A Comprehensive Evaluation of Convolutional Neural Networks Comparing Speed and Accuracy Performance on the Handwriting Digit Domain

Rahul Raja
Monroe Woodbury High School
155 Dunderberg Road
Central Valley, NY

Table of Contents

Acknowledgments

I would like to thank everyone who helped me be able to complete my research. I would like to thank my teachers Mrs. Flanagan and Mrs. LaCondi for assisting me during the course and formulating a plan for me to complete. I would also like to thank my mentor from Express Scripts, Dr. Naveen Chilumula, for helping to come up with a methodology so that I can execute it and always being available whenever I needed assistance.

Abstract

Machine learning is the field where models learn without being explicitly programmed. Deep learning is a subfield of that where complex models are developed to solve problems. Computer vision is one problem where models try to recognize or identify different objects. In this study, I developed a convolutional neural network, a type of deep learning model, to classify different sets of images. In this case, the samples are handwritten digits. I looked into whether the training time of deep learning models would affect the accuracy rates of them (speed vs time). This was tested with central processing units (CPU) and graphic processing units (GPU), which is an external hard drive. GPU's are known to accelerate the training time by 10 to even 100 times faster than the standard CPU. I used the Mnist dataset and my own test set to analyze the results. I used my own test set to see real-world applications as it has no training set to work with. My convolutional neural network model achieves state of the art status when tested with the Mnist dataset with above a 99% accuracy. The results differed when using my own test set, with the accuracy rate much lower. In all of the tests shown, the GPU demonstrated better results than the CPU, however it was only marginal. The GPU shortened the training time almost ten times, which led to faster predictions and most likely contributed to their success.

A comprehensive study on Convolutional Neural Networks comparing GPU's and CPU's Performance on handwritten digit domain

Introduction

Machine learning is a vast area of research that continues to be developed every day. The term machine learning is derived from computers being able to learn without being explicitly programmed. This is seen in almost everything that we do on a day to day basis One example are search engines with getting specific websites or recommendations that you may prefer. Another example involves the development of robots are even driver assistance in self-driving cars to be able to recognize specific objects placed. This is what is known as computer vision where computer analyzes data given from images or from videos. Machine learning can range from a variety of fields, ranging from the medical field involving genetics through computer vision involving image classification.

The goal of machine learning is to be able to solve a given problem. Therefore a machine learning algorithm should be used in order to solve these problems. Researchers used different machine learning algorithms for predicting whether a salamander was present in an ecosystem or not using soil data as training. Complex algorithms include artificial neural networks (Srivastava, 2014). They concluded that neural networks were able to outperform other traditional algorithms and part of the reason why is their ability to perform with a large amount of data.

Computer vision continues to be in demand as there is a tremendous rise in research done in the past few decades. As said earlier, computer vision consists primarily of images and videos. The basic application of computer vision is object detection of images. There are some challenges which include variations in viewpoint, the difference in illumination, hidden parts of images, and background clutter. Also, traditional models are more likely to perform better with less data while neural networks perform well with a large amount of data. There are also plenty of datasets used in the branch of computer vision. The most common ones are the Mnist, ImageNet challenges, and German traffic sign datasets.

Most algorithms that deal with computer vision include neural networks and convolutional neural networks. However convolutional neural networks have greater success in the field of computer vision (Krizhevsky, 2012). These researchers were able to develop a functional Convnet on a large image dataset known as ImageNet. This large dataset contains about 1.2 million training images. They were able to successfully implement GPU's or as known as graphics processing units. With this, they were able to achieve groundbreaking success on a challenging dataset and able to perform better than traditional models.

As illustrated with this research (Tabik, 2017), they have shown that pre-processing methods are highly important to the success of deep learning models. These data pre-processing methods automatically enlarge the dataset so it will create more training samples. These methods include rotations, translations, and centering. By artificially enlarging the dataset, the model gets exposed to various images where they experience in the training phase. This research shows that pre-trained convolutional neural networks are able to perform at a higher rate than original ConvNets.

