ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

University of Zurich[UZH]

Understanding the Vulnerability of Artificial Neural Networks to Adversarial Examples

Shuchen Wu

Master Thesis

Theortical Neuroscience Group
Institute of Neuroinformatics
University of Zürich and ETH Zürich,
Department of Physiology
Universität Bern

Supervision

Jannes Jegminat
Prof. Dr. Jean-Pascal Pfister

February 7, 2024

institute of neuroinformatics

u^b
UNIVERSITÄT BERN

Acknowledgement

I would like to thank Jean-Pascal for his supervision and advice. From him I learn to ask specific questions and what are the important qualities to be a scientist. His meticulous comment on restructuring this thesis was an enormous help to me. I am deeply grateful for Jannes for his supervision, advice, the active discussions, encouragement and support even on late nights. Qinhan for his love and support. Anna for her warm heart encouragement to be a strong woman in science. Angela for being a nice friend, Christian for giving me encouragement when I was down. I also had great pleasure for working in the master's room at INI, especially the active discussions during lunch breaks. Lastly, I owe an enormous amount of debt and have the deepest gratitude to my parents, who supported the pursuit of my passion unconditionally. Without them this thesis would not have been possible.

Contents

Abstract

Deep neural networks(DNNs) are the most successful image classification algorithms these days. They have prevailed in applications and inspired visual neuroscience research. However, adversarial attacks on such networks were discovered recently, posing potential threat to their application security and challenging on our presumed understanding of their mechanism. This thesis is devoted to investigate the reason behind why DNNs are vulnerable to adversarial attacks. To obtain an intuitive understanding of a simple DNN while not losing the image classification capability of the network, we adopt a 3-layer fully connected architecture that learns its parameters from the MNIST dataset.

By speculating the potential causes of adversarial attacks, we arrive at the hypothesis that adversarial vulnerability is a result of the linear transformation components of DNNs learning to disentangle classes within a low dimensional manifold. In the case of MNIST, the manifold dimension is estimated to be 6. For 90% of randomly sampled images, their distance to the learned network decision boundary is less than $\frac{1}{5}$ of their distance to the corresponding manifold class. This suggests that the DNN is learning a highly curved decision boundary which lies close to data in image space, rather than separating image classes on the manifold with the maximal margin. The implication of this hypothesis is empirically quantified by measuring the sparsity of hidden unit activation. It is observed that adversarial attacks induce dense, weak hidden activation supporting the attack class, and sparse hidden activation supporting the benign class. Another consequence of the linear transformation feature of DNNs is that large visually unrelated subspace of images can be equivalently perceived by the network.

We verified a simple attempt to increase the network's robustness by adding noise on training images and discussed its limitations. Building on top of this understanding, we suggest several characters that should be included in an alternative image classification model robust to adversarial attacks and can be generalized to deeper neural network models.

Nomenclature

Acronyms and Abbreviations

NN	Neural Network
FFNN	Feed Forward Neural Network
FGSM	Fast Gradient Sign Method
BIM	Basic Iterative Method
GAN	Generative Adversarial Network
VAE	Variational Autoencoder

Introduction

Recently, deep learning has prevailed in computer vision applications. Banks use deep neural networks(DNNs) for facial costumer identifications, and autonomous driving cars use deep neural nets to recognize road signs. On a particular dataset, deep learning models are shown to have comparable or even superior image classification accuracy than humans([1, 2]).

In addition to engineering applications, deep neural nets has also inspired visual neuroscience. Due to its inspiration from biological neural systems, studies have suggested that feed forward deep neural networks could serve as a model for the visual pathway and visual perceptual learning ([3, 4, 5, 6]). They highlighted similarities between trained DNNs and the visual cortex such as behavioral and physiological patterns in human and monkey experiments, and layer- and unit-specific changes during learning compared with fMRI and electrophysiological data.

As the potential power of neural networks attract media and public attention, adversarial attacks were discovered. Slightly perturbed image can be generated to alter the classification of a well trained neural network classifier. Figure 1 shows such an example. A state-of-the-art DNN classifies the left image as a panda. But after a visually imperceptible perturbation(middle) is superimposed on top of the original image to generate an adversarial attack on the right, the same neural network classifies the new image as a gibbon with 99.3 % confidence.

Images like this poses significant danger in existing neural network's applications. Maliciously induced attacks could threaten bank account safety and alter the behavior of autonomously driving vehicles. What is more important is adversarial images challenge our presumed understanding of the deep neural networks, especially their similarity with the visual system, since they are examples that differentiate the response between the two systems. Hence understanding why neural network are vulnerable to adversarial attacks become an important question for application security. It also gives an opportunity to understand the image classification strategy used by neural networks compared to the visual system.

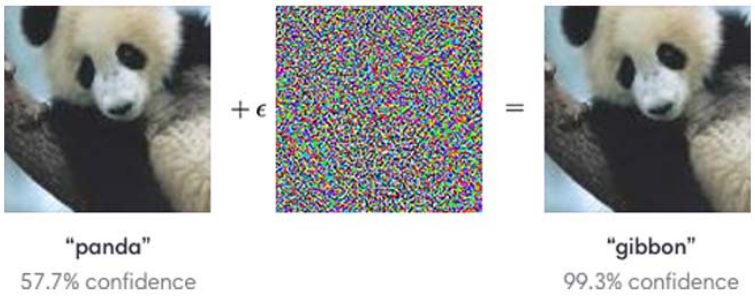

Figure 1: Example of an Adversarial Image on ImageNet data set, taken from [1]. **Left**: Original image, **Middle**: Scaled perturbation, **Right**: Adversarial Image

This thesis studies adversarial attacks of feed forward neural networks trained to recognize images. Although our ability to understand large scale computation is limited, we can build intuition about the how adversarial attack alters a neural network's decision by studying simple systems. Therefore we study the adversarial vulnerability of a simple 3-layer feed forward neural network trained on the MNIST hand written digits. Our discussion on adversarial vulnerability is organized with the following order:

Chapter 1 introduces feed forward neural networks. Chapter 2 defines adversarial images and gives an overview on common attack and defense methods. Chapter 3 goes over potential causes of adversarial images and narrows down the hypothesis to the linear transformation feature of DNN. At the end of Chapter 3 we introduce the network architecture and data set used for this investigation.

Then we approach the problem of adversarial vulnerability from two directions: one is understanding the data set's topology in input space and the other is how linear transformation influences the network decision. Chapter 4 studies the former aspect by estimating the dimension of data distribution. Chapter 5 then compares the data manifold topology with decision boundary topology learned by the neural network. In Chapter 6 we showed that small perturbation could be summed up to influence higher layers as a result of the linear transformation. In Chapter 7 we analyzed the visual resemblances of images within the subspace that equivalently activates a population of hidden units. In Chapter 8 we tried an attempt to increase the network's robustness and discussed its limitations.

Chapter 1

Feed Forward Neural Networks

In this chapter, essential knowledge of feed forward neural networks(FFNN) are introduced to give a sufficient foundation for readers to understand the following chapters. First, we introduce the definition of feed forward neural networks, followed by the learning process, then we discuss their applications and limitations.

1.1 Definition

A feed forward neural network with L layers is a function $f : \mathbf{x} \rightarrow y$ with input $\mathbf{x}$ and output y composed of L sequential information processing layers. To compute y, input $\mathbf{x}$, the 1^{st} layer, is processed in a "feed-forward" fashion. Each subsequent layer with N_l neurons, expressed by a vector $\mathbf{H}^l$, is computed from its previous layer of N_{l-1} neurons with their activity $\mathbf{H}^{l-1}$. As shown in figure 1.1, the computation of each layer involves first, a linear transformation:

$$\mathbf{h}^l = \mathbf{W}^l \mathbf{H}^{l-1} + \mathbf{b}^l \tag{1.1}$$

where $l \geq 2$, $\mathbf{W}^l \in \mathbb{R}^{N_{l-1} \times N_l}$, and $\mathbf{b}^l \in \mathbb{R}^{N_l}$. The output of this linear transformation, $\mathbf{h}^l$, is then passed through a nonlinear activation function, one example is:

$$\mathbf{H}^l = RELU(\mathbf{h}^l) \tag{1.2}$$

Activation function RELU is an element wise operation on $\mathbf{h}$:

$$RELU(h_i) = max(0, h_i) \tag{1.3}$$

Other instances of activation function include sigmoid,

$$\sigma(h_i) = \frac{1}{1 + \exp(-h_i)} \tag{1.4}$$

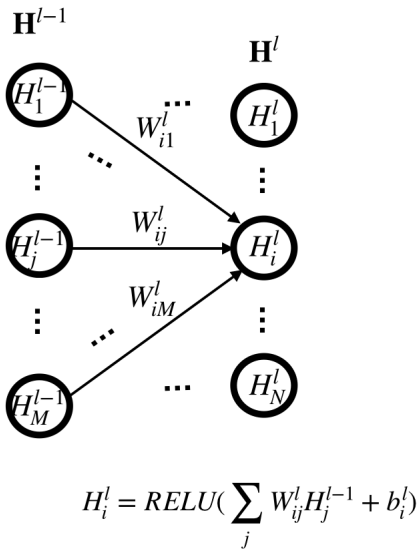

Figure 1.1: Layer wise processing of a feed forward neural network.

With each layer of information processing being a linear transformation followed by an element wise nonlinear operation, a neural network is essentially a composition function. The name suggests its resemblance to biological neurons. As the activation of a single neuron in the lth layer H_i^l computes a weighted sum from neural activation of its previous layer H^{l-1} and adds its own bias term b_i^l, before being passed to an activation function. An illustration can be found in 1.1. This computation process resembles Perceptron, a classical simplified model of single neuron computation [7].

1.2 Learning

Learning for a neural network model $f : x \to y$ means approximating a function $g : x \to y$ by adjusting its parameters including weights and biases $\theta = \{\mathbf{W^l}, \mathbf{b^l}$, for $l = 1, 2, ..., L\}$. The architecture, including the number of layers, the number of neurons in each layer in addition to the activation function, is set by the trainer and does not change during learning. In practice, g is an unknown function described by a data set $\mathcal{D}$ consists of N tuples corresponding to observed pairs of input and output from function g: $(x_i, y_i)_{i \in 1,...,N}$ where $y_i = g(x_i)$. For example, an image classification data set has x_i, the pixel-wise description of an image, and y_i, the corresponding human assigned label of that image.

A neural network learns by finding parameters that fits the data set. This is theoretically possible since neural network models are known as universal function approximators. Meaning that by adjusting its parameters, f can get very close to approximating any g. Some mathematical intuition behind the approximation ability of multi-layered percep-

trons has been suggested by Cybenko and others ([8, 9]). They proved that a single layer of neural network with infinitely many neurons with sigmoid activation function is able to approximate any nice function arbitrarily close. The intuition behind is that the function space spanned by the parameters θ of f is dense from $\mathbf{x}$ to y, hence there always exists a set of parameters that can get arbitrary close to $g : \mathbf{x} \to y$. Although theortically one-layered neural network with infinitely hidden units can approximate any dataset, in practice, deep neural networks with more than two layers are used more often rather than a single layer. This is in part because real world data set contains statistical redundency. For example, images from the natural world contains statistical redundancies that they are scale and location invariant. In such cases, the deeper layers of a deep neural network are able to recycle the computation results of their previous layers, which helps to save an exponentially number of computing units compared to learning using only one single shallow layer.

To adjust the parameters of f to approximate g described by a data set, learning is formulated into an optimization problem. In particular, we are looking for θ that minimizes a cost function $\mathcal{J}$. The cost function measures the deviation of network output $f(x)$ from $g(x)$. Usually the network output in the last layer(with K neurons) is normalized to sum up to one and seen as a distribution of the network's confidence in each class i from a total of K classes. Although it is highly questionable whether they actually represents a probability. A cost function can then penalizes the deviation of $\mathbf{P_f}(x)$ from $\mathbf{P_g}(x)$, which is a delta function concentrated on the truth label $g(x)$. The cross entropy cost function for one image x is formulated as:

$$\mathcal{J}(x) = -\sum_{i}^{K} \mathbf{P_f}(x)_i \log(\mathbf{P_g}(x)_i) \tag{1.5}$$

To measure generalization and prevent over-fitting, the data set $\mathcal{D}$ is randomly split into a training set $\mathcal{D}_{train}$, which is used to learn the parameters, and a test set $\mathcal{D}_{test}$, which is used to evaluate how well f approximates g.

In practice, the cost function are evaluated on mini batches of n data points $\{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$ randomly sampled from $\mathcal{D}_{train}$. The cost function for mini batch sums up cost function for each data point in the batch:

$$\mathcal{J}_{batch} = -\sum_{k}^{n}\sum_{i}^{K} \mathbf{P_f}(x_k)_i \log(\mathbf{P_g}(x_k)_i) \tag{1.6}$$

Due to our knowledge of g is limited to observations, the global landscape of $\mathcal{J}$ can not be mathematically formulated. So parameters θ are updated by the gradient information from randomly sampled mini batches with step α:

$$\mathbf{W}^l := \mathbf{W}^l - \alpha \nabla_{\mathbf{W}^l} \mathcal{J}_{batch} \tag{1.7}$$

$$\mathbf{b}^l = \mathbf{b}^l - \alpha \nabla_{\mathbf{W}^l} \mathcal{J}_{batch} \tag{1.8}$$

To train a neural network, this stochastic gradient descent process updates the θ until it converges and cost function do not go lower, corresponding to being in a local minimum.

The classification performance of a neural network is evaluated on $\mathcal{D}_{test}$ by the percentage of correctly classified images:

$$Accuracy = \frac{|\{f(x_i) = g(x_i) : (x_i, g(x_i)) \in \mathcal{D}_{test}\}|}{|\mathcal{D}_{test}|} \tag{1.9}$$

1.3 Applications and Limitations

Neural Network methods became popular since AlexNet, a 7-layer-feed-forward neural network, won the imagenet classification challenge in 2010 [**?**]. Since then, deeper and deeper feed forward neural networks are trained for image recognition. Their classification accuracy is reported to even exceed human classification performance on ImageNet data set ([2]). Because of their high classification accuracy and simple training procedure, deep FFNN models has become the state-of-the-art methods for image classification and other tasks. Variants of FFNN such as recurrent neural networks and long short-term memory networks have also been used for natural language processing, automated language translation and others. Neural network's popularity has dueled beyond academia into applications that were once unthinkable to be replaced by machines. Examples of such include road sign identification for autonomous driving (Chen2016Multi-ViewDriving), playing GO against human players ([10]), and using facial recognition for personal banking identification ([11]).

