COL774 Assignment 2.2

Rishit Jakharia, 2022CS11621

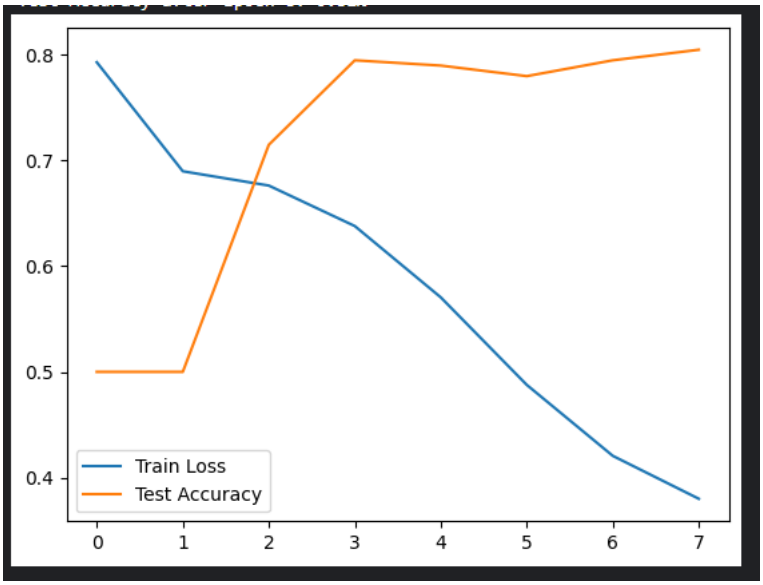
September 23, 2024

1 Part A: Binary Classification Using CNN

1.1 Accuracy on Public test set

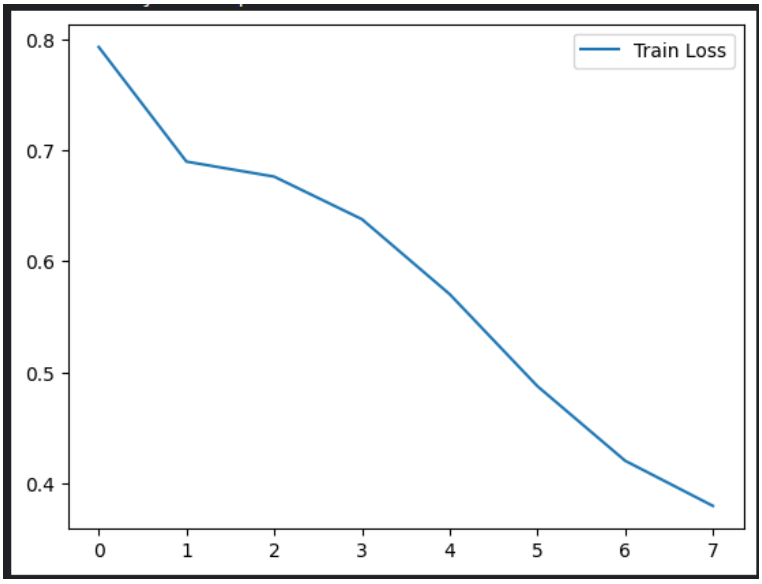
The accuracy on the public test set obtained was **80.5%**.

The training loss showed a steady decline across the 8 epochs, indicating effective learning by the model. Also, the rate of loss reduction did not slow down in the later epochs, suggesting that the model was not yet approaching convergence in 8 epochs.
Given below is the combined plot of train loss, and test accuracy.



1.2 Line plot of Training loss

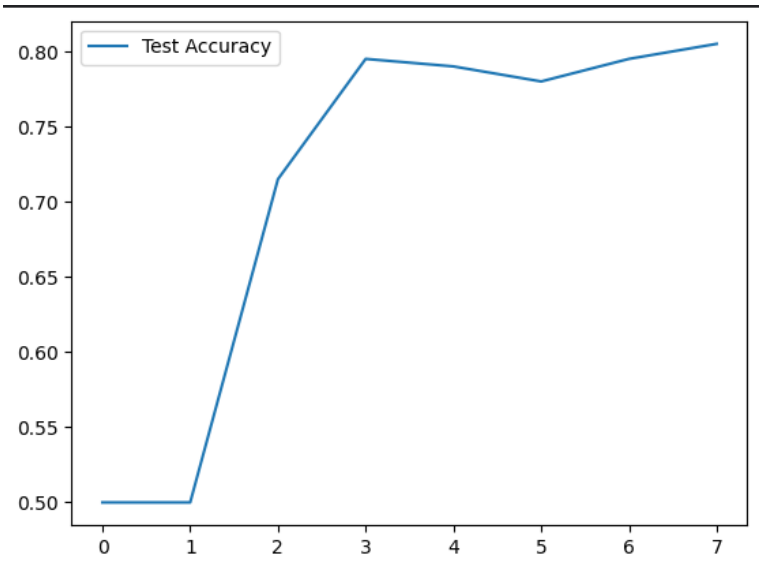
The Training loss per epoch graph was as below



As discussed above, we see a steady decrease, with no signs of convergence yet.

