# Convolutional Neural Network Comparison for MNIST Digit Classification

Kevin Heist, Liam Hehir, and Michael Fantini

*Villanova University, College of Engineering, PA, USA*

03/05/2023

LeNet-5: https://colab.research.google.com/drive/1FbaNrKo2b5PioDIsy46FjEver10xz_z4?usp=sharing

AlexNet Code: https://colab.research.google.com/drive/1FbaNrKo2b5PioDIsy46FjEver10xz_z4?usp=sharing

VGG11 Code: https://colab.research.google.com/drive/1FbaNrKo2b5PioDIsy46FjEver10xz_z4?usp=sharing

**Abstract**— In this report, we will be comparing the effectiveness of 3 different networks in classifying the MNIST dataset. The MNIST dataset contains a large base of handwritten number images, and we will be using it to train and test our models. The three different networks to be evaluated are LaNet5, AlexNet, and VGG11Net. In each of these networks, we will provide a description of the architecture, various computed metrics, in addition to providing a final recommendation for which network to use.

**Keywords**— Convolutional Neural Network: A neural network used in image classification containing mathematical convolution layers, pooling layers, and a fully connected layer to classify an input. Loss: A measure of how well a model is performing relative to the expected result; the model updates weights following the gradient of the loss function to minimize its magnitude. Accuracy: The percentage of of input data that is correctly classified in a CNN.

## I. Introduction

The Neural Networks are one of the most powerful tools in machine learning and can be applied to a myriad of problems. New neural network architectures are being tested and tried everyday for both linear and logistic regression. One application where a neural network thrives in is for image classification which often takes the form of a Convolutional Neural Network (CNN). A CNN is a useful tool for extracting key features from a data set to inform predictions against a set of test data in a classification context. The CNN is considered a supervised learning approach in which the input data is pre-labeled such that the true classification is known and compared to the classification output by the model.

In any CNN architecture, convolution is central to the process. The convolution layer consists of a kernel (can be thought of as a matrix of specified size and value) that moves across an image according to it's stride (how many pixels it moves from one matrix operation to the next) and produces an output image of smaller size that isolates certain features of the image. This reduced image then passes through additional layers that extract more features from an image by means of an activation function. Typically, the final layer of the CNN is the fully connected layer. In this layer, the results of the convolution and activation layers are reduced down to a number of nodes corresponding to the number of classes in the data set. The model's weights are applied to the input nodes and a final classification is output by the CNN.

We will be evaluating 3 different CNN architectures in this report based on they're layers and performance. The MNIST dataset will be used to train all of the networks, and a common test set will be applied to the architectures for even comparison. Based on the plots and metrics, we will provide a final determination of which architecture produces the best classification. The links to the code for the 3 models can be seen below in the reference sections.

## II. Network 1: LeNet-5

The LeNet-5 architecture is one of the earliest CNN architectures founded in machine learning. It was created and trained on the MNIST dataset. It consists of five layers, with three convolutional layers and two fully connected layers. There are two subsampling layers between the convolutional layers which can be seen in Figure 1. For this project we are using the pre-trained model of LeNet-5 on the MNIST dataset.

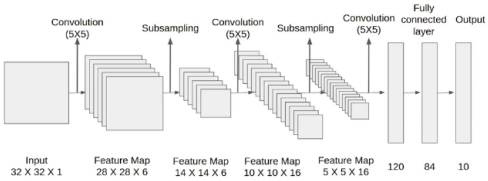


**Fig. 1:** Lenet-5 Architecture

The input to the first layer must be a 32x32 grayscale image. The MNIST dataset is comprised of 28x28 grayscale images, so they were resized in order to be passed through the first convolutional layer. The subsequent filtering layers

change the dimensions of the image from 32x32x1 to 5x5x16 as shown in Figure 1. All 400 nodes of the 5x5x16 image are then connected to each of the 120 hot features of the fully connected convolutional layer. Another fully connected convolutional layer is then used with 84 hot features, which then is connected to a number output to finally choose a digit between 0 and 9. The code for this network can be seen above, it was constructed using an article online [1].

