Problem statement:

Our project aims to develop a text recognition system using convolutional neural networks (CNNs) to convert handwritten or printed text images into editable digital text. The model will process images through multiple CNN layers to separate words and letters, ultimately reconstructing them into sentences. The final application will allow users to upload images of text and receive a copyable text output.

Data Preprocessing:

We are currently using the MNIST dataset instead of the originally proposed NIST Special Database 19 or EMNIST. MNIST consists of 70,000 grayscale images (28x28 pixels) of handwritten digits (0-9), split into 60,000 training samples and 10,000 test samples. The switch to MNIST was only a temporary change for this deliverable due to its simplicity and well-structured nature, making it easier to prototype the CNN model before moving to more complex datasets (we have also had more experience with MNIST than NIST or EMNIST).

Machine Learning Model:

Initially, we proposed using a sequence of three CNNs to process text images by segmenting words, breaking them into letters, and reconstructing sentences. However, so far, for prototyping, we started with a single CNN trained on the MNIST dataset to recognize individual characters before scaling up to full text recognition. This decision was made to simplify debugging and ensure a solid foundation before incorporating additional complexity.

For the framework we implemented our model using PyTorch for flexibility and ease of use. We used torch, torchvision and matplotlib so far. Current architecture is two convolutional layers to extract features from images (64 filters of size 3x3 per layer, ReLU activation for non-linearity and max pooling (2x2) to downsample) and fully connected layers for classification (128 neurons - 128 neurons - 10 output neurons)

We used the standard MNIST split of 60,000 training and 10,000 test images, without a separate validation set, relying on test performance for evaluation for now. To prevent overfitting, we applied L2 regularization (weight decay = 5e-4) and used Adam optimizer (lr = 5e-4). We tested the model by tracking training loss. A challenge we faced was tuning hyperparameters to balance generalization and convergence speed, which we addressed by adjusting learning rates and regularization strength.

Our model's performance was assessed using cross-entropy loss and accuracy as evaluation metrics. While the training loss decreased steadily, we were unable to verify test loss trends. Without explicit validation, potential overfitting remains a concern. Moving forward, we will refine our approach by tracking test loss and accuracy, and possibly introducing dropout or data augmentation if overfitting is observed. Additionally, we will move forward with the initial plan of using the NIST dataset as opposed to MNIST once we get comfortable with the dataset.

Note: Apologies for not including graphs due to time constraints!