Convolutional neural networks have implications other than images as they can work with 3D sensors with recognizing human actions, which has implications in the real-world application which is important to modern standards. (Ji et al. 2014) 3D convolutional neural networks take different continuous frames together in order to capture positions at x, y, z which is done by the hardware layer. 2D convolutional networks have their limitations and therefore can not perform as well as 3D's in this type of domain which suggests the promises of future work. With the growing number of sensors, more data can be attained and therefore CNN's will be exposed to this broad range of data. Currently, much of this research focuses on high-end sensors.

Researchers have formulated the algorithms as an optimization problem (Albelwi, 2017). This means that the algorithm would choose the best solution out of all the feasible ones. With their architecture, they used deconvert layers (Zeiler 2013), which does the reverse operations of a regular Convnet to see if models are learning. With their algorithm, they were able to stay competitive with the other state of the art results. One limitation of this study was that their model can not be used outside of the computer vision field.

Similar to this, researchers used evolutionary convents that are alike to the previous article (Baldominos, 2018). With evolutionary algorithms, an initial population is first generated. Then the performance is computed with each individual. Each individual receives a fitness score based on how long they would survive based on “survival of the fittest”. Once again their best performing architecture achieved state of the art results and one of the top performers in the Mnist Dataset.

With all of these techniques in the field of computer vision, there is also controversy between whether the depth of the algorithm actually affects the results (Albelwi, 2018). With their results, the researchers found that with some of their architecture they found that some pooling layers did not follow by convolutional layers. The more layers an algorithm has may not necessarily be the best one. One particular network named ResNet has achieved good success in the Imagenet dataset. (Wu, 2016)

One of the important aspects of developing a functional algorithm that is able to have low computational training time. Usually, this happens with the use of GPU's (CireşAn, 2012). Another way of cutting down training time is for the use of regularization methods. The purpose of regularization methods is to lower the rate of overfitting which is important because it is shown to increase accuracy. Overfitting is when the model or algorithm is really used to the training data however does poorly to the testing data. One of the common ways to lower the overfitting rate was to use dropout (Srivastava, 2014). The term dropout refers to removing specific units in a neural network. It would also remove temporary incoming and outcoming connections. The choice to use dropout on specific nodes is random use with a low probability.

There has not been much focus on whether the use of graphics processing units has any effect on the error rate. Much research applies GPU's directly to their neural network. Normally, researchers would apply GPU's directly because they can compute almost ten times faster than CPU’s. One particular journal article (Lee, Kim, 2010) looked at the correlation between the accuracies of GPUs and CPUs with Convolutional Neural Networks and test it on different kernel operations that occur during the convolutional phase, so it is not much of a computer vision task. Those researchers try to disprove the claim that GPU's are far superior to CPU with

50 to even 1000 times performance advantage. However, in this study, they claim that the difference is only 2.5 times and further testing may be needed.

Purpose

The purpose of this study to explore the uses of graphics processing units in neural networks. The use of them would be able to see a correlation between the speed/time during the training of a neural network to the accuracy of the algorithm. This objective is tested on the handwritten digits dataset known as Mnist.

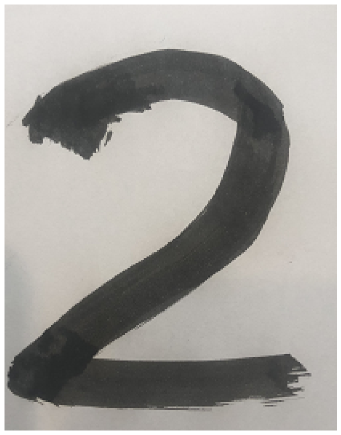
Methodology

The proposed plan is to create a convolutional neural network, one run implemented by a normal CPU and one run with a GPU The CPU acts as a control and I compared the Graphics Processing Unit to the control. I ran the model as many times as necessary using the CPU and then as many times as needed for the GPU. All of this is implemented in Python programming language. More specifically, I used Google Colab Notebooks, which is a Jupyter Open Source Project. What that means is that it comes prebuilt with some of the key libraries that I need in order to create and run my program. I used Keras for the deep learning software and features and Tensorflow as the backend. Important libraries I used are numpy and matplot. Another reason as to why I used the Google Colaboratory Notebooks is because it provides a free GPU. Normally the one from the notebook can cost around $300 to $400.

