A COMPARISON OF CNN AND DBN IMAGE CLASSIFICATION MODELS UNDER ADVERSARIAL CONDITIONS

by

Tao Yang

Thesis submitted in partial fulfillment of the

requirements for the Degree of

Bachelor of Science with

Honours

Acadia University

(December) 2020

This thesis by Tao Yang
is accepted in its present form by the
Jodrey School of Computer Science
as satisfying the thesis requirements for the degree of
Bachelor of Science with Honours

Approved by the Thesis Supervisor

Dr. Daniel Silver	Date

Approved by the Director of the School

Dr. Darcy Benoit	Date

Approved by the Honours Committee

Dr. Greg Lee	Date

Acknowledgements

First and foremost, I would like to show my deepest gratitude to my supervisor, Dr. Daniel Silver. Without his help, the goal of this work would not have been realized.

I shall extend my thanks to all professors in the computer science department for their help. I would also like to give a special thank to Dr. Jim Diamond, Dr. Greg Lee, Dr. Zhang Haiyi and Professor Duane Currie, as they gave me accurate and efficient help on many deep questions. They also provide extra convenience and help which is much more than I should get.

Last but not least, I would like to thank all my friends and family for their financial and moral support.

Contents

List of Tables

List of Figures

Abstract

We investigate and compare Convolutional Neural Networks (CNN) and Deep Belief Network's (DBN) ability to withstand several common attacks to limit their performance.

We propose that CNN makes a strong inductive bias assumption about the relationship between pixels that are proximal to each other that makes it valuable to adversarial attacks.

We implement several attacks using the MNIST and CIFAR-10 dataset where we modify pixels of the test images in different ways to challenge the CNN and DBN models. Each experiment is run multiple times to develop and test models. The accuracy of the models for each method is analyzed.

The results show that DBN models generally perform better under attack than the CNN models. When the assumption of the relationship between pixels is removed, the advantage of CNNs convolutional inductive bias no longer exist.

Chapter 1

Introduction

For the last twenty years, people have made great progress in implementing computer vision classification systems using high-performance hardware technology and new machine learning algorithms. Convolutional Neural Networks (CNN) are the dominant machine learning algorithms for computer vision in 2020 and people are starting to think that we have reached the peak, because CNN has higher accuracy than humans. However, people have noticed that CNN will successfully classify one picture but fail on a very similar one that is transformed from the first.

1.1 Definition of Problem

In this thesis, I will focus on CNN and Deep Belief Network and images that attack their performance. We cannot trust what the user will input to a computer program when we start to use classification algorithms in daily life and do important jobs. Therefore, we need to assume that the classification algorithm may receive input that is designed to have the AI fail to do its job.

1.2 Research Objectives & Approach

Deep learning has been applied to fields such as computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and some game programs. They have produced better results than human experts in some cases.

We wish to show that the CNN inductive bias, according to proximity of pixels, which assumes pixels adjacent to each other will tend to have the same colour and brightness, fails under certain adversarial attacks. DBNs, which learn features with a less specific inductive bias with no assumptions about the proximity of pixels, will not fail in the same way.

The primary undertaking in this research is to compare self-supervised methods (such as DBNs) to contemporary human-engineered machine learning methods (such as CNN) on classification problems that suffer from adversarial attacks.

1.3 Research Scope

This research will focus on small grey scale and colour images from the MNIST and CIFAR-10 datasets using CNN and DBN machine learning algorithms.

1.4 Overview

This thesis is structured as follows: Chapter 2 provides the necessary background information including relevant information about the history and architecture of the CNN and DBN algorithms. Chapter 3 presents our theory and approach as well as providing the model deep network architecture and the specific software packages, and hardware that will be used. Chapter 4 describes the experiments that compares CNN and DBN as per the theory of Chapter 3. Chapter 5 compares the model, and the results and provides analyses. Chapter 6 contains the conclusions and discusses future work.

Chapter 2

Background

Computer vision based on machine learning has spawned a large number of fast-growing, practical applications such as human face detection (Snapchat), image retrieval (Google Images), game and control (Alphastar, OpenAI, Kinect), surveillance, biometric technology and smart automatic-driving cars.

In this chapter, I will introduce two machine learning algorithms which are used in computer vision: Convolutional Neural Networks and Deep Belief Networks. Both algorithms are based on Artificial Neural Networks (ANNs) and the back-propagation algorithm.

2.1 Artificial Neural Networks

ANN refers to a complex network structure formed by the interconnection of many processing units called neurons. An ANN is an information processing system based on imitating the structure and function of mammalian neural networks (see Figure 2.1). It is an abstraction, simplification, and simulation of the human nervous system and its organizational structure and operating mechanism. ANNs simulate the activity of neurons with a mathematical model. The back-propagation (BP) method, short for the "backward propagation of error" is a gradient descent optimization method used to train an ANN. This method calculates the gradient of the loss function for each weight in the network to minimize the error of the network's predictions.[6]

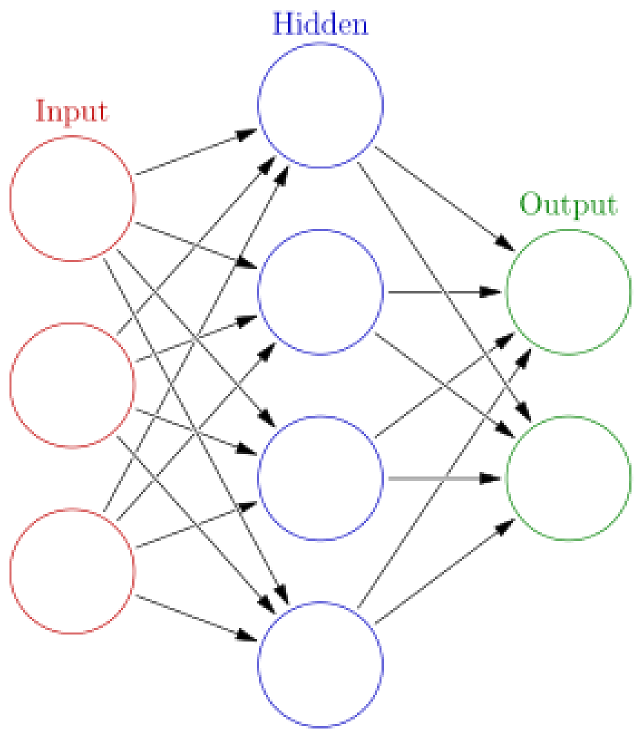


Figure 2.1: Basic ANN[6]

2.1.1 Back Propagation

Since backpropagation uses the gradient descent method, it is necessary to calculate the derivative of the squared error function with respect to the network weights. Suppose that for an output neuron, the squared error function is:

$$E = L(t, y) \tag{2.1}$$

where E is the loss between t the target output for a training example and y is the network output for that example. For every neuron j in the network, its output $\mathbf{o}_j$ is defined as

$$o_j = \varphi(\text{net}_j) = \varphi\left(\sum_{k=1}^{n} w_{kj} o_k\right) \tag{2.2}$$

The activation function φ is generally a nonlinear differentiable function such as the logistic function:

$$\varphi(net_j) = \frac{1}{1+e^{-net_j}} \tag{2.3}$$

which has a convenient derivative of:

$$\frac{d\varphi(net_j)}{dnet_j} = \varphi(net_j)(1-\varphi(net_j)) \tag{2.4}$$

if the error is the sum of squared error $E = \sum(o_j - t_j)^2$, then:

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net}_j} = \begin{cases} (o_j - t_j)o_j(1-o_j) \text{ if } j \text{ is an output neuron,} \\ (\sum_{\ell \in L} w_{j\ell}\delta_\ell)o_j(1-o_j) \text{ if } j \text{ is a hidden neuron.} \end{cases} \tag{2.5}$$

To update the weight $\mathbf{w}_{ij}$ using gradient descent, one must pick a learning rate which is η. The change in weight to be added to the original weight is equal to the product of the learning rate and the gradient, multiplied by -1.

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} = -\eta o_i \delta_j \tag{2.6}$$

2.2 Convolutional Neural Networks

A Convolutional Neural Network such as the one shown in Figure 2.2 is a feed-forward neural network with excellent performance in large-scale image processing. It has been widely used in image classification, object detection, target tracking and object counting. In comparison to other neural network structures, Convolutional Neural Networks require relatively few parameters, making them widely applicable.

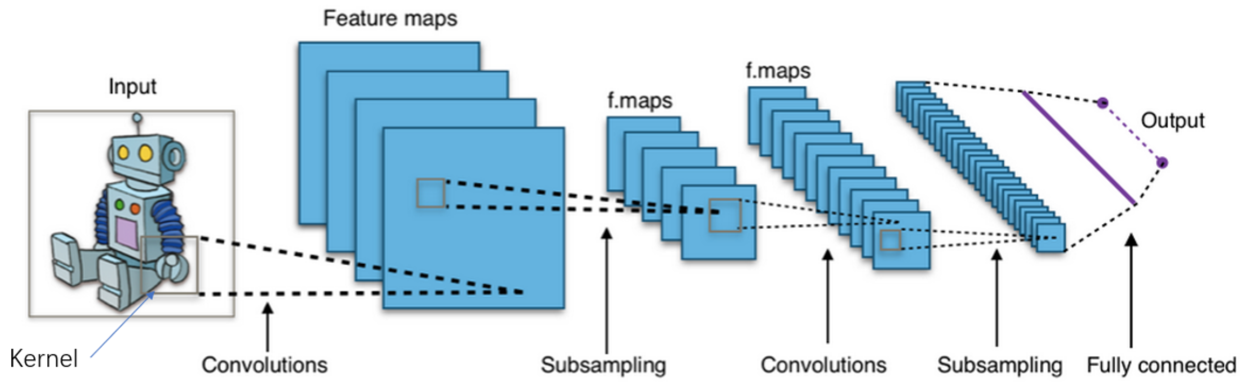


Figure 2.2: General Convolutional Neural Networks Structure[7]

2.2.1 History of CNNs

The Convolutional Neural Network is one of the most used neural networks in the field of deep learning technology. It has made many breakthroughs in the field of computational vision. Neural networks have reached many achievements such as image feature extraction, classification, and scene recognition. When compared with traditional image processing algorithms, one of the advantages of convolutional neural networks is that they avoids complex image pre-processing. Convolutional neural networks can directly input the original image for a series of experiments.