Besides engineering, neural network methods have also inspired the study of biological visual recognition. Researchers have discovered similarities between neural networks (NN) and primate vision in image recognition and visual perceptual learning ([3] [12] [4]). They highlighted representational similarities between hidden units of neural networks compared to monkey physiology and human fMRI data seeing images from different categories. Hence, many have come to see deep neural network as a framework of visual information processing ([6, 3]).

At the same time, other researchers highlighted differences between DNN and human visual recognition. For example, CNN image recognition performance deteriorates much faster with image distortions compared to human observers [13]. Computationally, deep convolutional networks lack neurally-inspired components of local gain control which are ubiquitous in biological sensory systems [14].

DNNs also have many unanswered questions that can be easily overlooked. One is architecture. In a neural network, the depth, the activation function, and the number

of neurons within each layer are arbitrarily preset by the trainer. Understanding on how those architectural specification influences their learning and performance, and what type of architecture is optimal for a specific type of data set is still a subject of research.

Another issue is DNN's interpretability. A neural network with L layers have a total of $\sum_l^L |\mathbf{W}^l| + |\mathbf{b}^l|$ number of parameters. This amount of parameters is most of the time, larger than the number of training examples. Unlike laws of physics with several variables, understanding how a big set of parameters influence the neural network's computation goes beyond the grasp of human intuition. Because of these reasons, neural networks have earned a reputation of being "black boxes".

Having a large number of parameters is also against the idea of using as little parameters as possible in statistical learning theory. This is because fitting data with a model that contains too many parameters could result in the problem of over-fitting, which is the situation that the model fit too closely to its training data, and fail to generalize to an unseen set of test data. But this problem of generalization does not seem to emerge for neural networks on the test set as their classification accuracy remains high. However, the reason behind why neural networks with a big set of parameters learn to generalize is still under investigation.

Another controversial feature of neural network is related to the training procedure. Since stochastic gradient descent is used to navigate the parameter space. Depending on the landscape of the loss function, rather than converging to the global minimum, parameters can easily get stuck in local minimum. Therefore, the same network architecture independently trained with different initial condition can converge to a completely different set of parameters.

Chapter 2

Adversarial Attacks and Defenses

Adversarial attacks on image classifying networks were discovered in 2014([1, 15]). They challenged the computational vision field's understanding on the classification ability of DNNs. As a visually indistinguishable perturbation, superimposed on a correctly classified image, is possible to alter a network's classification.

The possibility of generating images that fool neural networks poses threats to application security. A stop sign, maliciously attacked, could be regarded as a speed limit to an autonomous driving car; a face, maliciously attacked, could fake someone's identity. Any existing application that relies on the assumption that neural network recognizes images in the same way as humans do is vulnerable to this attack.

On the other side of this danger is an opportunity. Despite many people see deep neural networks as models of visual information processing ([4, 3]), adversarial images are examples of images that the two system respond differently to. Hence studying NN's vulnerability to adversarial attacks is a starting point to deepen our understanding of both NN and human vision.

Despite its urgency and importance, until today there does not exist a single defense method that is robust to all adversarial attacks, and the adversarial problem remains unsolved ([16, 12]). In this chapter, we define adversarial images, and go over commonly used attack and defence methods.

2.1 Definition

A neural network f has the goal of approximating some image classification function g: $\mathbb{R}^d \to k$, $k \in \mathcal{Z}$. g can some person, letâs say George, whose job is to look at some images

he identifies and assign each of them one category in $\mathcal{Z}$. At the end of many days, George produced a data set $\mathcal{D}$, which is divided into training data $\mathcal{D}_{train}$ and test data $\mathcal{D}_{test}$. Instances of $\mathcal{D}_{train}$ are used to train f, and how close f approximates g is evaluated on $\mathcal{D}_{test}$. The higher a model scores on the test set, the better the model is regarded to be.

The problem is: having a high classification accuracy on the test set is not a sufficient condition for f to classify all images to be the same class as George does. Adversarial images are examples of such cases. Hence the difference of classification between f and g is taken into consideration for the most general definition of adversarial image sets that is based on perceptual difference:

Definition 2.1.1. *Perceptual Difference Definition* An image x that g identifies to be z in a category set $\mathcal{Z}$ is an adversarial image of a neural network f if f classifies x as some category other than z. The set of adversarial images from this definition can be denoted as all images on a manifold $\mathbb{M}^d$: $\mathcal{A} = \{x : f(x) \neq g(x), x \in \mathbb{M}^d\}$. $\mathbb{M}^d \in \mathbb{R}^d$ is a subspace that contains all images that George identifies to be in one of the categories in $\mathcal{Z}$.

If the adversarial set $\mathcal{A}$ in this definition is empty for a neural network f, it implies that the neural network f classifies everything from the class categories $\mathcal{Z}$ in the same way that George does. This represents our fundamental wish to replace George with f. However, we don't know who George is, if he classifies images in the same way as another person do (although probably, based on our experience), and what are all the possible images $\mathbb{M}^d \in \mathbb{R}^d$ that George identifies with. Formulating the function g representing George's visual recognition in a more quantitative matter is a future investigation of psychophysics. But an ill formulated g which makes the adversarial set difficult to quantify and study under this perceptural difference definition 2.1.

An alternative definition of adversarial image is purposed based on the fact that our perceptual identification of an image does not change if the image altered within a small euclidean norm shift from itself([17]). Hence any image x' within a small euclidean distance ϵ from an image x in the data set should have the same perceptual classification for George as for any other person: $g(x') = g(x)$. With this idea we can define adversarial images with this notion of Euclidean Norm:

Definition 2.1.2. *Euclidean Norm Definition* An image x' is an adversarial image of f if

1. $f(x') \neq g(x)$
2. There exist a corresponding image in the data set $x \in \mathcal{D}$ that satisfies $\sqrt{\frac{\sum_i^d x'_i - x_i}{d}} \leq \epsilon$
3. ϵ is small.

The set of adversarial images within an ϵ distance from the original data set is: $\mathcal{A}_{\mathcal{D}}^{\epsilon} = \{x' : \exists x \in \mathcal{D}, \sqrt{\frac{\sum_i^d x_i' - x_i}{d}} \leq \epsilon, f(x') \neq g(x)\}$

Compared to the perceptual difference definition 2.1, this definition of adversarial images base on Euclidean norm is more rigorous as it only relies on knowing the data set. This is also the reason that most of the main stream adversarial literature uses the Euclidean Norm definition ([18, 15, 19]).

2.2 Common Attack Methods

Given a neural network f, there exist many attack methods to generate adversarial images of f. We group these attack methods into gradient based attack, optimization based attack, decision based attack, and transfer based attack.

2.2.1 Gradient Based Attack

The general idea behind gradient based attack is to apply gradient of the loss function with respect to the input to search for adversarial images. Gradient based attack methods include Fast Gradient Sign Method (FGSM) and Basic Iterative Method (BIM) ([20, 15]).

Fast Gradient Sign Method(FGSM)

We denote the input image as $\mathbf{X}$ to emphasis its vector property, a network f computes classification $y = f(\mathbf{X})$. FGSM generates adversarial image $\mathbf{X}'$ to be misclassified by the network as the target attack label $y' \neq y$ with a chosen perturbation step ϵ:

$$\mathbf{X}' = \mathbf{X} - \epsilon sign(\nabla_{\mathbf{X}} \mathcal{J}(\mathbf{X}, y')) \tag{2.1}$$

This attack method find a direction to change in input that decrease the loss function with respect to the attack label y'. If the parameters of network f is known, then FGSM can quickly generate adversarial attacks.

Basic Iterative Method (BIM)

Basic iterative method was developed to increase the success rate of adversarial attack, its iterative procedure also enables finding the smallest possible adversarial attack. The

iteration starts with the original image:

$$\mathbf{X'}^0 = \mathbf{X}$$

Then BIM takes iteration steps of size α until the network classifies $\mathbf{X'}^N$ as the attack label y':

$$\mathbf{X'}^{N+1} = Clip_{X,\epsilon}\left[\mathbf{X'}^N - \alpha sign(\nabla_{\mathbf{X'}^N}\mathcal{J}(\mathbf{X}, y'))\right] \tag{2.2}$$

$Clip_{X,\epsilon}(\cdot)$ is a projection function to make sure $\cdot$ stays within the specified image pixel range X_{min} and X_{max} of X and maximum deviation ϵ from $\mathbf{X}_i$:

$$Clip_{X,\epsilon}(X_i'^N) = \min(X_{max}, X_i + \epsilon, \max(X_{min}, X_i - \epsilon, X_i'^N)) \tag{2.3}$$

2.2.2 Optimization Based Attack

Optimization based attack transfers the adversarial generation problem into an optimization problem. Often the objective is to find the minimal perturbation that alters the network's decision. This include the Carlini-Wager attack and the decision based attack ([21] [22]).

Carlini-Wager Attack

By formulating an adversarial attack into a constraint optimization problem, Carlini-Wager Attack generates attack image $\mathbf{X'} = \mathbf{X} + \delta$ using an optimizer without taking the gradient.

$$\begin{aligned} \min_{\|\delta\|} \quad & \mathcal{D}(\mathbf{X}, \mathbf{X} + \delta) \\ \text{s.t.} \quad & f(\mathbf{X} + \delta)) = y', \quad (\mathbf{X} + \delta) \in [0, 1]^N \end{aligned} \tag{2.4}$$

$\mathcal{D}$ is a distance measure which can be L_{inf}, L_1, L_0 or L_2, y' is the attack target.

Decision Based Attack

This attack method preforms rejection sampling to find an adversarial attack with the smallest perturbation in some distance measure $\mathcal{D}$. The attack can be performed without knowing the parameters of the neural network ([22]). To attack $\mathbf{X}$, the algorithm start with $\mathbf{X'}_0$, which is an adversarial image or an image in the target class, i.e. $f(\mathbf{X'}_0) = y'$. For each iteration step N, the algorithm samples a new $\mathbf{X'}_N$ from a proposal distribution(such as a Gaussian) that satisfies the following conditions:

1. $\mathcal{D}(\mathbf{X'}_N, \mathbf{X}) \leq \mathcal{D}(\mathbf{X'}_{N-1}, \mathbf{X})$
2. $f(\mathbf{X'}_N) = y'$

2.2.3 Transfer Based Attack

This is a particularly interesting and intriguing class of attack. It relies on the fact that adversarial examples are usually transferable between neural network models ([1]). That is, adversarial images that fools one neural network model also fools other neural network models which are independently trained with different specified architectures. Based on this property, transfer based attack trains an ensemble of neural networks on the same data set. To attack an unknown neural network, it simply generates adversarial attacks that fools all neural networks within the trained ensemble. The substitute attack very often fool the unknown neural network classifier ([23], [24]). The success of this ensemble method suggests that adversarial vulnerability could be caused by a common problem shared among neural networks independent of their architectures.

2.3 Common Defense Methods

The common defense methods include defense by adversarial training, defense by gradient masking, defense by Generative Adversarial Network(GAN). Each of them suffer from their own limitations. Till today, there is no single defense method that can prevent a neural network from all possible attack methods.

2.3.1 Defense by Adversarial Training

This is the most common adversarial defense method. The idea behind is: adversarial images have different statistics from images in the data set. Therefore it is unfair for the network who has never seen adversarial images to recognize them correctly.

Defense method by adversarial training uses adversarial attacks to augment the training set of the network ([**?**]). While this method does generate networks that are robust (in terms of classification accuracy) to the attack method that generates the augmented training set(FGSM, for example),the adversarial robustness do not generalize to other attack methods such as decision based attack ([25], [26], [12], [27], [23], [28]). With this line of thought, one would need to use all adversarial attack methods to augment the training set to make a network robust, which is an in-feasible task.

2.3.2 Defense by Gradient Masking

Another idea to defend a network against gradient based attacks such as FGSM and BIM is to mask the gradients of that network. Gradient masking defenses drive the network to a saturated nonlinearity regime. One commonly used example is **Defensive Distillation**. Defensive Distillation is based on the idea of modifying the softmax function of the last layer by adding a temperature parameter:

$$softmax(x, T) = \frac{e^{\frac{z(x)_i}{T}}}{\sum_j e^{\frac{z(x)_j}{T}}} \tag{2.5}$$

$z(x)_i$ is the network's i^{th} neuronal output. Since softmax is usually the last layer of the network, it is also the first layer used in a gradient based attack. When temperature is high, the gradient of the network is very small. With this activation, the gradient based attack does not have nice guidance on direction of input perturbation to find adversarial attacks.

However, similar to defense by adversarial training, defensive distillation does not generalize to defend attacks that does not use the gradient information, such as boundary attack or transfer based attack([29, 30]).

Defense by Variational Autoencoder One of the most promising defense method on MNIST data set is to use a variational auto-encoder (Schott 2018 [16]). This idea comes from a Bayesian prospective. One would want to learn a causal relationship between y and $\mathbf{X}$, i.e. what in $\mathbf{X}$ is causing the classification of y. Using Bayes rule:

$$p(y|\mathbf{X}) = \frac{p(\mathbf{X}|y)p(y)}{p(\mathbf{X})} \propto p(\mathbf{X}|y)p(y) \tag{2.6}$$

The prior $p(y)$ is estimated from the training data. For each label $y \in \mathcal{Z}$, the model evaluates $p(y|\mathbf{X})$ which is the multiplication of its prior $p(y)$ and likelihood $p(\mathbf{X}|y)$. The log likelihood is calculated as:

$$l_y = log(p(\mathbf{x}|y)) = \mathbb{E}_{\mathbf{z}\ q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \mathcal{D}_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})] \tag{2.7}$$

$\mathbf{z}$ is the latent variable from which the generator $p_\theta(\mathbf{x}|\mathbf{z}$ generates $\mathbf{X}$. $q_\phi(\mathbf{z}|\mathbf{x}$ is a network that infers the latent variable, and By having those two networks, l_y can be calculated for each class, leading to the final decision y as the one which maximizes $\log p(y|\mathbf{X})$.

This model is robust to the state-of-the-art attack methods, also, when the model is fooled by some adversarial attack, the adversarial examples are meaningful to the human eye.

