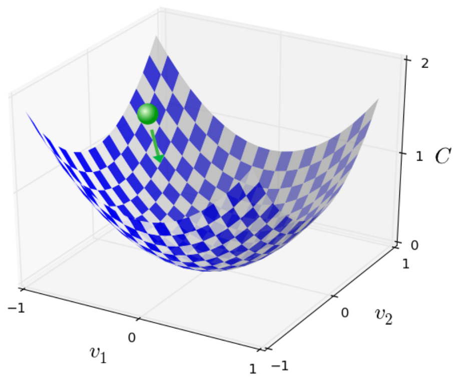


Figure 2.3: Gradient calculation representation [17]

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 [17]. 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 [10].

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 [17]. 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.

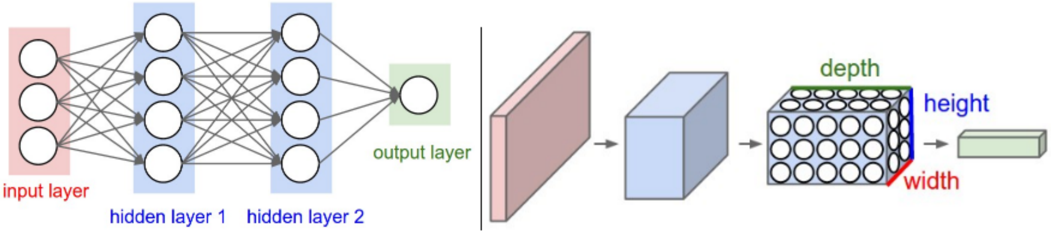


Figure 2.4: Shallow Neural Network vs Convolutional Neural Network
[10]

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 [8] 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.

2.3.1 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 [21]. 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 [7].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 [21]. 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 [12].

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 [7]. 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) [18]. 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

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 [6] 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.

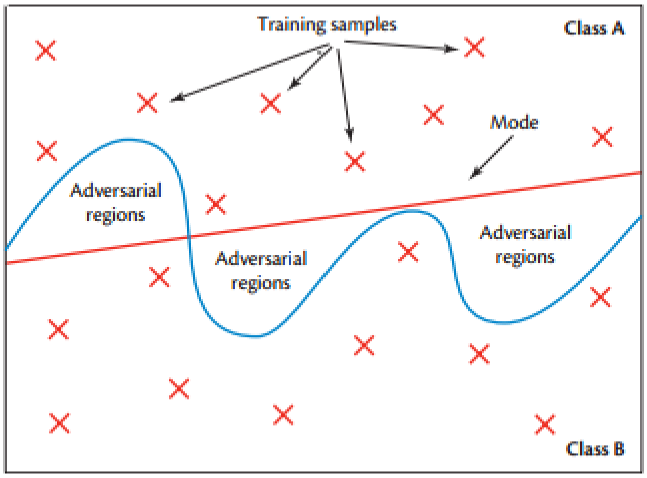


Figure 3.1: Two Dimensional Representation of unexplored adversarial regions [19]

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

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 [6]. The result of many small pixel changes is likely to generate an image with a wrong label in the network output.

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

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 hapeens when neural networks misses the top prediction of the original image but the adversarial example gets classified correctly with high confidence [2].

3.4 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

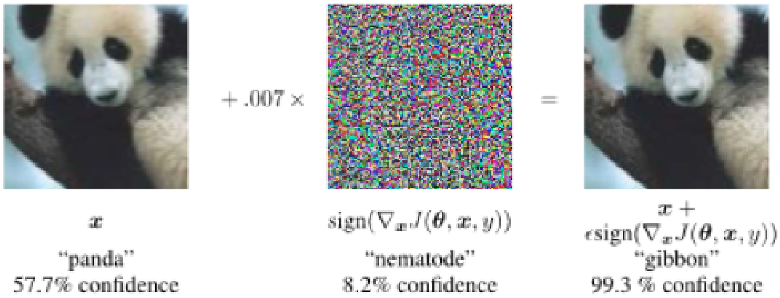

+ .007 ×
=
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} +$
$\epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 3.2: Adversarial example crafting with fast gradient sign [6].

are unrecognizable to humans, but are nonetheless labeled as recognizable objects by DNNs [16]. 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.

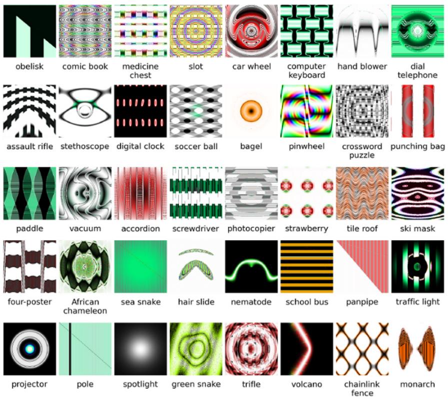


Figure 3.3: Examples of noisy images classified with high confidence [16].

3.5 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 [12] 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, [12] 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.

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 be mainly split into both Intra-Technique and Cross-Technique transferability. These are discussed in more details on the following sections.

4.1.0.1 Intra-technique transferability

The Intra-technique transferability is done by reproducing the learning process between two identical algorithms [20]. 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) [20] 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 [?]. Figure 4.1 shows classification performance when using intra-technique methods.

