Deep Learning Assignment

1 Data Cleaning and Preprocessing

The following preprocessing steps were performed:

- All the data given was loaded and concatenated.
- Upon inspection, each image contains 4 digits which are non-overlapping, have 1 channel (grayscale) and have size of 40×168 pixels.
- Created a dataloader and split it into training validation and test sets in the ratio 80:10:10 and batch size 32.

2 Phase 1: CNN Baselines

- Since a maximum of 4 digits are contained in any image, this was treated as a classification problem with 37 classes corresponding to the possible sums.
- Cross Entropy loss was used for this.
- Used a basic CNN Architecture with two convolution and pooling layers. Implemented early stopping based on the validation loss.
- Got an accuracy of ∼10%
- Weights for this model are stored in `best_model.pth`
- With a similar approach, instead of defining the architecture, a ResNet model was also tested. However, since pretrained ResNets have 3 channels (RGB) and use very different data, training the architecture from scratch on the given data seemed to be a better option.
- This performs better with an accuracy of about $\sim 50\%$
- Weights for this are stored in `best_resnet.pth`

3 Phase 2: Complex Architectures

- In this phase, a variety of architectures were tested for the task. Explanations of architectures that were tried out or not considered have been given below.

- Object Detection/Image Segmentation models - Such as YOLO were considered. Combining these with a simple regression model. However, this was not feasible because the data was not annotated.

- Sliding Window CNN - This works like a simple CNN but it utilizes a fixed size sliding window to scan through the image and identify the different digits. This achieved an accuracy of 15% with some hyperparameter tuning. The results are underwhelming because different digits have variable sizes and might even be placed on top of each other or have minor overlap causing these problems.

- Pretrained MNIST classifier - This was considered but again it did not work very well because in some data samples there was an overlap found.

- SWIN Transformer - This is similar to a UNET architecture combined with a vision transformer. Deeper layers correspond to better semantic segmentation of images. Training such an architecture from scratch was infeasible because of compute constraints. The pretrained SWIN transformers present in the OpenCV library required 224 x 224 size images so there was an additional data preprocessing step. However, there was not enough data when this architecture was trained. It caused massive overfitting leading to terrible performance on the test data 6 %

- ResNets with Hyperparameter Tuning - Finally, with some hyperparameter tuning, the resnets started giving extremely good results. Instead of training these architectures from scratch, the pretrained models were used. These have been trained on ImageNet etc. and hence had significantly better performance (93 percent with optimal hyperparameters for resnet34 and 89 percent for optimal hyperparameters with resnet101). There were random jumps in the training process, however after accounting for this by checkpointing the best models these gave the optimal performance. The labels that tend to get misclassi