One take from the success of this model is that a network need to know a generative model that attribute to a particular class y. The shortcoming of this model is that one variational autoencoder needs to be trained for each class, hence it is difficult to generalize this defense method to other, more complicated data sets such as ImageNet. It also uses a completely different architecture from the artificial neural network. However, one can still get some inspiration from the success of generative models.

2.3.3 Defense by Generative Adversarial Neural Networks(GAN)

Other defense methods use GAN to improve adversarial robustness including APE-GAN and MAG-NET ([31], [32]). The general idea is to have a discriminator network that tells whether an image is an adversarial or a benign example, and a generator network which projects the adversarial back to the data manifold. However, Carlini & Wagner showed that both of these method does not work. For example, MAG-NET is not robust to Carlini Wagner L_2 attack ([33], [21]).

Chapter 3

Speculating the causes

In this chapter, we go over speculated causes of neural network's vulnerability to adversarial attacks in order to narrow down our hypotheses. In the second part of this chapter, we introduce the data set and architecture used in this study.

3.1 Speculated Causes of Adversarial Attacks

3.1.1 The Bayesian Hypothesis

The Bayesian Hypothesis says that Adversarial examples exist because the network is not being Bayesian. A neural network learns by stochastic gradient descent. It adjusts parameters in little steps towards the direction of the steepest gradient that leads to a reduction of the cost function. At the end of learning, the network learns one set of parameters **theta** which has converged to some local minimum of the cost function. But this set of parameter is not unique, networks with the same architecture and different parameter initialization would end up learning different parameters despite training with the same data set.

The Bayesian idea says that instead of treating the parameters **theta** as a set of deterministic value, it should be seen as a set of random variables coming from a distribution. Any set of parameters sampled from this distribution evaluated on the data set would produce a low cost function. A classifier should integrate all the possible random parameters to make decision that is independent of whichever local minimum the network has converged to expressed by this equation: $P(y|\mathbf{X}) = \int_\theta P(y|\mathbf{X}, \theta)P(\theta)d\theta$.

A simple implementation of the Bayesian idea would be to train a set of networks with

different architecture and different initialization on parameters correspond to each architecture. Assuming for an ensemble of independently trained networks, $P(\theta)$ has equal probability, then the integration result $P(y|\mathbf{X})$ correspond to averaging this ensemble's output decision. Engstrom et. al. showed that this ensemble averaging process increases classification accuracy on adversarial examples generated by rotating and translating images ([**?**]). However, by the set up, this implementation is not robust to adversarial examples generated by transfer based attacks ([23]). Other than ensemble voting, another simple method to approximate Bayesian inference in neural networks is by dropout ([34]). If not doing Bayesian inference is the cause of the adversarial problem, those neural networks with dropout should be robust to adversarial attacks. However, existing results tested on neural networks with dropout do not support this hypothesis ([17]). Hence, provided that Bayesian inference has many computational benefits, we do not think that not having this feature attributes to network's adversarial vulnerability.

In addition, the Bayesian idea suggests that adversarial vulnerability is caused by having a neural network learning parameters converged until local minimum, instead of global minimum of the cost function. An alternative view of this problem without probability is to see the network's learning as doing some constraint satisfaction problem. In this sense, the goal of the learning process is to find a set of parameters that satisfy some constraint, i.e. leading to a low cost function. By this definition, there exist a subspace of parameters that equally leads to the same cost function/accuracy performance by the network. The learning process would help the network to find a set of parameter that lies in this subspace. But the definition of this problem does not specify that images that are close in euclidean distance should be classified as the same image. Hence there is no reason behind why solution to this problem should have this adversarially robust feature.

3.1.2 The manifold hypothesis

The manifold hypothesis states that Adversarial images exist because as the network gets deeper, it tends to classify object along a small line of manifold. Duvenaud et. al. ([35]) studied the infinitely wide neural network as layers of Gaussian process, and showed that the representation capability of the network tends to capture fewer degrees of freedom when the number of layer increases. However, pointed out by the paper, this is a pathology that emerges in very deep networks, since adversarial vulnerability is a problem for all kinds of network architectures, this should not be the direct cause of adversarial examples. Further, according to the manifold hypothesis, visually meaningful images lie in some low dimensional manifold in the input space. This hypothesis can be an explanation of why some very deep neural network achieves high classification accuracy.

3.1.3 The generative component hypothesis

The generative component hypothesis states that adversarial attacks exist because neural networks do not have a generative component specifying the distribution of $P(\mathbf{X}|y)$, rather, they learn a decision boundary that tiles the input space into the specified number of classes. Other than the decision boundary information, the network does not learn the distribution of the input data. Adversarial images, from this prospective, are images that is close to one class of distribution, but is falsely classified by the network. There are current neural network models implementation with generative component include Generative Adversarial Network (GAN) and Variational Autoencoder (VAE). GAN is a coupled network that contains a generator and a discriminator, both of which are deep neural networks. Defence methods by GAN have been proven not effective. There are some preliminary success of VAE which independently trains a VAE on each class of data, and then calculates log likelihood to determine the classification output([16]). This type of method will not classify a nonsense image into one of the labels, in addition, adversarial attacks generated on VAEs have resemblance to the visual shape of its target class. However, since training a VAE which involves a generative component is rather slow compared to feed forward neural networks, this method have problems to scale up for large image data set such as image net.

Although having a network that learns a generative model of the data is quite an appealing idea, current implementation of Bayesian inference in the deep neural network community does not produce adversarial robust models. Integration of the two ideas requires more advancements on the area of integrating generative model with connectionist networks. In the context of this thesis, as we want to understand better what feature in FFNN causes its vulnerability, it is still valuable to study FFNN.

3.1.4 The quantization hypothesis

The quantization hypothesis states that adversarial attacks exist because images that are being fed into the network is not quantized. Their motivation for quantization is mostly biological. They states that as neural activities in the brain are binary, images that are represented by the network should also go through some similar quantization process. Meanwhile, quantizing the image would also filter out attack perturbation automatically ([36]). Defense method of this type trains the neural network on quantized images, they support their argument by the fact that visual neurons represent and process images using binary(quantized) neuronal spikes. The drawback of this method is neural networks trained with quantized images also lose classification accuracy.

3.1.5 The linearity hypothesis

The linearity hypothesis states that adversarial attacks are caused by the current neural network being too linear. This speculated cause was briefly discussed in the earliest paper on adversarial attacks. Goodfellow pointed out that the linear transformation part of NN could attribute to its adversarial vulnerability, as a hidden neuron h_i compute $h_i = \sum_j \mathbf{W}_{ij} x_j + b_i$. A small magnitude of change in each x_j would accumulate into a big change in h_i, which is more significant if x has more pixels. He also showed that since linear transformation is a feature shared by many machine learning models, adversarial attack also exists for support vector machines, simple perceptions, and other models ([1]). This may hint to why adversarial attacks are transferable among many neural network models. Despite its simplicity, this explanation has not been generally accepted as satisfactory because there has been no empirical observations or quantizations of linearity's contribution to adversarial vulnerability.

3.2 Approach from this thesis

So far we have reviewed the essential concepts related to feed forward neural networks and surveyed the adversarial literature. Despite the importance of understanding the cause of neural network's vulnerability to adversarial attacks, this question has not been satisfactorily answered in the field. By going over speculated causes of NN's vulnerability, we primarily suspect that adversarial images are caused by the linear structure of neural network adapting its parameters to the training data set and fails to generalize to adversarial images that are visually equivalent. We hypothesize that this effect should be able to be quantified.

To study this problem, we set our definition of adversarial images to be the Euclidean space norm definition 2.1 rather than the more generalized perceptual closeness definition 2.1 since the Euclidean space norm definition only depends on the data set and circumvents the ambiguity of human psychophysics. We use a simple neural network architecture on a simple data set to study the adversarial problem in order to have an advantage of simple architecture while having a well performed image classifier.

For data set, we choose MNIST hand written digits, denoted as $\mathcal{D}$. It is the most commonly used data set in the machine learning community. It consists of 65,000 images of hand written digits (784 pixels in the range of [0,1] with 256 shadings of gray) and their corresponding human identified labels (between 0 to 9). $\mathcal{D}$ is divided into a training set $\mathcal{D}_{train}$ with 55,000 images and a test set $\mathcal{D}_{test}$ with 10,000 images. Sample images and their corresponding labels are shown in Figure 3.1.

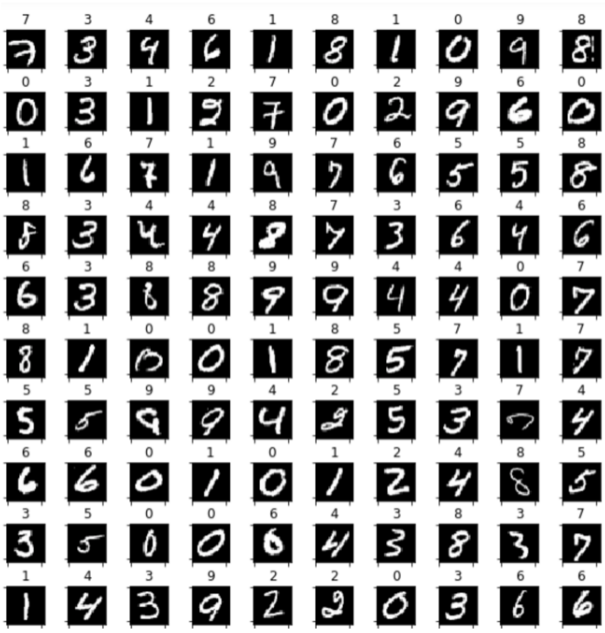

Figure 3.1: Example images of hand written digits with their labels in MNIST.

We use a simple 3-layer, fully connected architecture to study the neural network's vulnerability of adversarial attacks. Shown in Figure 3.2, the architecture consists of 784 input units, 1024 hidden units, and 10 output units. When input image $\mathbf{x}$ is passed onto the network, the classification output y is calculated as:

$$\mathbf{h^1} = \mathbf{W^1 x} + \mathbf{b^1} \tag{3.1}$$

$$\mathbf{H^1} = RELU(\mathbf{h^1}) \tag{3.2}$$

$$\mathbf{h^2} = \mathbf{W^2 H^1} + \mathbf{b^2} \tag{3.3}$$

$$\mathbf{H^2} = softmax(\mathbf{h^2}) \tag{3.4}$$

Softmax function is described by:

$$softmax(\mathbf{h^2}_j) = \frac{\exp \mathbf{h^2}_j}{\sum_{k=1}^{n} \exp \mathbf{h^2}_k} \tag{3.5}$$

As a result, $\sum_i \mathbf{H^2}_i = 1$. Because of this feature, $\mathbf{H^2}$ are often interpreted as a vector containing the network's choice probability of each class. Finally, the network's classification y is determined by the maximal entry in $\mathbf{H^2}_i$:

$$y = \underset{i}{\operatorname{argmax}} \mathbf{H^2}_i \tag{3.6}$$

The dimension of vectors and matrices are: $\mathbf{W^1} \in \mathbb{R}^{784 \times 1024}$, $\mathbf{W^2} \in \mathbb{R}^{1024 \times 10}$, $\mathbf{b^1} \in \mathbb{R}^{784}$, $\mathbf{b^2} \in \mathbb{R}^{1024}$.

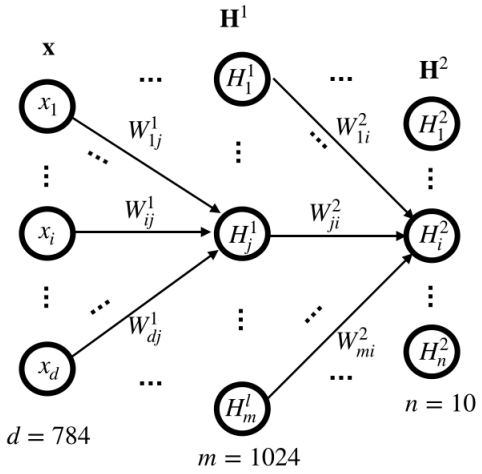

Figure 3.2: The 3 layer fully connected, feed forward neural network we used to train on MNIST data set.

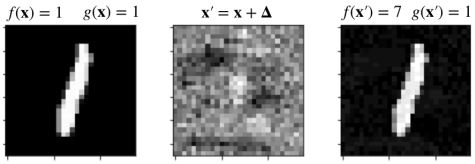

Figure 3.3: An example adversarial image. **Left**: One image from $\mathcal{D}$. **Middle**: Scaled adversarial perturbation generated by FGSM attack. **Right**: An adversarial image classified by the network as a 7.

The cross entropy cost function are used to learn the parameters $\mathbf{W^1}$, $\mathbf{W^2}$, $\mathbf{b^1}$, $\mathbf{b^2}$, updated on the training set by stochastic gradient descent using ADAM optimizer. After 20 epochs on the training data, the network classifies images with 98.7% accuracy on the test data. To study neural network's response to adversarial attacks, we mostly use gradient based attack because they have been shown to be the most efficient for finding the minimal perturbation needed to fool the network. An example of an attack generated by FGSM is shown in Figure 3.3.

Chapter 4

Estimating manifold dimension of MNIST Images

Neural networks are trained on images that are visually meaningful for humans. This is special as not all images appear to be visually meaningful for us. For example, we don't visually identify bar codes or QR codes as fast as a scanner, but we are really good at identifying a zebra in the field. Adversarial images are instances of images that are identified by the neural networks as one thing and human vision as another. To understanding the mechanism of adversarial attacks we need to understand two aspects: what are the characters of images that are visually meaningful, and what are the characteristics of classification mechanism of neural networks. This chapter studies what is special about visually meaningful images that composes the data set to get a better understanding of the former aspect. In the context of our data set, we want to understand the topology of visually meaningful hand written digits in input space, so that we can understand how decision of network separates classes of images with such topological feature in the following chapter.

4.1 Introduction

To understand the topology of visually recognizable images in a 784 dimensional input space is difficult as high dimensional data spaces goes beyond intuition. We associate this question with the long existing manifold hypothesis ([37]). The hypothesis states that visually meaningful images such as natural pictures or hand written digits lie in a low dimensional manifold in their embedding space. Some intuitive explanation for this hypothesis is that a lot of natural transformations such as translation, contrast changes, rotation or scaling form a continuous curve in image space. This means that images

identified to be the same class, for example, different scales of a hand written digit "2", lie along a continuous curve in image space. The manifold hypothesis motivates us to ask this question: what is the manifold dimension of MNIST dataset?