1.3 Public Test Data Accuracy Plot

The public test set accuracy vs epochs graph is as below.



1.4 Neural Networks vs CNN

Traditional neural networks, are generally less effective for image classification tasks compared to convolutional neural networks. As was observed by the 75% accuracy on the NNs and 80.5% accuracy on the CNNs.

In traditional NNs, each neuron in one layer is connected to every neuron in the next layer, leading to a **high number of parameters** when dealing with image data. This results in computational inefficiency and makes it harder for the network to capture spatial patterns in images, as all pixels are treated independently without considering their local relationships.

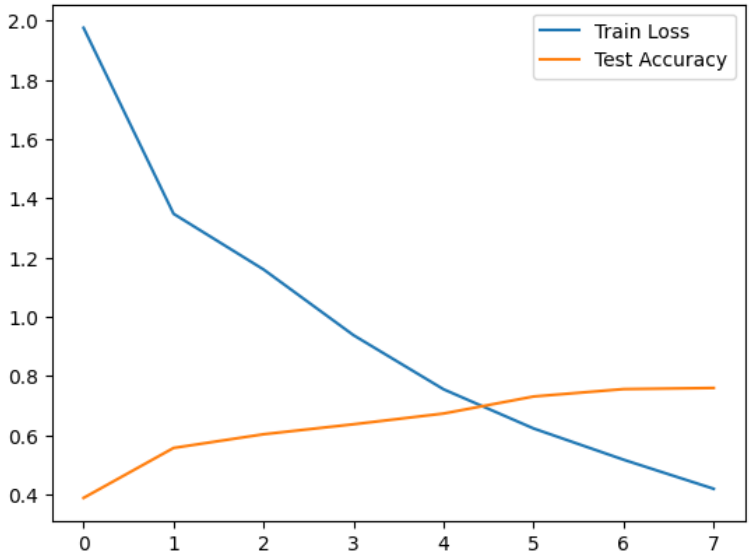
In contrast, CNNs are specifically designed for image-related tasks by leveraging convolutional layers, which apply filters to input images to detect spatial features like edges, textures, and shapes. These filters help reduce the number of parameters while retaining important spatial information. **Pooling layers in CNNs further reduce the dimensionality**, allowing for efficient learning of hierarchical patterns. As a result, CNNs outperform traditional NNs in image classification by providing better accuracy and generalization, particularly for large and complex image datasets.

2 Part B: Multi-Class Classification Using CNN

2.1 Accuracy on Public test set

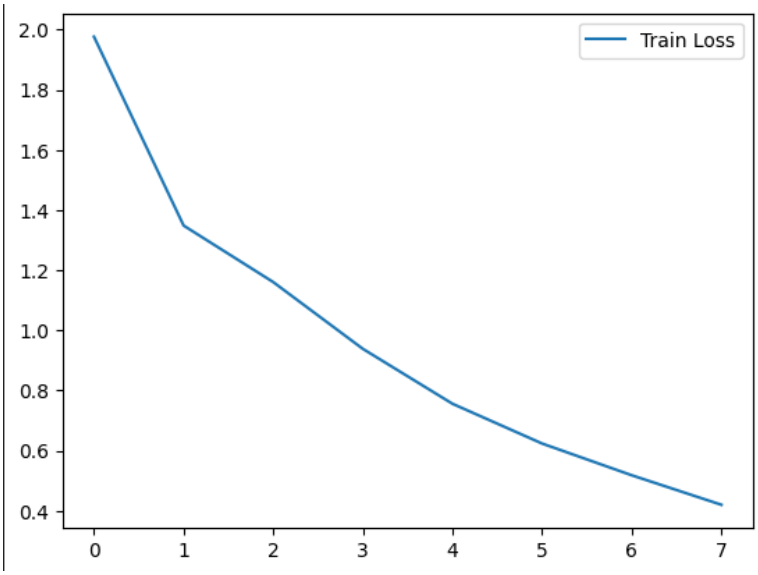
The accuracy on the public test set obtained was **76%**.

Similar to the above part, the training loss showed a steady decline across the 8 epochs, indicating effective learning by the model. Also, the rate of loss reduction did not slow down significantly in the later epochs, suggesting that the model was not yet approaching convergence in 8 epochs.
Given below is the combined plot of train loss, and test accuracy.