As mentioned before, the Keras deep learning library allows me to create the Convolutional neural network model. I developed the model to have three convolutional layers with two fully connected layers and finally a softmax layer. This does look much deeper than to 0what other researchers have proposed, however, I believe that more depth can be valuable in certain situations. Since the images from Mnist are black and white, a 2D convolutional layer is used so I do not have to account for RGB values. (Red, Green, Blue values that are commonly associated with color images so 3D convnets would be used in that case). The convolutional layer has a 5x5 kernel size (the sliding window operation) and strides in the x and y-direction (1,1). The activation unit is the default ReLUor (Rectified Linear Units), which can be used for optimization. The Max Pooling layer has a kernel size of 2x2 and a stride area of2x2. This layer acts as a statistical summary for the values from the convolutional layer. The next two layers is the same thing except for 32 input values, it is 64. Flatten out the Network with the fully connected layers and make it one-dimensional data. Add a dense 1000 neurons (nodes) layer with the Relu activation function. Lastly, a Dense layer states the number of classes with the Softmax activation function and make predictional values for each of the classes for that image. For the loss function, I used the Keras built-in cross-entropy. For data-preprocessing, I divided the pixel intensity values by 255 to give much smaller values and make the calculations much simpler for the computer to handle: computational power is finite. The batch size is 128 and hasl have 10 epochs (basically running through the training set 10 times in those batches). Dropout is used in this ConvNet as the rate is 0.5.

I ran my model first through the CPU and then through the GPU. The saving procedure is done with the pickle feature found through python documentation. The first phase is comparing the CPU and GPU performance on the Mnist data. The next phase is importing my own set of images similar to the Mnist dataset into the convolutional neural network. This is important as it tests how the model does in real-world scenarios. I imported 100 digits with each digit 10 times into the model. I would not only see how the model does in this scenario but also which hardware system has better performance. As shown below is an example.

Image 1



Results is done by looking through Matplot coding and visualizing how the model does in that case and see the loss function in depth. The loss function is huge because I can see if overfitting occurred or not. Results are plotting using many Excel spreadsheets.

