Report

Final submission

Overview:

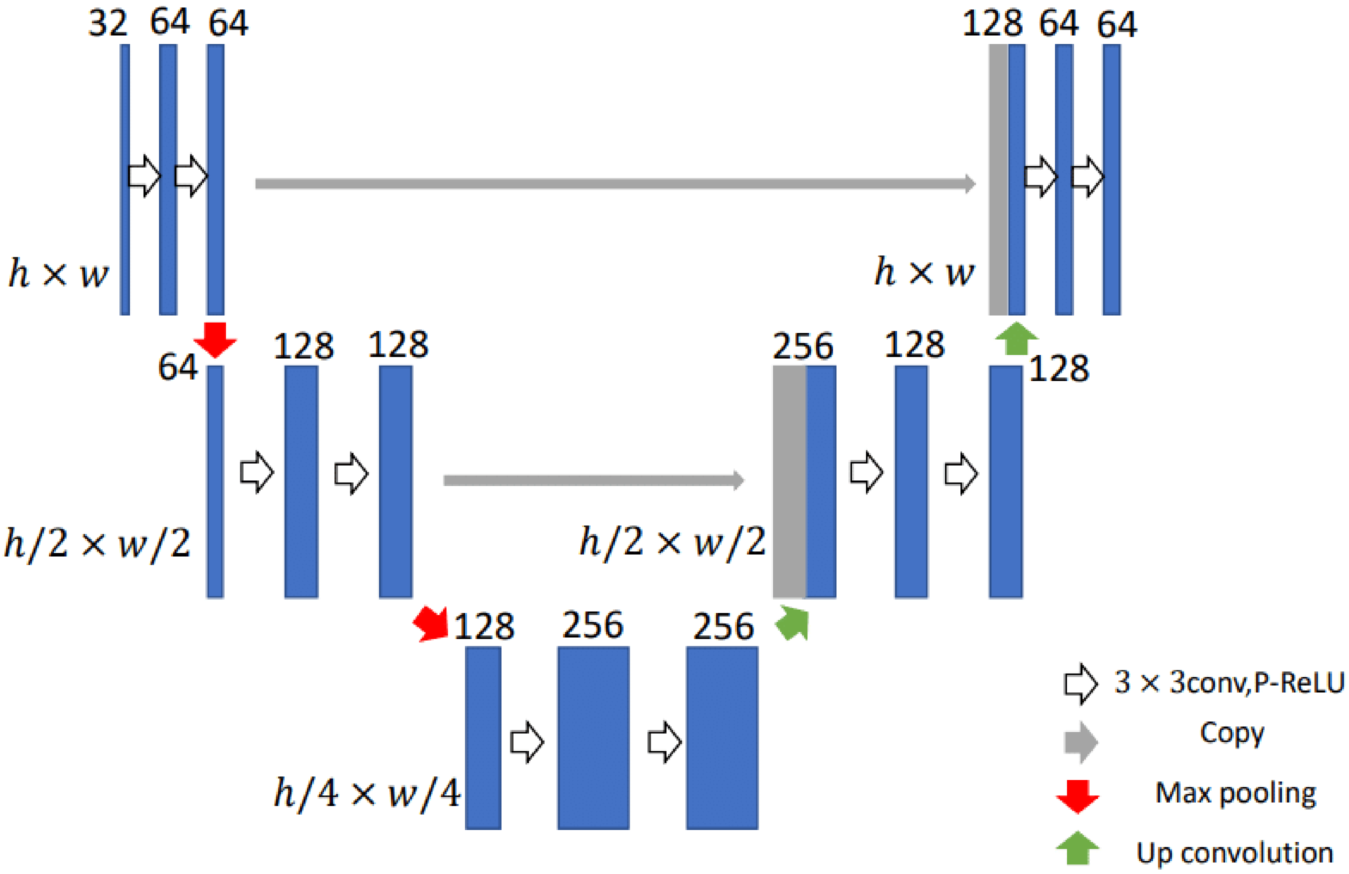
Tried 2 different strategies for digit sum task. Both of them were image segmentation strategies. First strategy, was to train a U-net, for semantic segmentation of the image. Based on this labelling of pixels, and the number of pixels per image, the model classifies whether a digit is there within that region or not.

