Ahmad Al-Jabi, Davi Gava Bittencourt, Rayyan Khan

Final Training Results

Model Changes Since Deliverable 2

Since our second deliverable, we have made several updates to our model architecture and training process. These include:

- Implemented the NIST Special Database 19 dataset (as originally intended) instead of the MNIST dataset. NIST is a lot more applicable to real world applications as the data isn't as clean and clearcut, and it includes 62 different characters and there are over 810,000 data points (128x128 images) in the dataset, instead of the 70,000 28x28 in MNIST

- We utilized OpenCV to take a picture of a document of written text and are able to segment a word into separate letters, take 128x128 images of those letters (using some padding and other resizing tools) and then feed the images into the CNN to predict.

- With the use of a script, this would essentially destruct images then reconstruct the words into text

These changes were implemented to make the model more fit for unseen images and potentially less clear data, and also allow the project to work as intended.

Final Model Architecture

Our final architecture consists of:

- 2 convolutional layers (64 channels, 3x3 size, stride and padding of 1) with ReLU activation, followed by max pooling

- 4 linear layers:
    - self.fc1 = nn.Linear(64 * 32 * 32, 256)
    - self.fc2 = nn.Linear(256, 256)
    - self.fc3 = nn.Linear(256, 128)
    - self.fc4 = nn.Linear(128, 62)

- We didn't use a softmax layer: This is intentional: during training, we used PyTorch's CrossEntropyLoss, which internally applies softmax, and during inference, we rely on torch.argmax to select the class with the highest logit. Since softmax preserves the ranking of logits, its omission at inference time does not affect prediction results.

Ahmad Al-Jabi, Davi Gava Bittencourt, Rayyan Khan

We used an 70/20/10 split for training, validation and testing respectively. We ran grid search to find the values of the best learning rate, weight decay (for l2 regularization) and model archetype

Training Details

- **Loss function:** Cross-entropy
- **Optimizer:** Adam (5e-4)
- **Regularization:** 5e-4
- **Epochs:** 10
- **Batch size:** 256

Performance Evaluation

We evaluated the model using the following metrics:

- **Accuracy:** 89.76%
- **Train Loss:** (see Figure 1)
- **Test Loss:** (see Figure 2)

Our final model achieved a test accuracy of **89.76%**, which actually represented a decrease in accuracy compared to when we used less data but with MNIST. We here hoping for more testing accuracy, but struggled to see noticeable improvements even with tuning hyperparameters on the validation set and playing around with different overfitting prevention measures (dropout, l2 regularization, using more data, early stoppage, etc.)

Figure 1: Training Loss Over Epochs

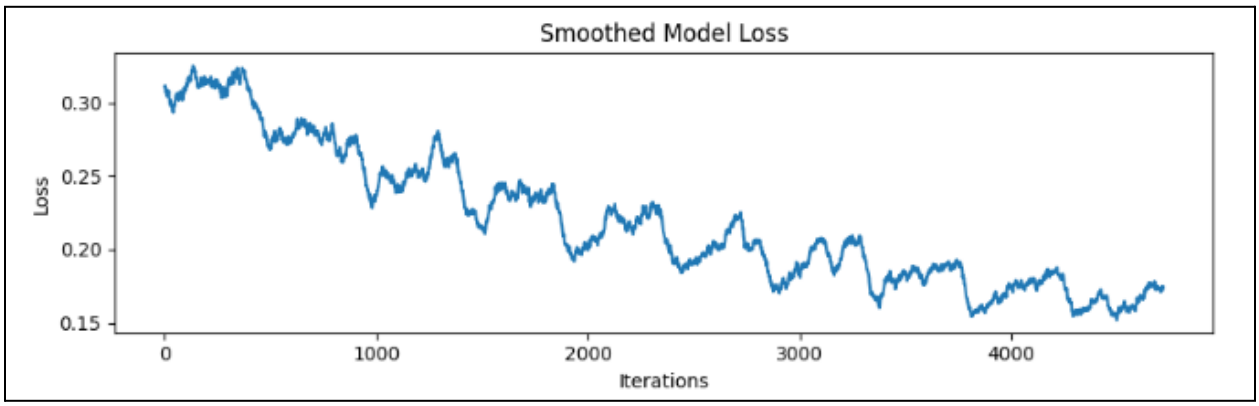
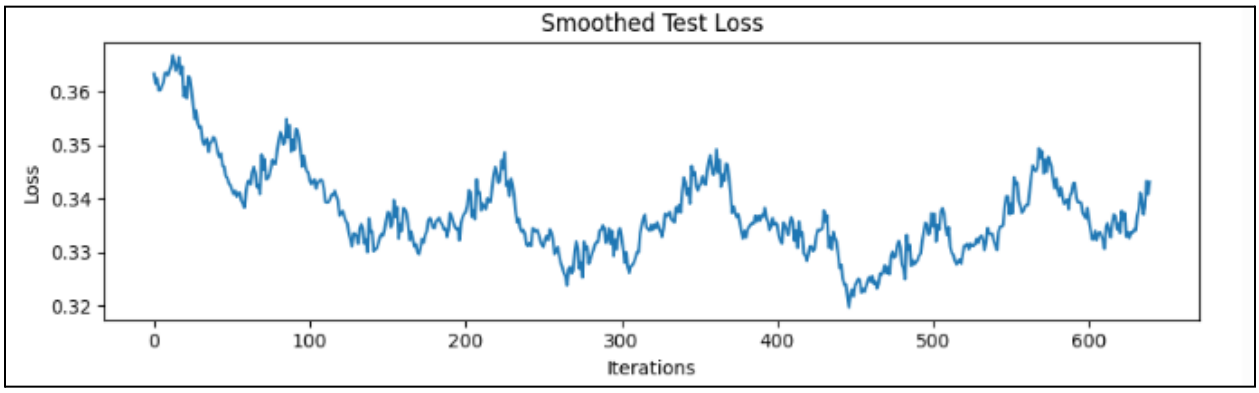


Figure 2: Test Loss



WEB APP:

We will be running a backend server and have a front-end done most likely in either flask or react which will allow us to either upload an image or draw it

We already have working python scripts that would automate the process of word segmentation, and it would only be a matter of hosting and unifying everything together.