MIDAS@IIITD

Multimodal Digital Media Analysis Lab

MIDAS Summer Internship 2021
Task 2

A report by Dhruv Patidar

The task:

1. Use this dataset (https://www.dropbox.com/s/pan6mutc5xj5kj0/trainPart1.zip) to train a CNN. Use no other data source or pretrained networks, and explain your design choices during preprocessing, model building and training. Also, cite the sources you used to borrow techniques. A test set will be provided later to judge the performance of your classifier. Please save your model checkpoints.

2. Next, select only 0-9 training images from the above dataset, and use the pretrained network to train on MNIST dataset. Use the standard MNIST train and test splits (http://yann.lecun.com/exdb/mnist/). How does this pretrained network perform in comparison to a randomly initialized network in terms of convergence time, final accuracy and other possible training quality metrics? Do a thorough analysis. Please save your model checkpoints.

3. Finally, take the following dataset (https://www.dropbox.com/s/otc12z2w7f7xm8z/mnistTask3.zip), train on this dataset and provide test accuracy on the MNIST test set, using the same test split from part 2. Train using scratch random initialization and using the pretrained network part 1. Do the same analysis as 2 and report what happens this time. Try and do qualitative analysis of what's different in this dataset. Please save your model checkpoints.

Part 1

The images in dataset 1 were 900 * 1200 * 3. I have downsampled it to 28 * 28. This decision was made to:

a. Make processing computationally cheaper.
b. For convenience, as other datasets had 28 * 28 as their dimensions.

After downsampling to 28 * 28, I have flattened the images and stored as a .csv file. All of the above is implemented in 'Preprocess_Dataset_1.ipynb'.

In the notebook 'CNN_Dataset1.ipynb', I have trained the CNN on dataset 1 and achieved the following:

```
Epoch 00027: ReduceLROnPlateau reducing learning rate to 3.1250001484295535e-05.
Epoch 28/30
69/69 - 68s - loss: 0.3266 - accuracy: 0.8827 - val_loss: 0.3824 - val_accuracy: 0.8669
```

Then in the same notebook, I had a look at the mistakes the NN makes. The performance was near to that of a human.

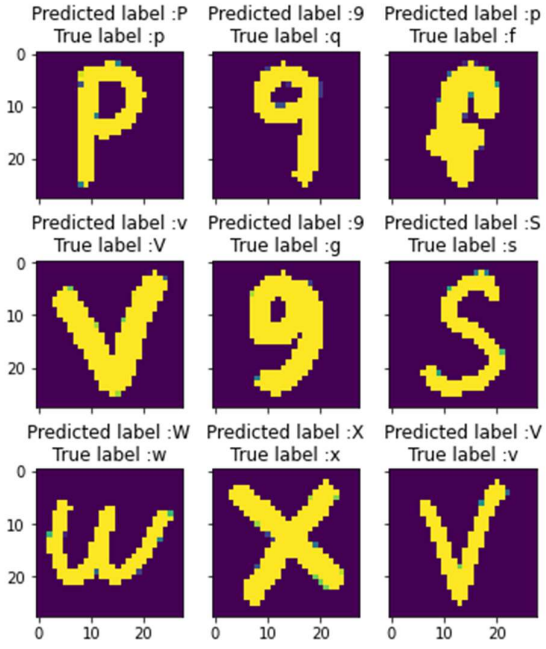

Fig: Some mistakes made by the CNN

In the figure at the left, only the top-right 'f' can be considered as a mistake, all the others are understandable.

If we have to label images from paragraphs, then we can make use of the context and be much more accurate.

I have analysed the mistakes in depth in 'CNN_Dataset1.ipynb'.

Part 2

In this part, I have experimented with the third-last and the second-last layers of the CNN. Originally it was 1000 dense neurons, followed by dropout of 0.5. I changed it to 500 dense neurons and a dropout of 0.7. This was done because:

1. Unlike Part 1, which had 62 classes, this time we had only 10.
2. Pretraining the network(on digits from dataset 1) with settings as that of Part 1, the NN overfit.

All notebooks with '_expt' in the end have this new network, while the normal ones have the old one.

With the experimental network:

1. Random Initialization overfits with the default settings, while the pretrained network just kept getting better.
2. The pretrained network outperformed the Random Initialization network. It has better final performance and less training time. The training is more stable.

```
Epoch 1/3
600/600 - 2324s - loss: 0.5642 - accuracy: 0.8434 - val_loss: 0.0925 - val_accuracy: 0.9754
Epoch 2/3
600/600 - 2329s - loss: 0.1160 - accuracy: 0.9683 - val_loss: 0.0388 - val_accuracy: 0.9876
Epoch 3/3
600/600 - 2318s - loss: 0.0978 - accuracy: 0.9742 - val_loss: 0.0892 - val_accuracy: 0.9757
```

