Assignment 4 Report

Implementation

1. Loading the data:
   The data was loaded into the colab notebook directly from Google Drive, after which it was copied to the local storage of the virtual machine. In order to simplify accessing the data from the local storage (for the augmentation phase), the images were organized into folders based on their labels/classes.
2. Pre-Processing:
   Upon inspecting the images, we saw that the pixels that made up the numbers were darker (closer to 255) while those that made up the background were significantly lighter (closer to 0). We utilized this observation and performed binary image thresholding on all the images, setting every pixel with a value of over 240 to a value of 255 and the rest to a value of 0. This is shown below in Figure 1.

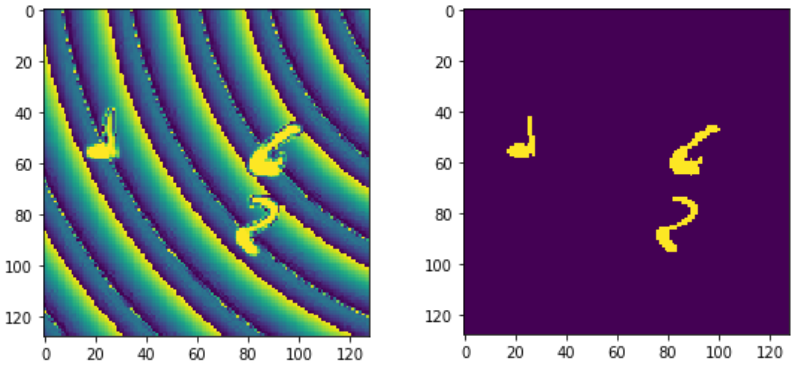

Figure 1: *Image before thresholding (left) and after thresholding (right)*

3. Data Augmentation:
   As an extension of the pre-processing, each image was rotated by a random degree between -20º and 20º as well randomly shifted vertically and horizontally by 1 pixel. This was done to improve the model's robustness by increasing the size of the dataset.
4. Constructing & Training:
   The Xception model architecture was imported from keras and adapted to the task (adjusting the input shape and number of classes). The Xception model was chosen because a member of our team had positive experiences with it when using it for his final project. The model was compiled using an Adam optimizer and categorical crossentropy for the loss calculation method, training was completed over 25 epochs.

Results

Our best model performed with a score of 0.97057 on the test data and our next best model performed with a score of 0.95828. Our best model was trained on an augmented version of the original dataset where we allowed the images to be rotated by up to 15 degrees and shifted 5 pixels to the left and right. This improved on our original models score by about 0.015 accuracy. The model was trained on 20 epochs as we had seen that after 20 epochs the models began to overfit on the training data and the validation accuracy of the model decreased towards 30 epochs.

Challenges

Our main challenge that we faced was getting the accuracy above 0.94. To overcome this challenge we used data augmentation to increase our dataset size and train on a wider variety of images. Another challenge was we kept running out of ram as we loaded the whole dataset into the colab notebook as a numpy array and then tried to train our model. To avoid keeping the whole array in the notebook, we stored each individual image into a folder with the class that the image belonged to and then used the keras flow_from_directory which creates an augmented dataset from a file directory. This allowed us to keep from using too much ram as well as augment the data.

Conclusion

The three big takeaways from this assignment were:

1. Witnessing how data augmentation can help improve model performance. Through applying rotation and translation transformations to the images, we were able to synthetically expand our data set. This reduced overfitting and, in turn, improved the model's accuracy.
2. Learning how to choose the best tool to solve the given ML problem. CNNs are generally dedicated to ML problems with features of the image type. With this in mind, existing CNN-based model architectures and pre-trained weights were experimented with. We found that the Xception model architecture (loaded through keras) was ideal for solving the challenge.
3. Learning techniques for reducing RAM consumption when using Colab. We learned that loading the data set into the local storage improved performance and helped reduce RAM consumption.

Link to Repository:

https://github.com/eyasyasir/Modified-MNIST-Digit-Classifier

Individual Contribution

Joseph:

- Wrote code for preprocessing data
- Helped optimize the hyperparameters of our model and train the model for submission
- Wrote Challenges and results section

Javier:

- Wrote code for testing various models and for data augmentation.
- Experimented with pretrained Retinanet digit recognition model.
- Trained the Xception model and optimized the hyperparameters.
- Edited implementation and conclusion sections.

Eyas:

- Experimented with yolov3 and MSER algorithms for bounding box approach to detect digits
- Created repo, readme instructions for training and testing the model
- Wrote implementation and conclusion sections