The 34th Canadian Conference on Artificial Intelligence
DOI: 0

The Disadvantage of CNN versus DBN Image Classification Under Adversarial Conditions

Tao (Andy) Yang[†], Daniel L. Silver [†,*]
[†] Acadia University

Abstract

We compare Convolutional Neural Networks (CNN) and Deep Belief Network's (DBN) ability to withstand common image classification attacks. CNNs makes a strong inductive bias assumption about the relationship between pixels that are proximal to each other. We propose that this bias makes CNNs vulnerable to adversarial attacks. We implement two attacks using the MNIST and CIFAR-10 dataset where we modify pixels of the training and test images in different ways to challenge the CNN and DBN models. The results show that the DBN models generally perform better under attack than the CNN models. The CNNs convolutional inductive bias is at a disadvantage when the assumption of the relationship between proximal pixels no longer holds.

Keywords: convolutional neural networks, deep belief networks, adversarial attacks

1. Introduction

Over the last twenty years, researchers have made great progress in implementing computer vision classification systems using high-performance computing and new machine learning algorithms. Convolutional Neural Networks (CNN) remain the dominant method for computer vision in 2021 [1]. CNNs have architectures that are highly engineered to take advantage of the relationship between pixels that make up an image. In comparison, older approaches such as Deep Belief Networks (DBN), which are pre-trained using unlabelled examples, have fallen out of favour [2]. This is unfortunate, as simple transformations on images used to train a CNN, can be used to fool that same CNN, even when the transformations are undetectable to the human eye [3]. This is a significant vulnerability that can be used to systematically trick systems that rely on CNN machine learning components.

In this paper we focus on images that have been created to attack the performance of CNN and DBN classification models. We use grey scale and colour images from the MNIST and CIFAR-10 datasets. We show that the CNN inductive bias, which assumes that important features can be extracted from adjacent pixels, fails under certain adversarial attacks. In contrast, DBNs, which are pre-trained using unsupervised examples, learn features with no assumption about the proximity of pixels, and do better under the same attacks.

2. Background

2.1. Convolutional Neural Networks

A CNN is a feed-forward neural network with excellent performance on image classification, object detection, tracking and counting. In comparison to standard feed-forward neural networks, CNNs have relatively few parameters, making them attractive for training with supervise labelled training examples [4]. A CNN's architecture is engineered to take advantage of the relationship between adjacent or proximal pixels in an image. The function of the convolutional layer is to extract features from small $n \times n$ portions of the input space using multiple convolution kernels. Each input to a convolution kernel is connected to a neuron in the next layer called a feature map. Each connection has an associated weight. After the weights of a convolution kernel have been trained, the kernel can sweep over the

[*]danny.silver@acadiau.ca

pixels of the image and compute the sum of the product of each kernel weight 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. In this way, a convolution layer detects simple features such as edges, corners, variations in color, and brightness that can be used to create higher order features[5]. This approach works based on the assumption (a form of inductive bias) that the $n \times n$ kernel receives pixels that are proximal to each other.

2.2. **Deep Belief Networks**

A DBN uses stacked autoencoders in it's lower layers to learn representations that are useful for later developing representations in the higher layers of the network using supervised back-propagation [6]. Specifically, a DBN can be composed of a stack of Restricted Boltzmann Machines (RBM) autoencoders. An RBM is an unsupervised generative neural network model that is trained to reconstruct whatever is presented at its inputs (visual layer) after passing these inputs to a hidden layer and back [2]. RBM networks form a bipartite graph; where a neuron in one layer is connected to all the neurons in the next layer, however connections between neurons in the same layer are not allowed. A RBM uses a gradient descent algorithm called *contrastive divergence* to minimize the error between the reconstructed values at the visual layer and the training examples.

In a DBN, layers of RBMs are stack one on top of the next and trained to improved the reconstruction accuracy at the input (visible) nodes of the network. After the RBMs are trained, addition layers can be added to the top of the RBM stack and the entire network can be trained for classification using the standard gradient descent, back-propagation algorithm. DBN networks can have their RBM stack pretrained with a large collection of unlabelled data thus learning a useful internal representation (a form of inductive bias). This can be used to reduce the number of labelled training examples needed to fine-tune the DBN to build accurate classification models.

2.3. **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, as shown in Figure 1 [7]. In this example, a small amount of random noise is added to each pixel. Researchers have worked to discover the lowest number of changes needed to fool a network. In 2018, it was shown that by only changing one pixel, a trained CNN network will incorrectly think a horse is a frog [7]. Most images in the CIFAR-10 dataset can be modified similarly to fool CNN models. Two examples are shown on the right of the Figure 1.