4.2 Method

Recall the definition of manifold dimension:

Definition 4.2.1. Manifold Dimension A topological space M is locally Euclidean of dimension k if every point x in M has a neighborhood U such that there is a homeomorphism ϕ from U onto an open subset of $\mathbb{R}^k$.

In other words, every point in a manifold of dimension k has a local neighbourhood that is the same as a open hypercube in R^k. Following this line of reasoning, to estimate manifold dimension on MNIST, we just need to estimate the dimension of local Euclidean space around the neighbourhood of each data point x in the dataset $\mathcal{D}$.

The dimension of Euclidean space can be described by the volume of points enclosed in a hyper-cube of dimension k. For example, in $\mathbb{R}^2$, a hyper cube of length d encloses an area of d^2; in $\mathbb{R}^3$, it encloses a volume of d^3.

If the dataset in $\mathbb{R}^m$ lies on a k dimension manifold, then a hypercube with length d centered around each data $x \in \mathcal{D}$ should enclose a volume of d^k data points. If we approximate the density of data to be ρ, then the volume equals to the density multiplied by the enclosed number of data points in the neighbourhood of this data:

$$d^k = \rho n \tag{4.1}$$

n is the number of data points enclosed in this hypercube:

$$n = |\{\tilde{x} : \sum_i |x - \tilde{x}| \leq d, \tilde{x} \in \mathcal{D}\}| \tag{4.2}$$

Taking the natural log on both side of equation 4.1, we arrive at:

$$\log(n) = \log(\rho) + k \log(d) \tag{4.3}$$

The dimension of data manifold k can be obtained by estimating the slope of $\log(n)$ versus $\log d$ around the neighborhood of an average data.

4.3 Implementation

We sampled 50 random data points x from the training set, n is measured for each x with increasing d according to equation 4.2. But to keep the neighbourhood small enough, we also fix $d \leq 25$. So there are on average 0.09% of data in the training set that are within distance d from x.

We also note that there is a minimal distance between data points on MNIST since the written digits are quantized with 256 gray values between 0 and 1. Further, having an n too small would violate the assumption about data density ρ. To make the condition in equation 4.1 meaningful, we set the minimal $d = 9.0$ to correspond to $n \geq 0.1$.

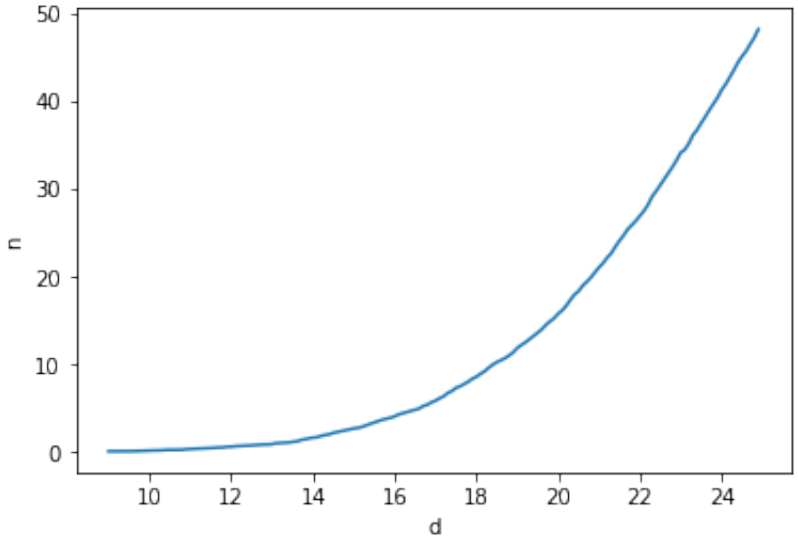

Figure 4.1: Average number of data points n within a hyper cube of length d centered around a random image$x$.

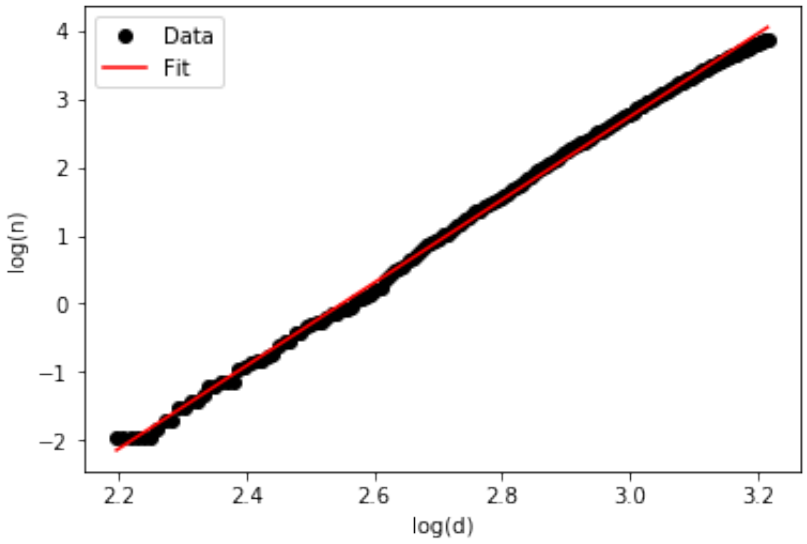

Figure 4.2: $\log n$ versus $\log d$. Black dot: $\log(n)$ versus $\log(d)$, Red line: linear fit of data, described by $\log(n) = 6.08 \log(d) - 15.51$.

4.4 Result and Conclusion

Figure 4.1 shows the averaged n across 50 randomly sampled x. As d increases, n takes a shape of power law. Figure 4.2 shows $log(n)$ versus $log(d)$, which is well fitted by the linear function: $\log(n) = 6.08 \log(d) - 15.51$. We thus estimate k to be 6.08. Since the manifold dimension is an integer, we can estimate that the data of MNIST lies in a 6 dimensional manifold in the 784 dimensional input space. As data in MNIST are randomly sampled instances of the continuous manifold, we may believe that rather than uniformly distributed, visually recognizable hand written digits assume some special topological structure as a low dimensional manifold embedded in input space. A classification done by the neural net thus needs to disentangle classes of hand written digits along this small line of manifold.

Chapter 5

Decision Boundary Topology of Neural Network compared to Data Distribution

5.1 Introduction

In the last chapter, we showed that hand written digit images that we visually identify lie in a low dimensional manifold. A neural network's classification goal is to disentangle digit classes along this small dimensional manifold. To understand why our neural network f disentangles instances of image manifold in the test set successfully yet is still vulnerable to adversarial images, we introduce a topological interpretation of adversarial sets in input space, then try to understand how network decision boundary separates inter-class margin.

5.1.1 Label sets

To measure and quantify the difference between neural network f and data set $\mathcal{D}$ which are discrete instances of g, we introduce the definition of label sets.

Definition 5.1.1. The label set $L_k(f)$ of a function $f(x)$ with classification label k is a set of the form:

$$L_k(f) = \{x : f(x) = k\}$$

In other words, for a classification function f, the label set $L_k(f)$ correspond to label k is the set of all images in input space that are classified by f as k. Similarly, $L_k(g)$ are

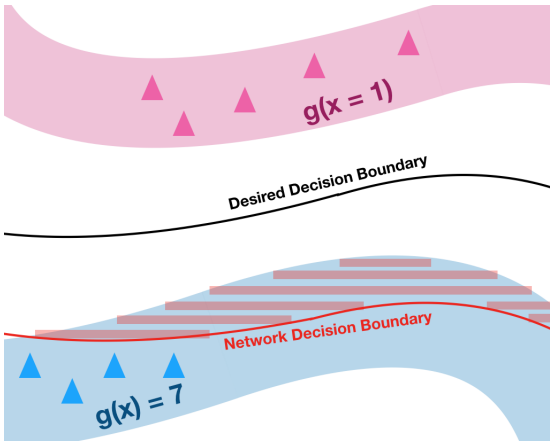

Figure 5.1: A cartoon visualizing image space, decision boundaries of f and and level set of g between class 1 and 7. Pink region and blue region correspond to manifold $L_1(g)$ and $L_7(g)$ in input space. Pink triangles and blue triangles are data points from $\mathcal{D}$ labeled as 1 and 7 that lies on the manifold. Solid red line is the network decision boundary bordering level set $L_1(f)$ and $L_7(f)$, i.e. points that lie in the region above red line are classified by $f(x)$ as 1, below solid line are classified by f as 7. Red stripped region, which differentiates level sets of f and g, are adversarial images. Solid line illustrates a decision boundary of free from adversarial regions.

all the images being visually recognized as label k. With the understanding from the last chapter, $L_k(g)$ lies in some low dimensional manifold in input space.

5.2 Adversarial Sets in Input Space

Having an understanding of level sets allows us to see adversarial images from a prospective of image space topology. By set up, neural network f tiles the input space by 10 disjoint label sets bordered by their decision boundary. For our image recognition function g, $L_k(g)$ lies in some low dimensional manifold and the label sets of g, which do not tile the image space. Decision boundary of g separates images that can be visually recognized as a digit from those that are not.

With this mental picture, we can then visualize the location of adversarial image sets. The adversarial image that visually appears to be a "7" but misclassified by the network as a "1" in Figure 3.3 is an instance from the dashed red region correspond to adversarial set in 5.1. Although the network decision boundary of f separates images from class "1" and class "7" in the data set(triangles) perfectly, those discrete observations of data from pink manifold $L_1(g)$ and blue manifold of $L_7(g)$ does not exclude location in input space that differentiate level sets of f and g. Alternatively, a neural network having the black

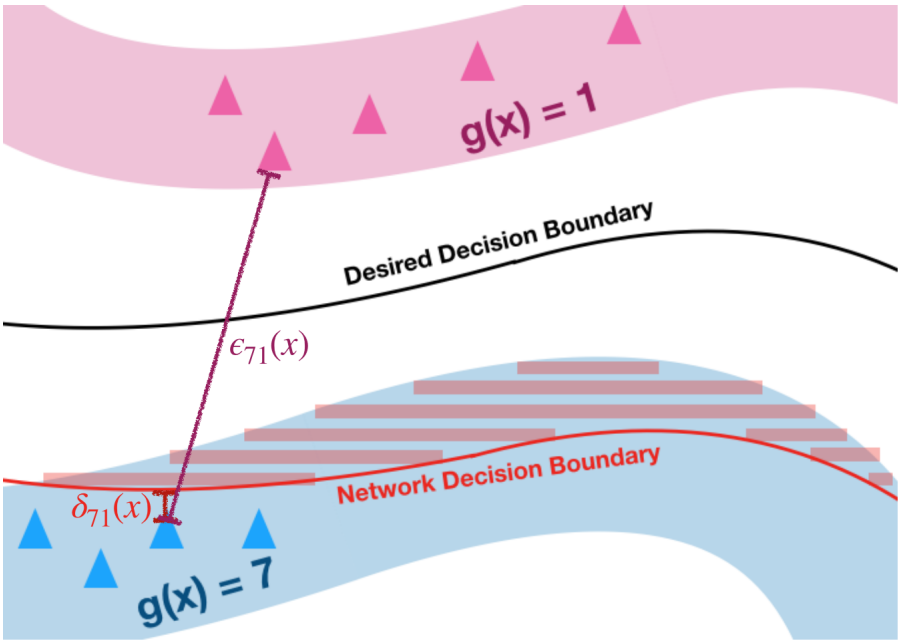

Figure 5.2: A cartoon visualizing $\epsilon_{71}(x)$ and $\delta_{71}(x)$ in input space. $\delta_{71}(x)$ measures the distance from this image x, to network decision boundary, $\epsilon_{71}(x)$ measures the distance from this image x in class 7 to level set $L_g(1)$.

decision boundary also classifies data perfectly, while separating level sets of g with the optimal margin, hence free from adversarial regions.

5.3 Quantifying deviation of decision boundary from the optimal margin

To compare how far the decision boundary learned by the neural network deviates from the optimal margin, we look at the distance from a data instance x from $\mathcal{D}$ in class j to label sets of all the other classes. $\delta_{jk}(x)$ measures the distance from x in class j to $L_k(f)$. $\epsilon_{jk}(x)$ measures the distance from x in class j distance to $L_k(g)$:

$$\delta_{jk}(x) = \inf_{x'}\{D(x', x) : f(x') = k, g(x) = j\} \tag{5.1}$$

$$\epsilon_{jk}(x) = \inf_{x'}\{D(x', x) : g(x') = k, g(x) = j\} \tag{5.2}$$

D is characterized by the normalized L_2 norm:

$$D(x', x) = \sqrt{\frac{\sum_{k=1}^{N}(x_k - x'_k)^2}{N}} \tag{5.3}$$

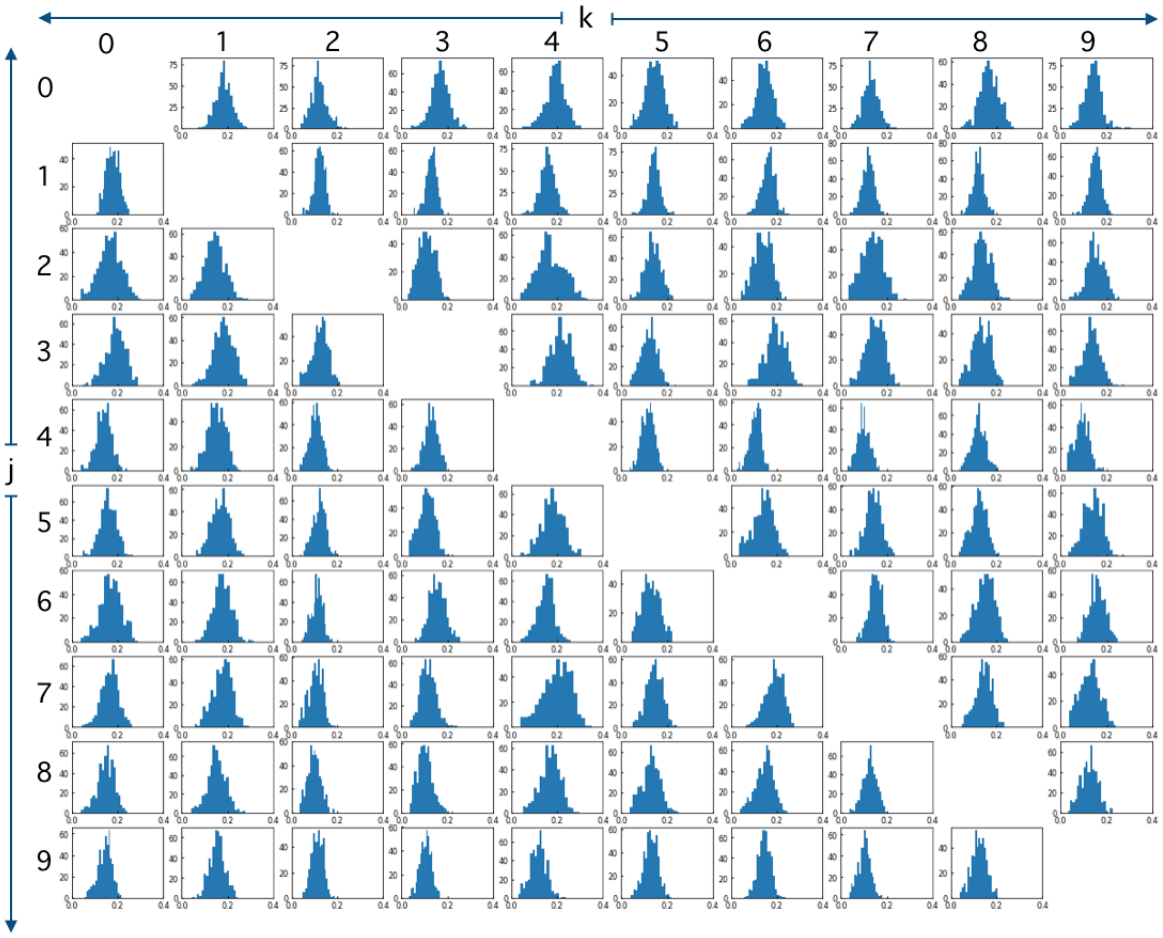

Figure 5.3: Distribution of $r_{jk}(x)$ from class j(row) to class k(column) obtained from 500 randomly sampled x in class j. x axis, percentage of $r_{jk}(x)$, ranging from 0 to 1; y axis, $r_{jk}(x)$. Small values of $r_{jk}(x)$ indicate that training samples are close to the decision boundary of the DNN, as compared to the similar distance between data classes.