2.2.2 The Embryonic Stage

In 1962, Hubel and Wiesel showed that the transmission of visual information from the retina to the brain was accomplished by multi-level receptive fields, and the concept of receptive fields was first proposed[22]. In 1980, Japanese scholar Fukushima proposed a neurocognitive machine based on the cognitive domain[16]. The neurocognitive machine is a neural network model with a self-organizing multi-layer. The response of each layer is stimulated by the local receptive field of the upper layer. The recognition of the pattern is not affected by the position, small shape changes, or scale. This machine can be understood as the first version of the convolutional neural network. The core point is to model the visual system and prove it is not affected by the position or size of the target objects.

2.2.3 Experimental Development

In 1998, computer scientist Yann LeCun and others proposed LeNet5 to use gradient-based back propagation algorithm to supervise a network. LeCun has made outstanding contributions in machine learning and computer vision and is known as the father of convolutional neural networks[23]. The LeNet5 network gradually converts the original image into a series of feature maps through alternately connected convolutional layers and down sampling layers (described in section 2.3). Then it transfers these features to a fully connected neural network to classify the images according to their characteristics[25]. The receptive field is the core of the convolutional neural network, and the convolution kernel of the convolutional neural network is the structural expression of the concept of the receptive field. Since the publication of LeNet5, the academic community has successfully applied CNN to problems such as on handwriting recognition, speech recognition, object detection and face recognition[17].

2.2.4 Large-scale Application and In-depth Research Stage

After the LeNet5 network, the Convolutional Neural Network was in the experimental development stage. It was not until 2012 that the proposal of the AlexNet network established the position of convolutional neural network in deep learning applications[2]. The network AlexNet won the championship of image classification on the training set of ImageNet, making the Convolutional Neural Network a key research object in computer vision. After AlexNet, new Convolutional Neural Networks were proposed, including Oxford University's VGG network, Microsoft's ResNet network[4] and Google's GoogLeNet network[3]. These networks have made Convolutional Neural Networks become a dominant method on image classification tasks in commercial applications.

2.2.5 Recent Use of CNNs

It is expected that Convolutional Neural Networks will continue to develop and be used for various application scenarios. For example, 3D convolutional neural networks for video understanding. It is worth noting that convolutional neural networks are not only applied to image-related networks, but also networks similar to images such as analyzing chessboards and Go boards[5].

2.3 CNN Architecture

2.3.1 Convolutional Kernel

The function of the convolutional layer is to extract features from input data, which contains multiple convolution kernels as shown in Figure 2.2. Each element of the convolution kernel corresponds to a weight coefficient and a bias input, similar to a feedforward neuron of a standard ANN. Each neuron in a kernel is connected to a neuron in the next layer called a feature map. The area of pixels covered by the convolution kernel is called the "receptive field". Its meaning can be compared to the receptive field of visual cortical cells. When the convolution kernel is working, it will sweep over the pixels of the image and compute the sum of the product of each kernel element times its respective pixel value. At each step as the kernel sweeps over the image, a new feature is computed and added to the feature map in the next layer of the network. The convolution layer's job is to detect simple features such as edges, corners, variations in color, and brightness that can be used to create higher order features[19].

2.3.2 Pooling or Down Sampling Layer

After feature extraction is done in the convolutional layer, the output feature map is passed to the pooling or down sampling layer for feature selection and information filtering. Pooling layers usually combine the outputs of neuron clusters at one layer into a single neuron and pass them to the next layer in order to reduce the dimensions. This is a fast method to down sample and still maintain the important aspects of the input. The selection of the pooling area in the pooling layer is the same as the steps of the convolution kernel scanning feature map.

2.3.3 Dropout

The dropout layer will deny the contribution of some neurons towards the next layer, otherwise the first batch of training samples influences the learning too much. This prevents overfitting since some learning of features only appear in later samples or batches.

2.3.4 Fully Connected Layer

The fully connected layer or dense layer is the same as the traditional feedforward neural network. The fully-connected layer is the last part of the convolutional neural network. The outputs of the last feature map in Figure 2.2 are converted into a vector, and passed to one or more dense layers. According to the representation learning perspective, the convolutional layer and the pooling layer in the convolutional neural network extract features from the input data, and the role of the fully connected layer is to nonlinearly combine the extracted features to obtain the output. That is, the fully connected layer tries to use the higher-order features created by the convolution and pooling layers to complete learning goals.

2.4 Deep Belief Networks

2.4.1 Deep Belief Networks Introduction

Deep Belief Networks (DBN) are structured autoencoders that are further trained to do classification or regression. Specifically, a DBN can be constructed of multiple Restricted Boltzmann Machines (RBM) and can be fine-tuned for classification using gradient descent and back-propagation algorithms after adding a fully connected layer at the top.

Researchers such as Geoffrey E. Hinton (British-Canadian cognitive psychologist and computer scientist) proposed RBM learning machines simultaneously from different fields with different motivations.[18] The RBM is composed of a visible neuron layer and a hidden neuron layer (see Figure 2.3). Since the hidden layer neurons are not connected to each other they are independent of each other, which makes a direct calculation of data-dependent expectations easy.

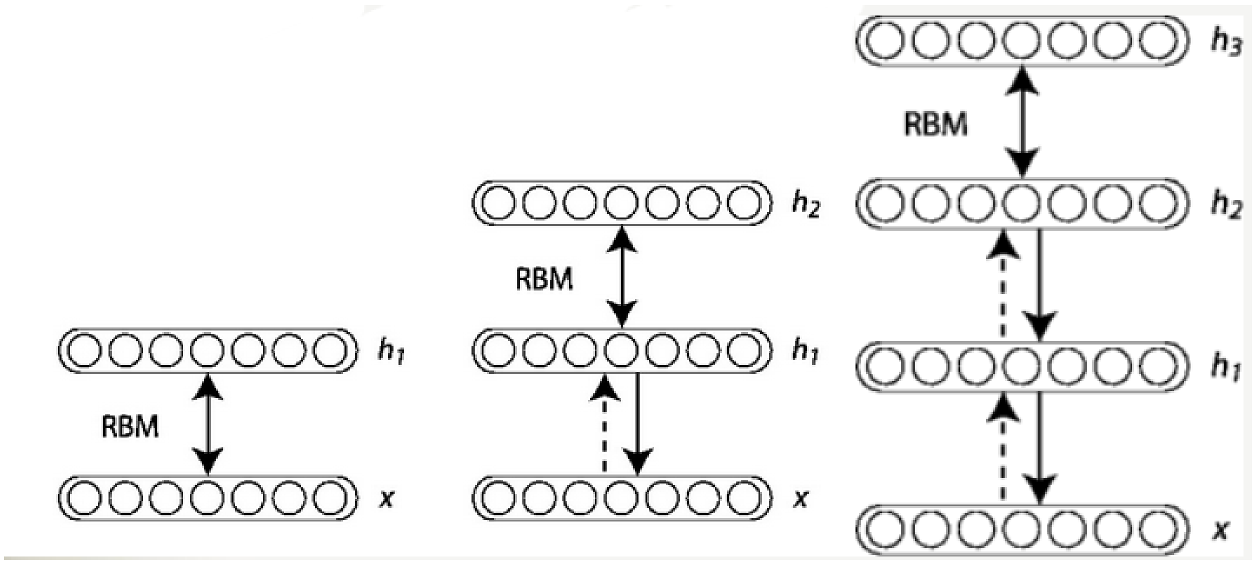


Figure 2.3: Training a Supervised DBN Using Multiple RBM Layers[1]

2.4.2 Boltzman Machine

A Boltzman Machine is a stochastic recurrent neural network proposed by Hinton and Sejnowski[18]. It can be regarded as a randomly generated Hopfield network[10]. It is one of the earliest artificial neural networks that can solve difficult learning problems through the inherent representation of learning data. The sample distribution follows the Boltzmann distribution in physics and so is called Boltzman Machine or BM. BMs are composed of binary neurons where each neuron only takes two states, 1 or 0. State 1 represents that the neuron is on, and state 0 represents that the neuron is off. In the following discussion, nodes will represent neurons.

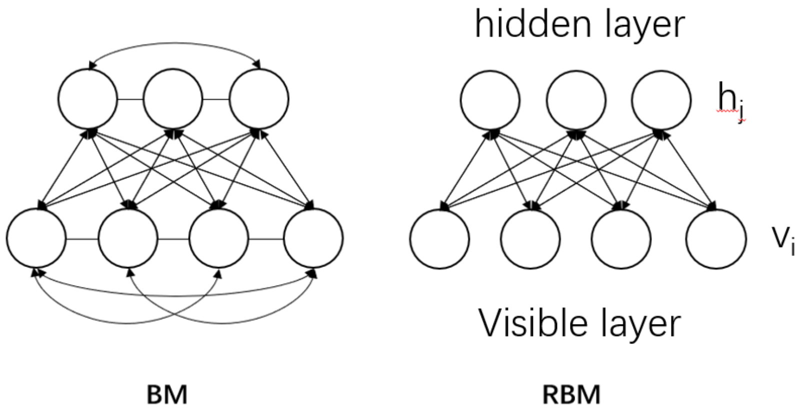


Figure 2.4: Difference between BM and RBM

2.4.3 Difference Between BM and RBM

An RBM is a restricted BM. The intra-layer connections of the visible layer and the intra-layer connections of the hidden layer are removed (see Figure 2.4). This reduces the training time of the intra-layer connections of the BM. It also creates a bipartite graph which allows for some nice mathematical properties.

In fact, it can be shown that some visible nodes x and hidden nodes h, the RBM learns $P(h|x) = \prod_j P(h_j|x)$ and $P(x|h) = \prod_i P(x_i|h)$[10].

2.4.4 Training Algorithm

The algorithm most used for training RBMs is the contrastive divergence (CD) algorithm proposed by Jeffrey Hinton[21]. This algorithm was first used to train the "expert product" model proposed by Hinton. This algorithm uses Gibbs sampling to update the weights during the gradient descent process, which is similar to the back-propagation algorithm used in training feedforward neural networks and it performs in the following equation:

$$W_{ij}(t+1) = W_{ij}(t) + \eta \frac{\partial log(p(v))}{\partial w_{ij}} \tag{2.7}$$

where $p(v)$ is the probability of a visible vector.