The second strategy was to train an object detection model, that also outputs the digit type, then pass the image through the model, see the digits detected in that model, and then add those up.

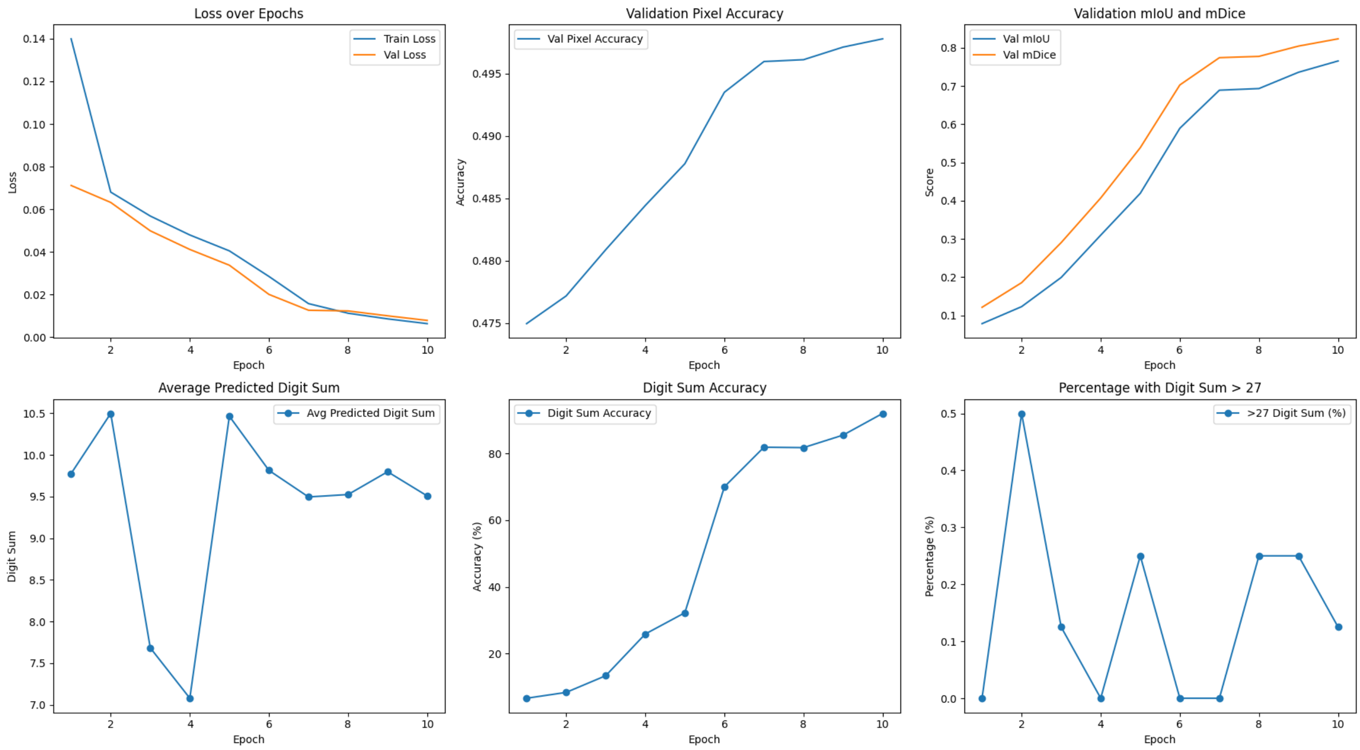
Unet Data processing

To train this U-net, i trained it on a dataset,(M2NIST) consisting of images, with multiple digits, and 11 binary masks, the 0th filter having ones for corresponding to pixels that belonged to the handwritten digit 0, and so on. This essentially is a pixel-wise classification task.