2.2 Line plot of Training loss

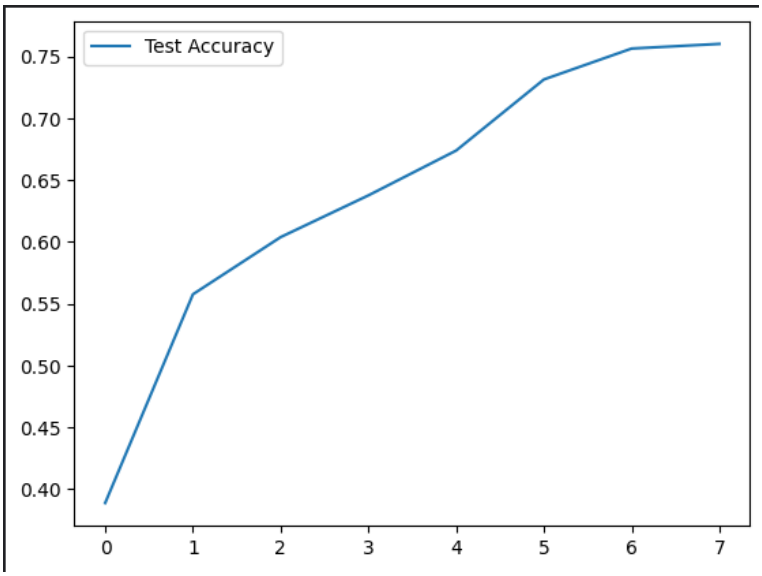
The Training loss per epoch graph was as below



As discussed above, we see a steady decrease, with little signs of convergence yet.

2.3 Public Test Data Accuracy Plot

The public test set accuracy vs epochs graph is as below.



2.4 Neural Networks vs CNN

Traditional neural networks, are generally less effective for image classification tasks compared to convolutional neural networks. As was observed by the 65% accuracy on the NNs and 76% accuracy on the CNNs.

Again this is likely due to the **high number of parameters** when dealing with image data slowing down the learning process. This results in computational inefficiency and makes it harder for the network to capture spatial patterns in images, as all pixels are treated independently without considering their local relationships.

3 Part C: CNN Model for Best Accuracy

This report details the development and experimentation of a Convolutional Neural Network (CNN) model for an 8-class handwritten language classification task. The model was trained and tested on a public dataset, and various architectures were explored to achieve the best performance. The final model was based on a residual convolutional architecture inspired by ResNet, which outperformed other baseline models in terms of accuracy and loss reduction.

3.1 Model Architecture

The best-performing model was a CNN model inspired by the ResNet architecture. The architecture can be summarized as follows:

- **Input:** Grayscale images (1 channel) of size 50×100.
- **Convolutional Layers:** The model consists of four residual blocks. Each block contains two convolutional layers with batch normalization and ReLU activations. The number of filters in each block increases progressively from 64 to 512.
- **Pooling:** A max pooling layer is applied after the first convolution layer, and adaptive average pooling is applied after the final residual block.
- **Fully Connected Layer:** The model ends with a fully connected layer mapping to 8 output classes.

This architecture was chosen because ResNet effectively mitigates the **vanishing gradient** problem by introducing identity mappings that allow gradients to propagate more efficiently during backpropagation.

3.2 Training and Feature Engineering

3.2.1 Batch Size

The model was trained using the Adam optimizer with a **learning rate of 0.001** and CrossEntropyLoss as the loss function. The training was performed for **16 epochs** with a **batch size of 128**.

Training with smaller batch sizes resulted in slower convergence and, upon reducing the number of epochs, turned into worse losses.

Training with larger batch sizes, resulted in worse losses, likely due to the loss being averaged out on a larger subset of train data, hence the errors were not getting propagated correctly, and getting averaged out.

3.2.2 Early Stopping

Tried the following epochs; **8, 16**
The model showed overfitting at **16**, and hence later, we tried **12**, which seemed to be the best fit.

