CNN 5 Fonts Classification

Banatwalla, Muhammad

April 14, 2021

# Introduction

For this homework, we will be working with Convolutional Neural Networks. We will perform image classification using a convolutional neural network into 5 classes. We will evaluate the performance of the various CNN models we build, varying the hidden layer size as well as implementing dropout techniques to find the most accurate model.

## Description of Data

For this assignment, we will again get our data from the ”University of California Irvine Repository for Machine Learning Datasets”. The data we are working with consists of digitized images of typed characters. The fonts we chose are Century, Ebrima, Gill, Leelawadee, and Proxy. Before treatment of the data, the data had the following sizes: Century - 7994, Ebrima - 6892, Gill - 5836. After selecting only the cases that are italic and with strength = .4, like we did in HW 2, they had the new following sizes: Century - 1999, Ebrima - 1723, Gill - 1459. The data consists of 400 features corresponding to a series of gray level image intensity values for different pixel positions.For this report, Century will correspond to Class 1, Ebrima will correspond to Class 2, Gill will correspond to Class 3, Proxy will correspond to Class 4 and Leelawadee will correspond to Class 5.

We then split the data into training and testing data, with a 80-20 split. The training set had 6671 cases and the testing set had 1670 cases. This means the total number of infos being fed into the model is 33,355.

For the purposes of CNN we reshaped the data to the 20x20 pixel image. Below is the letter ’A’ for each of the fonts we chose:



(a) Century (b) Ebrima (c) Gill (d) Proxy



(e) Gill

# Convolutional Neural Network

A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance using learnable weights and biases to various aspects/objects in the image and be able to differentiate one from the other. Each convolutional neural network is made up of one or many convolutional layers. Their goal is to find patterns from within images that can be used to classify the image or parts of it. The difference between a dense layer and a convolutional layer is that dense layers detect patterns globally while convolutional layers detect patterns locally. Our convolutional layers take feature maps as their input and return a new feature map that represents the presence of specific filters from the previous feature map. The filters can slide continuously through the image covering every possible position. This is common but sometimes we introduce the idea of a stride to our convolutional layer. The stride size represents how many rows/cols we will move the filter each time.

Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. The idea behind a pooling layer is to downsample our feature maps and reduce their dimensions. This is to decrease the computational power required to process the data while maintaining the process of effectively training the model. A common pooling method is max pooling that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map. A pooling layer is a new layer added after the convolutional layer. Specifically, after a nonlinearity (e.g. ReLU) has been applied to the feature maps output by a convolutional layer.

To help with a problem of overfitting, we implemented the dropout technique. Dropout randomly turns off a portion of neurons during training to possibly prevent dependency on the training set. For this CNN, we set our dropout percentage to 50%. Turning off 50% of the neurons can help prevent the model from memorizing the training data.

## Training CNN

The CNN architecture is a stack of convolutional layers followed by max pooling layers. The idea is that the stack of convolutional and maxPooling layers extract the features from the image. The first convolutional layer had 16 channels with a window size of 5x5 and a stride of one. Which means the convolutional layer is going to examine 5 x 5 blocks of pixels in each image. The stride size represents how many rows/cols we will move the filter each time. The next layer is a maxpool layer with 16 channels, window size of 2x2, and a stride of two. The next convolutional layer has 16 channels, window size of 3x3, and stride of one. The next maxpool layer has the same structure as the previous maxpool layer 16 channels, with window size 2x2, and a stride of two. Then these features are flattened and fed to densely connected layers that determine the class of an image based on the presence of features. After the hidden layer we implemented the dropout technique. The softmax function was applied to get the probability that each case belongs to each class. We used a batch size of 77 and implemented early stopping based on loss with a patience of 25.

We trained our CNN with a variety of hidden layer sizes. We used h values: 90,150,200,250,300,400, and 500. Below are the results for each CNN.