While training a supervised DBN using multiple RBM layers topped by multiple dense layers, Hinton adopted a layer-by-layer unsupervised method to learn the network weight bias weight parameters (see Figure 2.3). First, the data vector x and the first hidden layer are used as an RBM and the parameters of this RBM are trained (the weights connecting x and $h1$). Then the parameters of this RBM are fixed, and $h1$ is treated as a new visible vector and $h2$ as a hidden vector. This second RBM is then trained. The parameters of this RBM are then fixed as well and the next hidden layer is used to create a third RBM.

2.4.5 Application of DBNs and RBMs

DBN and RBM models have been successfully applied in collaborative filtering, classification, dimensionality reduction, image retrieval, information retrieval, language processing, automatic speech recognition, time series modeling, document classification, nonlinear embedded learning, transient data model learning, and signal and information processing. One significant advantage of DBNs is that they can be pre-trained using a large collection of unlabelled data, and then fine-tuned using a relatively small amount of labelled data.

2.5 Previous Confrontations Between DBN and CNN

Region-CNN(R-CNN) is the first algorithm to successfully apply deep learning to target detection[9]. Region-CNN(R-CNN) was introduced at CVPR 2014 with impressive results on PASCAL VOC detection. However R-CNN needs to pre-train on ImageNet before training on PASCAL VOC. Scholars finally embraced deep learning but a Russian-American computer scientist called Alyosha Efros wondered why PASCAL VOC had to be pre-trained first on the ImageNet data. From this, the Gelato Bet was born to answer the scientific question of whether one needs semantic supervision to learn a good representation[13]. It is called Gelato Bet because the wager is only one ice-cream. More details about this bet will be provided later in this section.

Alyosha Efros hypothesized that a method will exist that can outperform of R-CNN on PASCAL VOC without using a human annotations as pre-training before September 23th of 2015. Although he lost the bet that no method can match or beat the performance of R-CNN on Pascal VOC detection without extra human labeled data as pretraining, self-supervised methods now surpass the supervised algorithm at Pascal VOC detection.[20] Furthermore, the article pointed out that a large class of current machine learning methods rely on human provided-labels or rewards during the training process, which leads to several problems. Purely supervised algorithms often require a large number of samples to train, and converge to brittle solutions. People cannot rely on machine learning for high dimensional problems and the cost of acquiring labels is much higher in real life. The labeled training algorithm can only solve specific problems but not gain knowledge that can be repurposed. Self-Supervised Learning provides a promising alternative where the data itself provides supervision for a learning algorithm.[14]

It is shown in [14] that contrastive methods trained on unlabelled ImageNet data and evaluated with a linear classifier surpass the accuracy of supervised algorithms such as AlexNet. In this paper, LeCun points out that there are three types of learning. Reinforcement learning lets the machine predicting a scalar reward given once in a while, supervised learning lets the machine predicting a category for each input then check if it is correct or not, self-supervised learning lets the machine predict any part of its input for any observed part. These three types of learning correspond to weak, medium, and strong feedback respectively. In an experimental environment, mistakes are expected and acceptable, but in real life, mistakes can have real costs so we are trying to understand how people learn quickly in ways that require only limited labeled data to avoid damage when we need the machines to do things for humans. The author brings up the idea that self-supervised learning is trying to predict the future from the past by predicting any occluded part from all available parts. Their approach performs similar to the functionality of RBMs used in Deep Belief Network.[24]

PASCAL VOC

The PASCAL Visual Object Classes (PASCAL VOC) provides a set of standardized data sets for image recognition and classification. An image recognition challenge was held every year between 2005 and 2012[15].

Gelato Bet

After R-CNN came out in CVPR 2014 with impressive results on PASCAL VOC detection, Alyosha Efros questioned why R-CNN training can help the other if PASCAL and ImageNet have completely different label sets and biases. So he suggested that maybe the network did not need the ImageNet labels, just the ImageNet images to pre-train. The scientific question he wanted answered was: does one need semantic supervision to learn a good representation[13]? Alyosha Efros made a bet against his doctoral advisor, Indian-American engineer Jitendra Malik and declared that by the first day of autumn of 2015, there will be a method exist that can match or beat the performance of R-CNN on Pascal VOC detection, without the use of any extra, human annotations. If he wins, Mr. Malik promises to buy Mr. Efros one gelato.

2.6 Adversarial Attacks on CNN Vision Systems

It is possible to trick trained neural networks by carefully modifying the pixels of the images. This is called an adversarial attack, and if done well, it will make the network think an image of a panda is an image of a gibbon.[27] An example of applying noise to an image is shown in Figure 2.5.

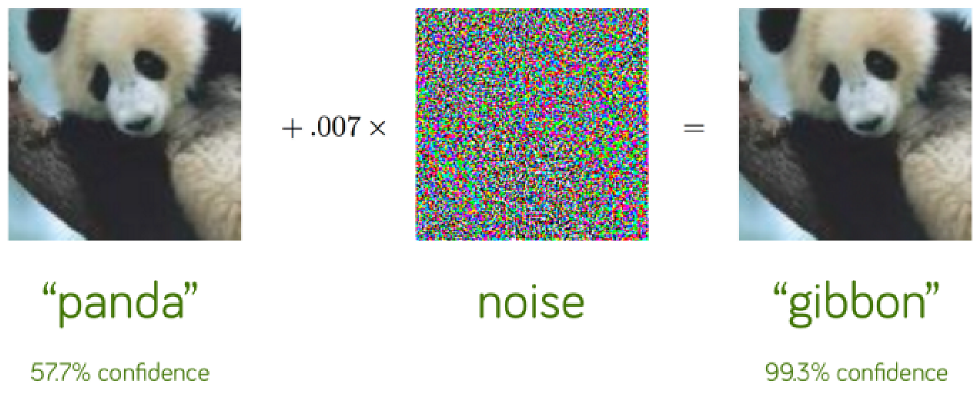


Figure 2.5: Adversarial Attack[27]

The example in Figure 2.5 requires a large number of pixel changes. Researchers have worked to discover the lowest number of pixel changes needed to fool a network. In 2018, Su and Vargas et al [27] showed that by only changing one pixel, a trained CNN network will incorrectly think a horse is a frog (see Figure 2.6). Most images in the CIFAR-10 dataset can be modified similarly to fool CNN models. Several examples are shown in the Figure 2.6.

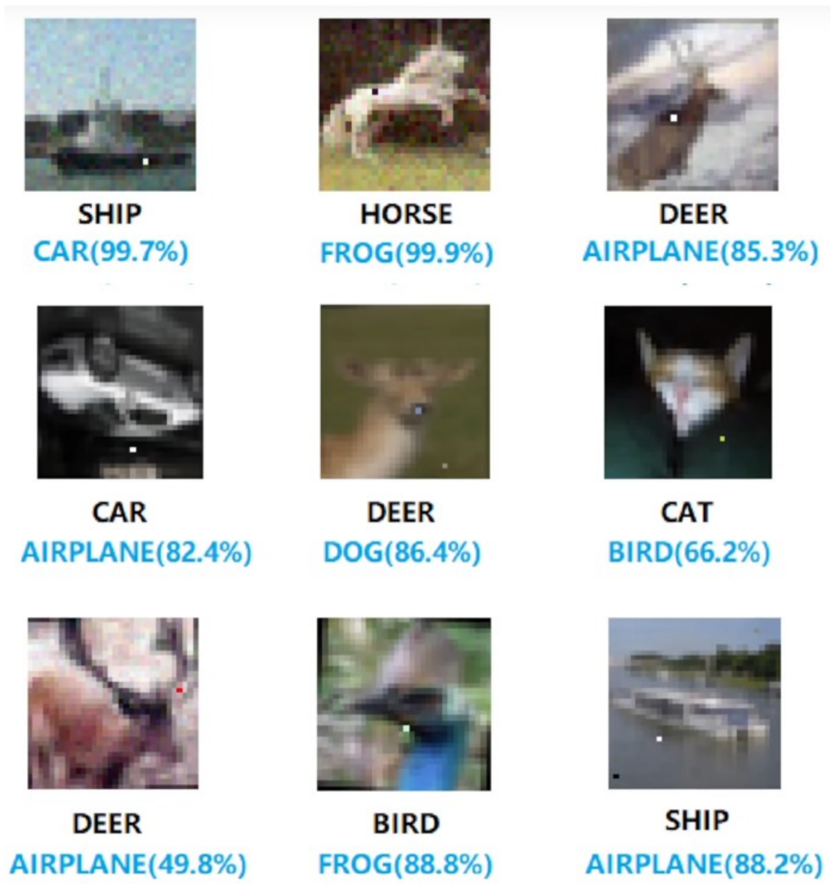


Figure 2.6: Examples of images that fooled deep learning models[27].

Chapter 3

Theory and Approach

This chapter will describe when the convolution assumption used by CNN models fails and why DBN models that do not rely on these assumptions should be superior. We then describe the fundamental approach we will take to develop and compare CNN, DBN and standard ANN models under challenging adversarial conditions.

3.1 Contrastive Self-Supervised Learning

Interest in using self-supervised methods to replace supervised methods in deep learning has been around for some time[14]. Recently, a self-supervised method has surpassed a supervised method (Mokai method proposed by He Kaiming in 2019) for Pascal VOC detection and has shown excellent results on several other tasks[14]. Behind the recent rise of self-supervised methods are DBNs and the contrastive learning algorithm discussed in Section 2.4.

Most current machine learning methods rely on human-annotated supervised target values. This over-reliance on annotation information has the risks that the underlying data has a much richer structure than sparse labels or rewards can provide. Thus, purely supervised learning algorithms often require large numbers of samples to train and converge to brittle solutions. It leads to task-specific solutions rather than knowledge that can be repurposed over a lifetime. Therefore, self-supervised learning has become a very promising method because the data itself provides supervised information for the learning algorithm.

I will demonstrate empirically that CNN will prove inferior to DBN when adversarial images are used to attack the CNN assumptions. Three experiments are undertaken: first the original MNIST is compared to mixed-pixel MNIST, second the original CIFAR-10 is compared against the modified CIFAR-10 under one-pixel attack, and third, the original CIFAR-10 compared against the modified CIFAR-10 under random noise attack.