Further, we use $r_{jk}(x)$ to characterize the distance ratio for x between $L_k(f)$ and $L_k(g)$

$$r_{jk}(x) = \frac{\delta_{jk}(x)}{\epsilon_{jk}(x)} \tag{5.4}$$

Shown in figure 5.2 is the schematic of $\delta_{71}(x)$ and $\epsilon_{71}(x)$ from an image in class 7. If a network f nicely separates level sets of g correspond to the black boundary in Figure 5.2, $r_{jk}(x)$ should be around 0.5 for any image x. We can further extract summary statistic by looking at the minimal inter-class distances between label sets of f and g. δ_{jk} measures the minimal distance from $L_j(f)$ to the manifold $L_k(f)$ across all data points:

$$\delta_{jk} = \inf_x \delta_{jk}(x) \tag{5.5}$$

Similarly, ϵ_{jk} measures the minimal distance from $L_j(g)$ to the manifold $L_k(g)$ across all

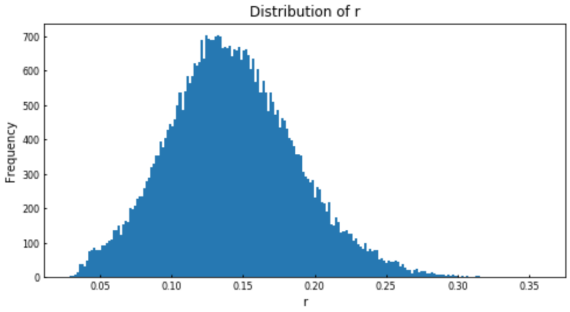

Figure 5.4: Histogram of r_{jk} over all j and k. x axis, of $r_{jk}(x)$, ranging from 0 to 1; y axis, $r_{jk}(x)$.

data points:

$$\epsilon_{jk} = \inf_{x} \{\epsilon_{jk}(x) : g(x) = j, g(x') = k\} \tag{5.6}$$

5.3.1 Experiment

We measure the those statistic on our default neural network mentioned in chapter 1, which learns the data set with 98 % test accuracy performance.

In practice, the manifold of g is described by finite amount of data observations $\mathcal{D}$. Due to computational constraints on large data sets, 500 random images are sampled from class j to obtain $\delta_{jk}(x)$ and $\epsilon_{jk}(x)$ on MNIST. $\delta_{jk}(x)$ is approximated by the minimal perturbed image x' from x generated by iterative gradient attack from x in class i to target j. $\epsilon_{jk}(x)$ is approximated by the minimal distance from x in class j to 500 random images in class k. Further, ϵ_{jk} and δ_{jk} are calculated by averaging the minimal 5 percent of sampled $\epsilon_{jk}(x)$ and $\delta jk(x)$ to avoid the influence of outlying data such as misclassified images.

Shown in figure 5.3 is the distribution of $r_{jk}(x)$ for all classes j and k from 0 to 9. Row j column k is the distribution of $r_{jk}(x)$ for random 500 images x in class j. Most $r_{jk}(x)$ have a normal shaped distribution with mean below 0.2, and a small standard deviation, implying a small variance of $\frac{\delta_{jk}(x)}{\epsilon_{jk}(x)}$ with a randomly sampled data.

This is confirmed by plotting the overall distribution of $r_{jk}(x)$ in Figure 5.4. All of $r_{jk}(x)$ falls below 0.4, 90% of $r_{jk}(x)$ is below 0.2. Most distance from a random data point x to neural network's decision boundary is less than one-fifth of that to the image manifold. Our neural network, despite classifying data well, does not separate the image manifold with close to optimal margin.

One more observation from figure 5.3 is that the distribution have different shapes with different j and k. For example, r_{93} have a more peaked distribution compare to r_{84}.

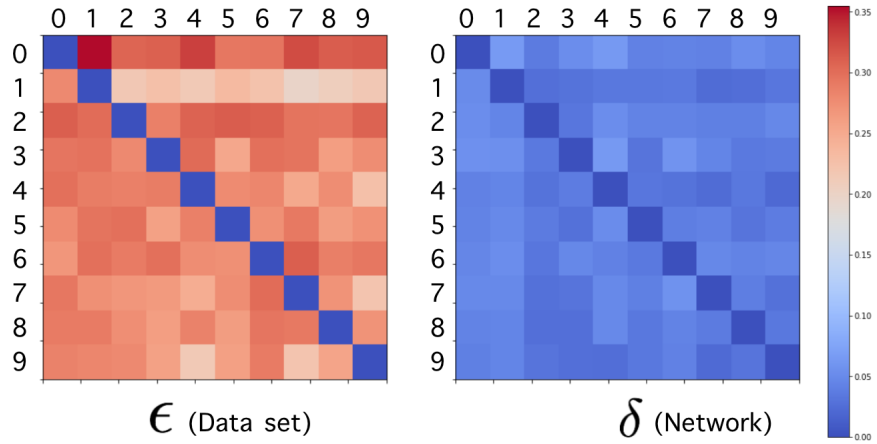

Figure 5.5: **a)** Depicted δ_{jk} and ϵ_{jk} on MNIST, where the row index belongs to class j.

In addition, r_{ij} does not seem to compensate r_{ji}. For example, r_{93} and r_{39} both are concentrated around 0.1. It is reasonable to assume the interclass distance in the manifold does not change between $\epsilon_{39}(x)$ and $\epsilon_{93}(x)$ with our measurement method, this means that the shortest distance from images in class "9" to network's decision boundary is similar to the shortest distance from images in class "1" to network's decision boundary. This suggests that the neural net is learning a decision boundary that are having a consistently small deviation from data compared to the separation of data specified by manifold. This observation is further confirmed by the summary statistic of the approximated minimal ϵ_{jk} and δ_{jk} across sampled images x in Figure 5.5 . While all entries of ϵ is above 0.2, all entries of δ is below 0.2. This means the approximated minimal distance from an image to the network's decision boundary is much smaller than the minimal inter class distance between manifolds.

The explanation for this is: the network is learning a highly curved decision boundary that are close to data from any class indicated by in figure 5.6. The neural net is adjust its large number of parameters to find a decision boundary that partitions the input space just enough to score well on classification accuracy, which leads to a highly curved decision boundary.

5.4 Conclusion and Discussion

We showed that adversarial sets of our FFNN on MNIST can be seen as regions in the hand written digit image manifold that correspond to a different network classification label. This adversarial region is not reflected by the classification accuracy on the data set.

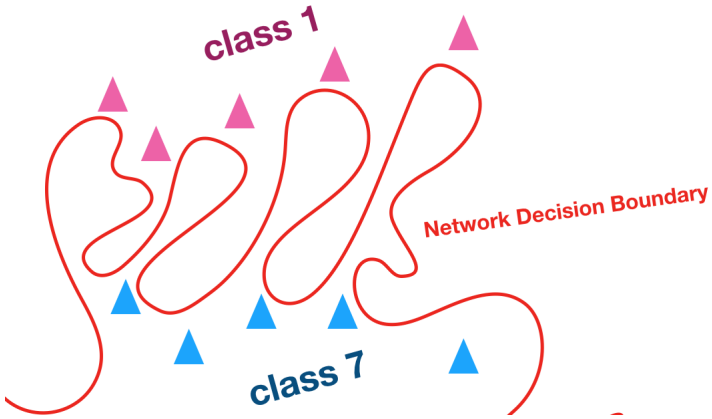

Figure 5.6: A 2D cartoon showing a likely learned decision boundary partition of the images.

We then quantified the deviation of network's decision boundary from an optimal margin by estimating the distance from an image to network's decision boundary and data manifold. Our method shows that the decision level sets learned by a well trained DNN is substantially different from the level set topology described by data. In particular, for a random image in the data set, the distance to the network's decision boundary is much smaller than the distance to images from an alternative class independent of which class the image belongs to. In particular, 90% of our sampled images have this distance ratio below 0.2. Further, the minimal distance from data to decision boundary learned by our network is significantly smaller than the minimal inter class distance described by data.

This suggest that despite high classification accuracy, the neural network does not learn to separate data manifolds with the maximal margin. Rather, it disentangles data along manifold using a highly curved decision boundary which lies very close to training data. As a result of this highly curved decision boundary, there exist adversarial regions which are very close to data points in the data set.

This highly curved decision boundary could result from learning. With a large number of parameters, the network specifies its decision boundary with a huge degrees of freedom. During the learning process, the gradient information from mini-batches simply moves the network's decision boundary away from incorrectly classified images with small steps. This process is similar to how a single perceptron updates its parameters.

Chapter 6

Quantifying the Effects of Linearity on Hidden Population during Adversarial Attacks

In Chapter 3, we went over the suspected causes and narrowed down our hypothesis that the linear transformation feature of neural network attributes to their adversarial vulnerability. Despite being suspected before, this hypothesis was not generally accepted due to its lack of quantitative evidence [1]. In this chapter, we study how linear activation function influences network decision under adversarial attack. Given our simple architecture, we can observe how adversarial attacks $\mathbf{x}' = \mathbf{x} + \Delta$ affect hidden population activities changes network decision from true label $y = f(x)$ to an adversarial label $y = f(x)$.

6.1 Grouping Hidden Units

Since it is hard to tell what is going on by looking at the activation of an entire hidden layer, we suggest to group the hidden units into their supporting classification label. Observed that for the last layer of our neural network:

$$\mathbf{h^2} = \mathbf{W^2 H^1} + \mathbf{b^2} \tag{6.1}$$

As the competition between entries of h_i^2 determines who wins out to be the network classification. Each h_i^2 is calculated by previous layer of hidden activation $\mathbf{H^1}$:

$$h_i^2 = \sum_k W_{ik}^2 H_k^1 + b_i^2 \tag{6.2}$$

How those output unit changes during an attack is dependent on their input $\mathbf{H^1}$. Since $H^1_k \geq 0$ for all k, for those H^1_k associated with positive weight $W^2_{ik} > 0$, activation of H^1_k increases h^2_i which translates to the competitiveness of the i^{th} class to be the network's output. Therefore, by the sign of W^2_{ik}, we can simply group the hidden neurons H^1_k into those that increases h^2_i and those that don't. For example, H^1_y represents activation of those hidden units supporting the benign label $f(\mathbf{x})$, and H^1_y represents the hidden units supporting the adversarial label $f(\mathbf{x}')$.

6.2 Studying the Hidden Unit Activation of One Example Attack

We can then observe activation of H^1_y and H^1_y responding to a random image $\mathbf{x}$ and its corresponding adversarial attack $\mathbf{x}$ in figure 6.1. Subplot **a** shows the original image, which is correctly classified by the network with 70% certainty, on the right is its adversarial attack, which is classified as an 8 with 77% confidence. Subplots **b**), **c**) and **d**) compares the histogram of hidden activation for different groups of units in $\mathbf{H^1}$ responding to input $\mathbf{x}$ and $\mathbf{x}'$. Specifically, **b** compares the distribution of hidden activation with original input, $\mathbf{H^1}_8(\mathbf{x})$, and adversarial input $\mathbf{H^1}_8(\mathbf{x}')$. Compared to $\mathbf{H^1}_8(\mathbf{x})$, $\mathbf{H^1}_8(\mathbf{x}')$ has many more of hidden units slightly activated. **c**) compares the histogram of hidden supporting benign label $\mathbf{H^1}_1(\mathbf{x})$ and adversarial label $\mathbf{H^1}_1(\mathbf{x}')$ Compared to $\mathbf{H^1}_1(\mathbf{x})$, $\mathbf{H^1}_1(\mathbf{x}')$ has hidden units more sparsely activated. For reference, **d** plots the comparison histogram of hidden supporting a reference label $\mathbf{H^1}_3(\mathbf{x})$ and $\mathbf{H^1}_3(\mathbf{x}')$. $\mathbf{H^1}_3(\mathbf{x}')$ has more units slightly activated, but not as significant as this change observed in **b**.

To change the network's decision from label 1 to label 8, adversarial perturbation increases the activation of many hidden units $\mathbf{H^1}_8$, and decreases the activation of many $\mathbf{H^1}_1$. The activation of $\mathbf{H^1}_3$ is also affected, but not as significant as $\mathbf{H^1}_8$. It seems that as many hidden units supporting adversarial label are slightly activated, the network switches its classification to be the adversarial label.

