Handwritten Letters Recognition

By Suyash Kathuria

Abstract

This report looks at 4 different Neural Networks that are trained and tested on the EMNIST handwritten letters dataset. The four different models used are Multilayer Perceptron(MLP), AlexNet, LeNet, LeNet with ReLU. It outlines the design for each model and then evaluates the results to see how they compare against each other. We picked these different models to analyse the differences between using convolutional layers and how to effects training time and test accuracy. We also evaluate the differences between the 2 activation functions, Tanh & ReLU. All of these models performed really well and achieved a test accuracy of higher than 90%. AlexNet had the highest accuracy with the longest training time whereas MLP had the fastest training time with the lowest accuracy.

Introduction

A Neural Network is a type of machine learning model that is inspired by the biological neurons in the human brain. We have designed 4 different neural network algorithms to classify Handwritten Letters from the EMNIST Database. It has a total of 145,600 images and 26 classes, 1 for each letter of the alphabet. This database is a mixture of lower-case and upper-case letters. We decided to choose the EMNIST database for our project because we wanted to try something simple as this is our first time working with machine learning. However, we still wanted it to be new to us, therefore not using the MINST handwritten digits. Being able to design a neural network that can classify handwritten letters, we can digitise handwritten books.

We chose 3 Convolutional Neural Networks (CNN) and one Multiplayer Perceptron (MLP) to make up our 4 models. MLP is a fairly simple neural network which does not have any convolutional layers. We wanted to see how it would perform against a CNN called AlexNet which has 5 convolutional layers. The other 2 CNN models are both, LeNet. One of them has been modified to use 'ReLU' as its activation function rather than 'Tanh'. We wanted to compare these 2 activation functions and also see what difference an activation function could make to a CNN model like LeNet.

Background

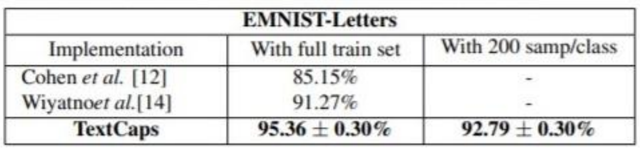
A neural network loosely copies how a brain's neuron system works, each neuron, or rather 'node' carries some sort of information. The links between different neurons is used to transfer that information. The system as a whole makes decisions based on this information from different 'nodes'. In a neural network, these nodes carry 'activation' values and have links that are weighted. A bunch of these nodes together make up what we call a layer. We are going to be working with 2 main layers types, a fully connected layer, and a convolutional layer. When an image is converted into a fully connected(FC) layer, each node represents a pixel from the image. Whereas a convolutional layer creates feature maps using the kernels to relate each pixel and its surroundings'. A convolutional layer is more complex which means it carries more accurate information than a FC layer. Activation functions are used after each layer to map the random values, information carried by the nodes, to a number that is more meaningful. In the training stage, the model learns to classify images by altering the 'weights' for the links between nodes. By training it a few times, the model gets more accurate as it finds those 'sweet' values for the weights which correctly classify images.

Literature review (state of art results)

The EMNIST's letters dataset is not as well-known but from the very few results online, here is two of them.

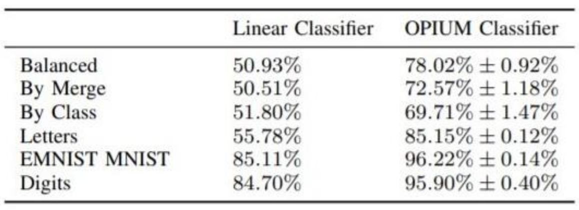
From the '**TextCaps** : Handwritten Character Recognition with Very Small Dataset':

Source: https://arxiv.org/pdf/1904.08095v1.pdf



EMNIST-Letters		
Implementation	With full train set	With 200 samp/class
Cohen *et al.* [12]	85.15%	-
Wiyatno*et al.*[14]	91.27%	-
TextCaps	**95.36 ± 0.30%**	**92.79 ± 0.30%**

From the '**EMNIST**: an extension of MNIST to handwritten letters'



	Linear Classifier	OPIUM Classifier
Balanced	50.93%	78.02% ± 0.92%
By Merge	50.51%	72.57% ± 1.18%
By Class	51.80%	69.71% ± 1.47%
Letters	55.78%	85.15% ± 0.12%
EMNIST MNIST	85.11%	96.22% ± 0.14%
Digits	84.70%	95.90% ± 0.40%

Source: https://arxiv.org/pdf/1702.05373.pdf?fbclid=IwAR0K2fJ
rmuJYGgzeJWWsnuFFRBGeJiAzbOd6BzEd37YRaw7QqY13zGcYMWU

Models' Design (Methodology)

Out of the four algorithms we have designed, one of them is a Multilayer Perceptron(MLP). It is one of the simplest neural networks. Our MLP consists of 3 FC layers, 2 of which are hidden layers and one output layer. The algorithm uses ReLU after every layer except the output layer. The image is first flattened and then passed into the network to be classified.