3.2 Model Development Evaluation

The CNNs and DBNs will have similar architectures of equivalent capacity, the same percentage of train samples, validation samples, test samples, and both will be well trained with each form to prevent overfitting. The accuracy models will be tested with 500, 1000, 5000, 10000 training samples, and we will examine the mean, 95% confidence interval, and highest and lowest accuracy expected. T-test hypothesis testing will be used to show if the two models have significant differences.

3.2.1 Network Architecture

The network architecture will have three tiers for both methods. CNN will have two convolutional layers and one fully-connect layer while DBN will have three RBM layers before the output layer. Figure 3.1 shows that the CNN model has two convolutional layers, followed by a pooling and a dropout layer to prevent overfitting then a dense layer then finally a softmax output layer. Figure 3.2 shows that the DBN has three RBM layers and finally a softmax output layer. More details will be provided in Section 4 under Data Used and Approach from each experiment.

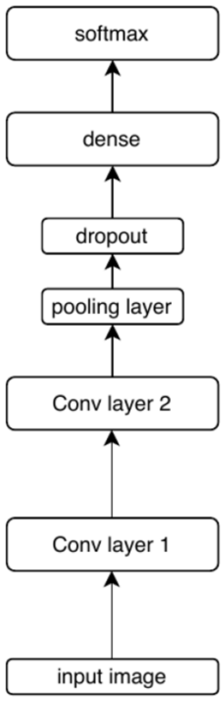


Figure 3.1: CNN Architecture

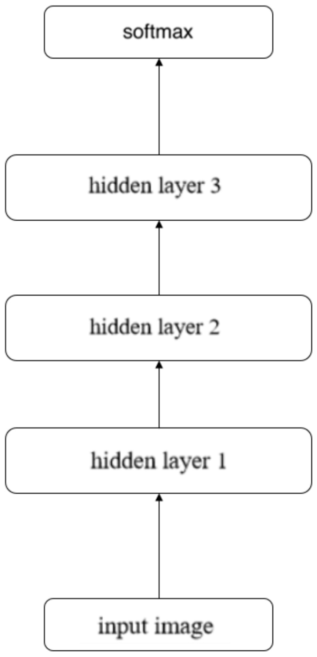


Figure 3.2: DBN Architecture

3.3 Software

This work relies on several well-known languages, libraries and packages for the creation, loading, and transformation of data as well as training and analyzing. This section shows details about each of the libraries and packages.

3.3.1 Colaboratory Cloud Computing Environment

Colaboratory (also known as Colab) is a free Jupyter Notebook environment that runs in the cloud and saves the notebook on Google Drive[8]. Colaboratory was a part of the Jupyter project at first but was eventually taken over by Google. In this research, a CNN algorithm will be implemented using Colab. The general environment is Python 3.6, Tensorflow 1.14.0, Keras 2.3.0-tf.

3.3.2 Python Data Science Stack

3.3.2.1 Tensorflow.keras

Keras is an open-source neural network database capable of running under TensorFlow, Microsoft Cognitive Toolkit, Theano and PlaidML. Keras is designed to be able to implement fast experimentation with deep neural networks while focused on being user-friendly, modular, and scalable[11].

3.3.2.2 Numpy

Numpy is an extension package of Python, it supports the operation of large, multi-dimensional arrays and matrices, and also provides many high-level mathematical functions to implement these operations.

3.3.2.3 Matplotlib

Matplotlib is a library of visual operation interface of Numpy.

3.3.2.4 Sklearn

Scikit-learn is a data preparation and machine learning library for Python. Scikit-learn is designed to interoperate with Numpy and Scipy while providing classification, regression, and clustering algorithms solution.

3.3.3 Deep Belief Network code

The Deep Belief Network we will use is a simple, clean, fast algorithm based on Python3 that follows scikit-learn guidelines that builds upon Numpy and TensorFlow libraries to take advantages of GPU computation[12].

Chapter 4

Empirical Studies

This chapter introduces, describes and analyzes the differences between CNN and DBN under three different adversarial attacks. The experiment in Section 4.1 attacks the assumption that CNN makes about the relationship between proximate pixels in an image. Section 4.2 tests the one-pixel attack on CNN and DBN models after the training of the models using normal images. Section 4.3 tests the ability of CNN and DBN models to deal with random noise applied to all pixels after training of the model using normal images. Finally, Section 4.4 discusses the results of all experiments and reflects on our theory.

4.1 Experiment 1: Mixed Pixel Attack on MNIST Dataset

4.1.1 Objective

The objective of this experiment is to prove the CNN's kernel based inductive bias, a strong assumption about the relationship between pixels can be a weakness. We scrambled all of the pixels of each picture in a consistent manner. We then trained CNN and DBN and standard ANN models and tested on the original images and the mixed-pixel images. We then compare the CNN, DBN and standard ANN models performances. The expected result is the CNN's accuracy will decreased, and DBN's accuracy will not. A standard ANN with the same architecture as DBN, but trained from the random initial weights instead of DBN pre-trained weights, is also compared to the CNN and DBN models. The standard ANN

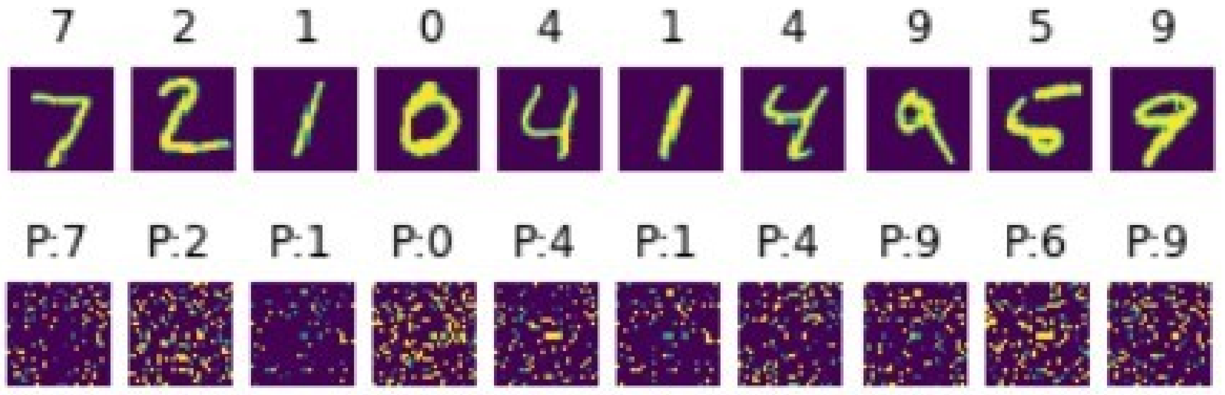


Figure 4.1: Some Examples of Experiment 1(P for Prediction)

comparison is to show that based on the same algorithm, pre-training via RBM will give an advantage to DBN. The RBM is self-supervised learning which does not require human-labeled data, which is very similar to human learning from daily observation of the world.

4.1.2 Data and Approach

Figure 4.1 shows how examples will look like when pixel is scrambled. We will make a comparison on how CNN and DBN performs on original pictures or scrambled pictures.

The DBN model will use 10,000 examples to pre-train the RBMs, with a 3 hidden layer structure of 512, 256, 512 nodes per layer. A learning rate of 0.01 will be used and each hidden layer of the DBN will be trained for 19 epochs. The model will be trained with 10,000 5,000 1,000 500 200 100 and 50 balanced sets of examples respectively. 20% of the samples will be used as a validation set. For example, with 200 samples, 160 will be training samples and 40 will be validation samples. 1500 epochs will be used to fine-tune the classification model. Early stopping will use the mean squared error of the validation set with a patient value of 100. For fine-tuning, the batch size is set to 100 and the learning rate is 0.1. Dropout is 17% and “relu” is the active function. The best model based on early stopping will be tested on 10000 test samples.

For the CNN, the structure is from VGGNET, who won the localization task in ILSVRC

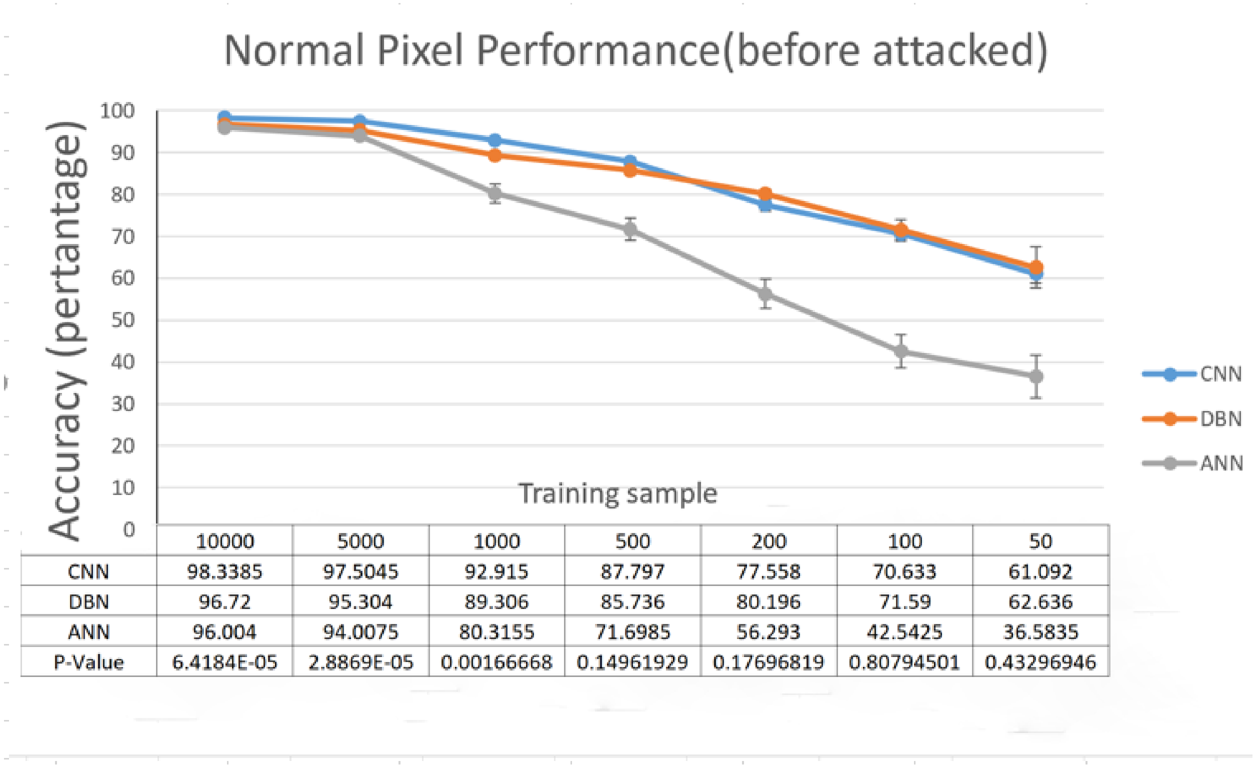
2014[26] with a dense layer which means the first conv2d layer has 64 filters with a kernel size of 3x3 and relu active function. The second conv2d layer has 16 filters with 2x2 kernel size and "relu" active function. Each convolutional layer has 2*2 maxpooling and 25% dropout. A 128 nodes dense (fully connected) layer with a 0.5 drop and softmax activation to make the prediction. CNN also uses 20% input data as a validation set and all input data are balanced 0s to 9s. The algorithm learns based on the cross-entropy and early-stop based on the validation accuracy with a patient of 100. The learning model has a 1.0 learning rate and the decay rate is set to 0.95, everything else is set to their default values.

