Machine Learning Algorithms for Low Data Environments

Motivation

Machine learning has an ever-growing impact on all facets of the modern world. It can solve a variety of complex problems such as image classification, speech/text recognition, and predictive financial analysis. With the influx of large datasets and the adoption of computationally powerful graphical processing units (GPUs), machine learning algorithms can accomplish such tasks with an accuracy comparable to, and many times higher than, human performance. Naturally, it is of vital importance to apply machine learning to all fields. However, machine learning is also limited; a reasonable dataset should contain a few thousand datapoints while an excellent dataset can have hundreds of thousands of points. In many fields such as medical or pharmaceutical industries, large quantities of data are scarce and are often expensive and difficult to obtain, especially if regulations are in place. Consequently, low-data learning is a growing field of great interest. This paper looks at the performance of many popular and commonly used algorithms on smaller-scale datasets. A focus on learning on datasets containing only 1 datapoint per class, known as "one-shot learning" is presented and three in-depth studies are conducted on an optimized one-shot learning algorithm called the Siamese Neural Network. Other specialized techniques such as Triplet Loss and CapsuleNet are briefly discussed. By understanding which algorithms perform well with variable-quantity datasets, humans may be able to apply machine learning to the fields that need it the most.

Datasets

The dataset primarily used in experiments is the Fashion MNIST dataset. This dataset contains 28x28 grayscale images of ten fashion articles (t-shirts, trousers, dresses, etc.). The training set contains 60,000 images while the testing set contains 10,000. A similar dataset called MNIST contains 28x28 grayscale images of ten handwritten digits (from 0 to 9) with the same training/testing distribution. State-of-the-art techniques perform image classification on MNIST with 99+% accuracy while achieving 90%-95% accuracy on Fashion MNIST.

These datasets are used because of their academic value and abundant documentation. In many real-world datasets, a preprocessing step is often needed to remove any corrupted or tainted data. This step can often be complicated and time-consuming. These datasets are clean and thus can safely avoid this step. Furthermore, these datasets are both image datasets, providing an environment that is more intuitive and understandable to test on rather than, for example, a text-based dataset. Finally, Fashion MNIST is used specifically to provide a better learning measure. While many techniques can classify with near-perfect accuracy on MNIST, a bigger gap is often present in the more difficult Fashion MNIST dataset.

Figure 1: MNIST Dataset

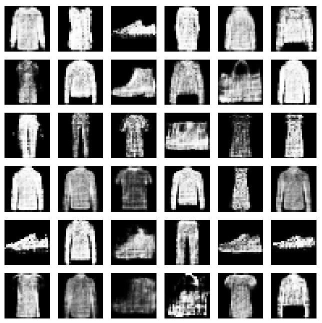


Figure 2: Fashion MNIST Dataset

<u>Experiment 1: Common Machine Learning Algorithms on a Variable Train Set</u>

This paper assumes relative familiarity with machine learning algorithms and concepts.

The first experiment focuses on the performance of seven popular and commonly used algorithms on the Fashion MNIST dataset. The following algorithms and their specifications are described:

- Convolutional Neural Network (CNN): 2 convolutional layers, 2 fully-connected (FC) layers, batch size of 128, 100 epochs
- Deep Neural Network (DNN): 3 layers, batch size of 128, 100 epochs
- Random Forest (RF): 10 trees
- k Nearest-Neighbor (kNN): 1 to 5 neighbors
- Support Vector Machine (SVM): default setting, uses L2 penalty
- Stochastic Gradient Descent (SGD): SVM with SGD training
- Logistic Regression (LR): default setting, uses L2 penalty

In the first preliminary run, the training set was varied from 10 images to the full 60000 images. The purpose of this run is to show the general trajectory of each algorithm in a high-data environment. Thus, algorithms were not optimized and only ran for 1 trial.