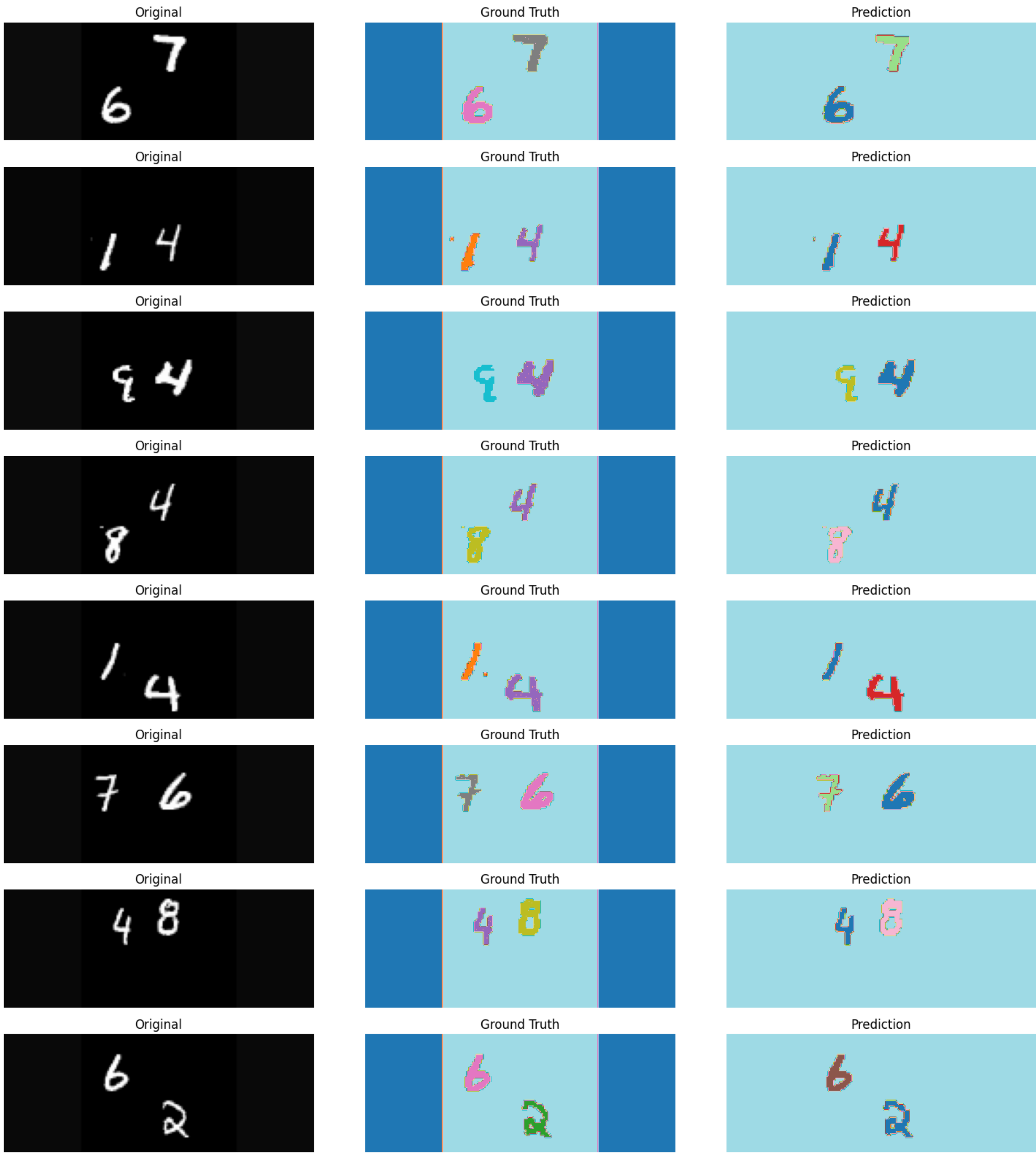
Architecture



Since this is essentially a pixelwise classification task, i used cross-entropy loss for optimization, over 10 epochs. The loss graphs looked like such. For the validation set, this model had obtained an accuracy of around 92%.



This image shows the predicted pixels, while displaying the class each pixel corresponded to. On M2Nist, the model had learnt in order to semantically segment the image into different classes.