A standard ANN is also tested based on the DBN architecture. The only difference in the ANN is that fine-tuning starts with random initial weights and not pretraining RBM weights. The weights are randomized with a standard deviation of 0.1 and the biases are randomized with the same function with a standard deviation of 1(default). The hidden layer structure is the same as the DBN network, which is 512, 256, 512 nodes per layer respectively.

4.1.3 Results

Figure 4.2 shows the mean accuracy over 5 runs for CNN, DBN and ANN to learn the MNIST dataset average. Both CNN and DBN significantly outperform the standard ANN. CNN has an advantage over DBN when the training samples are high, but then loses the advantage as the numbers of labelled training examples decreases. This is expected since CNN relies on a large number of human-labeled training samples learn an accurate model.

The p-values for T-tests of DBN and CNN models are shown in Figure 4.2. The p-value shows that for 10,000, 5,000, and 1,000 samples, alpha is less than 0.05 which means there is 95% confidence that CNN is statistical better than DBN. For 500, 200, 100, 50 train set sizes alpha is bigger than 0.05 so it shows that there is a 95% confidence that DBN and CNN have no statistical difference. It is worth noting that as the number of training example decreases, DBN's accuracy becomes better than CNN's. The p-values show that the CNN has an advantage, but then the accuracy of DBN overlaps with smaller training set sizes. This shows that CNN performs marginally better than DBN using its an assumption of the relationship between pixels.

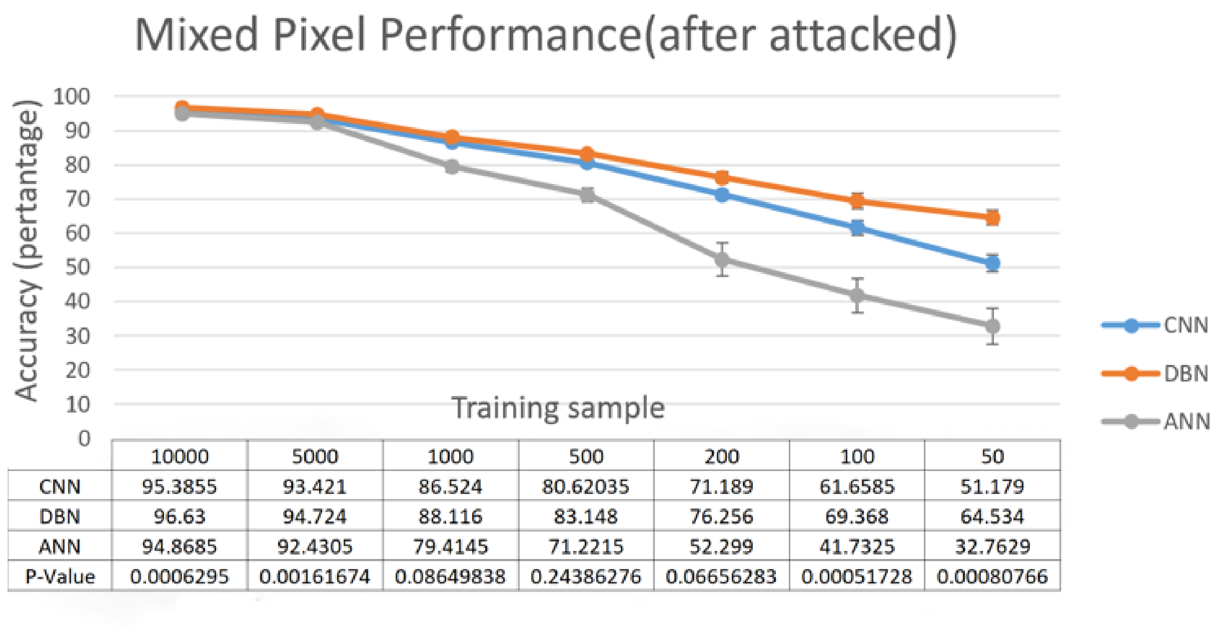


	10000	5000	1000	500	200	100	50
CNN	98.3385	97.5045	92.915	87.797	77.558	70.633	61.092
DBN	96.72	95.304	89.306	85.736	80.196	71.59	62.636
ANN	96.004	94.0075	80.3155	71.6985	56.293	42.5425	36.5835
P-Value	6.4184E-05	2.8869E-05	0.00166668	0.14961929	0.17696819	0.80794501	0.43296946

Figure 4.2: Accuracy Comparison Before Mix the Pixels

Figure 4.3 shows the performance of CNN, DBN and ANN on the MNIST dataset after mixing up the pixels of the image. Clearly, the mixing of the pixels attacks the performance of CNN, but not so for DBN whos results are stattistically the same as before the attack. The T-test for the CNN and DBN accuracy table is shown in Table 4.3. The DBN models are more accurate for all numbers of training examples. For 10,000 and 5,000 training examples, the accuracy from 5 runs of DBN and CNN has a statistical difference in favour of DBN. In the remaining 5 T-test results the mean differences for accuracy of DBN minus CNN are 1.592, 2.52765, 5.067, 7.7095, 13.355. As the number of training examples decrease, the difference between the methods increases. For 100 and 50 training examples, T-test values are smaller than 0.05 which means there are statistical differences between the accuracy of two models in favour of DBN.

The standard ANN results shows that the RBM pre-training is an important aspect of the DBN models. The pre-training examples are unlabeled data which do not come with the



	10000	5000	1000	500	200	100	50
CNN	95.3855	93.421	86.524	80.62035	71.189	61.6585	51.179
DBN	96.63	94.724	88.116	83.148	76.256	69.368	64.534
ANN	94.8685	92.4305	79.4145	71.2215	52.299	41.7325	32.7629
P-Value	0.0006295	0.00161674	0.08649838	0.24386276	0.06656283	0.00051728	0.00080766

Figure 4.3: Accuracy Comparison After Mix the Pixels

cost of labelling each image. CNN comes close to DBN performance only when larger labeled sets of training data are provided. DBN does better given a large collection of unlabeled data.

4.2 Experiment 2: One Pixel Attack on CIFAR-10 Dataset

4.2.1 Objective

The objective of this experiment is to compare CNN and DBN performance under the one-pixel attack originally porposed by Su, Vargas and Sakurai [27]. We use the CIFAR-10 dataset instead of the MNIST dataset for this classification task for both CNN and DBN in order to increase the attack success rate. All models are trained on the original CIFAR-10 images, but each test image will have one pixel deliberately set to black(0,0,0). The position of the black pixel will be moved from the first to the last pixel of the image. If the model is not able to predict the correct label, the attack is considered successful. The expected result is that the CNN will be affected more by this attack than the DBN. This will demonstrate that the CNN is too dependent on the local relationship between pixels.

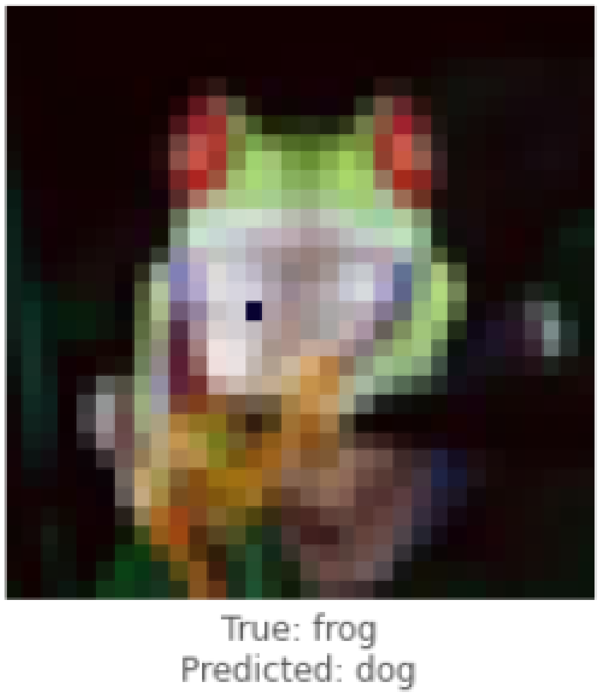


Figure 4.4: An example of an image from Experiment 2

4.2.2 Data Used and Approach

Figure 4.4 is the 103rd picture in the test dataset of CIFAR-10 under the one pixel attack. Only one pixel is changed near the centre of the picture. The CNN network thinks it is a dog instead of a frog. Each image is changed in a similar manner in an effort to have the models misclassify the image.

For the DBN model, we will use 10,000 examples to pre-train the RBMs with 3 hidden layers containing 512, 256, 128 nodes per layer, respectively. A learning rate of 0.01 will be used and each RBM hidden layer will be trained for 20 epochs. 10,000 training examples and 2,000 validation examples will be used. The classification model will be fine-tuned for 5000 epochs. Early stopping based on the mean squared error of the validation set will be used with a patience value of 40. For fine-tuning, the batch size is set to 100 and the learning rate is 0.02, dropout is 20%, and the "sigmoid" active function is used for all models. The model will be tested on 10000 samples provided by the CIFAR-10 dataset.

