MNIST Case Study

Name: Youseef Osama Ahmed ID:20190629
Name: Mohammad Alameen Abdilaziz ID:20190720

In this Notebook, we loaded the popular MNIST dataset and tried different techniques, architectures, activation functions …etc, to find the best model to capture the complexity of the problem.

Optimizers:
opt = SGD(learning_rate=0.0001, momentum=0.9)
opt1 = SGD(learning_rate=0.05, momentum=0.9)
opt2 = SGD(learning_rate=0.001, momentum=0.9)
opt3 = SGD(learning_rate=0.01, momentum=0.9)
opt4 = Adam(learning_rate=0.01, beta_1=0.9, beta_2=0.999)
opt5 = RMSprop(learning_rate=0.01, rho=0.9, momentum=0.1)

Trying different numbers of epochs

Model 1:
  Final accuracy: 0.9217 ,0.9180

```
Epoch 1/10
1594/1594 [==============================] - 10s 5ms/step - loss: 2.1314 - accuracy: 0.4138 - val_loss: 1.8247 - val_accuracy: 0.6716
Epoch 2/10
1594/1594 [==============================] - 10s 6ms/step - loss: 1.3077 - accuracy: 0.7370 - val_loss: 0.8850 - val_accuracy: 0.7961
Epoch 3/10
1594/1594 [==============================] - 8s 5ms/step - loss: 0.6993 - accuracy: 0.8255 - val_loss: 0.5736 - val_accuracy: 0.8510
Epoch 4/10
1594/1594 [==============================] - 8s 5ms/step - loss: 0.5114 - accuracy: 0.8624 - val_loss: 0.4612 - val_accuracy: 0.8751
Epoch 5/10
1594/1594 [==============================] - 8s 5ms/step - loss: 0.4318 - accuracy: 0.8793 - val_loss: 0.4025 - val_accuracy: 0.8856
```

Number of parameters: 149834
Aeravge epoch time: 9s ~ 10s
Model architecture:
  Conv2D: 64 each(5, 5), strides=(2,2), activation=relu
  MaxPooling2D: pool_size(2, 2), strides(2, 2)
  Dense: 64, activation='relu'
  Dense: 10, activation='softmax'
Optimizer: SGD, lr 0.0001, momentum 0.9, epochs 10, batch size 32

Model 2:

final accuracy: 0.9325 ,0.9370

```
Epoch 1/15
1594/1594 [==============================] - 8s 5ms/step - loss: 2.0256 - accuracy: 0.4934 - val_loss: 1.6045 - val_accuracy: 0.6970
Epoch 2/15
1594/1594 [==============================] - 10s 6ms/step - loss: 1.0736 - accuracy: 0.7930 - val_loss: 0.7248 - val_accuracy: 0.8326
Epoch 3/15
1594/1594 [==============================] - 9s 6ms/step - loss: 0.5953 - accuracy: 0.8544 - val_loss: 0.5121 - val_accuracy: 0.8660
Epoch 4/15
1594/1594 [==============================] - 9s 6ms/step - loss: 0.4646 - accuracy: 0.8758 - val_loss: 0.4275 - val_accuracy: 0.8799
Epoch 5/15
1594/1594 [==============================] - 9s 5ms/step - loss: 0.4053 - accuracy: 0.8872 - val_loss: 0.3822 - val_accuracy: 0.8899
```

Number of parameters: 149834

Average epoch time: 9s ~ 10s

Model architecture: Same as Model 1

Optimizers: #epochs: 15, everything else is the same

Increasing the epochs led to increasing the accuracy, meaning that with small epochs the model doesn't converge.