Results

Figure 1

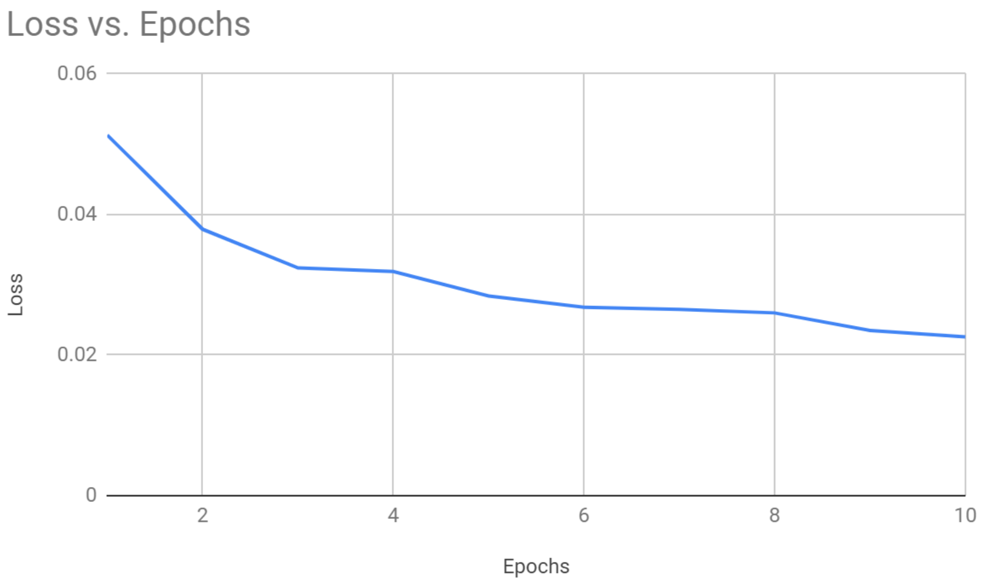
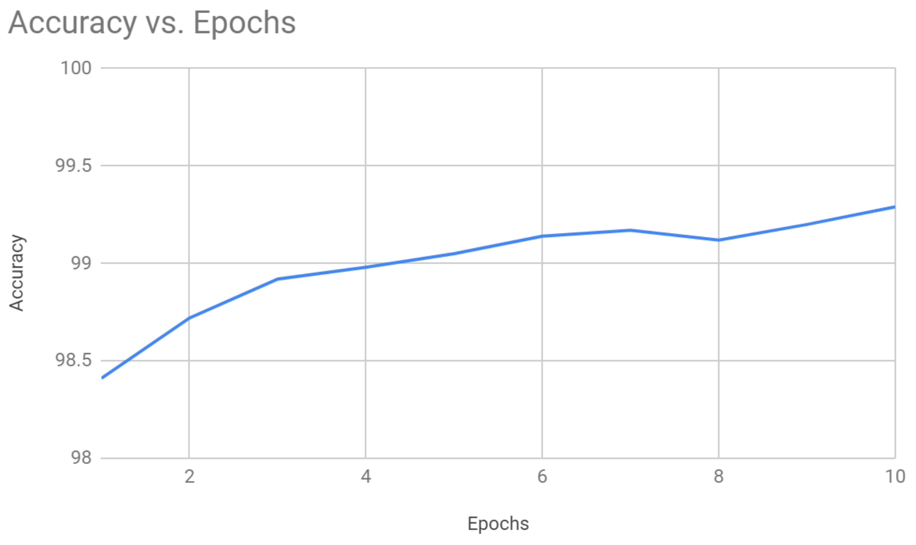
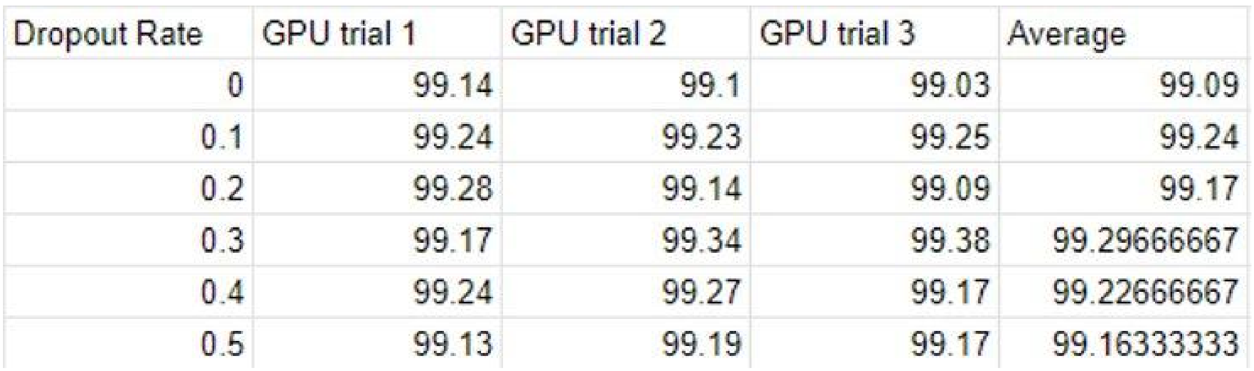