## III. Network 2: AlexNet

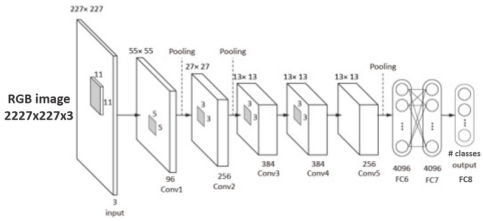


**Fig. 2:** AlexNet Architecture

AlexNet was a convulutional neural network architecture that was founded as a result of a large-scale image recognition challenge in 2012. The architecture consists of many more layers in comparison to the LeNet-5's architecture making it deemed as a deep neural network. AlexNet is typically used for RGB images, however for this demonstration we applied it to a gray-scale dataset. This architecture consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 3 fully connected layers, and 1 softmax layer. Each convolutional layer consists of convolutional filters and a nonlinear activation function ReLU. The AlexNet network works with image inputs of the size 256x256 pixels. For our dataset we adjusted it to take in 32x32 pixel images. Further details on the AlexNet architecture can be seen in Figure 2. The code for this network can be seen above, it was constructed using an online article [2].

## IV. Network 3: VGG11Net

The VGG-Net architecture (Visual Geometry Group) is a deep convolutional neural network that uses a smaller kernel size and more layers than LeNet-5 and AlexNet. Typically it is used in datasets on non-grayscale images, but we applied it to our MNIST dataset by changing the input from 3 channels to 1. The number of layers typically ranges from 16 to 19 and the kernel size is 3X3. The input goes through two initial convolution layers at depth=N, followed by a max pooling layer which reduces the size for the next convolution layer. This convolution has depth= N*2 and sometimes an additional convolutional before the max pooling. This cycle is repeated 4-5 times before entering the fully connected layer and the softmax function.

Since the depth of this network increases through the layers, training typically takes the longest out of the 3 networks discussed. The cascading depth of the filter does allow for high accuracy as a trade off. Figure 3 below illustrates the decreasing size and increasing depth of the layers in a VGG11 network. The code for this network can be seen above, it was constructed using an online article [3].

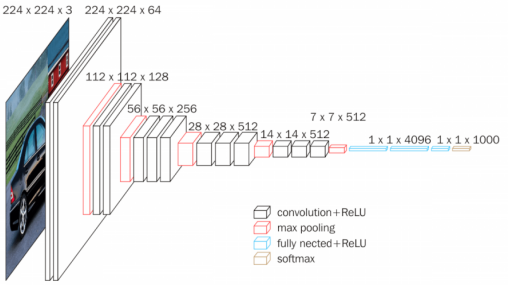


**Fig. 3:** VGGnet Architecture

## V. Approach and Implementation using Python

The dataset used for the modeling is the MNIST dataset. The dataset consists of 32x32 pixel gray-scale images of handwritten digits 0-9. The training set consists of 60,000 images while the testing set is 10,000 images. The input or features of our networks are the images being inputted often in batches. The labels or classes of the dataset are the digits 0-9.

When implementing the 3 CNN architectures in Python, various PyTorch methods were exercised. The cuda GPU was used to increase processing speed for the training of the network. First, the MNIST dataset was loaded through PyTorch's dataset library. Both the test and training set were acquired through the dataset library. Transformations were required to put the images in the correct data structure that can allow for it to be inputted into the models. The batch size remained constant at 32 for all models. Next, the neural network structures were designed in each of their respective ways. Next, the models were trained for 14 epochs apiece using entropy loss (Equation 1) as the cost function.Stochastic gradient descent (SGD) was the utilized optimizer for training. After each epoch the model was evaluated using training loss (Figure 4) and accuracy (Figure 5) using a test set. The accuracy was calculated by taking the max of the output of the network and validated if it matched the correct class. After training, the models would then be evaluated based off their accuracy and log loss in their predictions (Figure 6. The models' elapsed time while training was also tested over 3 iterations and recorded in Figure 7. Accuracy was found using max of the output and comparing the model's prediction to the image's expected class. Log loss was calculated using the sklearn function. Further details can be seen below in Equation 2. Finally, the results of the models' performances with relation to each class were explored.

