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 [1] to tackle more complex problems such as the one of labeling digital images. The work of Lecun (1989) [7] 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 [5]. 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 [4] and, more recently, recursive neural networks [2]. 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 [8]. 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 [6]. 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 [3][9][6][10]. 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 [3].

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 [**?**].

Although the biological inspiration for neural networks comes from understanding human vision, the work from Nguyen et al.(2015) [**?**] 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 [9]. 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 [1]. 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 [**?**] 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.

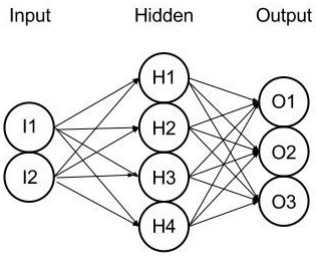

Figure 2.1: Multi-layer Perceptron

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* [**?**]. 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.

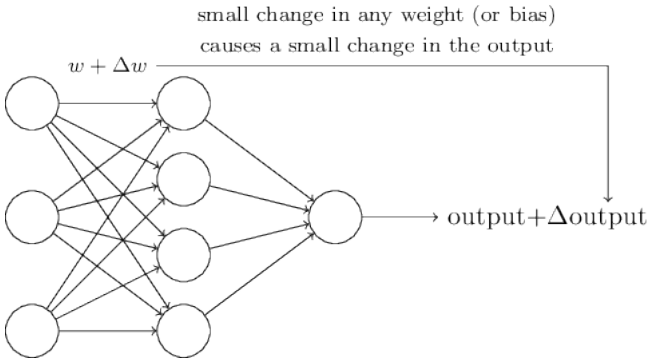

Figure 2.2: Output change with regards to layer(s) weight change [**?**]

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 [**?**]. 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 [**?**].

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

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.

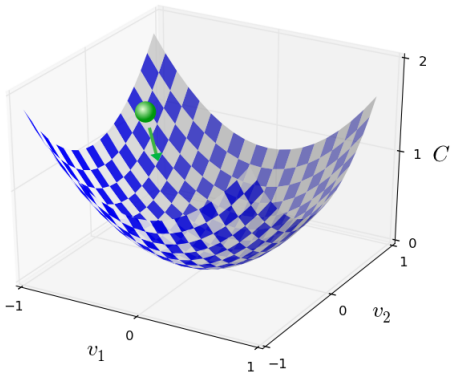

Figure 2.3: Gradient calculation representation [**?**]

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 [**?**]. 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 [**?**]. 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 [**?**].

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 [**?**]. 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.

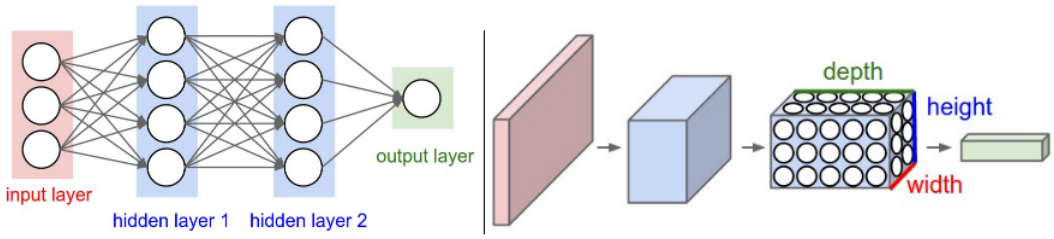

Figure 2.4: Shallow Neural Network vs Convolutional Neural Network [?]

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 [?] 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 [10]. 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 [**?**].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 [10]. 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 [6].

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 [**?**]. 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

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.

3.1 Neural Networks

In the context of machine learning Neural Networks are a class of differentiable algorithms [1]. 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 instance a network could have 10 neurons in the last layer that would map to a digit classification problem using the MNIST [**?**] 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.

3.2 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*

[**?**]. 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.

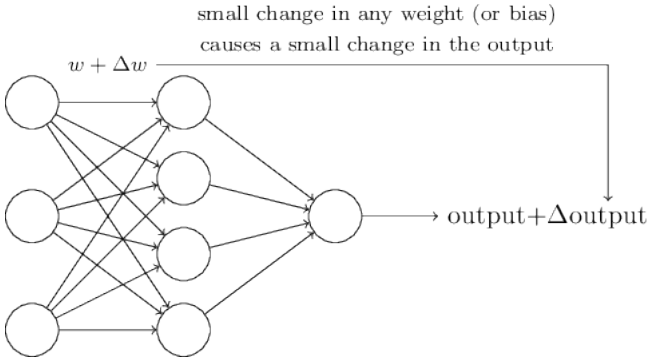

Figure 3.1: Output change with regards to layer(s) weight change [**?**]

A neural network learns from data using convex optimization techniques that involves calculating derivatives of linear and polynomial 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 [**?**]. 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 of a function, namely gradient [**?**].

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

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.

In order to learn the best representation of a given data set a training phase is firstly required. 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

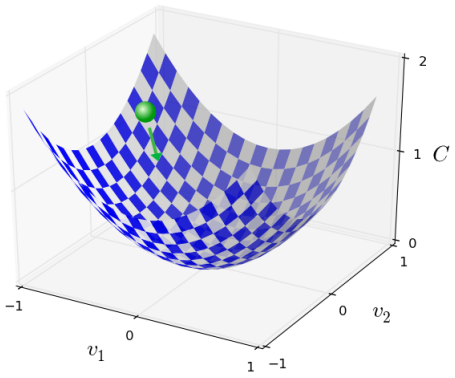

Figure 3.2: Gradient calculation representation [?]

function [?]. 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 dataset is too big for running one update after a complete loop over the entire data, one should use faster methods. 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.

3.3 Deep Neural Networks

Deep Neural Networks are a broad class of machine learning algorithms that can use many layers of nonlinear or linear processing in cascade for feature extraction and transformation. The 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 [**?**]. 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 [**?**].

3.4 Convolutional Neural Networks

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 [**?**]. 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 Conv-Nets 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 is 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.

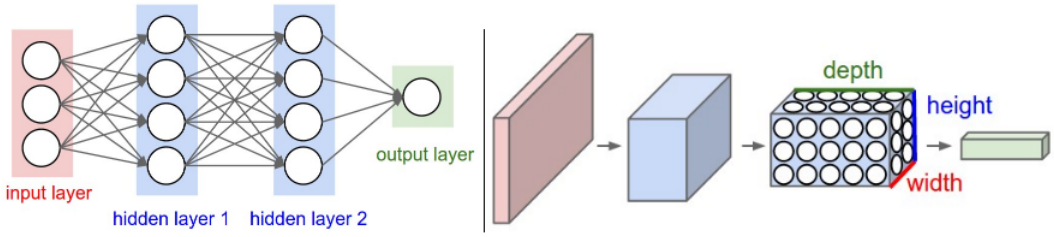

Figure 3.3: Shallow Neural Network vs Convolutional Neural Network [?]

Convolutional Neural Networks have recently been seen as the best approach 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 [?] 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.

3.4.1 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 [10]. 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 [?].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 [10]. 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 [6].

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 [**?**]. 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.

Chapter 4

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

[2] 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.

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

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.