As slight changes a hid of hidden units are induced by For any hidden unit $\mathbf{H^1}_i$, its positivity implies $\mathbf{h^1}_i = \mathbf{H^1}_i$, and with input $\mathbf{x}$ and $\mathbf{x}'$, $\mathbf{H^1}_i(\mathbf{x})$ and $\mathbf{H^1}_i(\mathbf{x})$ are:

$$\mathbf{H^1}_i(\mathbf{x}) = \sum_j \mathbf{W^1}_{ij}\mathbf{x}_j + \mathbf{b^1}_i \tag{6.3}$$

$$\mathbf{H^1}_i(\mathbf{x}') = \sum_j \mathbf{W^1}_{ij}\mathbf{x}'_j + \mathbf{b^1}_i \tag{6.4}$$

So the change from $\mathbf{H^1}_i(\mathbf{x})$ to $\mathbf{H^1}_i(\mathbf{x}')$ is induced by the linear transformation responding

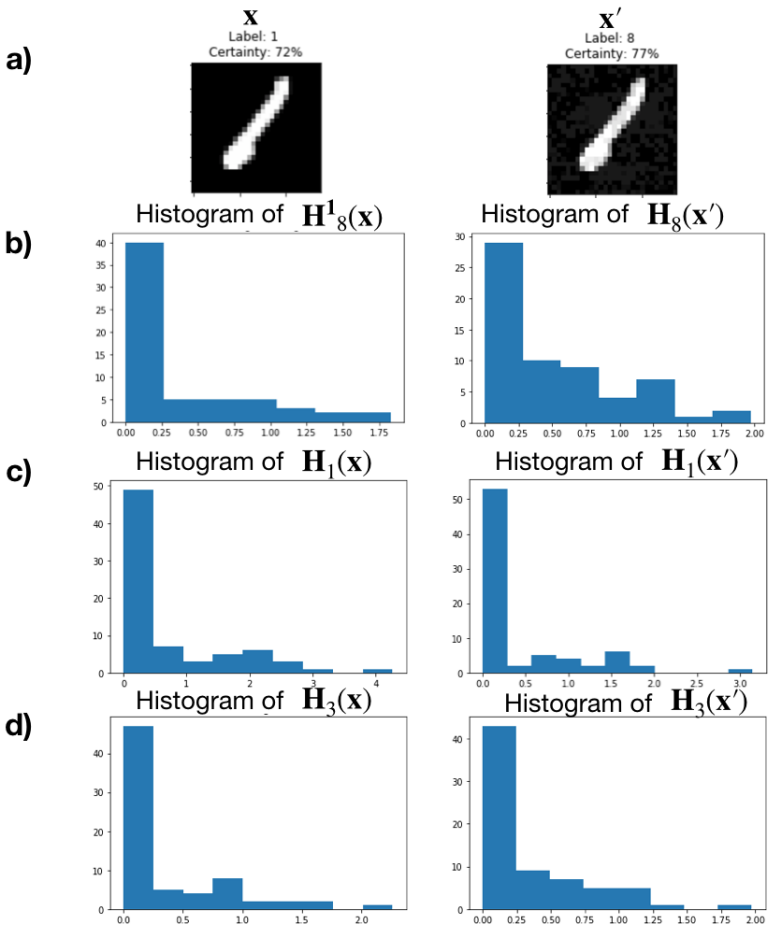

Figure 6.1: Observation of hidde

differently to perturbed $\mathbf{x}'$.

$$\mathbf{H^1}_i(\mathbf{x}) - \mathbf{H^1}_i(\mathbf{x}') = \sum_j \mathbf{W^1}_{ij}(\mathbf{x}_j - \mathbf{x'}_j) \tag{6.5}$$

Since so far we only discussed the hidden population activation change for one instance of image and adversarial attack, we check if this effect can be generalized to other adversarial images and attacks.

6.3 Generalized to Many Adversarial Attacks

For many images $\mathbf{x}$ and their adversarial $\mathbf{x}'$, we want to study the hidden populations supporting original label $y = f(\mathbf{x})$, and adversarial label $y' = f(\mathbf{x}')$, described as:

$$\mathbf{H^1}_y = \{\mathbf{H^1}_j : \mathbf{W^2}_{jy)} > 0\} \tag{6.6}$$

$$\mathbf{H^1}_{y'} : \{\mathbf{H^1}_j : \mathbf{W^2}_{jy'} > 0\} \tag{6.7}$$

and compare the response of $\mathbf{H^1}_y(\mathbf{x})$ with $\mathbf{H^1}_y mathbf x')$, and $\mathbf{H^1}_{y'}(\mathbf{x})$ with $\mathbf{H^1}_{y'} mathbf x')$, to see if the population's activation are affected.

Since it is hard to characterize histogram of hidden population for many images, there should be some way to reduce our observation of hidden population change to some quantification. To quantify population of hidden activation, we use GINI index as a measure of how sparse/populated activations with n units are:

$$GINI(\mathbf{H}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|H_i - H_j|}{2n\sum_{i=1}^{n} H_i} \tag{6.8}$$

Compared to other sparsity measures, it is demonstrated to be a preferred sparsity measure with properties of including scale invariance([38]). In addition, sparsity measure by GINI index has simple interpretation. When GINI index is 1, it correspond to super sparse distribution, i.e. one $\mathbf{H}_i$ has all the wealth. When GINI index is 0, it correspond to all $\mathbf{H}_i$'s being equal.

6.3.1 Experiment

We pick 1000 random images from the data set and generated 840 successful adversarial attacks by applying FGSM with step size 0.1. GINI index is calculated for hidden units supporting adversarial labels and benign labels. The sparsity activity of hidden supporting attack label responding to normal image $\mathbf{H}_{y'}(\mathbf{x})$, and adversarial image $\mathbf{H}_{y'}(\mathbf{x'})$ is compared in the left plot of figure 6.2. In addition, GINI index of hidden supporting benign/original label responding to normal image $\mathbf{H}_y(\mathbf{x})$, and adversarial image $\mathbf{H}_y(\mathbf{x'})$ is compared in the right subplot of figure 6.2.

6.3.2 Result

Compared to $\mathbf{H^1}_{y'}(\mathbf{x})$, sparsity of $\mathbf{H^1}_{y'}(\mathbf{x'})$ is reduced, as almost all points falls below the equal sparsity line. This implies that those attacks $\mathbf{x'}$ generated from $\mathbf{x}$ increases hidden sub-population's activation supporting adversarial decision units y'.

In the right image in figure 6.2, compared to $\mathbf{H^1}_y(\mathbf{x})$, sparsity of $\mathbf{H^1}_y(\mathbf{x'})$ is bigger, as almost all points falls above the equal sparsity line. This implies that those attacks $\mathbf{x'}$ generated from $\mathbf{x}$ increases hidden sub-population's sparsity supporting the original label y. The distribution of deviation of the left and right plot in figure 6.2 is shown in 6.3. Across most of those adversarial attacks, hiddens supporting attack label has negative sparsity change, and those supporting true label has positive sparsity change.

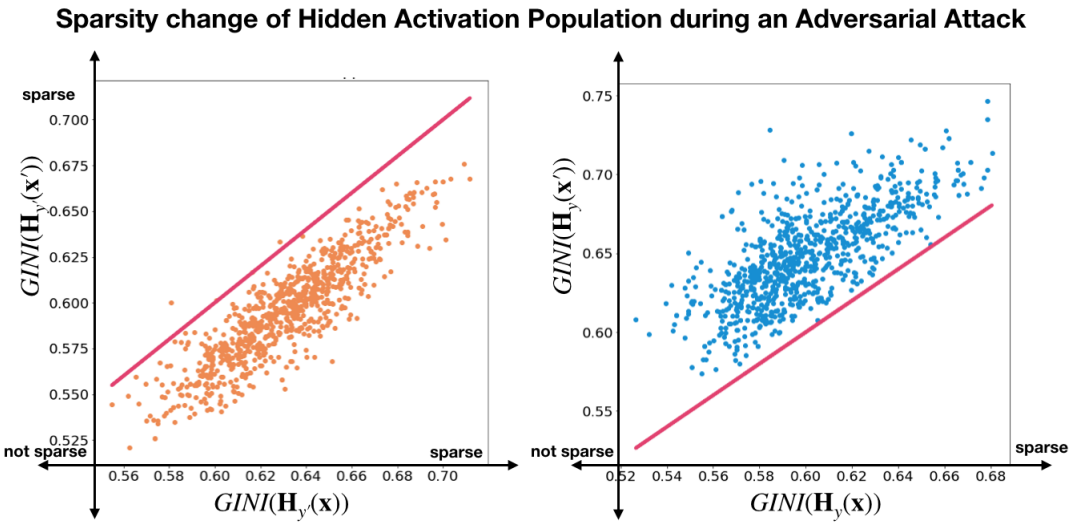

Figure 6.2: **Left**: Hidden sub population's activation sparsity supporting adversarial label y' responding to regular image(x axis), $\mathbf{H^1}_{y'}(\mathbf{x})$, and adversarial image(y axis), $\mathbf{H^1}_{y'}(\mathbf{x}')$. Each point correspond to one pair of image $\mathbf{x}$ and its adversarial dual: $\mathbf{x}'$. Red line: No sparsity difference between $\mathbf{H^1}_{y'}(\mathbf{x})$ and $\mathbf{H^1}_{y'}(\mathbf{x}')$. **Right**: Hidden sub population's activation sparsity supporting true label responding to regular image(x axis),$\mathbf{H^1}_{y}(\mathbf{x})$, and adversarial image(y axis), $\mathbf{H^1}_{y}\mathbf{x}'$.

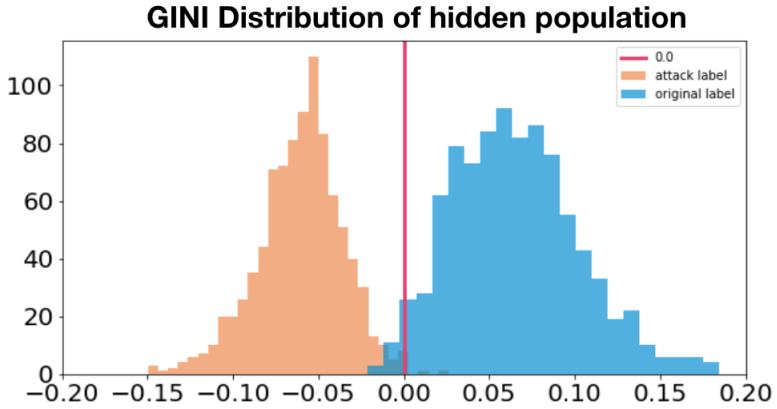

Figure 6.3: Histogram of sparsity difference between hidden subpopulation supporting network decision $\mathbf{H^1}_{f}$(orange), and human decision: $\mathbf{H^1}_{g}$(blue) responding to 840 random images $\mathbf{x}$ and their adversarial images $\mathbf{x}'$. Orange: distribution of $GINI(\mathbf{H^1}_{y}(\mathbf{x}')) - GINI(\mathbf{H^1}_{y}(\mathbf{x}))$. Blue: distribution of $GINI(\mathbf{H^1}_{y'}(\mathbf{x}')) - GINI(\mathbf{H^1}_{y'}(\mathbf{x}))$

Those observations imply the population activation change that we observed in one adversarial attacks can generalize to other adversarial attacks. This confirmed our hypothesis on the mechanism of how linear transformation influences adversarial vulnerability across many adversarial attacks.

As the network's confidence on class y, and y' is determined by:

$$h_y^2 = \sum_k W_{yk}^2 H_k^1 + b_y^2 \tag{6.9}$$

and

$$h_y^{2\prime} = \sum_k W_{y'k}^2 H_k^1 + b_{y'}^2 \tag{6.10}$$

Adversarial attacks sparsifies activation of those hidden units supporting the benign label, which are H_k^1 with $W_{yk}^2 > 0$, hence a more sparse acrivation decreases the network's confidence on the benign label h_y^2. At the same time, population of hidden units supporting adversarial label increases, which are those H_k^1 with $W_{y'k}^2 > 0$, as a result, h_y^2 increases. Small magnitude of pixel wise perturbation is summed up by the first linear layer to influence the population sparsity; the sequential layer then sums up population of affected hidden units. In this process, the network's classification changes from a correct classification to an adversarial classification.

6.4 Conclusion and Discussion

The influence of linear transformation on adversarial attacks, although suspected to be the cause from the earliest papers, have not had empirical observation or support for this claim to our awareness. We examined this problem in our simple neural network on MNIST. To quantify the influence of linearity on hidden populations being attacked, we purpose to look at the sparsity change of hidden population activation as a measure to quantify hidden population's activity influenced by linearity of the scalar dot product. We showed that compared to a benign image, the adversarial attacks decreases hidden population's sparsity due to the linear transformation of the first layer. Then the transformation of the 2^{nd} layer sums together those hidden populations to alter network decision to the attack label.

If one sees each hidden unit's activity as encoding the presence or absence of some visually meaningful feature, then the visually imperceptible adversarial perturbation is slightly similar to many features that hidden neurons are encoding, hence activating a population of hidden neurons. Since the network output is determined by the most "popular vote" of hiddens supporting their corresponding label, the effect of this imperceptible perturbation

end up influencing the network more than the perceptible shape of hand written digits. This can be seen as analogous to some social phenomenon. For example, we see many media news reporting about daily lives of some celebrities, but only 1 media covers the looming danger of climate change. The sheer number of news on trifling things is going to bias us toward ignoring what is, to our future, the most important.

Chapter 7

Analyzing Subspace of Images that lead to Equivalent Single and Layer-wise Hidden Neural Activation

7.1 Introduction

These days, many people see ANN as something analygous to our brain. Especially, some researchers showed that the activation of a hidden unit represents the presence of its preferred patterns similar to activation of V1 neurons detecting Gabor shaped edge patterns. In this case, the weight connecting the input to the hidden, $\mathbf{W^1}_i$, are seen as the pattern that the artificial neuron is encoding for analogous to V1 neurons. In convolutional neural network, those weights associated with hidden units in the first layer trained on natural images emerge to be edge like. This argument is used to support convolutional neural network as a model of the feed forward visual stream ([6, 39, 3]). However, adversarial images as examples that differentiate the classification between our perception and ANN, is a direct consequence of the different classification mechanisms between the two systems.

Here, we question and examine this popular belief and can ask similar questions in neuroscience to artificial neurons in our default network f: what is the pattern of image that hidden neuron activation is encoding, and an entire hidden layer? This question translates to looking for images that equivalently lead to strong activation of one hidden unit and a layer of hidden units. The linear layer-wise transformation of the neural network architecture allows us to use linear algebra methods to analyze subspace of images that leads to activation of a single neuron, and secondly, an entire layer of neurons.