Model 3:

Final accuracy: 0.9412, 0.9444

```
Epoch 1/20
1594/1594 [==============================] - 10s 6ms/step - loss: 2.1320 - accuracy: 0.3622 - val_loss: 1.8361 - val_accuracy: 0.6187
Epoch 2/20
1594/1594 [==============================] - 9s 5ms/step - loss: 1.3153 - accuracy: 0.7428 - val_loss: 0.8773 - val_accuracy: 0.8066
Epoch 3/20
1594/1594 [==============================] - 8s 5ms/step - loss: 0.6767 - accuracy: 0.8388 - val_loss: 0.5496 - val_accuracy: 0.8630
Epoch 4/20
1594/1594 [==============================] - 8s 5ms/step - loss: 0.4880 - accuracy: 0.8721 - val_loss: 0.4388 - val_accuracy: 0.8846
Epoch 5/20
1594/1594 [==============================] - 8s 5ms/step - loss: 0.4123 - accuracy: 0.8874 - val_loss: 0.3857 - val_accuracy: 0.8954
```

Number of parameters: 149834

Average epoch time: 9s ~ 10s

Model architecture: Same as Model 1

Optimizer: #epochs: 20, everything else is the same

also increasing it more lead to more improvement but we prefer to step here to stop the model from overfitting

We went with model 3 as it gave the best result so far.

Trying different learning rates

Model 4:

Final accuracy: 0.9959, 0.9848

```
Epoch 1/20
1594/1594 [==============================] - 9s 5ms/step - loss: 0.1611 - accuracy: 0.9499 - val_loss: 0.1067 - val_accuracy: 0.9678
Epoch 2/20
1594/1594 [==============================] - 9s 5ms/step - loss: 0.0618 - accuracy: 0.9815 - val_loss: 0.0602 - val_accuracy: 0.9819
Epoch 3/20
1594/1594 [==============================] - 8s 5ms/step - loss: 0.0420 - accuracy: 0.9868 - val_loss: 0.0636 - val_accuracy: 0.9816
Epoch 4/20
1594/1594 [==============================] - 8s 5ms/step - loss: 0.0335 - accuracy: 0.9891 - val_loss: 0.0948 - val_accuracy: 0.9781
Epoch 5/20
1594/1594 [==============================] - 9s 6ms/step - loss: 0.0281 - accuracy: 0.9908 - val_loss: 0.0565 - val_accuracy: 0.9852
```

Number of parameters: 149834

Average epoch time: 9s ~ 10s

Model architecture: Same as Model 1

Optimizer: learning rate = 0.05, everything else is the same

#Increasing the learning rate helped the model converge faster

Model 5:

Final accuracy: 0.9893, 0.9861

```
Epoch 1/20
1594/1594 [==============================] - 9s 6ms/step - loss: 0.6835 - accuracy: 0.8201 - val_loss: 0.2945 - val_accuracy: 0.9124
Epoch 2/20
1594/1594 [==============================] - 8s 5ms/step - loss: 0.2544 - accuracy: 0.9258 - val_loss: 0.2096 - val_accuracy: 0.9410
Epoch 3/20
1594/1594 [==============================] - 8s 5ms/step - loss: 0.1830 - accuracy: 0.9473 - val_loss: 0.1572 - val_accuracy: 0.9543
Epoch 4/20
1594/1594 [==============================] - 9s 5ms/step - loss: 0.1426 - accuracy: 0.9587 - val_loss: 0.1254 - val_accuracy: 0.9648
Epoch 5/20
1594/1594 [==============================] - 9s 6ms/step - loss: 0.1166 - accuracy: 0.9665 - val_loss: 0.1122 - val_accuracy: 0.9679
```

Number of parameters: 149834
Average epoch time: 9s ~ 10s
Model architecture: Same as Model 1
Optimizer: learning rate = 0.001, everything else is the same

<u># the model converged slower than model 4 and didn't achieve a better result</u>

Model 6:

Final accuracy: 0.9999, 0.9906

```
Epoch 1/20
1594/1594 [==============================] - 9s 5ms/step - loss: 0.2418 - accuracy: 0.9258 - val_loss: 0.0880 - val_accuracy: 0.9750
Epoch 2/20
1594/1594 [==============================] - 8s 5ms/step - loss: 0.0724 - accuracy: 0.9781 - val_loss: 0.0628 - val_accuracy: 0.9814
Epoch 3/20
1594/1594 [==============================] - 9s 6ms/step - loss: 0.0503 - accuracy: 0.9849 - val_loss: 0.0571 - val_accuracy: 0.9830
Epoch 4/20
1594/1594 [==============================] - 9s 5ms/step - loss: 0.0397 - accuracy: 0.9878 - val_loss: 0.0503 - val_accuracy: 0.9847
Epoch 5/20
1594/1594 [==============================] - 9s 5ms/step - loss: 0.0318 - accuracy: 0.9902 - val_loss: 0.0402 - val_accuracy: 0.9891
Epoch 6/20
```

Number of parameters: 149834
Average epoch time: 9s ~ 10s
Model architecture: Same as Model 1
Optimizer: learning rate = 0.01, everything else is the same

<u>#0.01 seems like the ideal learning rate as we have achieved our best result so far, meaning that we will go with model 6.</u>

Trying different architectures

Model 7:

Final accuracy: 1.000, 0.9912

```
Epoch 1/20
1594/1594 [==============================] - 21s 13ms/step - loss: 0.1854 - accuracy: 0.9411 - val_loss: 0.0796 - val_accuracy: 0.9773
Epoch 2/20
1594/1594 [==============================] - 20s 13ms/step - loss: 0.0611 - accuracy: 0.9813 - val_loss: 0.0471 - val_accuracy: 0.9872
Epoch 3/20
1594/1594 [==============================] - 21s 13ms/step - loss: 0.0439 - accuracy: 0.9862 - val_loss: 0.0431 - val_accuracy: 0.9871
Epoch 4/20
1594/1594 [==============================] - 21s 13ms/step - loss: 0.0343 - accuracy: 0.9893 - val_loss: 0.0402 - val_accuracy: 0.9889
Epoch 5/20
1594/1594 [==============================] - 20s 13ms/step - loss: 0.0249 - accuracy: 0.9918 - val_loss: 0.0357 - val_accuracy: 0.9898
```

Number of parameters: 63242
Average epoch time: 27s ~ 29s

Model architecture:

Conv2D: 32 each(3, 3), relu
MaxPooling2D: pool_size(2, 2), strides(2, 2)
Conv2D: 32, each(3, 3), relu
MaxPooling2D: pool_size(2, 2), strides(2, 2)
Dense: 64, relu
Dense: 32, relu
Dense: 10, softmax

Optimizer: same as before.

#adding an extra Convulational layer improved the model since the model now captuers more patterns in the dataset.

Model 8:

Final accuracy: 0.9837, 0.9710

```
Epoch 1/20
1594/1594 [==============================] - 6s 3ms/step - loss: 0.4185 - accuracy: 0.8664 - val_loss: 0.2485 - val_accuracy: 0.9246
Epoch 2/20
1594/1594 [==============================] - 5s 3ms/step - loss: 0.1789 - accuracy: 0.9443 - val_loss: 0.1633 - val_accuracy: 0.9477
Epoch 3/20
1594/1594 [==============================] - 5s 3ms/step - loss: 0.1384 - accuracy: 0.9574 - val_loss: 0.1646 - val_accuracy: 0.9516
Epoch 4/20
1594/1594 [==============================] - 5s 3ms/step - loss: 0.1185 - accuracy: 0.9638 - val_loss: 0.1313 - val_accuracy: 0.9598
Epoch 5/20
1594/1594 [==============================] - 5s 3ms/step - loss: 0.1057 - accuracy: 0.9674 - val_loss: 0.1494 - val_accuracy: 0.9541
```

Number of parameters: 9322
Average epoch time: 5s ~ 6s
Model architecture:

Conv2D: 16, each(3, 3), strides(2, 2), relu
MaxPooling2D: pool_size(2, 2), strides=(2, 2)
Conv2D: 32, each(3, 3), strides(2, 2), relu
MaxPooling2D: pool_size(2, 2), strides(2, 2)
Dense: 64, relu
Dense: 32, relu
Dense: 10, softmax

Optimizer: same as before.