The other models are all convolutional layers. Let us take a closer look into AlexNet. Our AlexNet model is modified slightly to accept an image size of 28x28x1 instead of the traditional input for AlexNet which is 256x256x3. This model consists of 5 convolutional layers and 3 fully connected layers, one of which is the output layer. We also had to modify the traditional kernel sizes of AlexNet to match the much smaller image that we are using. This model uses ReLU as its activation function and the max pool function after each layer. Since all this reduces the dimensionality of the input images, we also added padding (a layer of zeros) to the last 4 convolutional layers. The complexity of this model means that it will take a longer time, but we expect this model to also perform the best as the convolutional layers enhances the learning ability for the model.

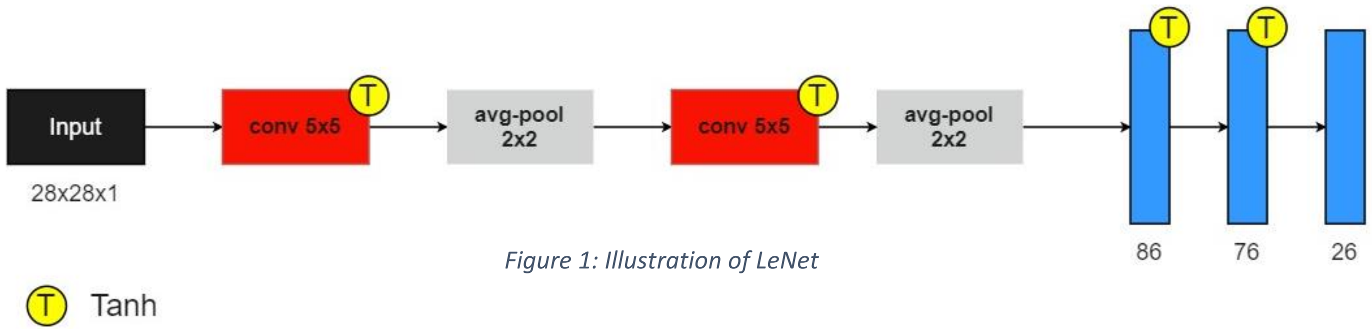


Figure 1: Illustration of LeNet

The other two models are both LeNet. They have 2 convolutional layers followed by the 3 fully connected layers.

Four our third algorithm, the LeNet model uses tanh as its activation function and average pool after every convolutional layer (Figure 1). In the last model, we have modified the LeNet to use ReLU instead of tanh as its activation function. This would help us evaluate the differences between these 2 activation functions on a convolutional neural network.

Evaluation

All of our models performed really well on the test dataset with all achieving an accuracy higher than 90%. We will now be looking deeper into the accuracy curves for each model and comparing them. First we will start with AlexNet and Multilayer Perceptron(MLP).

AlexNet vs Multilayer Perceptron

Figure 2: Learning Curve – AlexNet

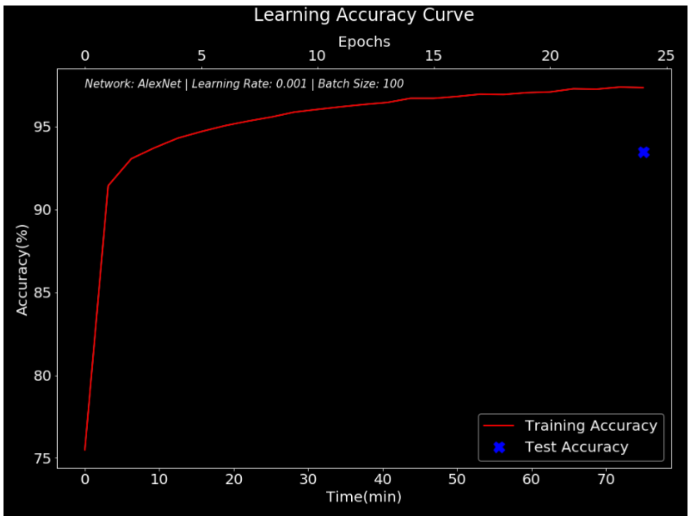
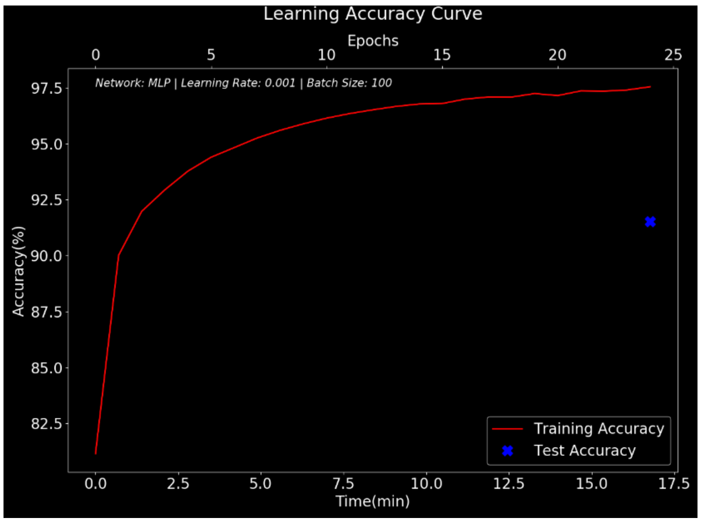