7.2 Subspace of image that leads to equal activation of a single artificial neuron

First we start with the question: what is one intermediate, hidden unit in the neural network representing? What do two images look like if they activate one neuron to the same extent? In our simple network, we look at this question in the first hidden layer. Recall in our hidden layer, on unit's activation H_i is:

$$H_i = RELU(h_i) \tag{7.1}$$

where,

$$h_i = \mathbf{W}_i\mathbf{x} + b_i \tag{7.2}$$

$\mathbf{x} \in [0,1]^M$, $\mathbf{W}_i : \mathbb{R}^M \to \mathbb{R}$ Since $H_i = 0$ does not affect subsequent layers, we look at the situation where $H_i > 0$ is observed, which leads to:

$$H_i = \mathbf{W}_i\mathbf{x} + b_i \tag{7.3}$$

We want to find the set A that contains all $\mathbf{x}$ that leads to the same $h_i > 0$:

$$A = \{\mathbf{x} \in [0,1]^M : h_i - b_i = \mathbf{W}_i\mathbf{x}\} \tag{7.4}$$

To know if one image $\mathbf{x}_1 \in A$ uniquely activates h_i, we assume there exist another image $\mathbf{x}_2 \neq \mathbf{x}_1$ such that

$$h_i = \mathbf{W}_i\mathbf{x}_1 + b_i = \mathbf{W}_i\mathbf{x}_2 + b_i \tag{7.5}$$

Subtracting b_i from both sides, we get:

$$\mathbf{W}_i(\mathbf{x}_1 - \mathbf{x}_2) = 0 \tag{7.6}$$

Denote:

$$\Delta\mathbf{x} = (\mathbf{x}_1 - \mathbf{x}_2)$$

We arrive at: $(\mathbf{x}_1 - \mathbf{x}_2) \in Null(\mathbf{W}_i)$. We can know if $\mathbf{x}_1$ is unique by analyzing the null space of $\mathbf{W}_i$. By rank nullity theorem:

$$M = Rank(\mathbf{W}_i) + Nullity(\mathbf{W}_i) \tag{7.7}$$

Since $Rank(\mathbf{W}_i) = 1$

$$Nullity(\mathbf{W}_i) = M - 1 \tag{7.8}$$

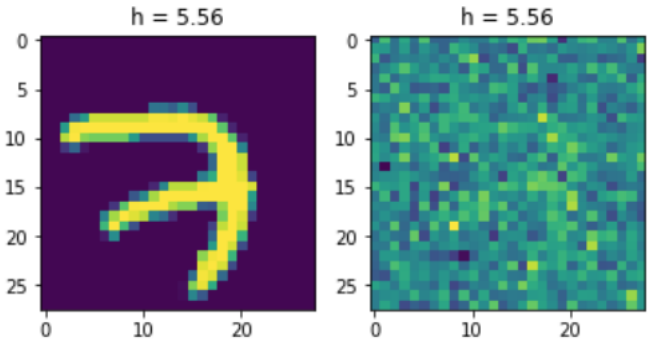

Figure 7.1: Two images that equivalently activates h_i, Left: a random image that activates h_i, Right: an equivalent image that equally activates h_i

Hence the null space of $\mathbf{W}_i$ is not empty but a subspace of dimension M-1.

$$h_i(\mathbf{x}_1) = h_i(\mathbf{x}_1 + \Delta\mathbf{x}) \; \forall \Delta\mathbf{x} \in Nullity(\mathbf{W_i}) \tag{7.9}$$

Therefore, for any image $\mathbf{x}_1$, it is possible to find an an entire subspace of images with dimension $M - 1$ that leads to the same activation value of the hidden.

If hidden neurons in our neural network behaves as pattern detectors as visual neurons , one should expect any $\mathbf{x}_1$ and $\mathbf{x}_2$ be visually equivalent, or at least similar to each other.

But we can easily find a contradiction. Two images with no visual resemblances to each other activates h_i to equally strongly. Figure 7.1 from our fully connected neural network verifies this case. On the left is written digit image from the training set, it activates one hidden neuron h_i to 5.56. On the right is another image that activates h_i to the exact same extent. There is no visual similarity between images on the left and right, which is plausible given the high dimension of null space. Hence just by looking at hidden activation of h_i, one cannot infer what pattern h_i is seeing, unlike neurons in V1.

This can be problematic as subsequent layers of neural network makes decisions based only on previous hidden activation. If the first layer of hidden units does not encode the presence of patterns, then a nonsense image could also influence the later layers of the network and subsequently leads to some strong decisions.

One may say that showing one hidden unit does not detect patterns does not exclude pattern detection ability of an entire layer. After all, a neural network transforms images into layer wise representation which can be a useful way for classification. In the next section, we analyze the possible images that leads to the same hidden layer activation to verify if they have visual resemblance to each other.

7.3 Subspace of images that leads to equivalent layer of hidden activation

As a subsequent layer of neural network process its previous layer by a linear transformation followed with a nonlinear activation function. We analyze the linear transformation $\mathbf{W^l}$: $\mathbb{R}^M \to \mathbb{R}^N$. of any subsequent layer from one layer $\mathbf{H^{l-1}}$ to the next $\mathbf{h^l}$ and investigate the space of images that leads to equal activation. This problem is divided into two cases: I. $M \geq N$; II, $M \leq N$.

7.3.1 $M \geq N$

For the case $M \geq N$, there is more neurons in the previous layer than the later layer. It is reasonable to assume the rank of W being N, as its column vectors are independently updated, this fact is also empirically verified. Then the activation of a hidden layer $\mathbf{h^l}$ is:

$$\mathbf{h^l} = \mathbf{W^l H^{l-1}} + \mathbf{b^l} \tag{7.10}$$

By our set up, $\mathbf{H^{l-1}} \in \mathbb{R}^M$ is the previous layer, and $\mathbf{h^l} \in \mathbb{R}^N$ is the next layer. In this case, $Nullity(\mathbf{W^l}) = M - N$. For any $\mathbf{H^{l-1}}$, there is a subspace of $\mathbf{H^{l-1\prime}}$ with dimension $M-N$ that lead to the same hidden layer activation, $\forall\, \mathbf{H^{l-1\prime}} = \mathbf{H^{l-1}} + \Delta\mathbf{H^{l-1}}$, $\Delta\mathbf{H^{l-1}} \in Nullity(\mathbf{W^l})$, then $\mathbf{h^l}(\mathbf{H^{l-1}}) = \mathbf{h}(\mathbf{H^{l-1\prime}})$.

7.3.2 $M \leq N$

For the case $M \leq N$ with less neurons in the previous layer than the later layer. Using the same argument as the previous case, as the rank of W is empirically observed being M, then $Nullity(\mathbf{W^l}) = 0$. The linear transformation $\mathbf{W^l}$ is a one to one mapping, meaning for any $\mathbf{h^l}$ correspond to a unique $\mathbf{H^{l-1}}$.

7.3.3 Subspace of images with the same H^2 in our default neural network

From the above discussion, we know that the transformation $\mathbf{W^1}$ in our neural network is a one to one mapping, meaning that any $\mathbf{h^1}$ correspond to a unique input $\mathbf{x}$. But $\mathbf{W^2}$ has a big subspace of $\mathbf{H^1}$ that leads to equivalent activation of $\mathbf{h^2}$.

Therefore, we can find a subspace in input that leads to equivalent activation in the final layer. One would expect two images that leads to the same activation value $\mathbf{H^2}$ should be visually indistinguishable from each other. However, this is not the case for our trained

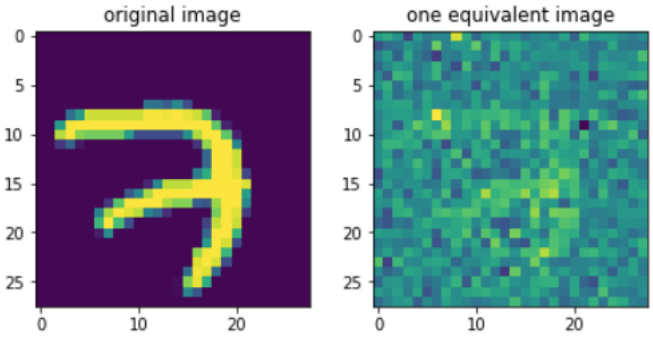

Figure 7.2: Two images that equivalently activates $\mathbf{H^2}$, Left: A random image from MNIST. Right: An image that leads to the same $\mathbf{H^2}$ activation. As a result, the network has the same classification of the left and right image.

network. Shown in figure 7.2 is a contradiction. The left and right image have the same $\mathbf{H^2}$. While one is a hand written digit, the other does not appear meaningful.

7.3.4 Conclusion and Discussion

To see if neural network's hidden unit represents the prescence or abscence of the pattern, just like neurons in V1, we examined images that leads to the same hidden unit activation, and the same layer wise activation. By analyzing the linear transformation subpart of the architecture, for an activation of one artificial neuron, we showed that there exists an entire subspace of images that lead to equally strong activation of this neuron.

Further, we analyzed the linear mappings from any layer of M neurons to its subsequent layer of N neurons, and showed that if the linear mapping is from more neurons in a layer to less neurons in the next layer, there is a subspace of images that leads to equivalent layer wise activation with dimension $M - N$. On the other hand, linear transformation from a layer of less neurons to more neurons is a one to one mapping.

Note that transformation done by any neural network cannot be a one to one mapping. One thing is, the activation function $RELU$ that we omit reduces every $h_i < 0$ to 0, so it transforms large spaces of images equally to 0. In addition, for any neural network, there is at least one layer with more neurons in the previous layer and less neurons in the next layer, which makes one to one mapping impossible. For example, in our naive neural network model, despite the first layer is has a one to one linear transformation, the second layer is not.

It is true that a function that maps an image space into a small set of labels has to learn some compressed information and hence would have subspaces of input that are equiva-

lently activate hidden units, but images that maps to the same network output should at least have visual resemblances to the network's decision. However, unlike expected, those subspace that leads to equivalent hidden activation of our last neural layer does not contain visual meanings. It is very easy to find a counter example of images that equally activates our last neuronal layer with no visual resemblances to each other. This is possibly due to the linear transformation feature of neural network, which leads to equivalent subspace spanned by basis functions, having a different topological character compared to visually meaningful image manifolds.

The fact that hidden activation does not represent the presence of visual patterns in images is problematic for both network's classification and understanding the network. As the consecutive layers of neural nets makes decision based on its previous hidden activation, a nonsense image, or an image visually identified to be an alternative label, could be seen equivalent to a hand written digit for a network classifier. In addition, although we have a network that performs well on classification of images in the test set, we do not understand the character of images that leads to the decisions.

From another prospective, if one would like to design an alternative architecture that has hidden units resembling the activation of visual neurons, each neuron should represent how similar one image is to the neuron's preferred pattern, so that images that equivalently activates one neuron and an entire layer have visual resemblances to each other. Translating this to input space, the subspace of equivalent hidden activation should correspond to manifold of visually meaningful images.

Chapter 8

Increasing Network Robustness by Stochasticity

8.1 Introduction

In Chapter 4 we showed that the neural network needs to learn to to disentangle classes of visually meaningful images lying on a low dimensional manifold embedded in input space. In Chapter 5, we showed that our naive feed forward neural network f, despite having good classification performance, have decision boundary very close to training images. As a result, adversarial sets correspond to images that are close to training images does not get classified the same as those training images. Building on top of our knowledge, we purpose a simple method to improve adversarial robustness by pushing the decision boundary away from training images. We do this by adding stochastic noise to the training images, although adding preprocessing to input image has been tried for other purposes, we verify this approach with our experimental understanding.

8.2 Method

In Chapter 5, we learned that adversarial sets lie in image space that are on the data manifold, but cross decision boundary learned by the network, mainly because of the sparsely distributed data points. Ideally, if we can move the decision boundary away from the data manifold, we should be able increase the robustness of a network f against adversarial attack. In order to achieve that, adding stochastic perturbation on images in the training set could help. Knowing that stochastic perturbation of a small magnitude does not change the class of image class, but may cross the decision boundary learned by

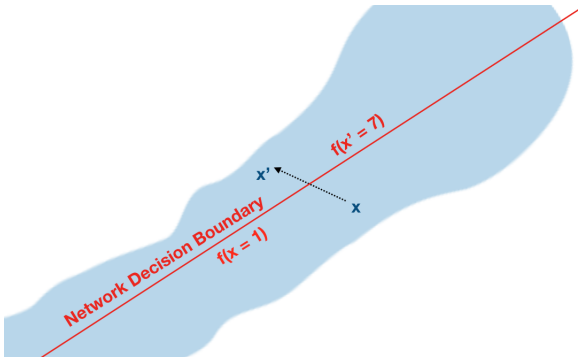

Figure 8.1: A cartoon of input space. Blue region is the manifold of all images classified as 1. By stochastically perturbing image x in class 1 to x' and train network with x' labeled as 1, x' lies still on the manifold of 1, but cross the decision boundary that partitions the data manifold. Hence, a network trained on an augmented data set should be more robust to adversarial attacks.

the network. One example is illustrated in figure 8.1, a data in the training set x is on the image manifold of 1, after stochastically perturbing x to x', x' stays on the manifold of 1, but crosses the network's red decision boundary. In this case, the network trained on x fails to accurately classify x', but a network trained on both x and x' would correctly classify both data.

By training with stochastically perturbed images, network is forced to classify images that are randomly a small distance away from images in the training data. So it should be expected that the robustness of network trained on augmented data should improve.