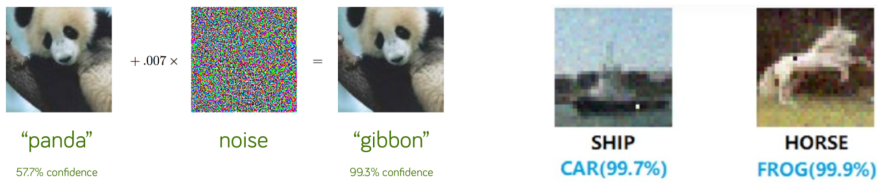


Figure 1. Left: Adversarial attack by adding noise to an image. Right: One-pixel attack images that fooled deep learning. [7].

3. Theory and Approach

CNNs rely heavily on the engineered inductive bias provided by the $n \times n$ kernel assumption discussed in Section 2.1. If the inputs to a CNN are not in accord with this assumption then the inductive bias may prove to be a disadvantage. More specifically, we propose that this well known inductive bias for discovering features leaves CNN models open to attack. In addition, CNNs rely on annotated target labels for supervised learning. This reliance prevents CNNs from taking advantage of the potentially rich features that could be extracted from the input values using unsupervised learning. For this reason, purely supervised learning algorithms often require large numbers of samples to train internal representations that tend to be task-specific rather than useful for learning over a lifetime.

In contrast, semi-supervised approaches, such as DBNs, develop internal representations from unsupervised examples from the same input space. The internal representation developed by the RBMs in a DBN make no assumption about the relative location of pixels in an image; the representation self-organizes to create features for reconstructing the image at the input nodes. These same features often prove useful for classification and regression problems that use the same input space. Therefore, at the price of pre-training using unlabelled examples, DBN models provide a promising method for lifelong learning [8]. Furthermore, we propose that models developed using DBN will be superior to CNN models when under attack by those who would take advantage of the knowledge of the algorithms.

4. Empirical Studies

This section aims to demonstrate empirically that a semi-supervised approach using DBNs is superior to the strictly supervised CNN approach when under attack. The first experiment uses the MNIST data set to show the weakness of the CNN algorithm, as compared to the DBN algorithm, when the kernel assumption no longer holds for the input data. The second experiment tests the performance of CNN and DBN models against noisy one-pixel attack images and shows the significant loss in model accuracy that CNN suffers versus DBN. A 5-fold cross-validation approach is taken for both experiments.

4.1. Experiment 1: Mixed Pixel Attack on MNIST Dataset

Objective. This experiment aims to show that CNN's kernel based inductive bias can be a weakness. The pixels of each MNIST picture are scrambled in a consistent manner, such that a human cannot classify an image. Figure 2 shows examples of the original and mixed-pixel images. CNN and DBN as well as standard ANN models (with equivalent capacity) are trained and tested on original images and then trained and tested on mixed-pixel images.

Data and Methods. The DBN models have three RBM hidden layers of 512, 256, 512 nodes and a final output layer of ten softmax nodes. The RBM layers are pre-trained on 10,000 examples for 19 epochs using the contrastive divergence algorithm with a learning rate of 0.01. Following this, the DBN model is fine-tuned in a supervised manner using 10000, 5000, 1000, 500, 200, 100 and 50 balanced sets of labeled examples with 20% of the examples used as a validation set to prevent over-fitting. The models were trained for 1500 epochs using standard back-propagation gradient descent with early-stopping and a patience of 100. The batch size is set to 100 and the learning rate is 0.1. Dropout is 17% and the ReLU active function is used at all nodes except the softmax output nodes. The best model, based on early stopping, is tested on 10,000 test examples. A standard ANN, based on the DBN architecture, is also developed from random initial weights and tested.

The CNN uses the VGGNET architecture [9]. 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 2x2 maxpooling and

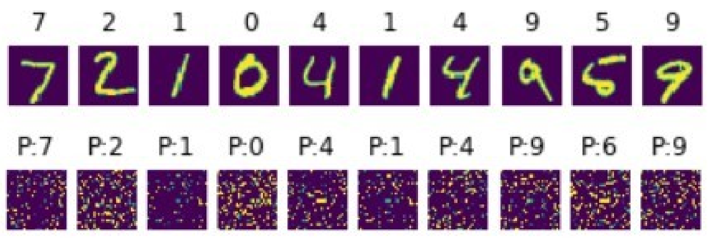
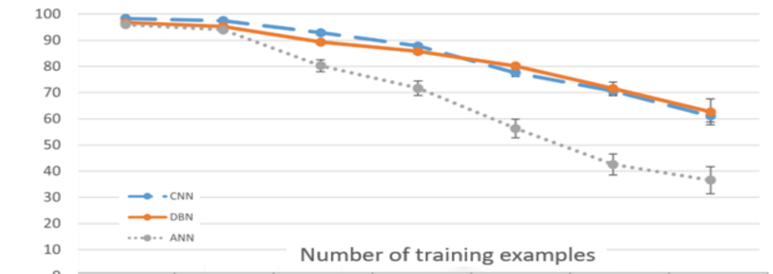


Figure 2. Examples of images used in the first experiment: top row shows original images, bottom row shows mixed-pixel images.