#decreasing the number of filters in the first Convolutional layer didn't hurt the model that much since it is only learning basic features.

Model 9:

Final accuracy: 1.0000, 0.9914

```
Epoch 1/20
1594/1594 [==============================] - 27s 17ms/step - loss: 0.1639 - accuracy: 0.9476 - val_loss: 0.0640 - val_accuracy: 0.9813
Epoch 2/20
1594/1594 [==============================] - 26s 16ms/step - loss: 0.0604 - accuracy: 0.9816 - val_loss: 0.0580 - val_accuracy: 0.9829
Epoch 3/20
1594/1594 [==============================] - 27s 17ms/step - loss: 0.0419 - accuracy: 0.9865 - val_loss: 0.0512 - val_accuracy: 0.9852
Epoch 4/20
1594/1594 [==============================] - 25s 16ms/step - loss: 0.0326 - accuracy: 0.9897 - val_loss: 0.0445 - val_accuracy: 0.9879
Epoch 5/20
1594/1594 [==============================] - 27s 17ms/step - loss: 0.0249 - accuracy: 0.9922 - val_loss: 0.0432 - val_accuracy: 0.9887
```

Number of parameters: 42698
Average epoch time: 30s ~ 31s

Model architecture:

Conv2D: 32, each(3, 3), relu
MaxPooling2D: pool_size(2, 2), strides(2, 2)
Conv2D: 32, each(5, 5), relu
MaxPooling2D: pool_size(2, 2), strides(2, 2)
Dense: 32, relu
Dense: 10, softmax
Optimizer: same as before.

<u>#increasing the size of the stride improved the model since it now looks at a bigger area of the image helping the model capture extra features.</u>

Model 10:

Final accuracy: 1.0000, 0.9881

```
Epoch 1/20
1594/1594 [==============================] - 38s 24ms/step - loss: 0.2286 - accuracy: 0.9293 - val_loss: 0.0931 - val_accuracy: 0.9716
Epoch 2/20
1594/1594 [==============================] - 37s 23ms/step - loss: 0.0761 - accuracy: 0.9774 - val_loss: 0.0691 - val_accuracy: 0.9791
Epoch 3/20
1594/1594 [==============================] - 37s 23ms/step - loss: 0.0518 - accuracy: 0.9844 - val_loss: 0.0602 - val_accuracy: 0.9821
Epoch 4/20
1594/1594 [==============================] - 37s 24ms/step - loss: 0.0368 - accuracy: 0.9886 - val_loss: 0.0624 - val_accuracy: 0.9822
Epoch 5/20
1594/1594 [==============================] - 38s 24ms/step - loss: 0.0285 - accuracy: 0.9907 - val_loss: 0.0635 - val_accuracy: 0.9809
```

Number of parameters: 693866
Average epoch time: 38s ~ 41s
Model architecture:

Conv2D: 128, each(3, 3), relu
MaxPooling2D: pool_size(2, 2), strides(2, 2)
Dense: 32, relu
Dense: 10, softmax

Optimizer: same as before.

<u>#Even though this model achieved some high accuracy bu it didn't generalize as well as the previous ones because there is a single convolutional layer that focuses on the simple features and doesn't give much attention to more complex ones. Choosing the right architecture was harder than previous iterations as it was a toss-up between models 7 and 9, but we chose 9 as it performed generally better on newer data.</u>

Trying different batch sizes

Model 11:

Final accuracy: 0.9992, 0.9902

```
Epoch 1/20
3188/3188 [==============================] - 28s 9ms/step - loss: 0.1456 - accuracy: 0.9548 - val_loss: 0.0603 - val_accuracy: 0.9814
Epoch 2/20
3188/3188 [==============================] - 28s 9ms/step - loss: 0.0521 - accuracy: 0.9837 - val_loss: 0.0505 - val_accuracy: 0.9856
Epoch 3/20
3188/3188 [==============================] - 28s 9ms/step - loss: 0.0366 - accuracy: 0.9886 - val_loss: 0.0564 - val_accuracy: 0.9823
Epoch 4/20
3188/3188 [==============================] - 28s 9ms/step - loss: 0.0292 - accuracy: 0.9909 - val_loss: 0.0569 - val_accuracy: 0.9834
Epoch 5/20
3188/3188 [==============================] - 28s 9ms/step - loss: 0.0238 - accuracy: 0.9928 - val_loss: 0.0394 - val_accuracy: 0.9883
```

Number of parameters: 42698
Average epoch time: 28s ~ 29s

Model architecture: same as our best model, model 9.
Optimizer: Batch size: 16, everything else is the same.

#No clear improvement over our best model so we stick with it.

Model 12:

Final accuracy: 0.9993, 0.9897

```
Epoch 1/20
797/797 [==============================] - 21s 26ms/step - loss: 0.2312 - accuracy: 0.9260 - val_loss: 0.0868 - val_accuracy: 0.9743
Epoch 2/20
797/797 [==============================] - 21s 26ms/step - loss: 0.0718 - accuracy: 0.9778 - val_loss: 0.0647 - val_accuracy: 0.9813
Epoch 3/20
797/797 [==============================] - 22s 27ms/step - loss: 0.0524 - accuracy: 0.9838 - val_loss: 0.0626 - val_accuracy: 0.9797
Epoch 4/20
797/797 [==============================] - 22s 28ms/step - loss: 0.0422 - accuracy: 0.9867 - val_loss: 0.0451 - val_accuracy: 0.9870
Epoch 5/20
797/797 [==============================] - 21s 26ms/step - loss: 0.0351 - accuracy: 0.9891 - val_loss: 0.0429 - val_accuracy: 0.9874
```

Number of parameters: 42698
Average epoch time: 20s ~ 21s
Model architecture: same as our best model, model 9.
Optimizer: Batch size: 64, everything else is the same.

#No clear improvement over our best model so we stick with it, the only clear difference is that training was shorter.

Trying different activation functions

Model 13:

Final accuracy: 0.9990, 0.9906

```
Epoch 1/20
1594/1594 [==============================] - 22s 13ms/step - loss: 2.3081 - accuracy: 0.1056 - val_loss: 2.3033 - val_accu
racy: 0.0976
Epoch 2/20
1594/1594 [==============================] - 21s 13ms/step - loss: 2.3033 - accuracy: 0.1099 - val_loss: 2.3005 - val_accu
racy: 0.1102
Epoch 3/20
1594/1594 [==============================] - 20s 13ms/step - loss: 2.0368 - accuracy: 0.2718 - val_loss: 1.1817 - val_accu
racy: 0.6211
Epoch 4/20
1594/1594 [==============================] - 20s 12ms/step - loss: 0.6618 - accuracy: 0.8040 - val_loss: 0.3823 - val_accu
racy: 0.8889
Epoch 5/20
1594/1594 [==============================] - 20s 13ms/step - loss: 0.2906 - accuracy: 0.9168 - val_loss: 0.2116 - val_accu
racy: 0.9401
```

Number of parameters: 63,242
Average epoch time: 19s ~ 20s
Model architecture:

- Conv2D: 32, each(3, 3), tanh
- MaxPooling2D: pool_size(2, 2), strides(2, 2)
- Conv2D: 32, each(3, 3), tanh
- MaxPooling2D: pool_size(2, 2), strides(2, 2)
- Dense: 64, tanh
- Dense: 32, tanh
- Dense: 10, softmax

Optimizer: same as before.