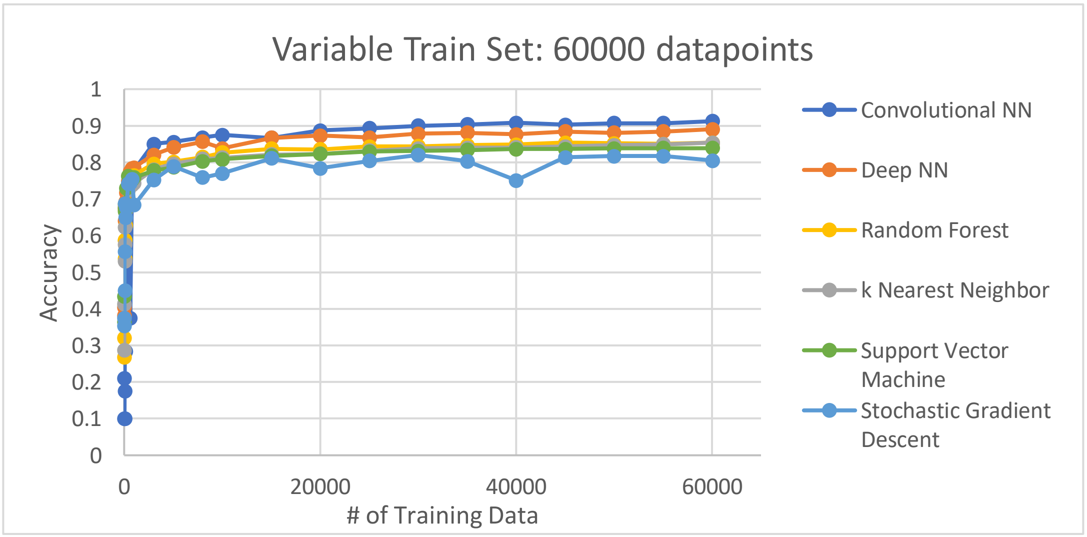


Figure 3: Performance of six algorithms on a 60000-image variable train set

As expected, CNN and DNN performed the best. Seeing the bumpiness and poor performance of SGD, it was quickly removed and replaced with LR, which had a better performance.

On the second run, the goal was to optimize the algorithms and plot them on a smaller scale to simulate a more realistic dataset. Instead of using total training examples, total samples per class were considered instead. This way, the 10 classes were each balanced with an equal number of images. In this trial, the datapoints ranged from 1 to 100 images per class.

Next, the original training set was split into a smaller training set and a validation set. In general, the validation set was 20% of the smaller training set except in the cases of very low datapoints. The validation set was used to optimize 4 of the 6 algorithms. The CNN/DNN saved the model that had the lowest validation loss in 100 epochs, the kNN used the best number of neighbors from 1 to 5, and RF chose the optimal number of trees from 10, 30, 50, 70, and 100 trees. LR and SVM remained unchanged. Finally, to remove irregularities or anomaly cases, an average was taken to cover the entire dataset. For example, if there were 50 training examples and 10 validation examples, then there were 60 points used out of 60000 total examples. In this case, 1000 iterations were ran and averaged. The results of the second run are shown:

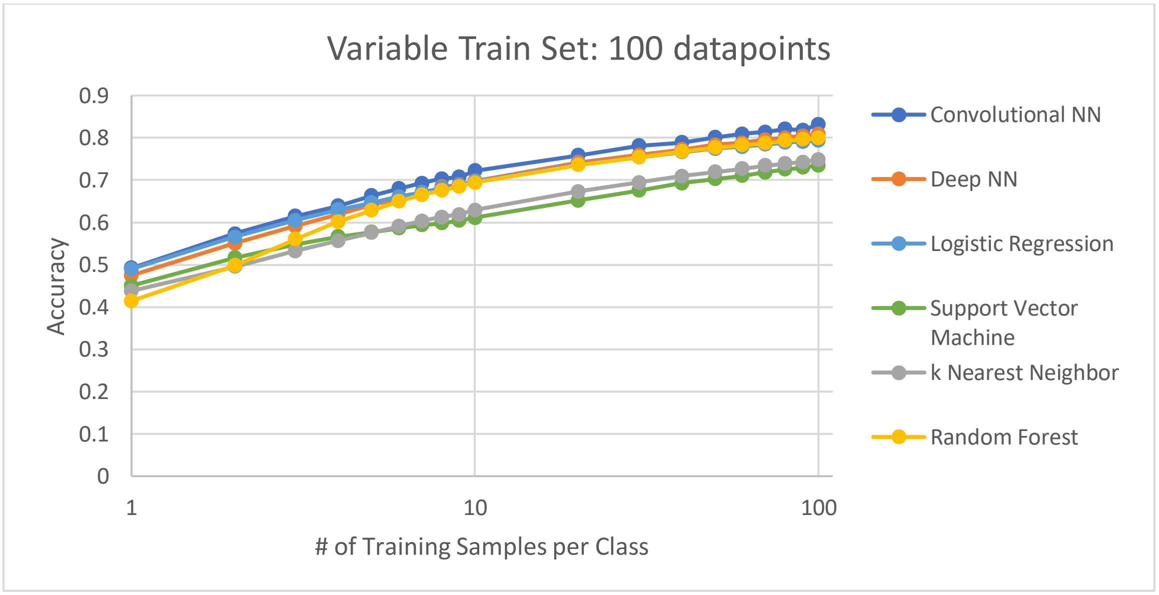


Figure 4: Performance of six algorithms on a 100-shot variable train set

The general trend remains the same; CNN and DNN still excel in higher-data cases, with other algorithms following behind. However, it is more interesting to see the cases of very low data: from 1 to 10 points.

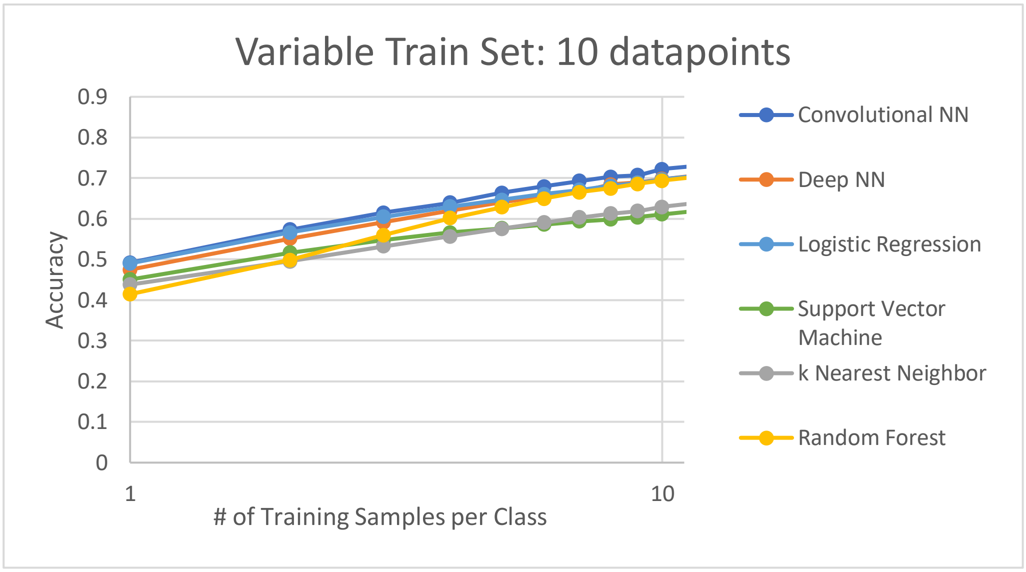


Figure 5: Performance of six algorithms on a 10-shot variable train set

Random Forest starts from the lowest accuracy at one sample per class but quickly advances. Logistic Regression starts with a relatively high accuracy but falls behind DNN and RF after a few points. These trends motivated the second experiment: one-shot learning.

Experiment 2: One-Shot Learning

Humans can do what is known as "one-shot learning". Given a single image or example, humans have the ability to reidentify another instance of it with high precision. This is how humans naturally learn from an early age, yet such a task is not as trivial for machines. As a major stepping stone for achieving general intelligence, one-shot learning is currently a hot topic of attention. In context of this paper, one-shot learning is the instance of having only one datapoint per class, which based on the aforementioned results, proves to be a difficult task. Many algorithms have been developed to specialize in one-shot learning.

