Deep Learning Assignment Part 1

Pragya Khanna 2022122003

1 Methodology and Rationale

1.1 Preprocessing

1. Normalization: The pixel values were normalized to the range [0, 1] by dividing by 255 to improve model convergence.
2. Reshaping: Since the images are grayscale, an additional channel dimension was added (X = np.expand_dims(X, axis=-1)) to make the input compatible with TensorFlow's CNN layers.
3. One-Hot Encoding: The labels were converted to categorical format using tf.keras.utils.to_categorical for multi-class classification.
4. Dataset Splitting: The dataset was split into training (60%), validation (20%), and test (20%) sets using train_test_split. This ensures a robust evaluation of the model's performance on unseen data.

1.2 Model Architecture

1. Input Layer:

   Accepts images with dimensions (40, 168, 1).
2. Convolutional Layers: (extract spatial features effectively)

   Three convolutional layers with increasing filters (32, 64, 128) to extract features of varying complexity. Each convolutional layer is followed by Batch Normalization (for stability) and Max Pooling (to reduce spatial dimensions).
3. Fully Connected Layers:

   Flattened feature maps are passed through a dense layer with 256 neurons. Batch Normalization is applied for stability, followed by a Dropout layer to reduce overfitting. The final output layer has 37 neurons with a softmax activation for multi-class classification.
4. Dropout Regularization:

   Dropout layers with a 50% rate are added after dense layers to prevent overfitting by disabling neurons randomly during training.

1.3 Training

1. Loss Function:
   The categorical cross-entropy loss function was used since the problem involves multi-class classification.
2. Optimizer:

   Adam optimizer was chosen for its adaptive learning rate and computational efficiency.
3. Batch Size and Epochs:

   A batch size of 32 was selected to balance memory usage and training speed. The model was trained for 100 epochs, with early stopping implemented to prevent overfitting.

2 Inferences

Accuracies at the end of training: Validation = 14.2% and Training = 83.6%

- The significant gap between training and validation accuracy suggests that the model is likely overfitting to the training data and failing to generalize well to unseen data.
- The increasing validation loss combined with stagnant validation accuracy further supports the hypothesis that the model is overfitting. The model is failing to generalize, as it memorizes training patterns instead of learning robust features.

3 Further Work

Low accuracies and my basic intuition (that the original model was primarily learning pixel-level features without comprehending the numerical relationships in the dataset) point that this is not sufficient, hence I modified my approach. The revised pipeline includes explicitly finding bounding boxes for the numbers, followed by prediction using model trained (from scratch) on MNIST digit dataset to find the constituent numbers and then summing them to find the final target value. I used gaussian smoothening (to tackle

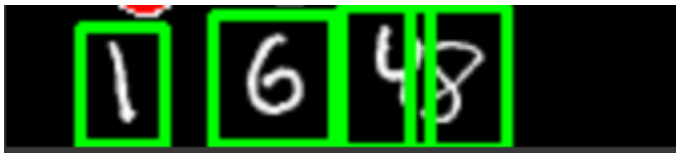

Figure 1: Sample of the digits recognised

digits connected to eachother since handwritten) followed by finding contour maps. The issue encountered here was multiple bounding boxes overlapping, to resolve which, I employed a heuristic to select the four bounding boxes with minimal overlap, ensuring each box corresponded to a distinct digit. The MNIST data was trained on a basic CNN model.

I would like to elaborate more on this approach in the second part.