First, we tried a hidden layer size of 90 neurons. This model had 16,241 weights and biases which is less than the total number of infos. We first show the cross entropy plot:

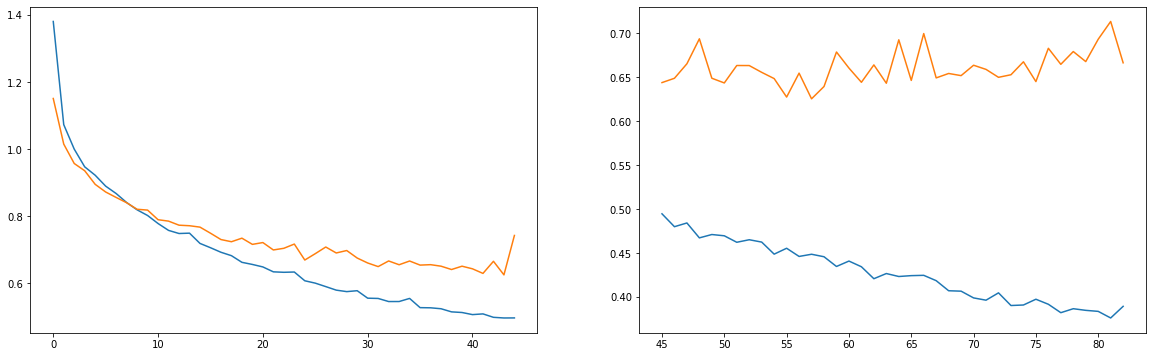


Figure 2: Cross Entropy H90

Cross-entropy loss measures the performance of a classification model. Since the aim is to minimize the loss, the smaller the loss the better the model the cross entropy vs. epochs plot can help us determine the best model and if there is any overfitting. The training curve and test curve should perform similarly if no overfitting is occurring. Figure 2 shows that the training and test set are performing similarly; the training set loss goes down more as the number of epochs increases but still stays at a similar range as the test set. The loss decreases drastically until 30 epochs then stabilizes. Around 70 epochs the training set loss is going down more than the test set so there is a possibility of overfitting. We implemented dropout 50% and early stopping to prevent this. Early stopping stops training once the model performance stops improving on the validation dataset. We had a patience of 25. Patience is the number of epochs to wait before early stop if no progress on the validation set. Early stopping stopped training here at 85 epochs.

Below is the graph of test and train accuracy as well as the confusion matrix:

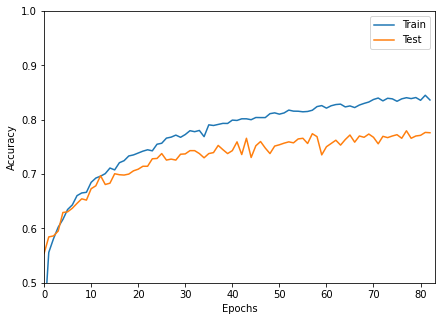


Figure 3: Accuracy H90

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL | Predicted: PROXY | Predicted: LEELAWADEE |
| Actual: CENTURY | 81% | 6% | 3% | 4% | 7% |
| Actual: EBRIMA | 9% | 66% | 7% | 2% | 16% |
| Actual: GILL | 6% | 7% | 78% | 3% | 6% |
| Actual: PROXY | 0% | 0% | 0% | 100% | 0% |
| Actual: LEELAWADEE | 13% | 21% | 2% | 3% | 62% |

Table 1: Confusion Matrix for CNN H90

Figure 3 shows the accuracy of our classification model versus number of epochs. It also shows that the accuracy increases drastically until 30 epochs then stabilizes. This model had an overall accuracy of 79.2%. According to table 1, the model with a hidden layer size of 90 neurons is classifying well for century, gill, and proxy font but it has a lower accuracy for ebrima and leelawadee. It seems to be confusing them as it inaccurately predicted 21% of true class leelawadee as ebrima and 16% vice versa. It is interesting to note that the Proxy font was classified perfectly. This could be because this font looks entirely different than all of the other fonts. Referring to the images in section 1.1, the proxy ‘A’ is a lot thinner and has a different shape than all of the other fonts.

We then tried a CNN with hidden layer size of 150 with a total number of weights and biases of 42,041. Below is the cross entropy plot:

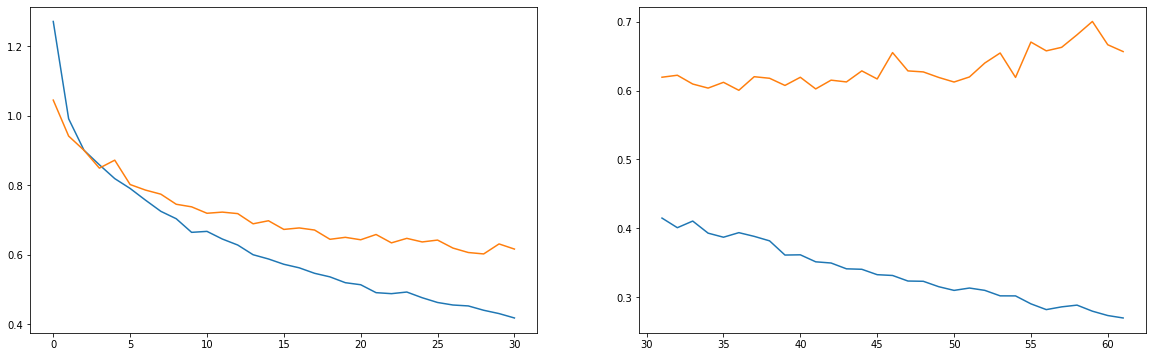


Figure 4: Cross Entropy H150

The cross entropy plot versus epochs, figure 4, for the model with a hidden layer size of 150 neurons shows that the loss decreased dramatically until around 30 epochs. Then the test set stabilized at a cross entropy loss of around 0.6 but the training set loss continued to decrease less than 0.3 after 55 epochs. This is worrisome because if the training set is performing much better than the test set then overfitting may be occurring. Overfitting is when the model performs well on the training set but not as well on new data it hasn’t seen before.

Below is the graph of test and train accuracy as well as the confusion matrix:

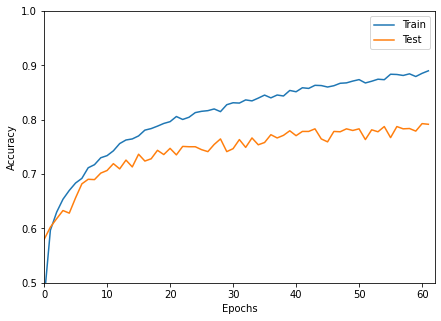


Figure 5: Accuracy H150

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL | Predicted: PROXY | Predicted: LEELAWADEE |
| Actual: CENTURY | 79% | 7% | 3% | 4% | 7% |
| Actual: EBRIMA | 6% | 72% | 7% | 2% | 12% |
| Actual: GILL | 4% | 7% | 82% | 3% | 3% |
| Actual: PROXY | 0% | 0% | 0% | 100% | 0% |
| Actual: LEELAWADEE | 6% | 21% | 4% | 3% | 67% |

Table 2: Confusion Matrix for CNN H150

Figure 5, the accuracy versus epochs plot shows that the accuracy increases until around 20 epochs then stabilizes. As in figure 4 it also shows the training set performing better and the test set stabilizing after 40 epochs. This model had an overall accuracy of 80.1%.Table 2 shows that increasing the hidden layer size to 150 neurons helped the performance of the model for classifying the fonts ebrima, gill, and leelawadee. There is still some improvement that could be made because 21% of true class leelawadee fonts were classified by the model as ebrima.

We then tried a CNN with hidden layer size of 200 with a total number of weights and biases of 55,141. Below is the cross entropy plot:

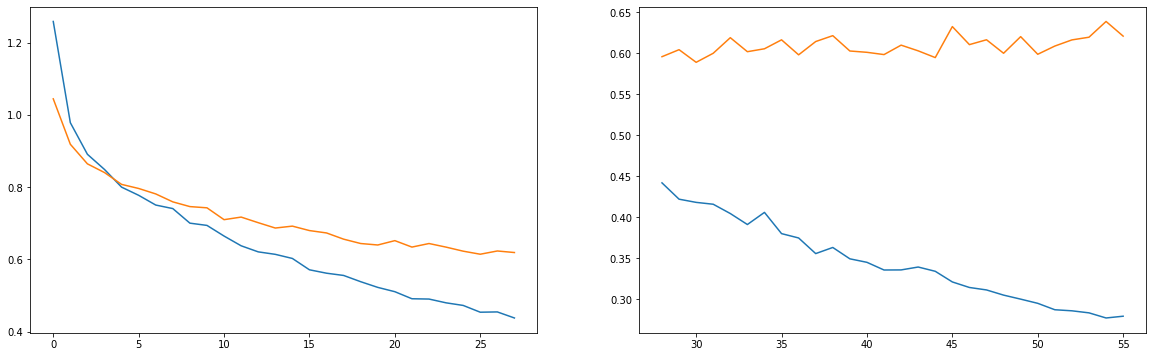


Figure 6: Cross Entropy H200

Figure 6 shows the cross entropy loss vs epochs of the model with a hidden layer size of 200 neurons. The loss decreases intensely for the training and test set until 20 epochs then stabilizes. It is interesting to note that after 30 epochs the test set loss stays around 0.6. However, the training set loss continues to decrease. This could be an indicator that the model is overfitting on the training set.