Figure 2



While looking at both figures 1 and 2, it is possible to analyze how the model performs. With the loss function, the goal of it is to minimize the "cost" value as much as possible to minimize the errors. From figure 1, it is shown that it is decreasing at a gradual rate with each passing epoch, showing that the model it learning from the mistakes. The final loss value for this particular instance is shown to be 0.0226. Figure 2 shows the accuracy rate which is the result of the loss function working properly as it is increasing for the majority. For this particular instance, the final accuracy rate is reported as 99.29%

Figure 3

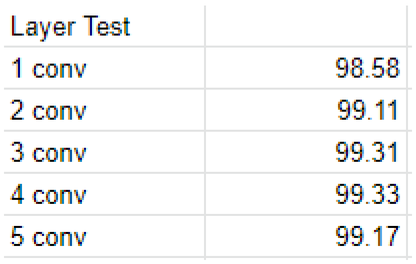


Dropout Rate	GPU trial 1	GPU trial 2	GPU trial 3	Average
0	99.14	99.1	99.03	99.09
0.1	99.24	99.23	99.25	99.24
0.2	99.28	99.14	99.09	99.17
0.3	99.17	99.34	99.38	99.29666667
0.4	99.24	99.27	99.17	99.22666667
0.5	99.13	99.19	99.17	99.16333333

As shown in figure 3, I ran multiple tests with my Convolutional Neural Network model in order to get a better analysis. The first test was with testing the dropout rate for both of the convolutional layers. I kept the rate constant for both of the layers to make things simpler to understand. As reported in figure 1, I kept the variation between 0.00 and 0.50 with 0.10 increments for testing. I repeated three times for each. I have found that when testing for the Mnist dataset, there was a considerable difference when looking at no dropout and with dropout. However, when comparing for each of the dropout rates separately, there was not a considerable

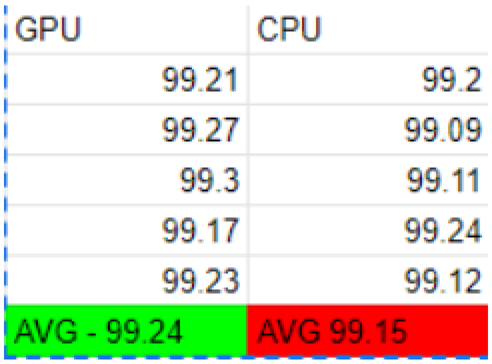
The difference at first glance. The top accuracy rate I found was 99.38% and the lowest was 99.03%, which was when applying no dropout. No dropout rate was considerably better than the other ones, which leads to the question as to why.

Figure 4



Layer Test	
1 conv	98.58
2 conv	99.11
3 conv	99.31
4 conv	99.33
5 conv	99.17

Another test I ran was with testing how model depth affects the results, as shown in figure 4. I tested with 1 through 4 convolutional layers only and repeated the test three times for each layer individually. Model depth has great influence as to whether a model can be able to function properly. A small one may not be satisfactory whereas a larger one can be accustomed to overfitting. It all depends greatly on the type of dataset being used and the metrics of the convolutional neural network itself. When testing with one layer, it was expected it would be the least achieving with a current average of 98.59%. With two convolutional layers, it was found to be a slight improvement with 99.01%. With the current setup of three convolutional layers, I found that the model has an accuracy rate of 99.30%. Lastly, with four convolutional layers, I found the rate to be only slightly worse with 99.26%. Possible explanations to this would definitely be the overfitting rate that the model got accustomed to training data. With the process, I ran through the graphic processing unit. I have noticed that the training time increases considerably with the greater number of convolutional layers.