## VI. Equations

Entropy Loss:

$$L = -\sum_{i=1}^{n} \hat{y}_i \log(p_i) \tag{1}$$

for n classes, where y-hat = truth label, and p = softmax probability of the class.

Log Loss:

$$LogLoss = -\frac{1}{q}\sum_{i=1}^{q}\sum_{j=1}^{l} y_{ij}\log(a_{ij}) \quad (2)$$

for l classes, where q is the number of elements in the sample, $a_{ij}$ is the answer (probability) of the algorithm on the i-th object to the question of whether it belongs to the j-th class, $y_{ij} = 1$ if the i-th object belongs to the j-th class, otherwise $y_{ij} = 0$. This equation was found using the source below[4].

## VII. Results and Analysis

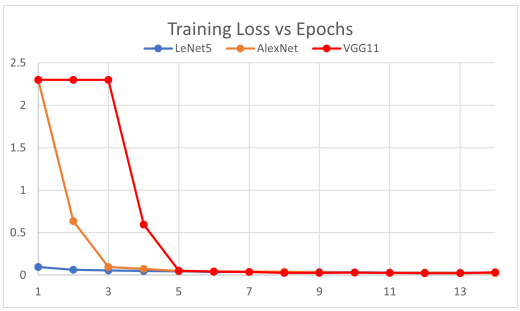


**Fig. 4:** Training Loss per Epoch

The first graph that is outputted is seen during the training phase of our models. The training loss of a neural network is a metric that checks how well the updated model fits the training data. A perfect model with respect to the training data would give a total training loss equal to zero. When looking at the 3 models that were evaluated, they all had a differing progressions with their training losses. The VGG11 net had a lot of training loss up until the 4th epoch then it began to converge at the 5th epoch with a training loss below .052. AlexNet followed a similar trend given, but had minimal training loss by the 3rd epoch. The LeNet-5 structure had very low training loss before the first epoch. This is to be expected due to the model being pre-trained for the MNIST dataset. Overall, all of the models had a training loss around .05 from epoch 5 and on meaning that models converged only about a third of the way through the training process.

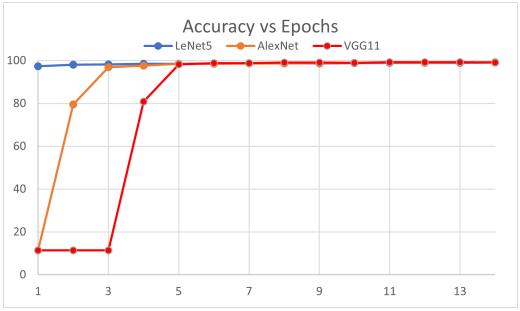


**Fig. 5:** Accuracy per Epoch

Along with the training loss, the accuracy of the models throughout the training process was calculated using a test set. The training loss graphs and accuracy graphs follow a similar pattern in which epochs they make the greatest jumps in. THe VGG11 net had an average accuracy of about 11% through the first 3 epochs. Then it reach over 98% by the 5th epoch. Finishing with an accuracy of 99.03%. Next, AlexNet had an accuracy of about 98% by the third epoch with it being significantly worse in the preceding epochs. It finished with an accuracy of 99.18%. Finally, the LeNet-5 model started with a 97% accuracy and finished with an accuracy of 99.08%. Its accuracy being high from the start is because the model is pre-trained.

| Model Accuracy / Log Loss | | |
|---|---|---|
| Model | Accuracy | LogLoss |
| LeNet-5 | 99.09% | .02796 |
| AlexNet | 99.18% | .02416 |
| VGG11 | 99.03% | .03278 |

**Fig. 6:** Accuracy and Log Loss

Accuracy for this report was measured by comparing the prediction of the model to the true class of the inputted image. All of the models resulted in very similar accuracies all being approximately 99% with AlexNet having the slight edge over the other two. Next, the log loss was calculated using the equation seen above in Equation 2. Log loss was the next metric that was calculated to measure the performances of all of the models. Lower log loss can be more indicative than accuracy at times due to it essentially calculating how poor a prediction can be. The AlexNet had the best log loss at .02416, with LeNet-5 being the next best, and VGG11 performing the worst.