Below is the graph of test and train accuracy as well as the confusion matrix:

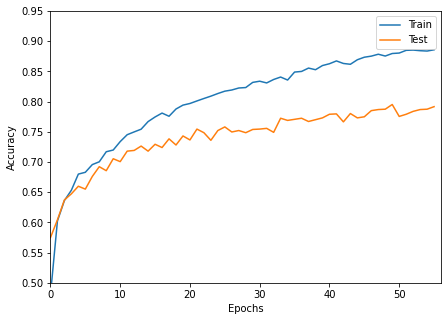


Figure 7: Accuracy H200

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL | Predicted: PROXY | Predicted: LEELAWADEE |
| Actual: CENTURY | 83% | 7% | 3% | 4% | 4% |
| Actual: EBRIMA | 7% | 73% | 4% | 2% | 14% |
| Actual: GILL | 7% | 5% | 80% | 3% | 4% |
| Actual: PROXY | 0% | 0% | 0% | 100% | 0% |
| Actual: LEELAWADEE | 8% | 20% | 3% | 3% | 67% |

Table 3: Confusion Matrix for CNN H200

Figure 7 shows that the accuracy for the training and test set increases until 20 epochs then stabilizes. The training set accuracy does continue to increase after 30 epochs while the test set accuracy stays about the same. This model had an overall accuracy of 78.9%. According to table 3, increasing the hidden layer size to 200 neurons kept the performance of the model for classifying all the fonts about the same. It increased the classification for century by 4% which was the highest increase in performance out of the 5 fonts.

We then tried a CNN with hidden layer size of 250 with a total number of weights and biases of 68,241. Below is the cross entropy plot:

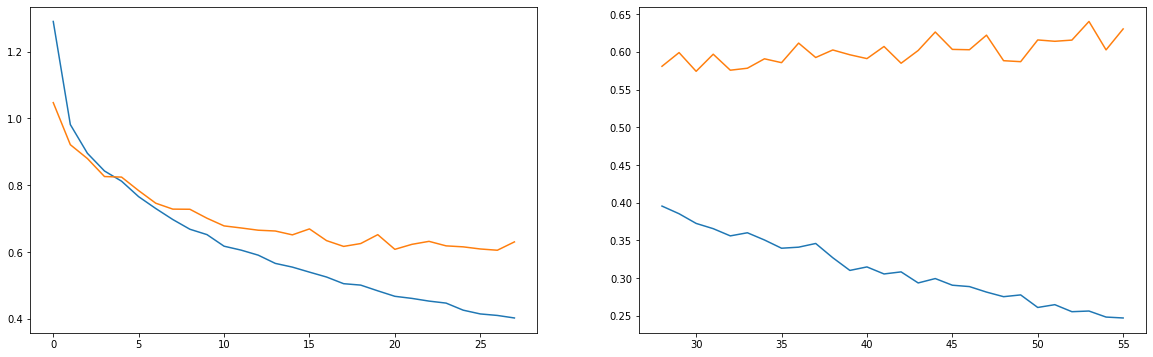


Figure 8: Cross Entropy H250

Figure 8 shows that the test set loss decreases until around 15 epochs then stabilizes around 0.6. However, the training set loss continues to decrease drastically as the number of epochs continues to increase.

Below is the graph of test and train accuracy as well as the confusion matrix:

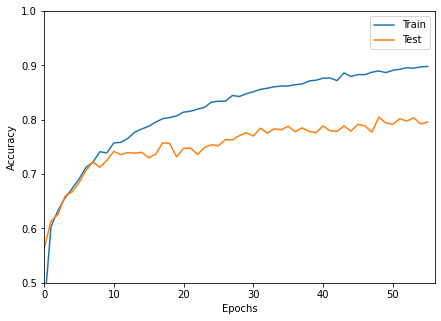


Figure 9: Accuracy H250

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL | Predicted: PROXY | Predicted: LEELAWADEE |
| Actual: CENTURY | 81% | 5% | 4% | 4% | 7% |
| Actual: EBRIMA | 9% | 67% | 5% | 2% | 17% |
| Actual: GILL | 5% | 3% | 84% | 4% | 4% |
| Actual: PROXY | 0% | 0% | 0% | 100% | 0% |
| Actual: LEELAWADEE | 9% | 19% | 3% | 3% | 66% |

Table 4: Confusion Matrix for CNN H250

Figure 9 which displays accuracy versus epochs shows that the accuracy for both the training and test set increases until about 10 epochs. The test set accuracy stabilizes at around 70% accuracy as the epochs continue to increase. However, the training set accuracy continues to increase reaching 90% around 50 epochs. The difference between the training and test set accuracy is alarming. This model had an overall accuracy of 80.5%. According to table 4 increasing the hidden layer size to 250 neurons only increased the classification of font gill. The rest of the fonts either stayed or decreased accuracy even if just a few percentage points.