Figure 3: Learning Curve – Multilayer Perceptron



The AlexNet model achieves the highest test accuracy of 93.5% whereas the MLP model is the fastest to reach a high training accuracy compared to all other models. It only takes the MLP model just over 2.5 minutes to reach a training accuracy of 93% whereas for the AlexNet model it takes nearly 10 minutes. This is expected as the MLP model has no convolution layers and the AlexNet model has 5 convolutional layers. Adding convolutional layers increases the complexity of the model as it adds more weights which in turn results in a longer training time. The MLP model reaches a high trianing accuracy near the end of the training but when tested, results in a much smaller testing accuracy. This is most likely due to the overfitting of the model. Since a such simple model, it only requires a short amount of epochs to reach its peak and then it starts to overfit. This means that the MLP model starts to memorise the training dataset rather than learning to classify and therefore a big drop in accuracy when introduced a compeltely new test set. In contrast, the convolutional layers in the AlexNet model add more features which result in a smaller but still high gap from training to test accuracy also a sign of overfitting.

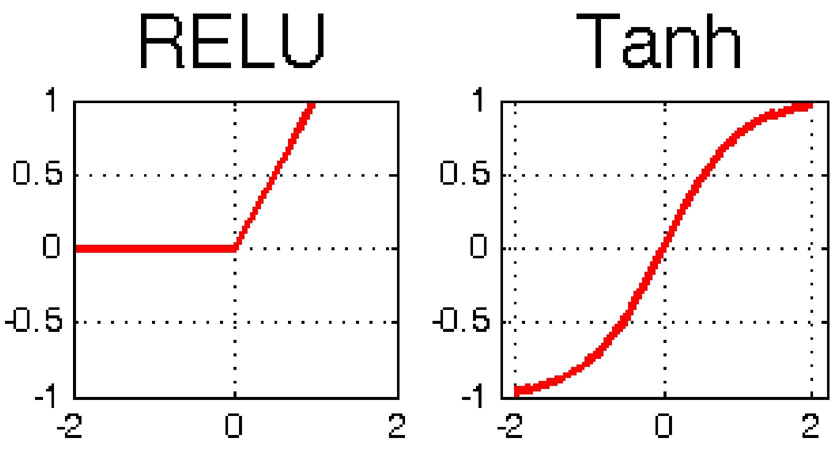
Now we will look at the other two models and look at the differeces between ReLu and Tanh.

ReLU vs Tanh (with LeNet)

An activation function is used to plot the activations from a neural network layer on a graph, mapped to a range of 0 to 1 or -1 to 1 depending on the function.

The ReLU, Rectified Linear Unit, function is half rectified at the bottom (Figure 3) which means that when used, all the activations that are negative become zero. It's range is 0 to 1. This decreases the computation power but can also decrease accuracy due to the negative activation values not being mapped appropriately on the graph.

Figure 4: ReLU vs Tanh



***Source**: https://www.researchgate.net/figure/Activation-Functions-ReLU-Tanh-Sigmoid_fig4_327435257*

The Tanh function whereas has a range from -1 to 1 and is a non-linear function unlike ReLU. Any negative values passed into this network are mapped strongly negative. This takes longer but can be more accurate especially when there's only 2 classes.

Now let's take a look at how these 2 different activation functions perform on a convolutional neural network.

