Notebook Overview: This notebook builds and trains a handwriting recognition model using both the IAMS public dataset and custom handwritten samples collected by having a sample.txt that individuals wrote on a piece of paper and then sent in images. Those were then converted to a jpg format and adding in the model to train. The goal is to train a deep learning model that can read and decode handwritten lines of text from images.

Steps in Code:

Dataset Preparation

The IAM dataset was used. It has thousands of English handwriting samples and combined it with our own scanned handwritten images and transcriptions. Images were resized and normalized. Texts were converted into sequences of characters using a tokenizer. All text sequences were padded to a uniform length and merged into one dataset with the IAM data.

Model Building

We built a CNN + Bidirectional LSTM model with a CTC loss, which is ideal for recognizing unsegmented handwritten text. The model processes each image and learns to predict sequences of characters from left to right.

Model Testing

The code trained two models: an earlier version with a simpler CNN-LSTM setup and a newer, improved model with better architecture, more training time, and prediction callbacks. The earlier model (at the bottom of the notebook) was a basic attempt to validate the data pipeline. It showed weaker prediction performance. The final model was trained for up to 60 epochs with callbacks for early stopping and learning rate adjustment. Each epoch printed a sample prediction to track visual progress. Training/validation loss showed improvement. A few validation samples were used and visualized predictions vs ground truth to check model behavior. The newer model showed improving predictions with lower validation loss over time. This is expected because of our delays with a third group member, early predictions are not going to be the best accuracy. We can explain the limitations and next steps in our presentation though not worried ☺.

To do next!

Evaluate performance using Character Error Rate CER and Word Error Rate WER