| Training Time per Model | |
|---|---|
| Model | Time (h:m:s) |
| LeNet-5 | 6:22 |
| AlexNet | 5:27 |
| VGG11 | 1:44:5 |

**Fig. 7:** Accuracy and Log Loss

Each model was trained 3 times and the average time elapsed during the training phase was recorded. All models were trained using the cuda GPU. The VGG11 took nearly 2 hours to train which is to be expected due to the large number of layers and nodes it contains when compared to the other two models. The AlexNet and LeNet-5 models both took roughly 5-6 minutes to train. One thing to note is that the AlexNet consists of more layers and it still is more efficiently trained compared to the LeNet-5 model which was not expected. Also, VGG11 required much more time to make its predictions when compared to the other networks.

| Digit | Accuracy |
|---|---|
| 0 | 4.06% |
| 1 | 3.69% |
| 2 | 5.54% |
| 3 | 8.86% |
| 4 | 5.17% |
| 5 | 11.07% |
| 6 | 9.59% |
| 7 | 12.55% |
| 8 | 13.65% |
| 9 | 25.83% |

**Fig. 8:** Percent of Misclassification by Digit

The final relationship that was explored in this data set was analyzing the pattern of incorrect image classification. Based off of the wrong predictions made by the 3 models, we found which digits were the most often misidentified. The models struggled most with images containing the number 9 making up about 25% of the false predictions made up collectively by the models. Next, numbers 8, 7, and 5, accounted for about 12% of the inaccurate classifications. The other numbers made up for the rest of the errors with digits 0 and 1 being the the most easily identifiable digits in the dataset making up about 3.5% each of the mistaken predictions.

## VIII. Conclusion

The goal of this project was to see how different convulutional neural networks compared to the MNIST dataset. Through the founded results it is evident that a smaller network was more than enough to correctly classify the digit images that were inputted into our models. While all 3 of the models gave an accuracy of approximately 99%, the LeNet-5 and AlexNet architectures where able to acheive this with much lower time and resources necessary to train. This project also allows us to see that the deeper or more complex the neural network, does not necessarily mean the more effective it is. Often with the deeper networks you exchange computation time for accuracy, but in this case it seemed as if the VGG11 network was complicated to a fault resulting in lower accuracies and log losses. The AlexNet model produced the lowest log loss while also being the quickest to train which makes it the clear best option for the MNIST dataset.

Other things to note about this experimentation was each of the models only needed about 6 epochs of training to be able to produce an accuracy of about 98%. While the extra epochs helped get the accuracy closer to perfect, if this model were to be implemented the slight increase in accuracy may not be worth the extra computing time. Another evaluation that was explored was the relationship between each of the digit classes and how often their images were responsible for an error. Images of the number '9' were responsible for approximately 25% of the misclassification errors over all 3 of the models with the next closest making up about 13% of the mistaken labeling. If these models were to be implemented for practical use it is evident that the models would benefit from additional calculations when an image has potential to be '9'.

One final thing to consider are as to how a model like this could be applied to real world processes. Due to the nature of the MNIST dataset, the models trained in this project could be used for a multitude of applications. One example is for the practice of being able to scan any sort of numerical value via pictures and convert it to text. This may take a few different forms such as converting a handwritten paper to text, finding the correct dollar amounts on checks, or even being able to harvest the information from a written math test in a form that can be easily manipulated to evaluate its score. Overall, the applications of a digit classifying model are widespread and can be advantageous if further explored. Talk about their accuracies (comparable accuracies), log loss(AlexNet was best) why its important, time for compilation, epochs that should be trained up till. Bigger neural networks does not always mean better what network to choose. possible applications for this set (any written numerical scanning - checks, grading, etc) . Potential reasons for dataset skew towards 9

## References


1. https://blog.paperspace.com/writing-lenet5-from-scratch-in-python/
2. https://github.com/l5shi/Image-Recognition-on-MNIST-dataset/blob/master/AlexNet.ipynb
3. https://debuggercafe.com/training-vgg11-from-scratch-using-pytorch/
4. https://dasha.ai/en-us/blog/log-loss-function