Since the provided dataset for the assignment had no such semantic labels, i was only able to test the model on the assignment dataset. For digit identification, i had looked at different nonzero-regions on the image, and had seen if the digit's pixel density was greater than a threshold i had obtained from eda (40 pixels). With this, on the validation dataset, from M2NIST, i had obtained a 92% accuracy, and on the assignment dataset, I had obtained a 75% accuracy. The reason I think this happened, was simply because of how digits were written in the assignment dataset, and how they were different from the M2NIST dataset, and how they intersected, due to which the U-net, had misclassified certain digits.

An example of a mistake was the below image:

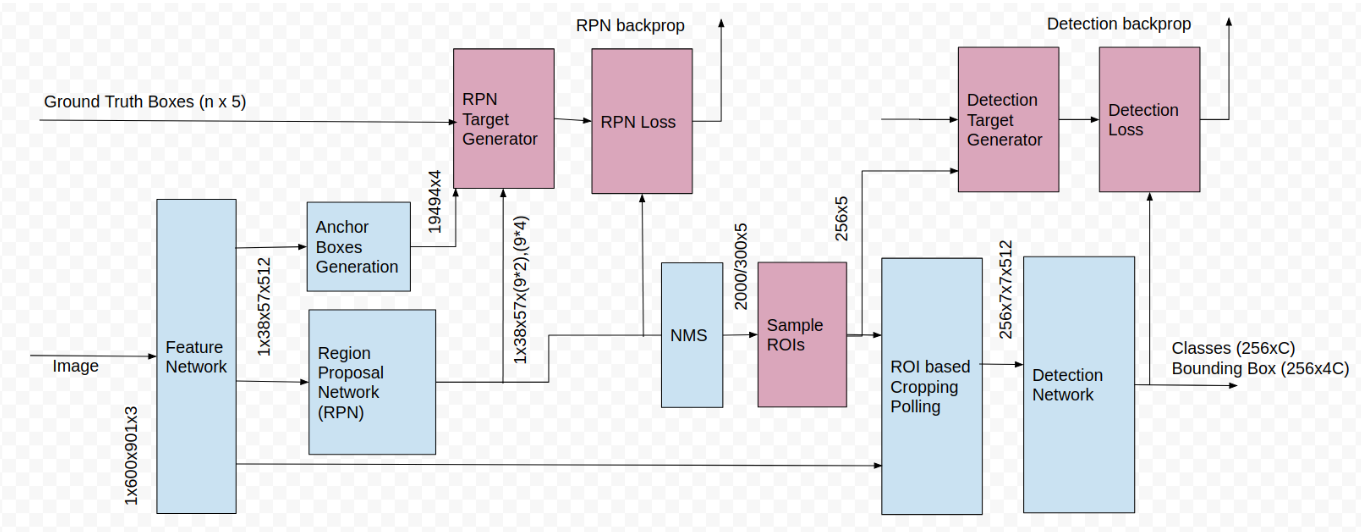
Here the 5, was misclassified as a 9.

Now my second strategy, was to train an object recognition model, to recognize images