Siamese Neural Network (SNN) Background

One popular algorithm is known as the Siamese neural network (SNN). The method of applying a neural network on one-shot learning comes from the necessity of extracting useful features from data. Using traditional algorithms such as a kNN with a L2 distance would result in poor generalizations. For example, if there were two identical images and one was shifted over by a few pixels, a kNN could possibly recognize them as completely different images. The idea of truly "learning" two similar or different objects is introduced in the idea of Siamese neural networks.

A SNN inputs two images through the same convolutional neural network. The CNN would output two corresponding vectors called encodings. The difference of these encodings would be found and passed into a fully-connected layer to obtain a probability of whether the images are the same or different. The similarity function of a SNN is the squared difference of the encodings and is small if the two images are the same and large if the two images are different.

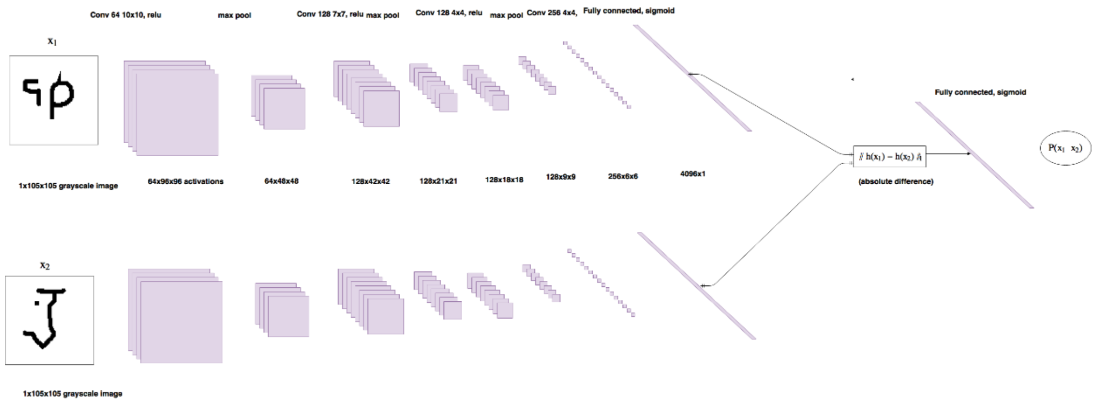


Figure 6: Siamese Neural Network Architecture

The way SNNs were used in the following studies essentially followed 4 steps:

1. Preprocessing: A certain number of images were selected per class. These images would be split into a balanced or unbalanced dataset (explained below).
2. Training: The dataset would be used to train the SNN and the best model would be selected.
3. Encoding: A separate testing set of images would be selected and the encodings of these points would be obtained by running them through the SNN.
4. Testing: The original regular images and encoded images would be run through an optimized kNN and the results would be compared.

SNN Study 1: Variable Train Set

This paper's main use of the SNN was to utilize the encodings. If the encoded images performed better than the regular images, then the algorithm learned new, useful information from the Siamese net.

In the first study, the goal was to apply a SNN on a variable train set in the same way as the first experiment. The number of training samples per class ranged from 2 to 50. These images were preprocessed as similar and different pairs. For example, given 3 images per class, the total number of images is 30. Thus, the total number of pairs is $\binom{30}{2} = 435$ pairs, $\binom{3}{2} * 10 = 30$ of which are similar pairs and 405 of which are different pairs. Because of this reason, the one-shot learning case was not actually considered. A one-shot learning case would result in 45 total pairs, none of which are similar. This would have to include data augmentation to generate similar pairs. Nonetheless, the low-shot cases are well-inferred.

For the 3-shot case, if the network simply guessed that all the pairs were different, it would achieve over 93% accuracy. With larger numbers of images per class, this accuracy would be even higher. Thus, to ensure that the model was truly learning similar pairs, the data was first balanced to have the same number of similar and different pairs, establishing a 50% baseline. The table below shows the results.

Mean Scores:		
Train	Val	Hold
98.67%	86.62%	86.44%

Using a balanced training set, the network trained with 85+% accuracy, meaning that the algorithm was tangibly learning.