#Training is faster, but the results are not better

Model 14:

Final accuracy: 1.0000, 0.9914

```
Epoch 1/20
1594/1594 [==============================] - 26s 16ms/step - loss: 0.1503 - accuracy: 0.9530 - val_loss: 0.0910 - val_accu
racy: 0.9707
Epoch 2/20
1594/1594 [==============================] - 23s 15ms/step - loss: 0.0534 - accuracy: 0.9836 - val_loss: 0.0550 - val_accu
racy: 0.9836
Epoch 3/20
1594/1594 [==============================] - 23s 14ms/step - loss: 0.0370 - accuracy: 0.9881 - val_loss: 0.0404 - val_accu
racy: 0.9891
Epoch 4/20
1594/1594 [==============================] - 23s 15ms/step - loss: 0.0262 - accuracy: 0.9913 - val_loss: 0.0501 - val_accu
racy: 0.9861
Epoch 5/20
1594/1594 [==============================] - 23s 14ms/step - loss: 0.0199 - accuracy: 0.9933 - val_loss: 0.0433 - val_accu
racy: 0.9884
```

Number of parameters: 63,242

Average epoch time: 20s ~ 21s

Model architecture:

- Conv2D: 32, each(3, 3), SELU
- MaxPooling2D: pool_size(2, 2), strides(2, 2)
- Conv2D: 32, each(3, 3), SELU
- MaxPooling2D: pool_size(2, 2), strides(2, 2)
- Dense: 64, SELU
- Dense: 32, SELU
- Dense: 10, softmax

Optimizer: same as before

<u># SELU didn't improve much.</u>

Model15:

Final accuracy: 0.9977, 0.9887

```
Epoch 1/20
1594/1594 [==============================] - 21s 13ms/step - loss: 0.1609 - accuracy: 0.9493 - val_loss: 0.0644 - val_accu
racy: 0.9809
Epoch 2/20
1594/1594 [==============================] - 20s 13ms/step - loss: 0.0524 - accuracy: 0.9835 - val_loss: 0.0521 - val_accu
racy: 0.9846
Epoch 3/20
1594/1594 [==============================] - 20s 12ms/step - loss: 0.0357 - accuracy: 0.9885 - val_loss: 0.0449 - val_accu
racy: 0.9878
Epoch 4/20
1594/1594 [==============================] - 20s 13ms/step - loss: 0.0272 - accuracy: 0.9911 - val_loss: 0.0472 - val_accu
racy: 0.9859
Epoch 5/20
1594/1594 [==============================] - 22s 14ms/step - loss: 0.0205 - accuracy: 0.9935 - val_loss: 0.0428 - val_accu
racy: 0.9901
```

Number of parameters: 63,242

Average epoch time: 20s ~ 22s

Model architecture:

- Conv2D: 32, each(3, 3), LeakyRELU
- MaxPooling2D: pool_size(2, 2), strides(2, 2)
- Conv2D: 32, each(3, 3), LeakyRELU
- MaxPooling2D: pool_size(2, 2), strides(2, 2)
- Dense: 64, LeakyRELU
- Dense: 32, LeakyRELU
- Dense: 10, softmax

Optimizer: same as before

#Results between RELU and LeakyRELU were close, but it seems that RELU always came on top, it still seems that RELU works best with our data.

Trying different optimizers

Model 16:

Final accuracy: 0.9822, 0.9749

```
Epoch 1/20
1594/1594 [==============================] - 20s 12ms/step - loss: 0.1942 - accuracy: 0.9413 - val_loss: 0.1241 - val_accu
racy: 0.9649
Epoch 2/20
1594/1594 [==============================] - 18s 12ms/step - loss: 0.1102 - accuracy: 0.9686 - val_loss: 0.0918 - val_accu
racy: 0.9746
Epoch 3/20
1594/1594 [==============================] - 18s 12ms/step - loss: 0.0981 - accuracy: 0.9723 - val_loss: 0.0923 - val_accu
racy: 0.9743
Epoch 4/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.0929 - accuracy: 0.9755 - val_loss: 0.0963 - val_accu
racy: 0.9769
Epoch 5/20
1594/1594 [==============================] - 18s 12ms/step - loss: 0.0875 - accuracy: 0.9780 - val_loss: 0.0872 - val_accu
racy: 0.9791
```