Figure 5: Learning Curve – LeNet with ReLU

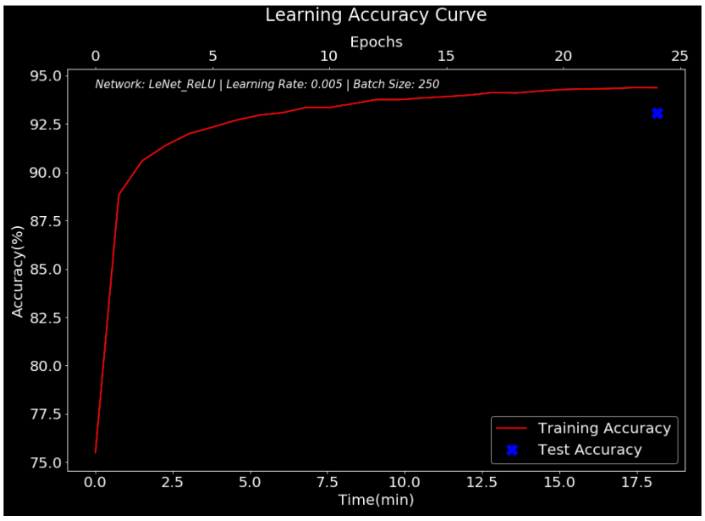
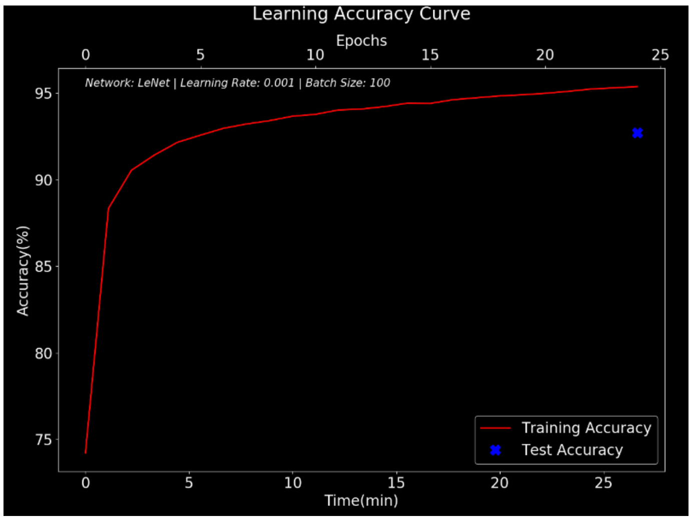
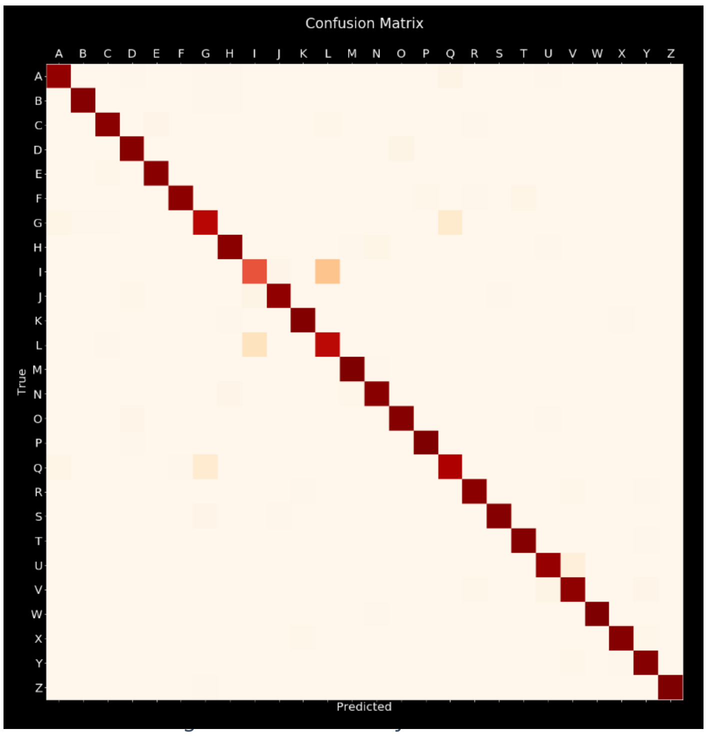


Figure 6: Learning Curve – LeNet with Tanh



The graphs above show the accuracy curves from the same model, LeNet but with different activation functions. The activation function is used after every layer, except the output layer, in each model. Now we can see that the model that used ReLU was faster than the model using tanh. This is expected as ReLU only focuses on the positive numbers which decreases the time taken at each epoch. If we look at the training accuracy, the model with tanh has a slightly higher training accuracy than the ReLU function but when tested on the test set, ReLU performs better. This is most likely because of tanh function getting confused due to their being 26 classes.