We then tried a CNN with hidden layer size of 300 with a total number of weights and biases of 81,341. Below is the cross entropy plot:

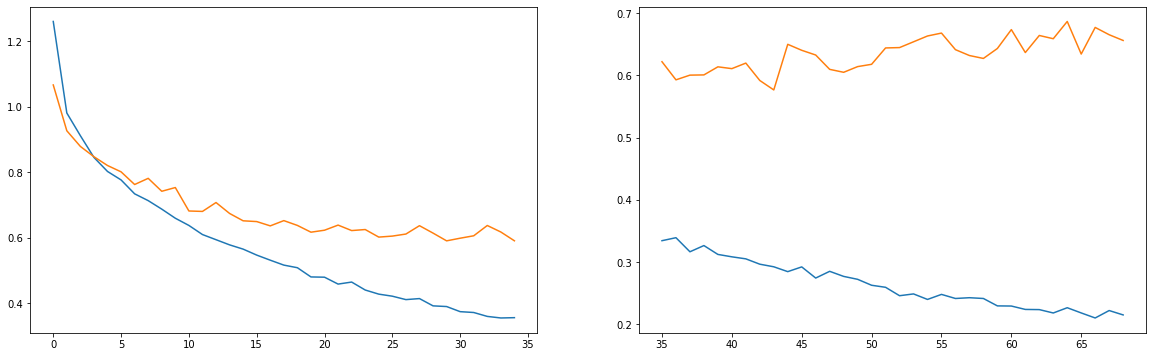


Figure 10: Cross Entropy H300

Figure 10, the cross entropy loss versus epochs shows that for the test set the loss decreases until 15 epochs then stabilizes at 0.6 after that. On the other hand, the training set loss continued to decrease reaching 0.2 at 65 epochs. At this point it is possible that the model is overfitting to the training set since there is a large difference in the performance of the training set and test set.

Below is the graph of test and train accuracy as well as the confusion matrix:

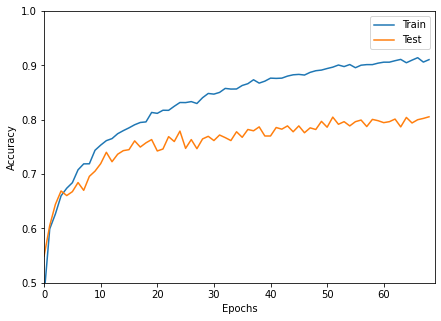


Figure 11: Accuracy H300

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL | Predicted: PROXY | Predicted: LEELAWADEE |
| Actual: CENTURY | 82% | 7% | 3% | 4% | 5% |
| Actual: EBRIMA | 7% | 69% | 4% | 2% | 17% |
| Actual: GILL | 6% | 6% | 79% | 3% | 5% |
| Actual: PROXY | 0% | 0% | 0% | 100% | 0% |
| Actual: LEELAWADEE | 6% | 21% | 0% | 3% | 69% |

Table 5: Confusion Matrix for CNN H300

Figure 11, shows that the test set accuracy increased until 20 epochs then the accuracy stayed between 70-80%. Yet, the training set accuracy continued to increase reaching 90% accuracy at 60 epochs. This model had an overall accuracy of 80.3%. Table 5 shows that increasing the hidden layer size to 300 neurons increased the performance of the classification of the fonts century, ebrima and leelawadee by the model. Unfortunately, still many true class leelawadee fonts are being inccuractly predicted by the model as ebrima fonts and vice versa.