25% dropout. This followed by a 128 node fully-connected hidden layer with a 50% dropout and then softmax output layer. The network is trained using the same sets of balanced examples as the DBN, for 1500 epochs, using the Adadelta optimizer with early-stopping and a patience of 100. The optimizer has an initial learning rate of 1.0 and a decay rate of 0.95; everything else is set to their default values for Tensorflow.

Results and Discussion. Figure 3 shows the mean accuracy and 95% confidence interval from the 5 cross-validation runs using the original image data. 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 to learn an accurate model. The p-values from the T-Tests comparing the CNN and DBN models are shown in Figure 3.

Figure 4 shows the performance of CNN, DBN and ANN models trained and tested on the MNIST mixed-pixel images. Clearly, mixing up the pixels of the images adversely affects the performance of the CNN models, but not so for the DBN models that maintain the same level of accuracy as before the attack. The T-tests p-values comparing the CNN and DBN model accuracies shows that the DBN models are more accurate for all training set sizes and statistically significant at the 90% level for all but the 500 training set size. The standard ANN results continue to show that pre-training is an important aspect of the DBN models.

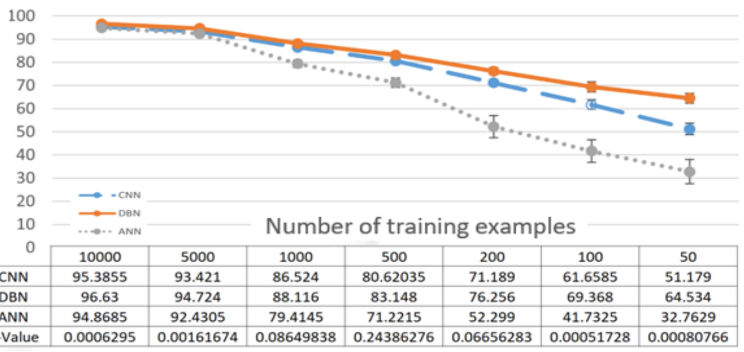


	10000	5000	1000	500	200	100	50
CNN	98.3385	97.5045	92.915	87.797	77.558	70.633	61.092
DBN	96.72	95.304	89.306	85.736	80.196	71.59	62.636
ANN	96.004	94.0075	80.3155	71.6985	56.293	42.5425	36.5835
P-Value	6.4184E-05	2.8869E-05	0.00166668	0.14961929	0.17696819	0.80794501	0.43296946

Figure 3. CNN and DBN accuracy on original test images by number of training examples.

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

Objective. The objective of this experiment is to compare CNN and DBN performance under the one-pixel attack originally proposed by [7]. We use the CIFAR-10 dataset for this classification task. Figure 1 shows examples of CIFAR-10 images under the one pixel



	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. CNN and DBN accuracy on mixed-pixel test images by number of training examples.

attack. All models are trained on the original CIFAR-10 images, but each test image will have one pixel deliberately set to black, RGB(0,0,0).

Data and Methods. The DBN models have three RBM layers containing 512, 256, 128 nodes per layer. They are each pretrained using 10,000 unlabelled examples from the CIFAR-10 dataset for 20 epochs, using a learning rate of 0.01. Following this, the DBN is fine-tuned for up to 5000 epochs using 10,000 labelled training examples. The sigmoid activation function is used for all hidden nodes, and the output nodes use the softmax function. The batch size is set to 100, the learning rate to 0.02, and dropout to 20%. Early stopping using 2,000 validation examples is employed to prevent over-fitting with a patience of 40. The model is then tested on 10,000 independent examples from the CIFAR-10 dataset.

The CNN models have 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 fully-connected hidden layer and then a softmax output layer. The CNN uses the same training, validation and test data as described for the DBN.

The models are trained and tested using the original image data. Then, for all test images the model successfully predicts, we attempt to fool the DBN and CNN models by replacing one pixel of the image with the value (0,0,0). The position of the attack pixel is moved from the first to the last pixel of each image in an effort to fool the methods. If a model is unable to classify an image correctly, we consider the attack successful and record the error. The models will be evaluated on how much their accuracy changes based on the attack.