Number of parameters: 42698
Average epoch time: 23s ~ 24s
Model architecture: Same as our best model
Optimizer: we used Adam optimizer with the following parameters
(learning_rate=0.01, beta_1=0.9, beta_2=0.999)

#The normal SGD performs better than Adam optimizer

Model 17:

final train: 0.9812 0.9702

```
Epoch 1/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.2097 - accuracy: 0.9453 - val_loss: 0.1469 - val_accu
racy: 0.9624
Epoch 2/20
1594/1594 [==============================] - 18s 12ms/step - loss: 0.1462 - accuracy: 0.9669 - val_loss: 0.1216 - val_accu
racy: 0.9700
Epoch 3/20
1594/1594 [==============================] - 18s 12ms/step - loss: 0.1490 - accuracy: 0.9685 - val_loss: 0.1965 - val_accu
racy: 0.9547
Epoch 4/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.1542 - accuracy: 0.9679 - val_loss: 0.1602 - val_accu
racy: 0.9739
Epoch 5/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.1637 - accuracy: 0.9668 - val_loss: 0.2736 - val_accu
racy: 0.9601
```

Number of parameters: 42698
Average epoch time: 23s ~ 24s
Model architecture: Same as our best model
Optimizer: we used RMSprop optimizer with the following parameters
(learning_rate=0.01, rho=0.9, momentum=0.1)

#The normal SGD performs better than RMSprop optimizer, we are going with our normal SGD optimizer with learning rate = 0.01 and momentum = 0.9

Trying different dropout rates

Model 18:

Final accuracy: 0.9973, 0.9908

```
Epoch 1/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.1695 - accuracy: 0.9466 - val_loss: 0.0602 - val_accu
racy: 0.9813
Epoch 2/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.0556 - accuracy: 0.9825 - val_loss: 0.0494 - val_accu
racy: 0.9848
Epoch 3/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.0387 - accuracy: 0.9878 - val_loss: 0.0366 - val_accu
racy: 0.9897
Epoch 4/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.0296 - accuracy: 0.9902 - val_loss: 0.0429 - val_accu
racy: 0.9878
Epoch 5/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.0234 - accuracy: 0.9924 - val_loss: 0.0393 - val_accu
racy: 0.9891
```

Number of parameters: 63242

Average epoch time: 19s ~ 20s

Model architecture:

- Conv2D: 32, each(3, 3), relu
- MaxPooling2D: pool_size=(2, 2), strides=(2, 2)
- Dropout: 0.4
- Conv2D: 32, each(3, 3), relu
- MaxPooling2D: pool_size(2, 2), strides(2, 2)
- Dense: 64, relu
- Dense: 32, relu
- Dropout: 0.4
- Dense: 10, softmax

Optimizer: SGD with learning rate = 0.01 and momentum = 0.9

#The model performed worse than the previous models due to the dropout layer.

Model 19:

Final accuracy: 0.9600, 0.9851

```
Epoch 1/20
1594/1594 [==============================] - 20s 12ms/step - loss: 0.6229 - accuracy: 0.8037 - val_loss: 0.0971 - val_accu
racy: 0.9741
Epoch 2/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.2819 - accuracy: 0.9249 - val_loss: 0.0781 - val_accu
racy: 0.9794
Epoch 3/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.2294 - accuracy: 0.9400 - val_loss: 0.0645 - val_accu
racy: 0.9834
Epoch 4/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.1998 - accuracy: 0.9471 - val_loss: 0.0584 - val_accu
racy: 0.9832
Epoch 5/20
1594/1594 [==============================] - 19s 12ms/step - loss: 0.1856 - accuracy: 0.9508 - val_loss: 0.0532 - val_accu
racy: 0.9858
```

Number of parameters: 63242

Average epoch time: 20s ~ 22s

Model architecture:

- Conv2D: 32, each(3, 3), relu
- MaxPooling2D: pool_size=(2, 2), strides=(2, 2)
- Conv2D: 32, each(3, 3), relu
- MaxPooling2D: pool_size=(2, 2), strides=(2, 2)

Dropout: 0.4
Dense: 64, relu
Dropout: 0.4
Dense: 32, relu
Dropout: 0.4
Dense: 10, softmax
Optimizer: same as the previous model.