Next, the data was again unbalanced. This was to ensure that the model was not limited by a lack of different pairs. To counteract the model guessing only "different", a weight of 0.915 was set for similar pairs and was found to produce the best accuracy for guessing similar pairs.

The Siamese neural network architecture was a very simple CNN consisting of only 2 convolutional layers followed by 2 fully-connected layers with a softmax activation function. The graph below shows the performance of the unbalanced dataset.

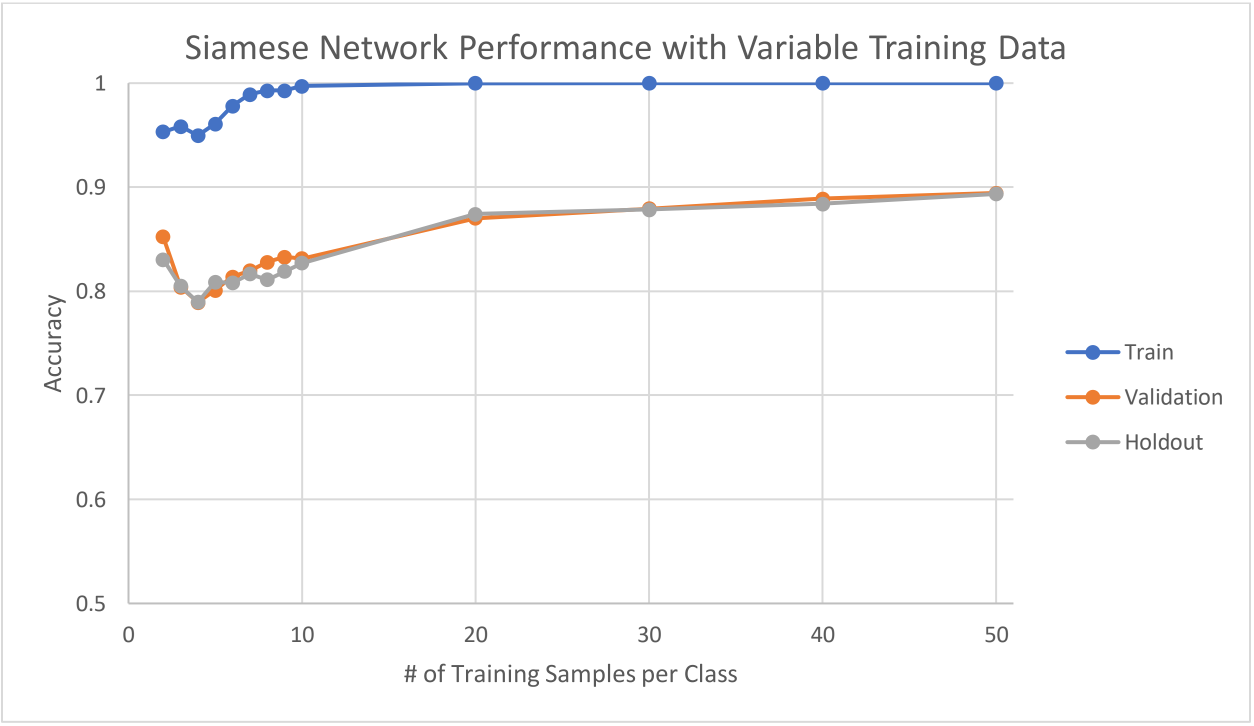


Figure 7: Performance of a Siamese Network on a 50-shot Variable Train Set

The Siamese network performed below the baseline for guessing "different" every time. Yet, at the end, this model was used because it was deemed more important to provide the SNN with more information than to establish a baseline. The bumpy plots with the low-data points are likely the result of not having enough iterations. While previous experiments had hundreds or even thousands of iterations, due to time constraints, only 15 iterations were averaged. The testing set and its corresponding encodings were passed through an optimized kNN (1-5 neighbors). Comparing the encoded and regular accuracies of the original images:

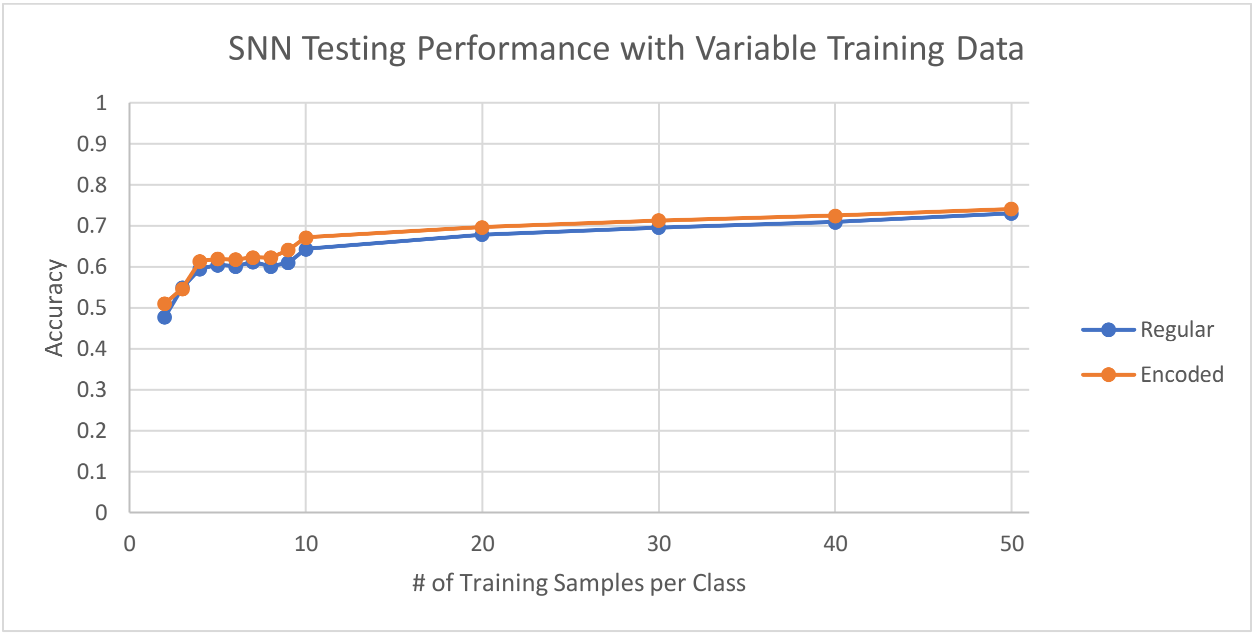


Figure 8: Performance of regular vs encoded testing on a 50-shot Variable Train Set

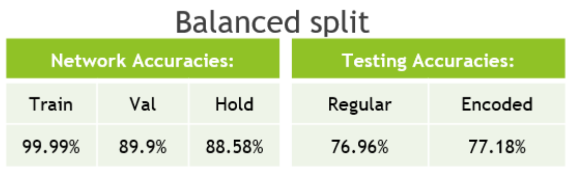
The encoded images performed slightly better (1-2%) than the regular images. This result was underwhelming; a very small increase in performance motivated the second study.

<u>SNN Study 2: Using Entire Dataset</u>

Although the encoded images performed better than the regular images across all trials, it was not a very large increase of accuracy. The goal was to show that SNN had a clearer and bigger impact on accuracy. The second study used the entire Fashion MNIST dataset, using half of the 60000-image training dataset for training the Siamese network and the other half for testing on the kNN. A validation and holdout set were constructed from the testing dataset and were used for optimizing the Siamese network and kNN (from 1 to 5 neighbors). The Siamese neural network architecture was slightly different in this study. The CNN consisted of 4 convolutional layers followed by 3 fully-connected layers with a softmax activation function.

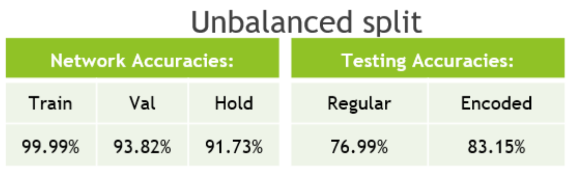
The data used for training the Siamese network were again preprocessed in balanced and unbalanced methods. The first method was to select 150 images per class out of the total 30000 images. These 1500 total images would generate $\binom{150}{2} * 10 = 111750$ similar pairs, much more than was previously used. The same number of different pairs were used to balance the number of similar and different pairs in the training set, establishing a baseline of 50%. Next, the unbalanced model was considered. This would generate $\binom{1500}{2} = 1123250$ pairs, almost 10 times more than was previously used in the 50-shot case.