We then tried a CNN with hidden layer size of 400 with a total number of weights and biases of 107,541. Below is the cross entropy plot:

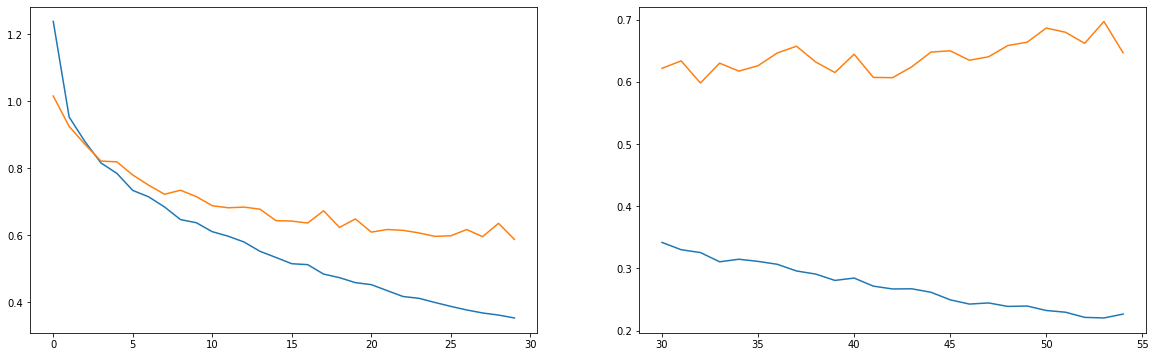


Figure 12: Cross Entropy H400

Figure 12 shows that cross entropy loss decreases for both the test and training set until about 20 epochs. After that the test set loss stabilizes at 0.6 and the training set loss continues to decrease reaching almost 0.2 at 55 epochs.

Below is the graph of test and train accuracy as well as the confusion matrix:

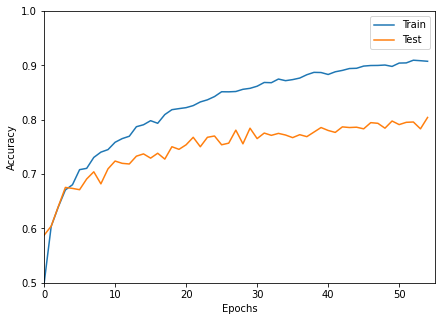


Figure 13: Accuracy H400

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL | Predicted: PROXY | Predicted: LEELAWADEE |
| Actual: CENTURY | 82% | 6% | 3% | 4% | 6% |
| Actual: EBRIMA | 7% | 69% | 4% | 2% | 17% |
| Actual: GILL | 7% | 4% | 83% | 3% | 3% |
| Actual: PROXY | 0% | 0% | 0% | 100% | 0% |
| Actual: LEELAWADEE | 8% | 20% | 2% | 3% | 68% |

Table 6: Confusion Matrix for CNN H400

Figure 13 shows that the accuracy of the test set increases until 20 epochs then stays between 70-80%. the training set accuracy continues to increase as the number of epochs increases reaching almost 90% at 50 epochs. This model had an overall accuracy of 80.1%. According to table 6, increasing the hidden layer size to 400 neurons only increased the classification accuracy for gill by 4% and the leelawadee font by 1%. The other fonts stayed the same or decreased in classification accuracy by our model.

We then tried a CNN with hidden layer size of 500 with a total number of weights and biases of 133,741. Below is the cross entropy plot:

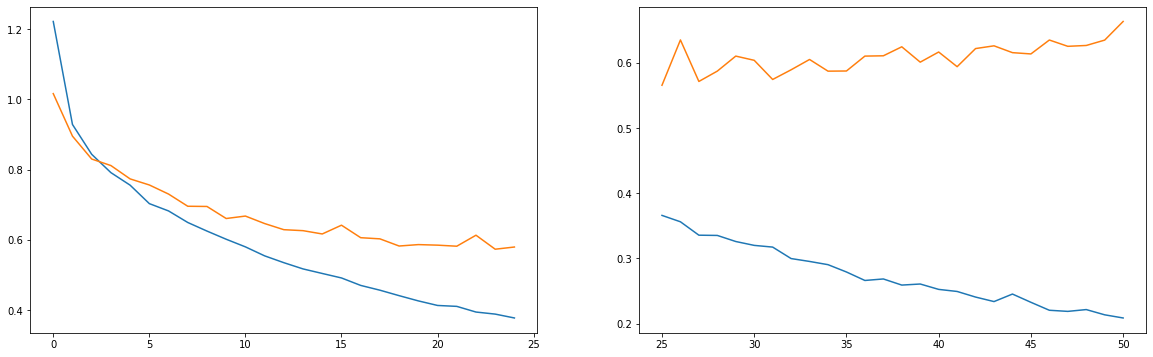


Figure 14: Cross Entropy H500

Figure 14 shows that the cross entropy loss decreased until about 20 epochs for both the training and test set. The test set loss stabilized after that but the training set loss continued to decrease.