The CNN model has three layers. The first conv2d layer has 64 filters and a kernel size of 3x3 and uses the relu active function. The second has conv2d layer with 16 filters and a 2x2 kernel size and uses the "relu" active function, 2x2 maxpooling. This is followed by a 128 node dense fully-connected layer then, and then a softmax layer as the output. The CNN uses the same data as DBN.

After training and testing the model using the original data, for all of the test examples that the model successfully predicted, we replace one pixel with the value (0,0,0) and test if the network can still classify the image correctly.

4.2.3 Results

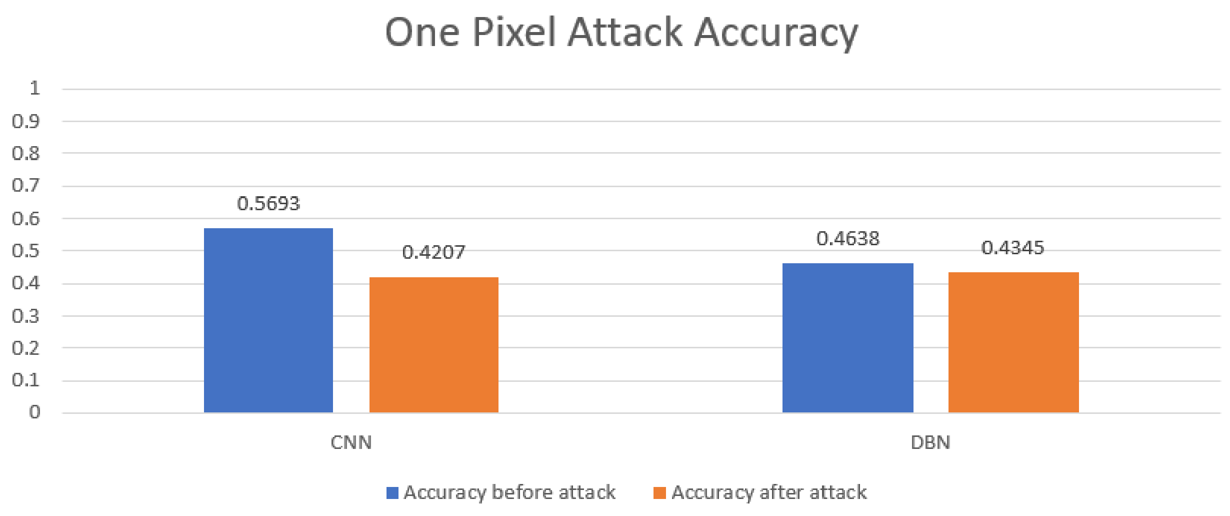


Figure 4.5: The Average Performance on 10,000 Tested Samples by the CNN and DBN Models Before and After the One-Pixel Attack

Figure 4.5 compares the CNN and DBN performance on the CIFAR-10 dataset before and after the attack. CNN models correctly classified 5693 images on average out of 10,000 test examples of normal images. It then misclassified 1486 after the one-pixel attack, with the remaining 4207 correctly classified. This shows that 26.1% of the attacked images were misclassified. The DBN models correctly classified 4638 of images of the original 10000 test samples of normal images. It then misclassified only 293 of these under one-pixel attack, or only 6.3%.

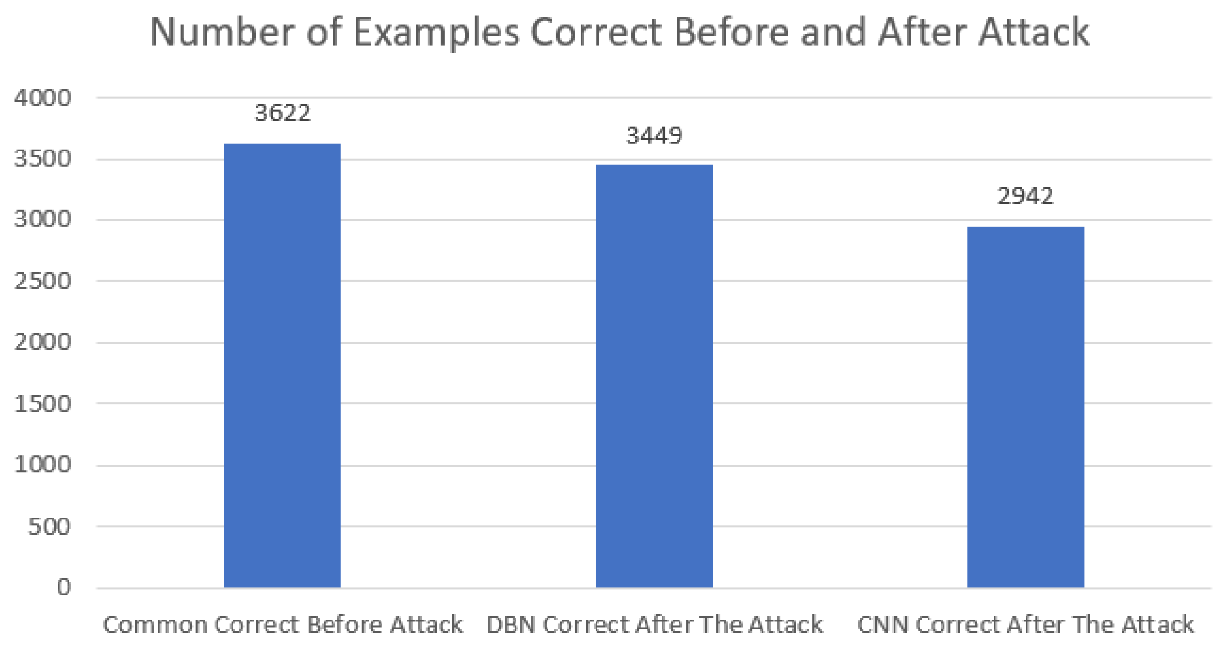


Figure 4.6: Performance on Common Correct Before and After the One-Pixel Attack

Figure 4.6 shows the common correct examples before the attack is 3622. DBN still has 3449 (=3622-173) correct after the attack, or only 4.78% error, while the CNN has only 2942 (=3622-680) correct after the attack, which is 18.77% error.

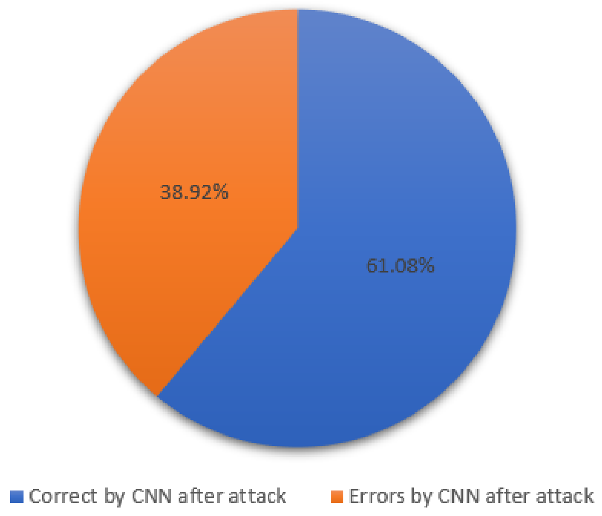


Figure 4.7: CNN's performance on examples that DBN gets incorrect.

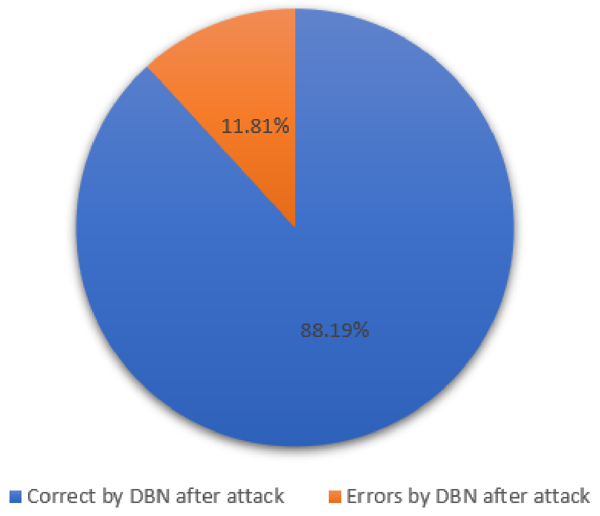


Figure 4.8: DBN's performance on examples that CNN gets incorrect.

Figure 4.7 and 4.8 shows how CNN and DBN models perform under attack on examples that the other algorithm did not correctly identify before the attack. It shows that for 1016 examples that CNN uniquely misclassified before the attack, that DBN correctly classifies 896 of these after the attack, or 88.19%. Whereas, of the 2062 examples that DBN uniquely

Algorithm	CNN	DBN
Correct examples before attack	5693	4638
Error after attack	1486	293
Correct after attack	4207	4345
Error after attack	0.2610	0.6317
Common correct before	3622	3622
Common correct before	0.6362	0.7809
Common errors after	41	41
Unique errors by each	639	132
Total errors after	680	173
Total errors after	0.1877	0.0477
Algorithm	CNN	DBN
Errors examples before attack	4307	5362
Common errors before	3291	3291
Common errors before	0.7641	0.6138
Still correct by other method after attack	896	1265
Now error by other method after attack	120	806
Check sum	4307	5362
Stilll correct by other method after attack	0.8819	0.6108
Now error by other method after attack	0.1181	0.3892

Table 4.1: Analysis of CNN and DBN before and after one pixel attack

misclassified before the attack, that CNN correctly classifies 1265 of these after the attack, or just 61.08%.

Table 4.1 provides details about how one algorithm performs on the examples that the other algorithm fails to identify before the attack. It shows that CNN failed to label 4307 examples in total and there were 3291 examples in common that both algorithms failed to properly identify. DBN is capable of correctly labeling 896 samples while under attack, there were also 120 samples that DBN identified correctly at first then fails on the picture is attacked.

DBN failed to label 5362 examples in total. CNN is able to correctly labeling 1265 of these examples while under attack, there were also 806 samples that CNN identified correctly at first then fails on the picture is attacked.

Overall, DBN only makes 11.8% mistakes (120 out of 1016) and CNN makes 38.9%

mistakes (806 out of 2071). Therefore, DBN performs better than CNN on the examples which the other algorithm failed to label in the one-pixel attack.

