MAIS 202 Kaggle Competition : Modified MNIST Challenge

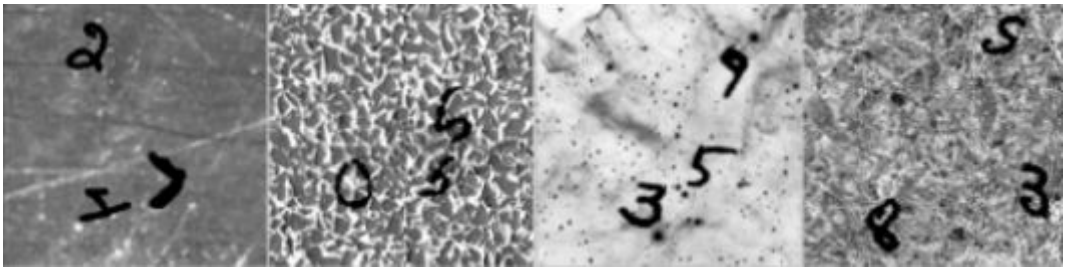
Team member: Mia Tran, Ricky Chen, Peter Guanhua Rong

1. INTRODUCTION

In this assignment, our goal is to conduct classification on a modified MNIST like dataset. The goal of this assignment is to identify the largest numerical value ranging from 0-9 from an image containing 3 random numbers. Our algorithm is a Convolution Neural Network with 5 layers. After several rounds of fine tuning, we were able to accomplish an accuracy rate of 97.9% on the test data set.

2. DATA SET

An example of the data set is attached below



From careful analysis, we observed that each entry of the dataset is an 128x128 array with elements ranging between 0 and 255 representing the intensity of that pixel. Therefore we assume that the images are automatically grayscale. We also observed that the data entries all have a decent amount of background noise. In addition, the numerical digits are extremely similar to the standard MNIST digits.

3. APPROACH

a. Preprocessing

We have previously implemented image denoising using an intensity filter and a median filter, to retrieve filtered images consisting of the digits only. This has eventually proven to be useless, because we achieved higher accuracy without this step. The second step is additional image data generation. By Implementing a data generator from the keras library, we were able to generate additional data using the same dataset, with tweaked features such as image rotation, image zooming, image shifting and shearing. As a side note, we made the important observation that image rotation and flipping were strictly forbidden in this specific dataset, because any rotation bigger than 45 degrees or image flipping will turn the number “6” into “9”. For moderation, we picked a 20 degrees rotation range.

b. Implementation of model

Since we are dealing with images, we figure that it is in our best interest to use Convolutional Neural Network (CNN). We refer to some

resources online and decide to go with the adjusted version of VGG16 architecture. To avoid overfitting and to improve our model, we batch normalized after each convolutional step and add dropout layers as much as possible. We have tried with several optimizers but Adam optimizer stands out because it is the most stable and has the best performance. Modified MNIST is a multi-class classification task so we stick with categorical cross-entropy loss function.
Since we only trained with a smaller learning rate of 1e-4 to ensure convergence. We then need to speed things up and to do that we have changed the call back in our function so that it reduces the learning rate by 10% every epoch.
Before we came up with the final model, we have tried different rotations for the Image Data Generator and also tried different learning rate, dropout rate with different number of epochs to decide which are the best parameters.

c. Result

After performing numerous trials, we achieve a 98% prediction rate on the test set with pre denoise mode (refer to pre_denoise colab)l. With the post denoise model, we are able to reach a 97% prediction rate (refer to report colab). We perform grid search on post denoise to find the best parameters; the best learning rate is 1e-4 with epochs of 41 and 0.2 dropout rate.

d. Challenges

At first, we have trouble to denoise the dataset since the value of the images is given as intensity value, but we manage to convert the intensity value to only 0 and 255 and then utilizes median filter from the external library scipy to denoise the white dots in the image (To see the process, please refer to the screenshots below). Furthermore, we find out that training without any external resources will take a lot of time, so we use the limited GPU provided by Google Colab. After we get our first CNN model done and to avoid overfitting as much as possible, we figure that we should use grid search to find the best parameters in order to have the best result; however, it is time consuming and will surpass the limit of GPU, so we only perform grid search on epochs, dropout rate and learning rate.

4. CONCLUSION

a. Pre- denoise result:

- For the pre-denosie, we have reached the accuracy of 98% with using our CNN model with parameters of epoch = 55, learning rate = 1e-4, dropout rate = 0.2

b. Post denoise result:

MAIS 202 Kaggle Competition : Modified MNIST Challenge

- We have denoise our dataset using the following code:

```
for train_pointer in range(len(train_images)):
    train_images[train_pointer] = data_process(train_images[train_pointer])
for test_pointer in range(len(x_test)):
    x_test[test_pointer] = data_process(x_test[test_pointer])
```

```
from google.colab.patches import cv2_imshow
from scipy import ndimage
def data_process(image):
    processed = (image > 210) * 255
    #cv2_imshow(processed)
    processed = ndimage.median_filter(processed, size=2)
    #cv2_imshow(processed)
    processed = (processed - np.mean(processed) / processed.std())
    #cv2_imshow(processed)
    return processed
```

- The accuracy of our post-denosie model is lower, at about 97% and we think that the reason for this is our CNN models have captured features well enough and by denoising, we have deleted some important features. Also the denoise might cause overfitting for our dataset cause our model is learning useless features which filtered by median filter. However with the pre-denoise, the model has learnt the original features without median filter.

5. INDIVIDUAL CONTRIBUTION

- **Mia:** Contributed to choosing the best CNN architecture and also came up with the callback and optimizers with a learning rate of 1e-4.
  Worked with Peter on the Grid Search CV to determine the best parameters for our CNN model.
  Trained the model with different choices of parameters.
  Submitted the prediction csv file with the first 97.9% of the pre-denoise model and 96.8% for the post-denoise.
- **Ricky:** Convert intensity pixel value to black and white image, perform median filter on image to denoise, construct the modified VGG16 architecture of CNN, train the model with several other parameters, put predicted values in the csv file for submission.
- **Peter:** Implemented Data Image generator, performed Gridsearch and model training. Performed several bug fixings. Unfortunately a lot of algorithms that I have implemented were not chosen for the final model, examples being MNIST digit extraction, MNIST digit filtering, and generation.