Below is the graph of test and train accuracy as well as the confusion matrix:

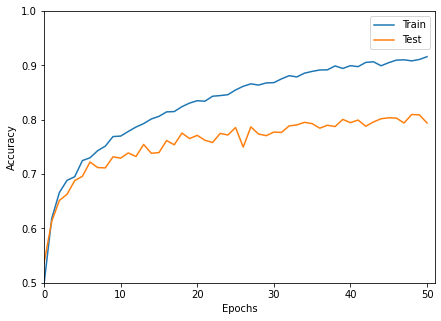


Figure 15: Accuracy H500

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL | Predicted: PROXY | Predicted: LEELAWADEE |
| Actual: CENTURY | 83% | 7% | 3% | 4% | 4% |
| Actual: EBRIMA | 7% | 77% | 3% | 2% | 11% |
| Actual: GILL | 5% | 4% | 82% | 3% | 3% |
| Actual: PROXY | 0% | 0% | 0% | 100% | 0% |
| Actual: LEELAWADEE | 8% | 23% | 2% | 3% | 64% |

Table 7: Confusion Matrix for CNN H500

Figure 15, demonstrates that the accuracy of the test set increased until about 10 epochs then stayed between 70-80%. The accuracy of the training set on the other hand continued to increase as the epochs increased. After 20 epochs the training set accuracy was between 80-90% and it seemed that if the number of epochs continued to increase the accuracy of the training set would too. The problem is that we would like our model to perform similarly on data it’s trained on with data it has never seen before. The difference between the fits of the training and test set is called variance. Ideally in our model we would like to have low variance. We also want to capture the true relationship for the training set - bias. Because of the bias-variance trade off we try to find a point that we get good predictions, maybe not great ones, but that our model will give consistently good predictions. The test set helps us see how well our model will perform on new data. In the next step we will try to find the best CNN that will hopefully give consistently good predictions.

This model had an overall accuracy of 80.7%. The confusion matrix for the CNN with hidden layer size of 500 shown in table 7 had an increase in classification accuracy for ebrima. The amount of true class ebrima wrongly predicted as leelawadee fonts decreased therefore increasing the accuracy for ebrima. However, the other fonts stayed the same or decreased in accuracy.

## Finding Best CNN

To compare performances of the CNN models we tried, we will look at the graph of H versus accuracy as well as confidence intervals.

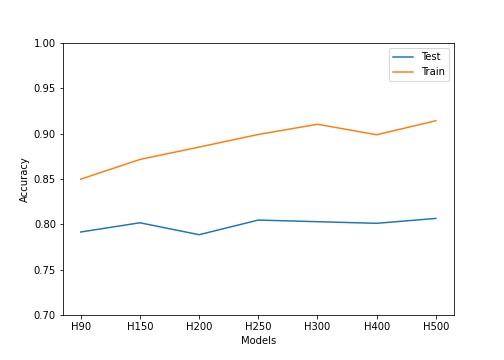


Figure 16: H v. Accuracy

Below are the 95% confidence intervals for the font accuracy’s for each model:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | H90 | H150 | H200 | H250 | H300 | H400 | H500 |
| CENTURY | (77%,85%) | (75%,83%) | (79%,87%) | (77%,85%) | (78%,86%) | (75%,85%) | (79%,87%) |
| EBRIMA | (61%,71%) | (67%,77%) | (68%,78%) | (62%,72%) | (64%,74%) | (64%,74%) | (73%,81%) |
| GILL | (73%,83%) | (76%,86%) | (75%,85%) | (80%,88%) | (74%,84%) | (79%,87%) | (78%,86%) |
| PROXY | (100%) | (100%) | (100%) | (100%) | (100%) | (100%) | (100%) |
| LEELAWADEE | (56%,68%) | (61%,73%) | (62%,73%) | (60%,72%) | (64%,74%) | (63%,73%) | (59%,69%) |

Table 8: Font Confidence Intervals

Looking at the intervals for the Century font, all of these intervals overlap, showing that in general all of the models did a relatively good job of classifying the Century font. For Ebrima, the H500 interval is completely disjoint from the H90 and H250 models, showing it is significantly better at classifying this font compared to those models. The intervals for Gill are also all overlapping, showing that the models did a relatively good job of classifying Gill. For Proxy, all of the models correctly classified this class. We believe this is due to the fact that this font is drastically different than the other fonts, being a thinner font as well as one with sharp edges opposed to rounded. And lastly, for Leelawadee, all of the intervals overlap showing that most of the models performed the same with classification of Class 5.