Confusion Matrix & Classification reports



The confusion matrix for our all 4 models was very similar, we will be looking at the confusion matrix produced by the AlexNet model (Figure 6). Overall, the model recognises most of the letters as it should. There's a few letters where our model gets confused at. It's clear to see from the matrix that our model sometimes fails to correctly diiferentiate between an 'L' and 'I'. This is reasonable since the dataset is a mix of lowercase and uppercase letters, the lowercase 'L' is written the same as uppercase 'I'. The matrix also shows that the model also gets confused between 'G' and 'Q' which again is reasonable due to the strokes for both letters being very similar.

Now if we compare all the four differnet classification reports, we can list our models from best to worst in terms of precision and recall. The four coloumns represent precision, recall, f1-score and support respectively.

AlexNet

```
    accuracy                         0.938     20800
   macro avg      0.939     0.938     0.938     20800
weighted avg      0.939     0.938     0.938     20800
```

Multilayer Perceptron

```
    accuracy                         0.912     20800
   macro avg      0.913     0.912     0.912     20800
weighted avg      0.913     0.912     0.912     20800
```

LeNet

```
    accuracy                         0.927     20800
   macro avg      0.928     0.927     0.927     20800
weighted avg      0.928     0.927     0.927     20800
```

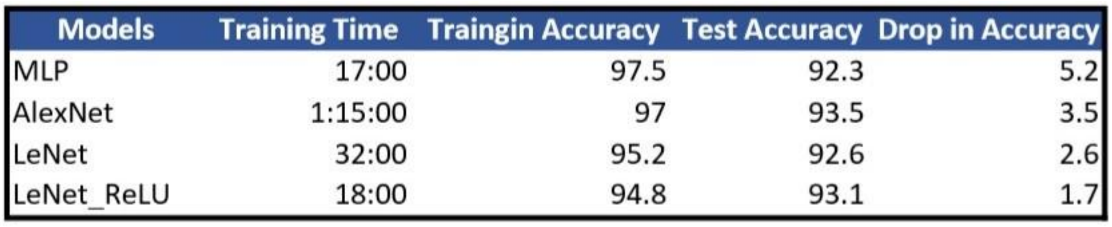
LeNet ReLU

```
    accuracy                         0.929     20800
   macro avg      0.930     0.929     0.929     20800
weighted avg      0.930     0.929     0.929     20800
```

Since this is a balanced dataset we don't see much of a different in precision and recall rate between classes (not shown in the tables above) and the average gives a pretty good look at the model's performance. Its evident that the AlexNet model performs the best in terms of classifying the different letters, with the LeNet ReLU and LeNet coming up very close to it. The MLP model performs the worst but still has an impressive performance.

Results & Future Work

Figure 8: Results



Models	Training Time	Traingin Accuracy	Test Accuracy	Drop in Accuracy
MLP	17:00	97.5	92.3	5.2
AlexNet	1:15:00	97	93.5	3.5
LeNet	32:00	95.2	92.6	2.6
LeNet_ReLU	18:00	94.8	93.1	1.7

Overall, all four of our models performed really well on the EMNIST handwritten letters dataset. We can see form the table above and as mentioned earlier in the report, AlexNet was the most accurate model as expected due to the number of convolutional layers that add more features and help the model learn the dataset better but also the slowest as it takes more computional power which in return takes more time. The MLP was the fastest model as it doesn't have any convolution layers and gives the lowest, but still high accuracy. We can conclude that adding convolutional layers to a model does increase the accuracy of the model but increases the time it takes for the training. There's always a trade off with using convolutional layers between the accuracy and the training speed of the model.

From the comparison between the activation functions, we can see that using ReLU improves the training time due to decreasing the computational power whereas the tanh function takes longer and also can get confused when used with too many classes as it has a bigger range to map the activations to. We can conclude that the ReLu funtion performs better than a tanh function on a convolution network.

Something to notice would be that the LeNet ReLU model takes slightly more time than the MLP model and achieves the second highest accuracy out of all four models. We can conclude that LeNet ReLU is the best model out of all the four models to be used on the EMNIST handwritten leters dataset as it achieves a high accuracy with a fairly short amount of time.

For future, we could try out using tanh function on the MLP model instead of the ReLU function. Since the MLP is fast we could use a more complex activation function to improve the accuracy. The tanh function could slow the model down but also increases the accuracy resulting in a similar performance to that of from the other CNN models.

On the other hand we could look into adding dropout layers to our AlexNet model. Its clear from the results above that our AlexNet model is overfitting since the drop in accuracy from training to test is the second highest. Adding dropout layers might help us counter that.