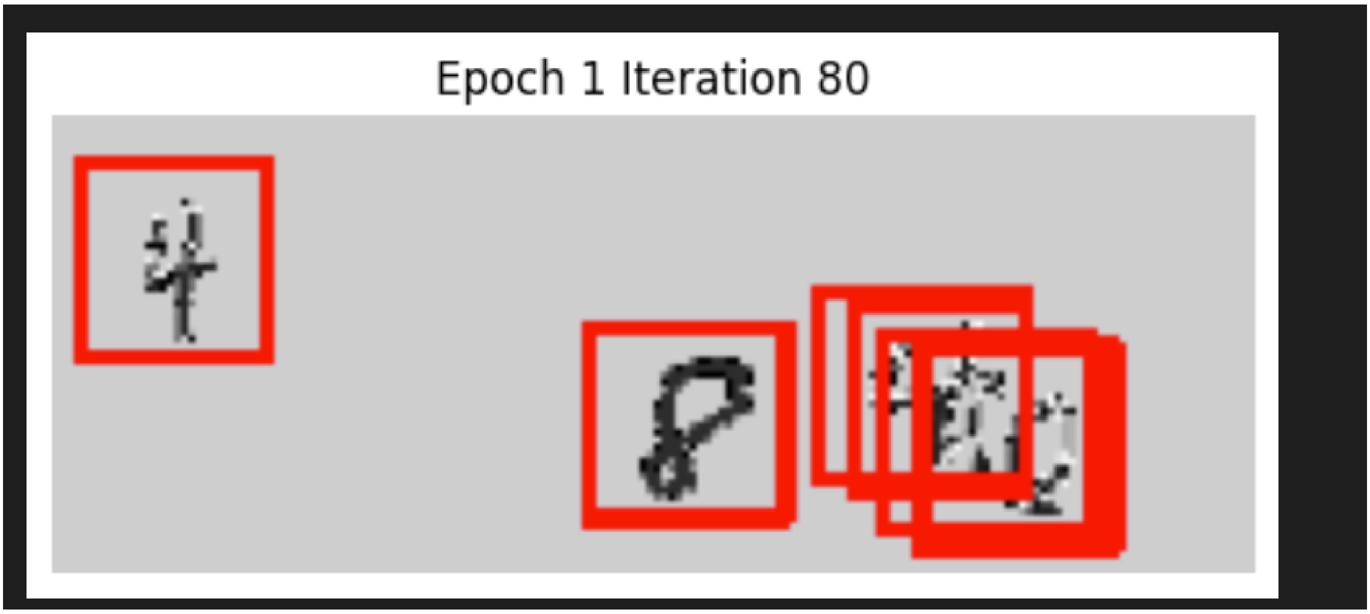
Faster-RCNN

Since in order to train such an object recognition model, one would need two things for an image with multiple digits in them, a set of bounding boxes as to where each digit was, and a label to each bounding box. Since this wasnt available for the provided dataset, I constructed it myself. I took MNIST single digit handwritten images, and had essentially constructed a dataset, by randomly placing them in bounding boxes of the size of the image, and by then copying them into the image, with some thresholds for overlap. From this , i now had a dataset, of images consisting of multiple digits, with the bounding box for each digit, and the label of the digit within the bounding box. I then trained a faster_rcnn using these.