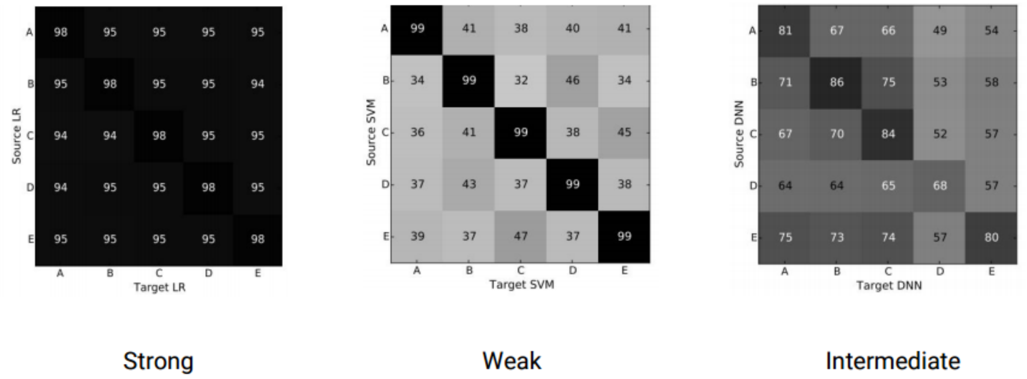
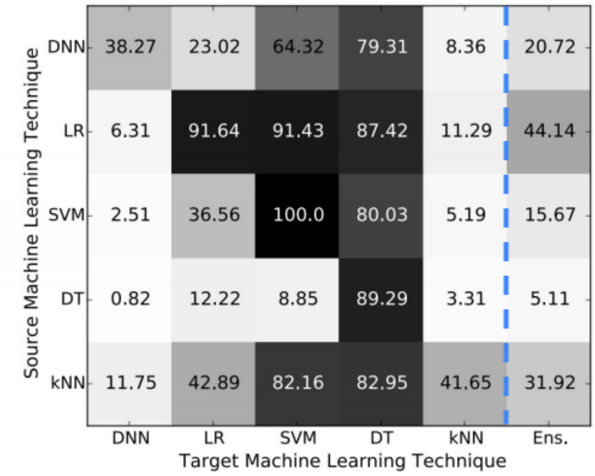


Figure 4.1: Knowledge transfer level using Intra-Technique Transferability [20]

4.1.0.2 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 4.1.0.1 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 [20].

Papernot N. et. al [20] 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.



	DNN	LR	SVM	DT	kNN	Ens.
DNN	38.27	23.02	64.32	79.31	8.36	20.72
LR	6.31	91.64	91.43	87.42	11.29	44.14
SVM	2.51	36.56	100.0	80.03	5.19	15.67
DT	0.82	12.22	8.85	89.29	3.31	5.11
kNN	11.75	42.89	82.16	82.95	41.65	31.92

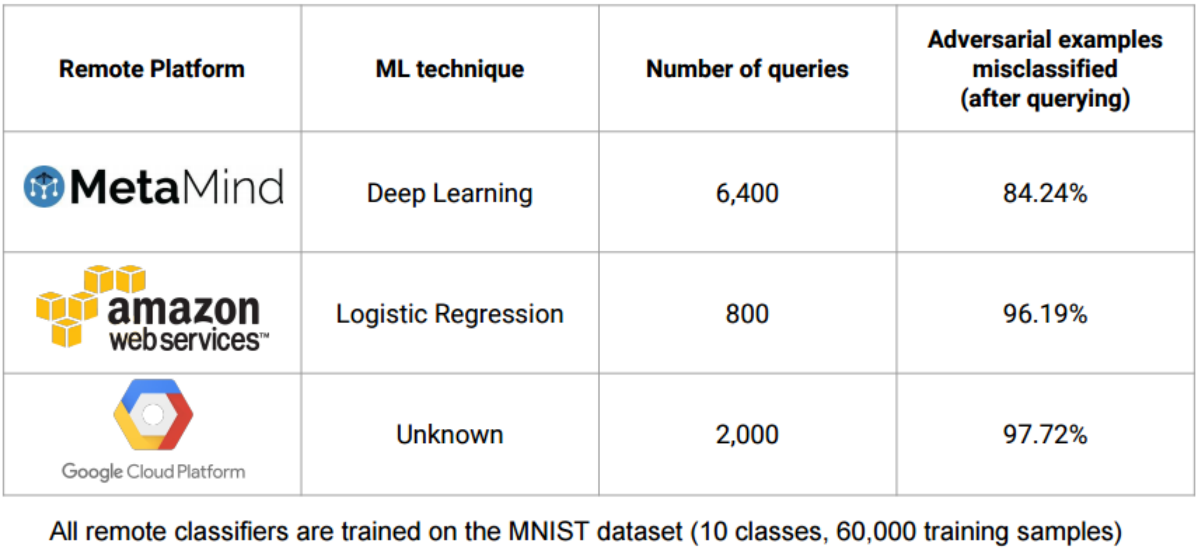
Figure 4.2: Knowledge transfer using Cross-Technique Transferability and Ensemble Methods [20]

4.2 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 [?].

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 webservices™	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)

Figure 4.3: Black-Box attacks results against real world systems

4.3 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% [6]. 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 [2].

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

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

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

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

[4] 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.

[5] 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.

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

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

[8] 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.

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

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

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

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

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

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

[20] 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.

[21] 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.