4.3 Experiment 3: Random Noise Attack

4.3.1 Objective

The objective of this attack is to compare CNN and DBN performance under a random noise attack. That is, random noisy pixels are added to the test set images in an effect to try to confuse the machine learning models. The purpose is to show that CNN is disadvantaged by its bias based on the relationship between proximal pixels. This assumption will be attacked by random noise enough to decrease the accuracy of the model. The expected result is that CNN will be affected by the noise more than DBN.

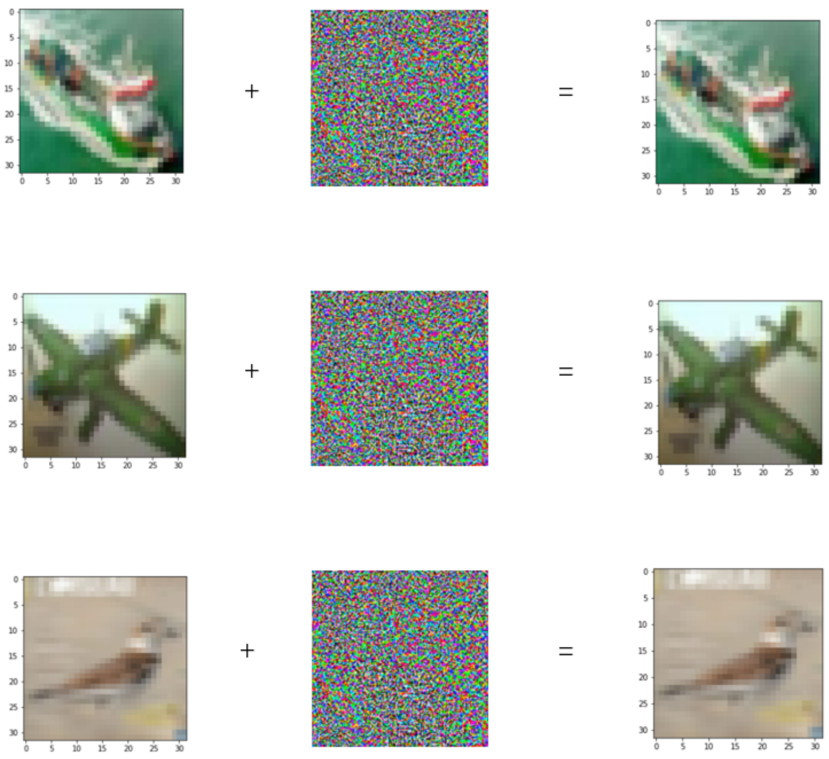


Figure 4.9: Some images used in the test set of the Random Noise Attack.

4.3.2 Data Used and Approach

Figure 4.9 shows that the 17th (ship), 28th (plane) and 71st (bird) under attack by random noise. Human will think the pictures before and after the attack are the same, but machine learning models may label them differently. In this attack, I used the CIFAR-10 dataset and set the same hyperparameter settings as the “One Pixel Attack” experiment for both DBN and CNN. The only difference is the change in the red, green and blue values of all of the pixels versus of picking one pixel and changing it to black, ie to (0,0,0). The random noise is generated by the function random.uniform() between -0.1 and 0.1 while the range of the red, green, blue value has a range of 0 to 1. So this means the pixel values change between -25 and +25.

4.3.3 Results

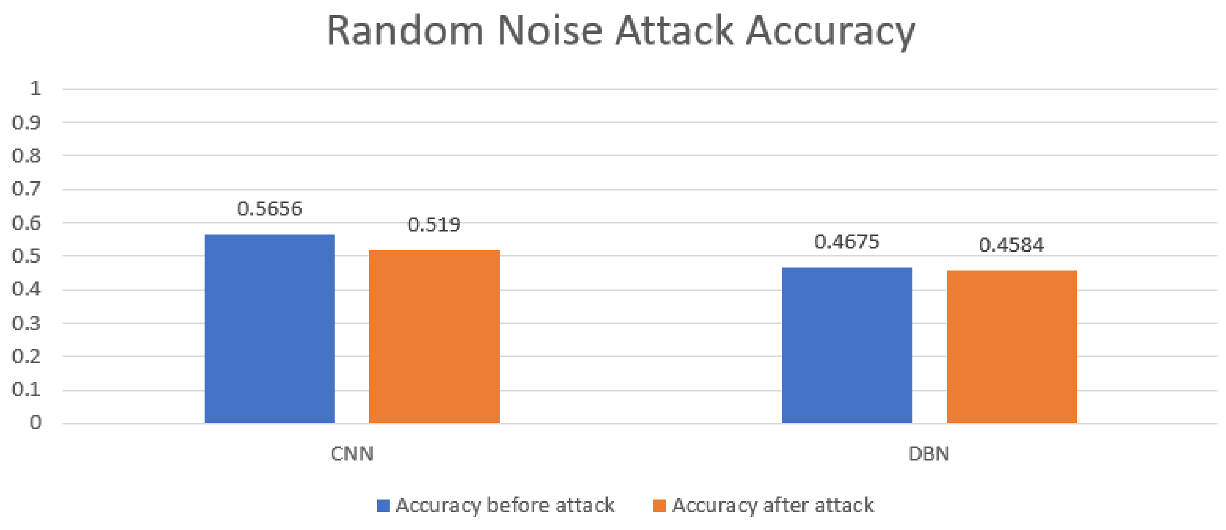


Figure 4.10: The average performance with 10000 test samples for both algorithms to show the accuracy before and after the Random Noise Attack

Figure 4.10 compares the CNN and DBN performance on the CIFAR-10 dataset. The CNN models correctly classified images with an average of 5656 out of 10000 tested samples of normal images. It then misclassified 466 after the random noise attack. The remaining 5190 were correctly classified. This means that 4.8% of the attack images were misclassified by the CNN models on average. The DBN models correctly classified 4675 images of the original 10000 test examples of normal images. It then misclassified only 91 of these under random noise attack or only 1.9%.

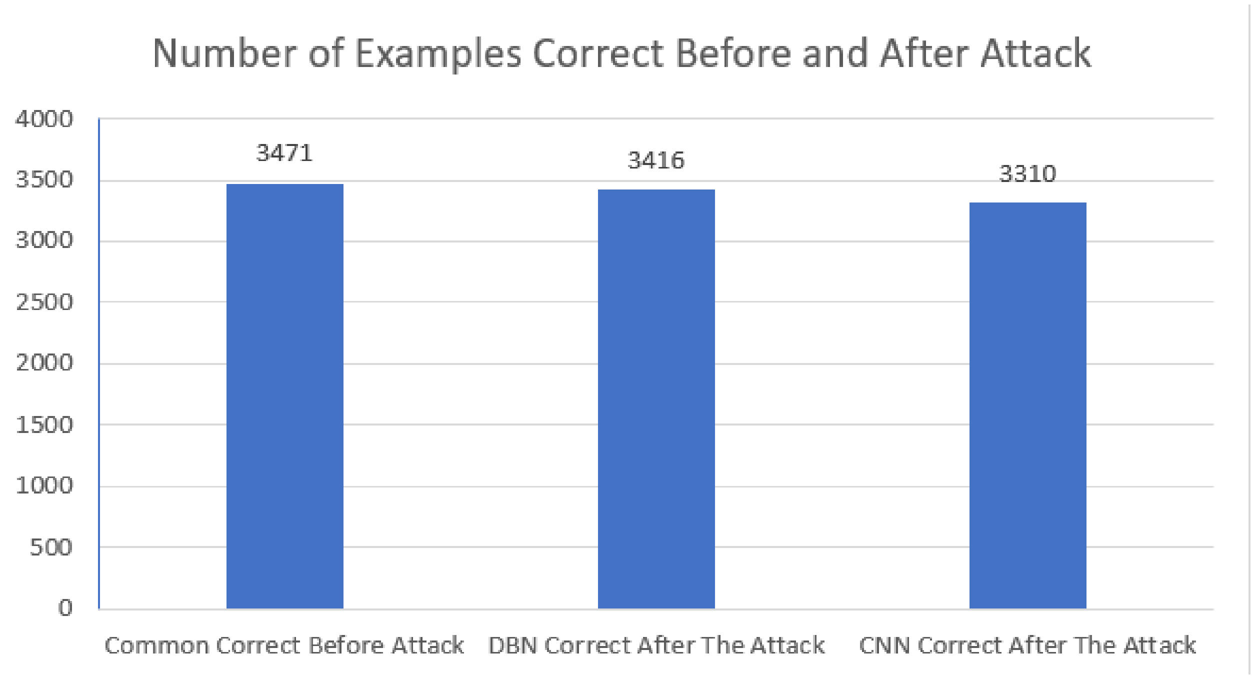


Figure 4.11: The average performance with 10000 test samples for both algorithms to show the accuracy before and after random noise attack

Figure 4.11 shows the common correct examples before attack is 3471. DBN still has 3416 (=3471-55) correct classified, or only 1.58% error, while the CNN has only 3310 (=3471-167) correct after the attack, which is 4.81% error.