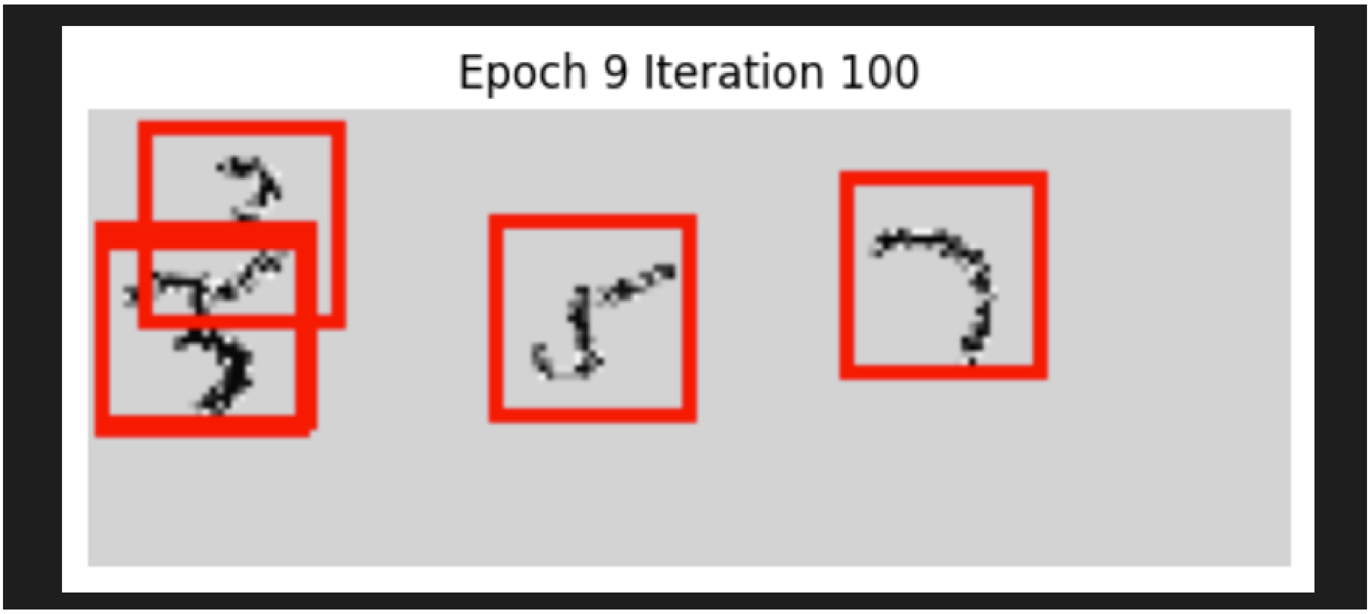
Architecture



During the first few epochs of training, the model struggled within bounding boxes, of nearby digits.

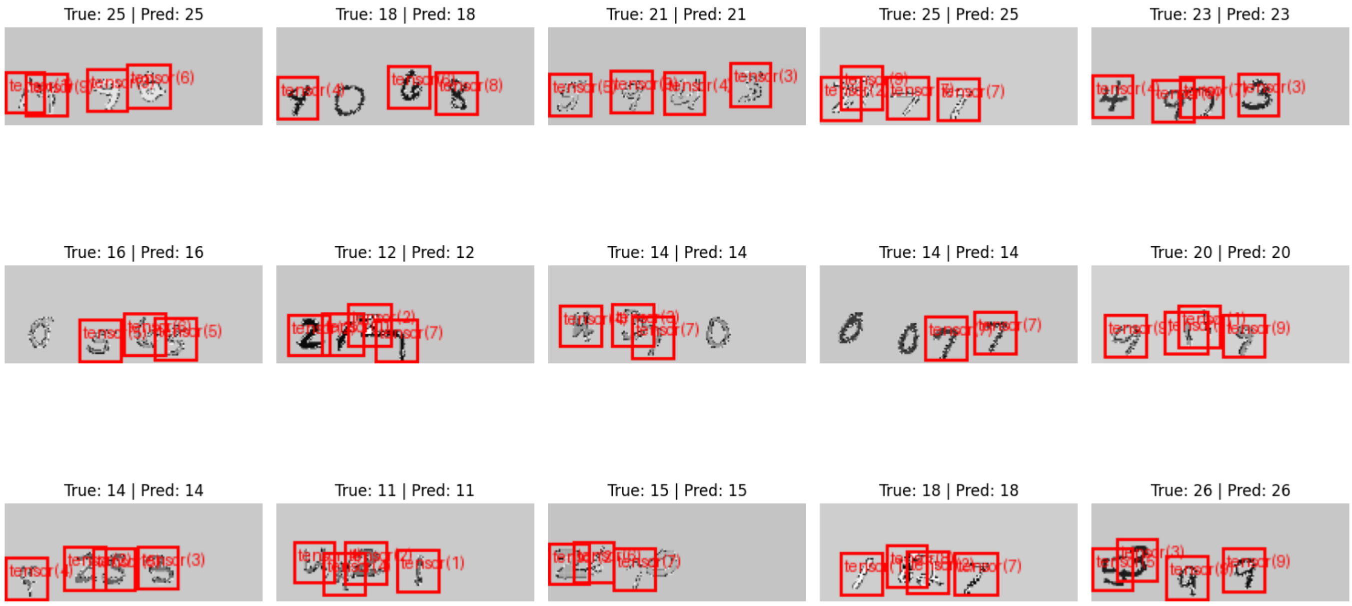


As time passed however, the model was able to output, more precise bounding boxes.