8.3 Robustness Measure

To evaluate the robustness of a network f against an adversarial perturbation of average L_2 norm of magnitude ϵ, we utilize the fact that perceptual classification does not change for x and x', i.e. $g(x) = g(x')$. Each data x' in $\mathcal{D}'_{test}$ is generated from x in $\mathcal{D}_{test}$ that are ϵ distance away from x measured by:

$$\epsilon = \sqrt{\frac{\sum_{k=1}^{N}(x_k - x'_k)^2}{N}} \tag{8.1}$$

The classification accuracy can be generalized to adversarial test set with perturbation ϵ: $\mathcal{D}'_{test}(\epsilon)$. Each image in $\mathcal{D}'_{test}$ with each of its images x' being attack images of its corresponding x from the original test set $\mathcal{D}_{test}$ with perturbation magnitude ϵ generated using FGSM. Essentially, $\mathcal{D}'_{test}(\epsilon)$ is an attack set of $\mathcal{D}_{test}$. Accuracy of a network f

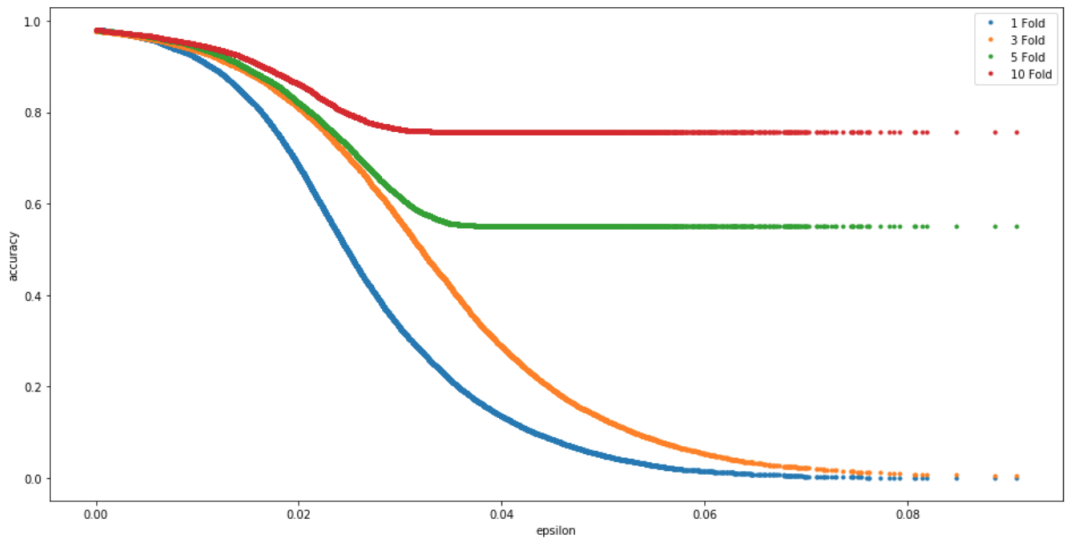

Figure 8.2: Accuracy versus epsilon robustness measure on network trained with 1, 3, 5 and 10 folds of stochastically augmented training set.

evaluated on $\mathcal{D}'_{test}(\epsilon)$ can be calculated by:

$$Accuracy = \frac{|f(x') = g(x)|}{|\mathcal{D}'_{test}|} \tag{8.2}$$

We know $g(x)$ from the data set. We can use this augmented accuracy evaluation as a robustness measure.

8.4 Experiment

To augment the training set $\mathcal{D}_{train}$, a perturbed images x' from x is generated by adding stochastic noise magnitude α on every pixel of x:

$$x' = clip_x(x + \alpha) \tag{8.3}$$

The $clip_x$ function make sure x' stays in the range specified in input space. Further,

$$\alpha \sim 0.1 \times 2 \times (Bernoulli(0.5) - 0.5) \tag{8.4}$$

We generate 1, 3, 5, and 10 folds of perturbation on the training set $\mathcal{D}_{train}$. For each n fold perturbation, it means that $n - 1$ stochastic images x' is generated for each image x in $\mathcal{D}_{train}$, as a result, the number of images used to train the network increases by 0, 2, 4, and 9 folds.

We fix the network architecture to the fully connected, feed forward architecture (784 x 1024 x 10) described in Chapter 3. Then network f_1, f_3, f_5, and f_{10} with this architecture

and the same initial parameters independently trains on their corresponding augmented training data with 1, 3, 5, and 10 folds of perturbation until convergence. We then evaluated the robustness of networks f_1, f_3, f_5, and f_{10} using extended accuracy measure with equation 8.2. Their performance with an increasing ϵ is shown in figure 8.2.

8.5 Result and Conclusion

The performance of f_1, f_3, f_5, and f_{10} with an increasing ϵ are shown in blue, orange, green and red lines. With our naive network f_1 depicted with blue line, we see that when ϵ is 0, accuracy is the classification accuracy on the test set. For f_1, classification accuracy in the adversarial attacked test set $\mathcal{D}'_{test}(\epsilon)$ drops quickly with an increasing ϵ, when ϵ is above 0.06, accuracy of f_1 drops down to 0. This means that if a network is not trained with noise, then its learned decision boundary can be so close to data that a perturbation magnitude with 0.06 is able to fool all of the network's classification on the test set. Meanwhile, the performance of networks trained with more folds of noise become better. While accuracy of f_3 also converges to 0, it is at a much slower pace, further, accuracy of f_5 converges to around 0.5 when ϵ increases to above 0.05. In addition, accuracy of f_{10} converges to around 0.8% as ϵ increases to be above 0.04. This means that above 50% of test image is not vulnerable to adversarial attack with magnitude above 0.05, and for f_{10}, around 80% of test image is not vulnerable to adversarial attack with magnitude above 0.04. This implies that by adding noise perturbation away from training images, we can make sure that the test data stays within sufficient distances away from the decision boundary, which fullfills our intention in 8.2.

However, there are two main limitation of this method: first, one needs a lot of noise perturbed data to increase the robustness of network. This issue is not so apparent in our MNIST data set, but can be a problem in larger data sets. Secondly, we augmented training image with noise magnitude of 0.1, and are able to push the decision boundary further. However, it does not rule out the possibility that when ϵ is above 0.1, the accuracy measure for f_{10} and f_5 might also drop to 0, since those neural nets are trained with noise augmented magnitude of 0.1, it does not rule out the possibility of adversarially classifiy images that are more than 0.1 ϵ away from the data set.

Conclusion

To get an intuitive understanding on why neural network image classifiers are vulnerable to adversarial attacks, we base our study on a simple 3 layer fully connected feed forward architecture learning to classify hand written digit images on the MNIST data set. By speculating on causes of adversarial vulnerability, we narrowed our hypothesis down to the linear transformation feature of neural network learning to disentangle visually meaningful image classes along a small dimension of manifold. Although previous literature also suggested that NN's linear transformation feature can be related to its adversarial vulnerability. This hypothesis was not generally accepted due to its lack of empirical evidence.

As understanding the topological characters of high dimensional image data is difficult, we associate this problem with the manifold hypothesis which states that those visually meaningful hand written digits lie on a low dimensional manifold. Using a box counting method, we estimated that the dimension of the MNIST data to be in a 6-dimensional manifold embedded in the 784-dimensional image space.

By its set up, a neural network separates the input space into label sets bordered by its decision boundary. To merge this with our understanding of visually meaningful images, we suggest to see the adversarial image sets as regions of input space that differentiate network decision and manifold class.

This understanding allows us to ask the question: how are decision boundary located around image manifolds. To study this, we sampled images from the data set and compares its distance to the network's decision boundary and the corresponding manifold. We showed that the distance ratio between the two is less than 0.20 for 90 % of sampled images. This result suggests that rather than finding the margin that farthest separates classes of image manifolds, the learned network decision boundaries are highly curved and lie very close to images in the data set. We suggest that this highly curved structure could be due to having a high degrees of freedom allowed by the large parameter set. The decision boundary's proximity to images in the data set can be caused by the gradient descent algorithm adjusting parameters, which assumes a high degree of freedom, in small steps away from incorrectly classified images.

To study how linear transformation feature affects adversarial vulnerability, we divided the hidden units into sub-population supporting adversarial class and benign class and compared their responses between one pair of benign image and its adversarial attack image. We observed the small visually indistinguishable perturbation on adversarial image influence denser weak population activation for sub-population supporting adversarial class, and sparser activation for hidden sub-population supporting benign class. To quantify this effect, we use GINI index as a sparsity measure and showed that this effect is observed in most adversarial attacks. We showed that the small visually indistinguishable perturbation on adversarial image, by linear transformation, induces dense, weak sub-population activation of hidden units supporting the adversarial label and sparsifies activations of those sub-population supporting the benign label. This effect is then being summed up by subsequent layer in an additive fashion, altering the network decision from the benign label to the attack label.

The linear transformation feature allows us to study subspace of images that leads to equivalent activation for one neuron and an entire layer of neurons. Another effect of the linear transformation by the network is the subspace of images that equivalently activates one neuron and an entire layer of neurons. We showed that rather than being visually equivalent or similar, some nonsense image are perceived by the network the same as a hand written digit image. This suggests that we should not interpret artificial neural activation as encoding the prescience or absence of specific patterns, as compared to visual neurons.

Our suspect of linear transformation causes adversarial vulnerability could be used to explain the reason adversarial attacks are transferrable across independently trained neural networks even with different architectures. It could be that linear transformation with different architectures learns similar linear subspace of image space that partitions the decision boundary of the network to disentangle data manifolds.

Finally, building on our understanding of manifold and decision boundary, we verified a simple attempt to increase the robustness of neural network by pushing the decision boundary away from data instances through augmenting the training set with stochastically perturbed images. We showed that this can effectively increase the robustness to FGSM with the specified perturbation margin. Specifically, by adding 10 fold stochastically perturbed images for each original data with pixel-wise perturbation magnitude 0.1, the robustness of neural network towards adversarial attacks with average L_2 norm converges to around 80%, compared to the less than 5% of the nonstochastically trained original neural network.

We acknowledge that our attempts to increase robustness for linear networks simply by pushing the decision boundary away from data is preliminary and has limitations. Rather,

we suggest that future study of finding architecture robust to adversarial attacks should think about replacing the linear transformation part of the architecture to an alternative layer wise transformation that is more similar to how the visual neurons identify patterns in images. Specially, they should have the following characteristics:

- Small perturbations in image should not be able to influence hidden neural population in an accumulative manner.
- Subspace of images that equivalently perceived by the neural network should resemble visual similarities to each other
- In the input space, the decision boundary of an adversarial robust neural network should have large margins between manifold classes.

As the adversarial problem is still unsolved today, those insights could be a guidance toward building a robust neural network against adversarial attacks.

Bibliography

[1] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. pages 1–11, 2014.

[2] Yoshua Bengio. Learning Deep Architectures for AI.

[3] Nikolaus Kriegeskorte. Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science*, 1(1):417–446, 2015.

[4] Li K Wenliang and Aaron R Seitz. Deep Neural Networks for Modeling Visual Perceptual Learning. *The Journal of Neuroscience*, 38(27):6028 LP – 6044, 7 2018.

[5] Lane T Mcintosh, Niru Maheswaranathan, and Aran Nayebi. Deep Learning Models of the Retinal Response to Natural Scenes. *Advances in Neural Information Processing Systems 29 (NIPS)*, 2016.

[6] Rishi Rajalingham, Elias B. Issa, Pouya Bashivan, Kohitij Kar, Kailyn Schmidt, and James J. DiCarlo. Large-scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural networks. *The Journal of Neuroscience*, 2018.

[7] F Rosenblatt and Contract Nonr. THE PERCEPTRON : A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION. 65(6):386–408.

[8] Cybenko. Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 1989.

[9] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 1991.

[10] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel,

and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

[11] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. DeepReID: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014.

[12] Robert Geirhos, Carlos R. Medina Temme, Jonas Rauber, Heiko H. Schuett, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. 2018.

[13] Robert Geirhos. Generalisation in humans and deep neural networks.

[14] Matteo Carandini and David J. Heeger. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1):51–62, 2012.

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. 7 2016.

[16] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. 3:1–16, 2018.

[17] 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*, 2016.

[18] Xin Li and Fuxin Li. Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[19] Gamaleldin F. Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alex Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial Examples that Fool both Human and Computer Vision. *arXiv Preprint*, 2018.

[20] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. E XPLAINING AND H ARNESSING. pages 1–11, 2015.

[21] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proceedings - IEEE Symposium on Security and Privacy*, 2017.

[22] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. 12 2017.

[23] Christian Szegedy, Ian Goodfellow, Joan Bruna, Rob Fergus, and Dumitru Erhan. Intriguing properties of neural networks. pages 1–10.

[24] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, Pieter Abbeel, and Jack Clark. Attacking machine learning with adversarial examples. *Www.Openai.Com*, 2017.

[25] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. 5 2018.

[26] Yash Sharma and Pin-Yu Chen. Attacking the Madry Defense Model with L_1-based Adversarial Examples. 10 2017.

[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, (2017):1–23, 2018.

[28] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses. pages 1–20, 2017.

[29] Nicholas Carlini and David Wagner. Defensive Distillation is Not Robust to Adversarial Examples. 7 2016.

[30] Wieland Brendel and Matthias Bethge. Comment on "Biologically inspired protection of deep networks from adversarial attacks". pages 1–4, 2017.

[31] Shiwei Shen, Guoqing Jin, Ke Gao, and Yongdong Zhang. APE-GAN: Adversarial Perturbation Elimination with GAN. 7 2017.

[32] Dongyu Meng and Hao Chen. MagNet: a Two-Pronged Defense against Adversarial Examples. 2017.

[33] Nicholas Carlini and David Wagner. MagNet and "Efficient Defenses Against Adversarial Attacks" are Not Robust to Adversarial Examples. 11 2017.

[34] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. 48, 2015.

[35] David Duvenaud, Oren Rippel, Ryan P. Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. pages 1–20, 2014.

[36] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering Adversarial Images using Input Transformations. 10 2017.

[37] Pratik Prabhanjan Brahma, Dapeng Wu, and Yiyuan She. Why Deep Learning Works: A Manifold Disentanglement Perspective. *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[38] Niall P. Hurley and Scott T. Rickard. Comparing Measures of Sparsity. 11 2008.

[39] Randall C. O'Reilly, Dean R. Wyatte, and John Rohrlich. Deep Predictive Learning: A Comprehensive Model of Three Visual Streams. 2017.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Neuroinformatics
Prof. Dr.Jean-Pascal Pfister

Title of work:

Understanding the Vulnerability of Artificial Neural Networks to Adversarial Examples

Thesis type and date:

Master Thesis, February 7, 2024

Supervision:

Jannes Jegminat
Prof. Dr. Jean-Pascal Pfister

Student:

Name:	Shuchen Wu
E-mail:	shwu@ini.uzh.ch
Legi-Nr.:	17-781-743
Semester:	4

Statement regarding plagiarism:

By signing this statement, I affirm that I have read and signed the Declaration of Originality, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Declaration of Originality:

`http://www.ethz.ch/faculty/exams/plagiarism/confirmation_en.pdf`

Zurich, 7. 2. 2024: _____