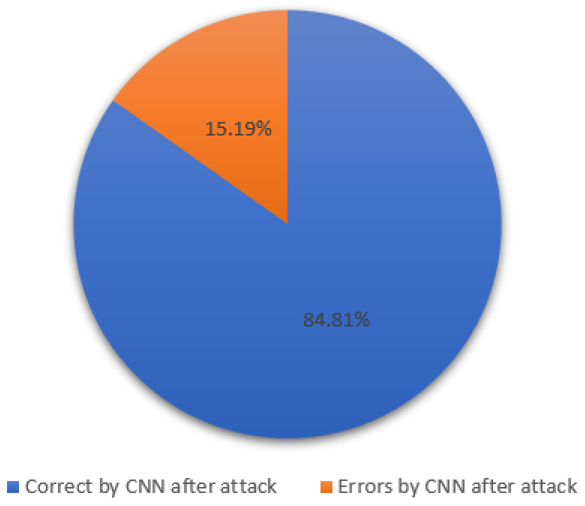


Figure 4.12: CNN's performance on samples that DBN gets incorrect

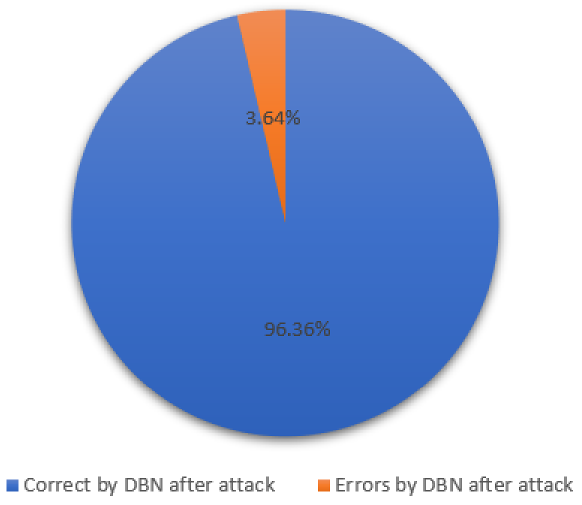


Figure 4.13: DBN's performance on samples that CNN gets incorrect

Figures 4.12 and 4.13 show how CNN and DBN models perform under attack on examples that the other algorithm does not correctly identify before the attack. It shows that for 988 examples that CNN uniquely misclassified before the attack, that DBN correctly classified 952 of these after the attack, or 96.36%. Whereas, of the 1969 examples that DBN uniquely misclassified before the attack, that CNN correctly classified 1670 of these after the attack,

Algorithm	CNN	DBN
Correct before	5656	4675
Error after attack	466	91
Correct after attack	5190	4584
Error after attack	0.0824	0.0195
Common correct before	3471	3471
Common correct before	0.6137	0.7425
Common errors after	6	6
Unique errors by each	161	49
Total errors after	167	55
Total errors after	0.04811	0.1585
Algorithm	CNN	DBN
Errors before	4344	5325
Common errors before	3356	3356
Common errors before	0.7726	0.6302
Still correct by other method after attack	952	1670
Now error by other method after attack	36	299
Check sum	4344	5325
Stilll correct by other method after attack	0.9636	0.8481
Now error by other method after attack	0.0364	0.1519

Table 4.2: Analysis Table of CNN and DBN Before and After Attack

or just 84.81%.

Table 4.2 has more details about how one algorithm performs on the examples that the other algorithm fails to identify before the attack. It is shown that CNN failed to label 4344 examples in total, there were 3356 examples that both algorithms did not properly identify. DBN is capable of correctly labeling 952 samples while under attack, there were also 36 samples that DBN identified correctly at first then fails if the picture is attacked.

DBN failed to label 5325 examples in total, there were 3356 examples that both algorithms did not properly identify. CNN is capable of correctly labeling 1670 samples while under attack, there were also 299 samples that CNN identified correctly at first then fails if the picture is attacked.

Overall, DBN only makes 3.64% mistakes (36 out of 988) and CNN makes 15.18% mistakes (299 out of 1969). DBN performs better than CNN on the samples that the other algorithm failed to label under the random noise attack.

4.4 Analysis and Discussion

Based on the experiment in Section 4.1, we can see that for a standard MNIST dataset before the attack, CNN and DBN outperform standard ANN models. Furthermore, the experiment shows that when the training examples fall below 100 that DBN does marginally better. The performance of the DBN shows that RBM pre-training equips the DBN network with learned features that rival the features created by the CNN network. Most importantly, the mixed-pixel attack experiment shows that DBN has a significant advantage over CNN in 5 out of 7 of the training sets. The CNN models accuracies suffer when the pixels of the image are mixed, but the DBN models remain at the same level as before the attack.

The 'one-pixel attack' experiment in Section 4.2 shows DBN models have higher accuracy in comparison to the CNN. For images that DBN can identify, the CNN has an 18.7% accuracy drop, which is about 4 times the percentage errors of DBN, which has a 4.77% accuracy drop. The comparison for one algorithm's performance on examples that the other algorithm misclassifies shows that CNN has 3 times the errors after an attack. This suggests that the CNN's inductive bias assumption has a disadvantage with this attack.

The 'random noise attack' in Section 4.3 describes the performance for CNN and DBN when the images suffer from small changes to each pixel. The result shows that CNN has an error percentage rate that is 4 times higher than DBN after this form of attack. It has shown that CNN loses significant accuracy after the attack and this is because the random noise interferes with the strong assumption made by the conv kernel that pixels proximal to each other have a relationships, such as they will have similar colour or brightness. Noise hurts this assumption and DBN is not affected to the same degree.

We propose that CNN's problems stem from its convolution kernel bias. If the input image attacks this bias, CNN models lose accuracy while DBN models are less affected. The DBN models also show that the self-supervised learning method takes advantage of readily available unlabeled data and does not need large quantities of human-labeled examples.

Chapter 5

Summary and Conclusion

5.1 Summary

This research investigated three comparative experiments that attacked the CNN's inductive bias based on the use of a small convolution kernel that assumes a relationship between the pixels that are proximal to each other.

This thesis has shown that the strong built-in bias of the CNN and pre-training bias of DBNs can provide an advantage for image classification problems over standard ANNs. This research has demonstrated with several empirical studies that DBNs, that learn their internal representation using an unsupervised method directly from the inputs, create a bias for classification learning that is not adversely affected in the same way as the CNN bias.

5.2 Conclusions and Contributions

This study draws the following conclusions and contributions: CNN's inductive bias advantages become disadvantages if the convolution kernel assumption is not justified. DBN has an advantage over CNN when attacked using knowledge of this CNN weakness. DBN's self-supervised learning does not rely on such strict assumptions and therefore does not suffer from the same loss of accuracy.

This research has made the following new contributions:

- It has demonstrated that the assumption of the relationship between pixels is removed, the advantage of CNN's convolution kernels and pooling no longer hold.
- The one-pixel and random noise attack is enough to make the CNN accuracy rate drop significantly, while the impact on DBN is far less.

This research has reinforced the following prior contributions:

- CNN is overly dependent on human-labeled data while DBN is not.
- A DBN can be pre-trained using unlabeled data to produce significantly better models than ANN's trained with only labelled examples.
- It is important to consider attacks as artificial intelligence becomes a part of everyday life. The advantages of one method versus another under the attack will increase in its relevance.

5.3 Future Work

The following is a list of potential future work.

Discovering Higher Accuracy with Self-Supervised Methods. In this research the DBN models had lower initial accuracy with large training sets compare to CNN. With more unlabelled examples and longer RBM training times the DBN may do as well as CNN before the attack, and continue to do better after the attack.

More Complex Real-World Images. A logical next step for this research is to use more complex real-world examples to test how each algorithm scales up.

Smarter Attack Algorithm. There are advanced attack methods introduced in Section 2.6 that have not been tested in this research effort. This could be the subject of future work.

Hidden Features Visualization. An analysis of hidden layer weights and neuron activations would be a valuable area to study. By showing the details inside the hidden layers, we can analyze why some attacks cause the CNN to fail and DBN to succeed.

Applications Beyond Image Classification. This thesis only focused on image classification. Many domains like object tracking, voice detection or analysis or text analysis could also be used in future studies.

Adversarial Attacks Designed for DBNs. This thesis showed the adversarial attacks against the kernel's bias from CNN worked efficiently. It will be meaningful to think of an attack that can specifically hurt DBNs.

Bibliography

[1] Restricted boltzmann machines. URL https://www.todaysoftmag.com/article/747/restricted-boltzmann-machines.

[2] Large scale visual recognition challenge 2012 (ilsvrc2012), . URL http://www.image-net.org/challenges/LSVRC/2012/results.html.

[3] Large scale visual recognition challenge 2014 (ilsvrc2014), . URL http://image-net.org/challenges/LSVRC/2014/results.

[4] Large scale visual recognition challenge 2015 (ilsvrc2015), . URL http://image-net.org/challenges/LSVRC/2015/results.

[5] Alphago: Google deepmind, Jan 2016. URL https://web.archive.org/web/20160130230207/http://www.deepmind.com/alpha-go.html.

[6] Artificial neural network, Nov 2020. URL https://en.wikipedia.org/wiki/Artificial_neural_network.

[7] Convolutional neural network, Dec 2020. URL https://en.wikipedia.org/wiki/Convolutional_neural_network.

[8] Colab, Nov 2020. URL https://en.wikipedia.org/wiki/Colab.

[9] Region based convolutional neural networks, Sep 2020. URL https://en.wikipedia.org/wiki/Region_Based_Convolutional_Neural_Networks.

[10] Restricted boltzmann machine, Nov 2020. URL https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine.

Bibliography

[11] Keras, Nov 2020. URL `https://en.wikipedia.org/wiki/Keras`.

[12] Albertbup. A python implementation of deep belief networks built upon numpy and tensorflow with scikit-learn compatibility, 2017. URL `https://github.com/albertbup/deep-belief-network`.

[13] Efros Alyosha, Mar 2019. URL `https://people.eecs.berkeley.edu/~efros/gelato_bet.html`.

[14] Ankesh Anand. Contrastive self-supervised learning, 2020. `https://ankeshanand.com/blog/2020/01/26/contrative-self-supervised-learning.html`.

[15] J.. Demsar, A.. Zisserman M.. Everingham, R.. Fergus L.. Fei-Fei, P.. Perona R.. Fergus, L.. Fevrier V.. Ferrari, D.. Lowe, D.. Cox N.. Pinto, A.. Torralba B.. Russell, M. J.. McGill G.. Salton, R.. Szeliski D.. Scharstein, and et al. The pascal visual object classes (voc) challenge, Jan 1970. URL `https://link.springer.com/article/10.1007/s11263-009-0275-4`.

[16] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36 (4):193–202, 1980. doi: 10.1007/bf00344251.

[17] C. Garcia and M. Delakis. Convolutional face finder: a neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004. doi: 10.1109/tpami.2004.97.

[18] Terry Sejnowski Geoffrey Hinton. Learning and relearning in boltzmann machines. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.*, 1: 282–317, 1986.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. Springer, 2016.

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, Mar 2020. URL `https://arxiv.org/abs/1911.05722`.

[21] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. doi: 10.1162/089976602760128018.

[22] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cats visual cortex. *The Journal of Physiology*, 160(1):106–154, Jan 1962. doi: 10.1113/jphysiol.1962.sp006837.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[24] Yann LeCun. Self-supervised learning. `https://drive.google.com/file/d/1r-mDL4IX_hzZLDBKp8_e8VZqD7fOzBkF/view`.

[25] Rizwan Muhammad RizwanI am an electrical engineer. Lenet-5 - a classic cnn architecture, Apr 2020. URL `https://engmrk.com/lenet-5-a-classic-cnn-architecture/`.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[27] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019. doi: 10.1109/tevc.2019.2890858.