Results and Discussion. The left side of Figure 5 compares the CNN and DBN performance on the 10,000 example CIFAR-10 test dataset before and after the attack. The CNN models perform better than the DBN models prior to the attack, correctly classifying 5693 images on average. But following the one-pixel attack the CNN models misclassify 1486 of these images, with the remaining 4207 correctly classified. This is a 26.1% drop in performance due to the attacked. In contrast, the DBN models correctly classified 4638 images prior to the attack and 4345 of these images after the attack, This is a difference of only 293 images, or a 6.3% drop in performance. This means that following the attack, the DBN models were slightly more accurate than the CNN models.

The right side of Figure 5 shows that the DBN and CNN models correctly classify 3622 images in common before the attack. It also shows that after the attack, the DBN models still correctly classify 3449 (173 errors) of these images after the attack, for a 4.78% drop in accuracy. In comparison, the CNN models correctly classify only 2942 (680 errors) of these images after the attack, which is a 18.77% drop in accuracy. Clearly, the DBN models respond better than the CNN models to the one-pixel attack.

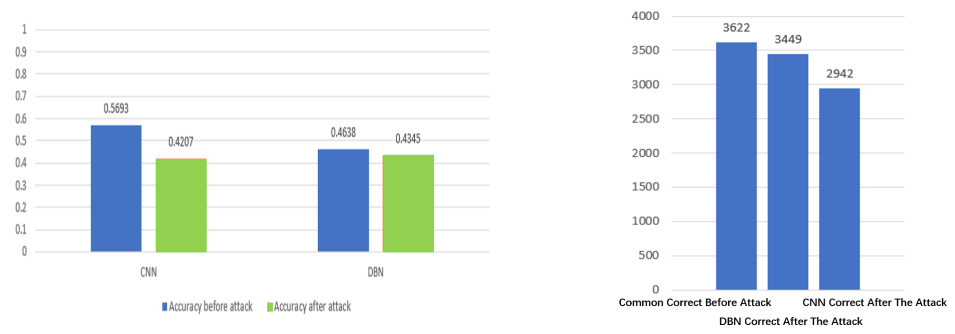


Figure 5. Left: Performance on test examples before and after the one-pixel attack. Right: Performance on test images before and after the one-pixel attack.

5. Conclusion

The CNN algorithm, by using convolutional kernels, makes a strong inductive bias assumption about how pixels that are proximal to each other can develop internal representation crucial to learning. However, our experiments show that the same bias can be a significant weakness that makes CNN models vulnerable to adversarial attacks. In comparison, the DBN algorithm assumes that the best internal representation can be developed by pre-training the network using large sets of unlabelled examples from the same input space without other *a priori* assumptions. CNNs inductive bias becomes a disadvantage when the assumption of the relationship between proximal pixels no longer holds.

The DBN models also demonstrated that semi-supervised learning methods take advantage of readily available unlabeled data and do not need large quantities of labeled examples as compared to strictly supervised approaches. This suggests that in the context of lifelong machine learning (or continual learning) systems that the DBN approach of learning the underlying internal representation is superior to the highly engineered CNN approach.

There are several directions to pursue in future work. First, more advanced methods of unsupervised training using variational autoencoders may increase DBN model performance. Second, there are more advanced attack methods that could be explored in the context of CNN and DBM networks [7]. Finally, a logical next step for this research is to use more complex real-world image examples to test how our findings scale up.

References

[1] Y. Lecun and Y. Bengio. "Conv. Networks for Images, Speech, Time Series". In: *The Handbook of Brain Theory and Neural Networks*. Ed. by M. A. Arbib. The MIT Press, 1995, pp. 255–258.

[2] G. E. Hinton. "A Practical Guide to Training Restricted Boltzmann Machines". In: *Neural Networks: Tricks of the Trade, 2nd Edition, LNCS* 7700 (2012), pp. 599–619.

[3] A. Ilyas et al. *Adversarial Examples Are Not Bugs They Are Features*. 2019. arXiv: 1905.02175.

[4] S. Saha. *A Comprehensive Guide to Convolutional Neural Networks-the ELI5 way*. Dec. 2018. URL: https://towardsdatascience.com.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. Vol. 1. Springer, 2016.

[6] G. E. Hinton, S. Osindero, and Y. W. Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18 (2006), pp. 1527–1554.

[7] J. Su, D. V. Vargas, and K. Sakurai. "One Pixel Attack for Fooling Deep Neural Networks". In: *IEEE Trans. on Evol. Comp.* 23.5 (2019), pp. 828–841. DOI: 10.1109/tevc.2019.2890858.

[8] A. Anand. *Contrastive Self-Supervised Learning*. https://ankeshanand.com/blog/2020/01/26/contrative-self-supervised-learning.html. 2020.

[9] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].