3.2.3 Architecture

The Final Model Architecture was as shown below

Layer	Output Shape	Number of Parameters
Input (Grayscale Image)	$1 \times 50 \times 100$	0
Conv2D (7x7, stride=2)	$64 \times 25 \times 50$	3,200
BatchNorm2D	$64 \times 25 \times 50$	128
ReLU(inplace)	$64 \times 25 \times 50$	0
MaxPool2D (3x3, stride=2)	$64 \times 13 \times 25$	0
Conv2D (3x3, stride=1)	$64 \times 13 \times 25$	36,928
BatchNorm2D	$64 \times 13 \times 25$	128
ReLU(inplace)	$64 \times 13 \times 25$	0
Conv2D (3x3, stride=1)	$64 \times 13 \times 25$	36,928
BatchNorm2D	$64 \times 13 \times 25$	128
ReLU(inplace)	$64 \times 13 \times 25$	0
Conv2D (3x3, stride=2)	$128 \times 7 \times 13$	73,856
BatchNorm2D	$128 \times 7 \times 13$	256
ReLU(inplace)	$128 \times 7 \times 13$	0
Conv2D (3x3, stride=1)	$128 \times 7 \times 13$	147,584
BatchNorm2D	$128 \times 7 \times 13$	256
ReLU(inplace)	$128 \times 7 \times 13$	0
Conv2D (3x3, stride=2)	$256 \times 4 \times 7$	295,168
BatchNorm2D	$256 \times 4 \times 7$	512
ReLU(inplace)	$256 \times 4 \times 7$	0
Conv2D (3x3, stride=1)	$256 \times 4 \times 7$	590,080
BatchNorm2D	$256 \times 4 \times 7$	512
ReLU(inplace)	$256 \times 4 \times 7$	0
Conv2D (3x3, stride=2)	$512 \times 2 \times 4$	1,180,160
BatchNorm2D	$512 \times 2 \times 4$	1,024
ReLU(inplace)	$512 \times 2 \times 4$	0
Conv2D (3x3, stride=1)	$512 \times 2 \times 4$	2,359,808
BatchNorm2D	$512 \times 2 \times 4$	1,024
ReLU(inplace)	$512 \times 2 \times 4$	0
AdaptiveAvgPool2D (1x1)	$512 \times 1 \times 1$	0
Flatten	512	0
Linear ($512 \rightarrow 8$)	8	4,104
Total Parameters		**4,731,786**

Table 1: Architecture of the ResNet-based CNN Model with Number of Parameters

3.2.4 Learning Rate Scheduler

While not using a learning rate scheduler, the below was the test accuracy observed, clearly overfitting the data. Hence, a scheduler was implemented.

After the Implementation of the scheduler

```
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.1)
```

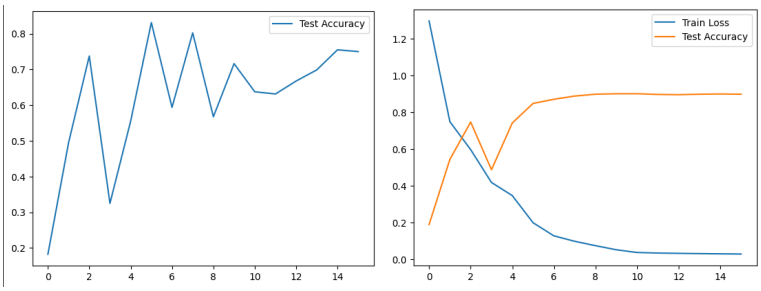


Figure 1: (Left) before Scheduler — (Right) after Scheduler

3.3 Experimentation and Results

3.3.1 Baseline Model: Simple CNN from part b

We initially experimented with a simple CNN architecture consisting of two convolutional layers followed by max pooling and a fully connected layer. This model achieved a training loss of **0.4168** after 8 epochs but failed to generalize well on the test set.

3.3.2 Larger CNN

Increasing the complexity of the CNN by adding more filters and layers resulted in overfitting. The training loss after 8 epochs was **0.6804** likely due to higher number of params resulting in slower convergence, and the model did not perform well on the test set.

3.3.3 AlexNet and VGGNet

We experimented with the AlexNet and VGGNet architectures. AlexNet performed better than the baseline CNN, achieving a training loss of **0.4063** after 8 epochs. However, VGGNet was computationally expensive and too slow for my PC.

3.3.4 ResNet

The ResNet-inspired model was the most successful, achieving a final training loss of **0.1796** after 8 epochs. Introducing bias terms and training for 16 epochs further improved performance, reducing the training loss to **0.0249** with early stopping **0.0389**.

3.4 Loss vs. Epochs

The following plot shows the loss values for the best-performing ResNet model over the course of 16 epochs.

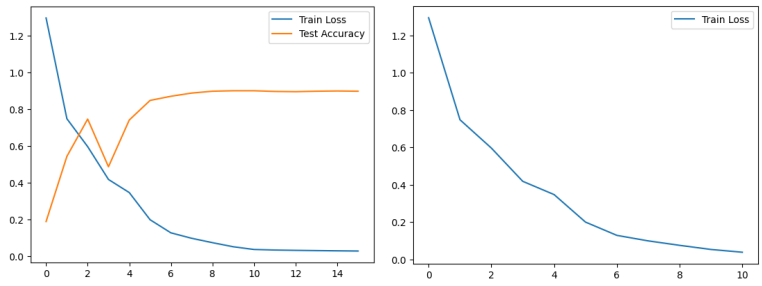


Figure 2: Training Loss and Test Acc. vs Epochs for the ResNet Model