Below are the 95% confidence intervals for overall accuracy:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | H90 | H150 | H200 | H250 | H300 | H400 | H500 |
| Accuracy C.I. | (77.1%,81.1%) | (78.2%,82.1%) | (76.8%,80.7%) | (78.5%,83.4%) | (78.3%,82.2%) | (78.1%,82.0%) | (78.7%,82.5%) |

Table 9: Overall Accuracy Confidence Intervals

Since all of these intervals for the overall accuracy overlap, it shows the performance was not significantly different for any one model. However, just based off the overall accuracy, H500 had the highest accuracy of 80.7%. From Figure 16, we can see that the performances of the models are all relatively close.

# Conclusion and Further Suggestions

Despite getting an accuracy of 80.7% from the 5 classes, we are curious to see if we can improve on the performance of our CNN. To this end we had a couple of suggestions we could try to improve results. We first attempted deepening our network using additional hidden layers, we started with two hidden layers, h1 of 500 and h2 of 250, this gave us an accuracy of 81.32%. Below are the graphs obtained from this experiment:

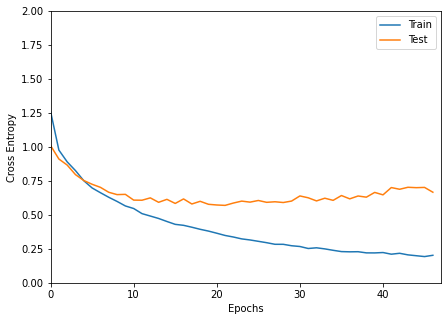


Figure 17: Cross Entropy H500-250

We used early stopping to prevent our model from overfitting, our model stopped at 47 epochs this means the lowest val loss was obtained on the 22 epoch.

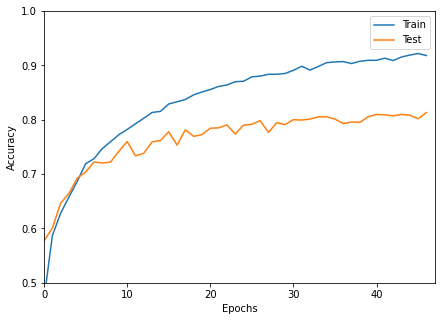


Figure 18: Accuracy H500-250

Our accuracy plots show that our accuracy continues to grow as the epochs increase, however since we wanted our model to be flexible early stopping stopped our model from getting much higher.

Since increasing layer size yielded improvements we continued to play along a bit more with the number of layers and layer sizes to see if we can get better results. Our findings are documented in the table below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | H1 | H2 | H3 | H4 | Batch Size | 3 of parameters | Epochs | Accuracy |
| 1 | 400 | 200 | NA | NA | 77 | 186,741 | 47 | 81.32% |
| 2 | 400 | 200 | NA | NA | 40 | 186,741 | 42 | 80.90% |
| 3 | 400 | 200 | 100 | NA | 40 | 206,341 | 44 | 79.70% |
| 4 | 500 | 300 | 100 | 50 | 40 | 316,941 | 41 | 80.83% |
| 5 | 500 | 250 | 125 | 65 | 25 | 296,381 | 44 | 79.58% |

We had hoped playing with these parameters and adding hidden layers would drastically improve the performance of our CNN, unfortunately it did not work the way we had hoped and despite the different models our accuracy seems to be maxing out around the 81% mark. We believe that this might be due to the similarities between few characters across the fonts. We wanted to continue to explore some other ways we could have improved upon the accuracy of our model. While we were not able to use the following techniques due to time limitations we have thought it prudent to mention them below to encourage future research.

As with all deep learning models we believe we can improve accuracy if we had more data, unfortunately with our fonts data set we don’t have a way to increase our cases but we can use Image Data Augmentation. Image augmentation parameters that are generally used to increase the data sample count are zoom, shear, rotation, preprocessing function and so on. Usage of these parameters results in generation of images having these attributes during training of Deep Learning models. Also since CNN is not rotation invariant, we can add images in the dataset using rotation.

Another technique we would have liked to try would have been to increase the resolution of our images. we would like to try to see if improved resolution would improve results.

Finally we realize that it is not very wise to rely on a single model. It would be better for us to try to use model ensemble techniques to devise a voting like structure to classify our cases better. We hope to use these methods as listed to arrive upon better results in the future.