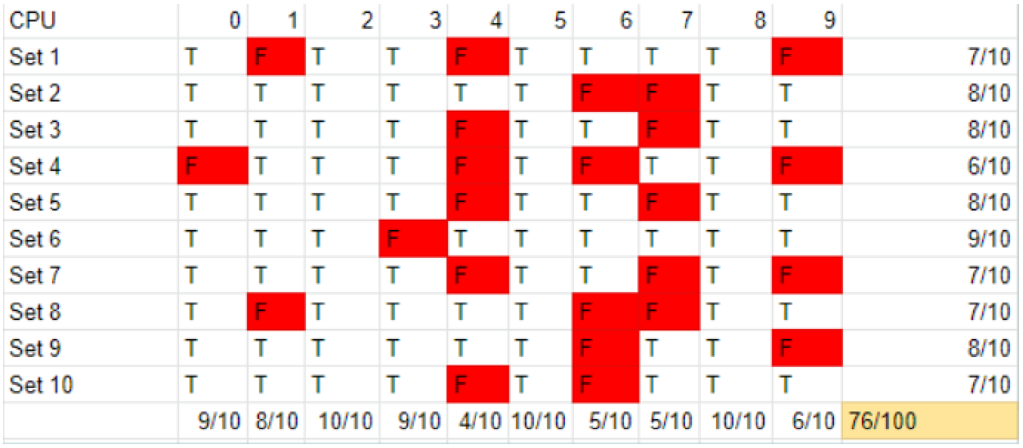


GPU	CPU
99.21	99.2
99.27	99.09
99.3	99.11
99.17	99.24
99.23	99.12
AVG - 99.24	AVG 99.15

Figure 5

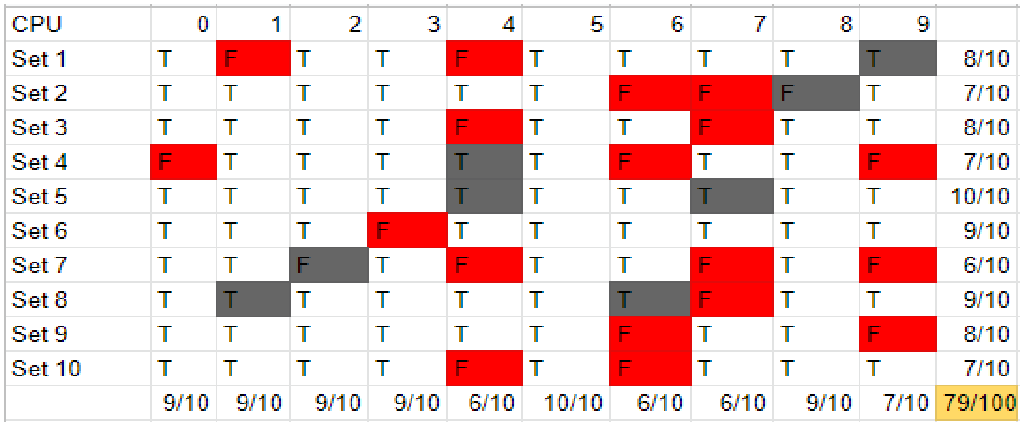
As for the main test I did, the results proved to be much different than when comparing with the Mnist Dataset. When comparing with the GPU and CPU separately, both did seem to provide similar types of results. When comparing with the Mnist dataset, both did manage to break 99%. The graphic processing unit did have generally better results.

Figure 6



CPU	0	1	2	3	4	5	6	7	8	9	
Set 1	T	F	T	T	F	T	T	T	T	F	7/10
Set 2	T	T	T	T	T	T	F	F	T	T	8/10
Set 3	T	T	T	T	F	T	T	F	T	T	8/10
Set 4	F	T	T	T	F	T	F	T	T	F	6/10
Set 5	T	T	T	T	F	T	T	F	T	T	8/10
Set 6	T	T	T	F	T	T	T	T	T	T	9/10
Set 7	T	T	T	T	F	T	T	F	T	F	7/10
Set 8	T	F	T	T	T	T	F	F	T	T	7/10
Set 9	T	T	T	T	T	T	F	T	T	F	8/10
Set 10	T	T	T	T	F	T	F	T	T	T	7/10
	9/10	8/10	10/10	9/10	4/10	10/10	5/10	5/10	10/10	6/10	76/100