Both the balanced model and the unbalanced model were used to calculate the testing accuracy. Below the results are shown:



Balanced split

Network Accuracies:			Testing Accuracies:	
Train	Val	Hold	Regular	Encoded
99.99%	89.9%	88.58%	76.96%	77.18%



Unbalanced split

Network Accuracies:			Testing Accuracies:	
Train	Val	Hold	Regular	Encoded
99.99%	93.82%	91.73%	76.99%	83.15%

The balanced dataset performs well above baseline and the unbalanced dataset fails to meet baseline again. However, while looking at the testing accuracies, the balanced dataset only increases the regular accuracy by a 0.2% while the unbalanced split improves accuracy by 6-7%. The conclusion is that the unbalanced model does much better for improving the encoded accuracy while the balanced model does better for achieving a high Siamese network accuracy.

SNN Study 3: Generalization of Features

The third study of Siamese neural networks was inspired by "Siamese Neural Networks for One-shot Image Recognition" by Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov of Univeristy of Toronto. In their experiment, they trained a convolutional SNN on the Omniglot dataset, a dataset containing 1623 different handwritten characters from 50 different alphabets, with 20 examples of each character. In many ways, it is the transpose of MNIST, having more classes but less examples. After testing the MNIST dataset with the SNN model, they achieved an impressive 70.3% accuracy.

This paper features a similar study: training on the MNIST dataset and testing on the Fashion MNIST dataset. In theory, this task may be more difficult than the one performed by the University of Toronto. While that experiment trained and tested on handwritten characters, this one trained on handwritten characters and tested on images of clothing. If the encoded images performed with a higher accuracy than the regular images, then the MNIST-trained SNN provided useful information. If no information was learned, then the encoded images and regular images were expected to have the same accuracy.

The results were clear – thus no visual aid is provided. The encoded accuracies did not perform better than the regular accuracies. In many ways, this result was expected. Transfer learning with a more comparable dataset could perhaps provide a more tangible increase.

Future Work

There is still much work to be done. As the first step in a larger application, the purpose of this paper is to introduce the problem of low-data learning and a single algorithm that could be used to address it.

A useful next step could be to include another SNN Study: training with a large dataset and testing on a variable, small dataset. This is the intersection point between the first and second study.

Next, it would be useful to apply other specialized low-data algorithms such as triplet loss and CapsuleNet. The basic intuition behind triplet loss is similar to the Siamese network. Triplet loss chooses 3 images: one anchor image, one “positive” image, and one “negative” image. A positive image is an image that is the same as the anchor image while the negative image is different from the anchor image. A small amount of images would thus generate much more data (n choose 3 examples!). CapsuleNet involves the idea of outputting a vector rather than a probability.

Finally, it would be useful to apply these algorithms and studies on another dataset such as the Omniglot Dataset. As this dataset has many classes and few examples, it would work much better in low-data environments.

Acknowledgements

This project was made possible by the Department of Energy and Argonne National Laboratories who created, organized, and funded the SULI program. Additional thanks to Dr. Fangfang Xia and Neil Getty for their continuous advice, expertise, and mentorship throughout the project.

References

Bouma, S. (2017, March 29). One Shot Learning and Siamese Networks in Keras. Retrieved August 1, 2018, from https://sorenbouma.github.io/blog/oneshot

K, F. (2018, Feburary 02). [Deep Learning Lab] Episode-1: Fashion-MNIST – Deep Learning Turkey – Medium. Retrieved July 26,2018, from https://medium.com/deep-learning-turkey/deep-learning-lab-episode-1-fashion-mnnist-c7a60029836

Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop* (Vol. 2).

Wikipedia contributors. (2018, July 23). MNIST database. In *Wikipedia, The Free Encyclopedia*. Retrieved 17:08, July 26, 2018, from https://en.wikipedia.org/w/index.php?title=MNIST_database&oldid=851591265