Fig: Results from 'CNN_MNIST_RandomInit_Expt.ipynb'

```
Epoch 1/3
600/600 - 2716s - loss: 0.3943 - accuracy: 0.8761 - val_loss: 1.0055 - val_accuracy: 0.8141
Epoch 2/3
600/600 - 2670s - loss: 0.1289 - accuracy: 0.9647 - val_loss: 0.0443 - val_accuracy: 0.9863
Epoch 3/3
600/600 - 2660s - loss: 0.0999 - accuracy: 0.9732 - val_loss: 0.0256 - val_accuracy: 0.9911
```

Fig: Results from 'CNN_MNIST_Pretrained_Expt.ipynb'

The original network:

1. Never overfit.
2. Achieved better results with Random Initialization than the experimental network.
3. This time, I tuned the learning rate while pretraining the network, and pretrained more(achieved better accuracy. Validation loss remained similar.) than the experimental network. However, the previous pretrained network outperforms.

```
Epoch 1/3
600/600 - 2367s - loss: 0.4278 - accuracy: 0.8984 - val_loss: 1.1957 - val_accuracy: 0.7597
Epoch 2/3
600/600 - 2409s - loss: 0.0888 - accuracy: 0.9733 - val_loss: 0.0435 - val_accuracy: 0.9866
Epoch 3/3
600/600 - 2362s - loss: 0.0798 - accuracy: 0.9775 - val_loss: 0.0387 - val_accuracy: 0.9875
```

Fig: Results from 'CNN_MNIST_RandomInit.ipynb'

```
Epoch 1/3
600/600 - 2722s - loss: 0.2609 - accuracy: 0.9270 - val_loss: 0.1044 - val_accuracy: 0.9733
Epoch 2/3
600/600 - 2726s - loss: 0.0994 - accuracy: 0.9721 - val_loss: 0.0482 - val_accuracy: 0.9848
Epoch 3/3
600/600 - 2704s - loss: 0.0815 - accuracy: 0.9780 - val_loss: 0.0447 - val_accuracy: 0.9871
```

Fig: Results from 'CNN_MNIST_Pretrained.ipynb'

Part 3:

The dataset provided is very funny in the sense that the data points are exactly opposite of what I expected. For example, the folder '0' has all other digits but 0.

In Preprocessing_Dataset_3.ipynb, I have preprocessed the dataset.

Both the networks, obviously, did not perform well on the MNIST test set. They were being trained to predict anything but the right answer.

```
Epoch 1/3
600/600 - 2730s - loss: 2.4558 - accuracy: 0.1004 - val_loss: 7.4234 - val_accuracy: 0.0508
Epoch 2/3
600/600 - 2727s - loss: 2.3067 - accuracy: 0.1010 - val_loss: 2.3035 - val_accuracy: 0.0892
Epoch 3/3
600/600 - 2697s - loss: 2.3031 - accuracy: 0.0985 - val_loss: 2.3035 - val_accuracy: 0.0974
```

Fig: Results from 'CNN_DS3_RandomInit.ipynb'

```
Epoch 1/3
600/600 - 2750s - loss: 2.3469 - accuracy: 0.1016 - val_loss: 2.4488 - val_accuracy: 0.0445
Epoch 2/3
600/600 - 2771s - loss: 2.3007 - accuracy: 0.1074 - val_loss: 2.5298 - val_accuracy: 0.0105
Epoch 3/3
600/600 - 2709s - loss: 2.2979 - accuracy: 0.1082 - val_loss: 2.6014 - val_accuracy: 0.0039
```

Fig: Results from 'CNN_DS3_Pretrained.ipynb'

```
Epoch 1/3
600/600 - 2726s - loss: 2.3888 - accuracy: 0.1015 - val_loss: 2.5911 - val_accuracy: 0.0886
Epoch 2/3
600/600 - 2690s - loss: 2.3034 - accuracy: 0.1009 - val_loss: 2.3041 - val_accuracy: 0.0913
Epoch 3/3
600/600 - 2589s - loss: 2.3029 - accuracy: 0.1011 - val_loss: 2.3035 - val_accuracy: 0.0892
```

Fig: Results from 'CNN_DS3_RandomInit_Expt.ipynb'

```
Epoch 1/3
600/600 - 2645s - loss: 2.3474 - accuracy: 0.1001 - val_loss: 2.3033 - val_accuracy: 0.0958
Epoch 2/3
600/600 - 2633s - loss: 2.3026 - accuracy: 0.0993 - val_loss: 2.3034 - val_accuracy: 0.0974
Epoch 3/3
600/600 - 2592s - loss: 2.3026 - accuracy: 0.1017 - val_loss: 2.3036 - val_accuracy: 0.0892
```

Fig: Results from 'CNN_DS3_Pretrained_Expt.ipynb'

In the experimental network, there isn't much difference after 3 epochs. In the original network however, the pretrained network's validation loss increases steadily.

Note:

I had trained all my networks on Google Colab for faster training. However, this means that not all model checkpoints could be retrieved.

Also, all the notebooks having the CNN were similar. So I have explained the code in only one of them: 'CNN_Dataset1.ipynb'. I have also explained the code in notebooks used for preprocessing.