Figure 7



CPU	0	1	2	3	4	5	6	7	8	9	
Set 1	T	F	T	T	F	T	T	T	T	T	8/10
Set 2	T	T	T	T	T	T	F	F	F	T	7/10
Set 3	T	T	T	T	F	T	T	F	T	T	8/10
Set 4	F	T	T	T	T	T	F	T	T	F	7/10
Set 5	T	T	T	T	T	T	T	T	T	T	10/10
Set 6	T	T	T	F	T	T	T	T	T	T	9/10
Set 7	T	T	F	T	F	T	T	F	T	F	6/10
Set 8	T	T	T	T	T	T	T	F	T	T	9/10
Set 9	T	T	T	T	T	T	F	T	T	F	8/10
Set 10	T	T	T	T	F	T	F	T	T	T	7/10
	9/10	9/10	9/10	9/10	6/10	10/10	6/10	6/10	9/10	7/10	79/100

The accuracy rates dropped, however, when looking at my own set of instances as we see in both figures 6 and 7, where figure 6 shows the results with the CPU and figure 7 shows the results with the GPU. The CPU reported an accuracy rate of 76%, whereas the GPU had an accuracy rate of 79%. The grey values in figure 7 show the difference between what images were classified differently and varied from the graphic processing unit and the CPU. Some of the digits that were heavily misclassified were the numbers 4,7, and 9. The most classified digits were the numbers 0,2,3,5. It makes sense even when looking at other papers (Salah 2016), who reported ones who had a distinct or an easy shape to be the easier ones whereas ones where digits that are written in a variety of ways can be quite difficult to classify.

Discussion

It is important to note the training time for both the CPU and GPU. On average, the CPU took almost 2-3 hours to train, whereas the GPU took around 20-30 minutes. All of it depends on the convolutional neural network architecture and the number of layers that are used. It has been reported that the number of layers does not have a positive correlation with the accuracy rate. As shown in figure 4, there is a considerable dip in the accuracy rate with four and five convolutional layers but not for the other layers. This theory most likely is true with larger datasets that are used.

Another question that arises is if the deep learning library could affect the results. This generally does not hold true as most libraries will contain the same features such as dropout and epochs instilled. However, an advanced library, such as Theano or Pytorch, could provide more advanced features and enhance results as reported in numerous journal articles.

Data pre-processing is important because it keeps the samples much simpler. If this step was not performed, the images would not fit the Mnist criteria with 28x28 pixel sizes and with the black and white colors, making it harder for the model to classify.

There could be many reasons as to why the model could not classify as well as the Mnist digits. For one, the format of how my digits were created was different to how the Mnist digits are. With the Mnist dataset, there are 60,000 training images for the model to learn from and is why the convolutional neural network model has a high accuracy rate. I introduced new sets of images which are similar, but yet still distinct from the other set images the model was accustomed to seeing. If I had a training set for the model to learn and test with that, the network could yield better results. It is why it is imperative to have more data for the model to learn from. Another reason maybe because of the photos that I took should be in high precision as it was taken with a phone camera. It is true that the clarity of my samples on screen would not be the same as showing Mnist samples so that could play a huge factor as to why the model underachieved.

Also, as stated prior, I used three convolutional and pooling layers throughout all of the experiments and tests I had done. The results may differ a bit when using more or less, but overall network length typically does not influence the success of deep learning models with medium-sized datasets like the MNIST one.

As far as the dropout rate, reported in figure 3, the results would be similar if it was tested with the GPU. This theory may not hold true when looking at larger datasets so it is important to test multiple dropout rates to see which is the best. However, it can be concluded that any rate of dropout will enhance the accuracy rate in the long term. Dropout works well in this instance because it essentially closes out the nodes on a given probability based on the rate that is applied. This essentially makes the model much simpler than the previous model with no dropout.