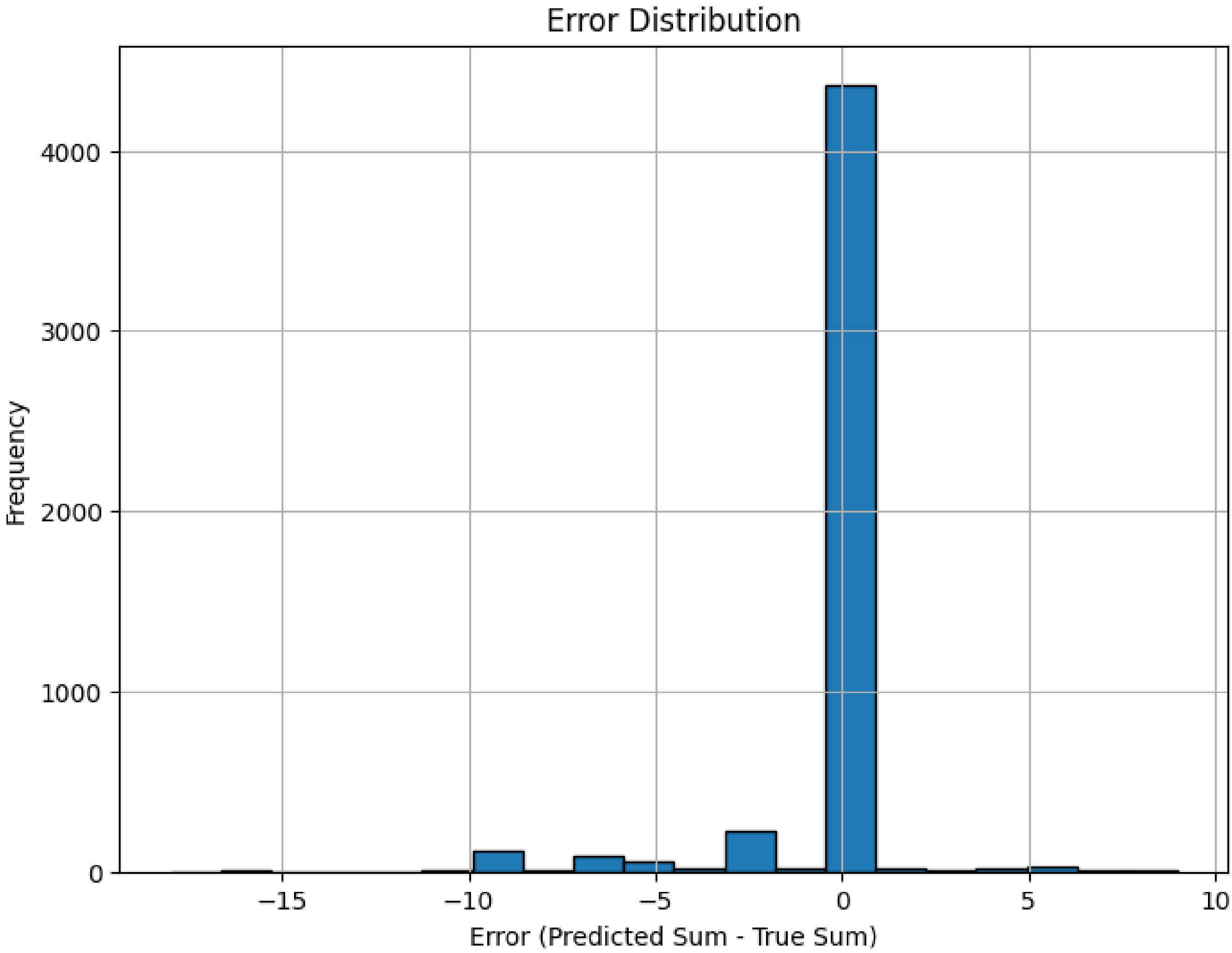


To obtain a sum, i pass an image through the model, see the bounding boxes with scores greater than a threshold, and the digit within the bounding box, corresponds to the label assigned to said bounding box. For the assignment dataset, I had obtained a validation accuracy of 89%.

This is how the model performed on the assignment dataset, here i plotted the bounding box obtained, and the digit label of the bounding box, and compared the true sum vs the predicted sum. Here i plotted bounding boxes for digits with values greater than 0.



Below is a histogram of error distribution, most of the times, the model has zero error(around 90%).



These were the statistics of the test set(assignment dataset):

Overall Accuracy: 0.87 Mean Error: -0.53 Standard Deviation of Error: 2.06