Through these tests, I approached to the conclusion that the GPU does not severely impact deep learning algorithms' performance, especially when the dataset is small in scale compared to ones like ImageNet where the size is roughly 1.2 million images. Especially with how small my test set was with only 100 images, the results may differ when using a much larger dataset to validate this theory. As far-right now, there is no real reason as to why the model guesses differently for when the GPU is used and when the CPU is being used with my own test set, as shown in figure 7 with the highlighted boxes in grey. One possible explanation would be that the GPU processes information quicker and so the guesses become a little bit more precise and more confident results compared to the CPU where it is much slower and the model as more time to analyze results with less confident results.

Conclusion

In sum, I created a working convolutional neural network model capable of achieving state of the art results. We tested this on the mnist dataset and then on my own test-set so that we can be able to see the applications in the real world. These types of scenarios are important for testing in the real world because there are many applications that do result from this, such as autonomous cars. Our own test set served as an example that it is difficult for models to achieve success on a variety of datasets given the set of circumstances. No special pre-processing techniques have been used except for the ones to format my test-set images. It is possible that with techniques such as rotation or translation, it would generate much more training samples and therefore provide more results.

Our convolutional neural network model achieved state of the art status with the results that were produced. Almost all of the tests showed accuracy rates above 99 percent. This shows that convolutional neural networks are the most valuable type of architecture for these types of problems related to computer vision. This is not the same case when applying this logic to our test set as the network barely achieved 80 percent accuracy. As said earlier, much of the reason is that there was no training set for this use, which was really set up for testing real-world success, as it is nearly impossible to train the model almost everything in the world that is possible.

The correlation between speed and accuracy is present; however, it is not deemed significant in smaller-scaled datasets like the one I used, which supports many of the theories other studies have shown. Further testing of this theory should be applied to much larger datasets such as ImageNet.

Future Research

For future research, I should include a training set for my own test set and see how the results are and if the results are similar to the Mnist ones. As or right now, research is trending towards the use of sensors. Sensors provide a lot more data from the environment and one prominent source are cameras and smartphones. Sensors are valuable because of the amount of data that can be retrieved and is a really good way of testing applications for the real world. With the use of sensors, they can record anything from any time, making them a viable source for information gathering. It is proven that with convolutional neural networks, more data is better for accuracy rates.

References

[1] Albelwi, Saleh, and Ausif Mahmood. "A framework for designing the architectures of deep convolutional neural networks." Entropy 19.6 (2017): 242.

[2]CireşAn, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. Neural networks, 32, 333-338.

[3]Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[4]Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[5]Ritambhara, et al. "Deepchrome: deep-learning for predicting gene expression from histone modifications." Bioinformatics 32.17 (2016): i639-i648.

[6]Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[7]Talbert, Doug, et al. "Using Machine Learning to Understand Top-Down Effects in an system: Opportunities, Challenges, and Lessons Learned." (2017).

[8]Turing, Alan M. "Computing machinery and intelligence." Parsing the Turing Test. Springer, Dordrecht, 2009. 23-65.

[9]Baldominos, Alejandro, Yago Saez, and Pedro Isasi. "Evolutionary convolutional neural networks: An application to handwriting recognition." Neurocomputing 283 (2018): 38-52.

[10]Wu, Zifeng, Chunhua Shen, and Anton van den Hengel. "Wider or deeper: Revisiting the resnet model for visual recognition." arXiv preprint arXiv:1611.10080 (2016).

[11]Yim, Junho, et al. "Image classification using convolutional neural networks with multi-stage feature." Robot Intelligence Technology and Applications 3. Springer, Cham, 2015. 587-594.

[12]Tabik, Siham, et al. “A Snapshot of Image Pre-Processing for Convolutional Neural Networks:Case Study of MNIST.” International Journal of Computational Intelligence Systems, vol. 10, no. 1, 2017, p. 555., doi:10.2991/ijcis.2017.10.1.38.