#The model’s performance even dropped more.

Model 20:

Final accuracy: 0.7687, 0.9658

```
Epoch 1/20
1594/1594 [==============================] - 34s 21ms/step - loss: 2.2051 - accuracy: 0.1628 - val_loss: 1.5272 - val_accu
racy: 0.5069
Epoch 2/20
1594/1594 [==============================] - 40s 25ms/step - loss: 1.2013 - accuracy: 0.5635 - val_loss: 0.4409 - val_accu
racy: 0.8629
Epoch 3/20
1594/1594 [==============================] - 40s 25ms/step - loss: 0.8447 - accuracy: 0.7053 - val_loss: 0.3483 - val_accu
racy: 0.8730
Epoch 4/20
1594/1594 [==============================] - 35s 22ms/step - loss: 0.7811 - accuracy: 0.7425 - val_loss: 0.2595 - val_accu
racy: 0.9653
Epoch 5/20
1594/1594 [==============================] - 36s 23ms/step - loss: 0.7185 - accuracy: 0.7694 - val_loss: 0.2151 - val_accu
racy: 0.9663
```

Number of parameters: 63242
Average epoch time: 20s ~ 22s
Model architecture:
Conv2D: 32, each(3, 3), relu
MaxPooling2D: pool_size=(2, 2), strides=(2, 2)
Dropout: 0.75
Conv2D: 32, each(3, 3), relu
MaxPooling2D: pool_size=(2, 2), strides=(2, 2)
Dense: 64, relu
Dense: 32, relu
Dropout: 0.75
Dense: 10, softmax
Optimizer: same as the optimizer used by Model 18.

#One thing to note is that although accuracy on the training set was bad, it performed way better on unseen data than our previous model.

Model 21:

Final accuracy: 0.5111, 0.7243

```
Epoch 1/20
1594/1594 [==============================] - 33s 20ms/step - loss: 2.2771 - accuracy: 0.1328 - val_loss: 2.1170 - val_accu
racy: 0.3033
Epoch 2/20
1594/1594 [==============================] - 33s 21ms/step - loss: 2.0881 - accuracy: 0.2293 - val_loss: 1.7329 - val_accu
racy: 0.5182
Epoch 3/20
1594/1594 [==============================] - 31s 20ms/step - loss: 1.9132 - accuracy: 0.3047 - val_loss: 1.3778 - val_accu
racy: 0.5361
Epoch 4/20
1594/1594 [==============================] - 32s 20ms/step - loss: 1.7589 - accuracy: 0.3626 - val_loss: 1.2273 - val_accu
racy: 0.5953
Epoch 5/20
1594/1594 [==============================] - 31s 19ms/step - loss: 1.6701 - accuracy: 0.3981 - val_loss: 1.0783 - val_accu
racy: 0.6412
```

Number of parameters: 63242

Average epoch time: 20s ~ 22s

Model architecture:

- Conv2D: 32, each(3, 3), relu
- MaxPooling2D: pool_size=(2, 2), strides=(2, 2)
- Conv2D: 32, each(3, 3), relu
- MaxPooling2D: pool_size=(2, 2), strides=(2, 2)
- Dropout: 0.75
- Dense: 64, relu
- Dropout: 0.75
- Dense: 32, relu
- Dropout: 0.75
- Dense: 10, softmax

Optimizer: same as the optimizer used by Model 18

#this model performs the same way as our last model, the key difference is that having a big dropout probability means regularizing the model way too much, stopping it from functioning well.

Summary of our best model, model 9:-

The number of epochs is 20

The batch size is 32

Optimizer: Stochastic gradient Decent with a learning rate of 0.1 and a momentum of 0.9

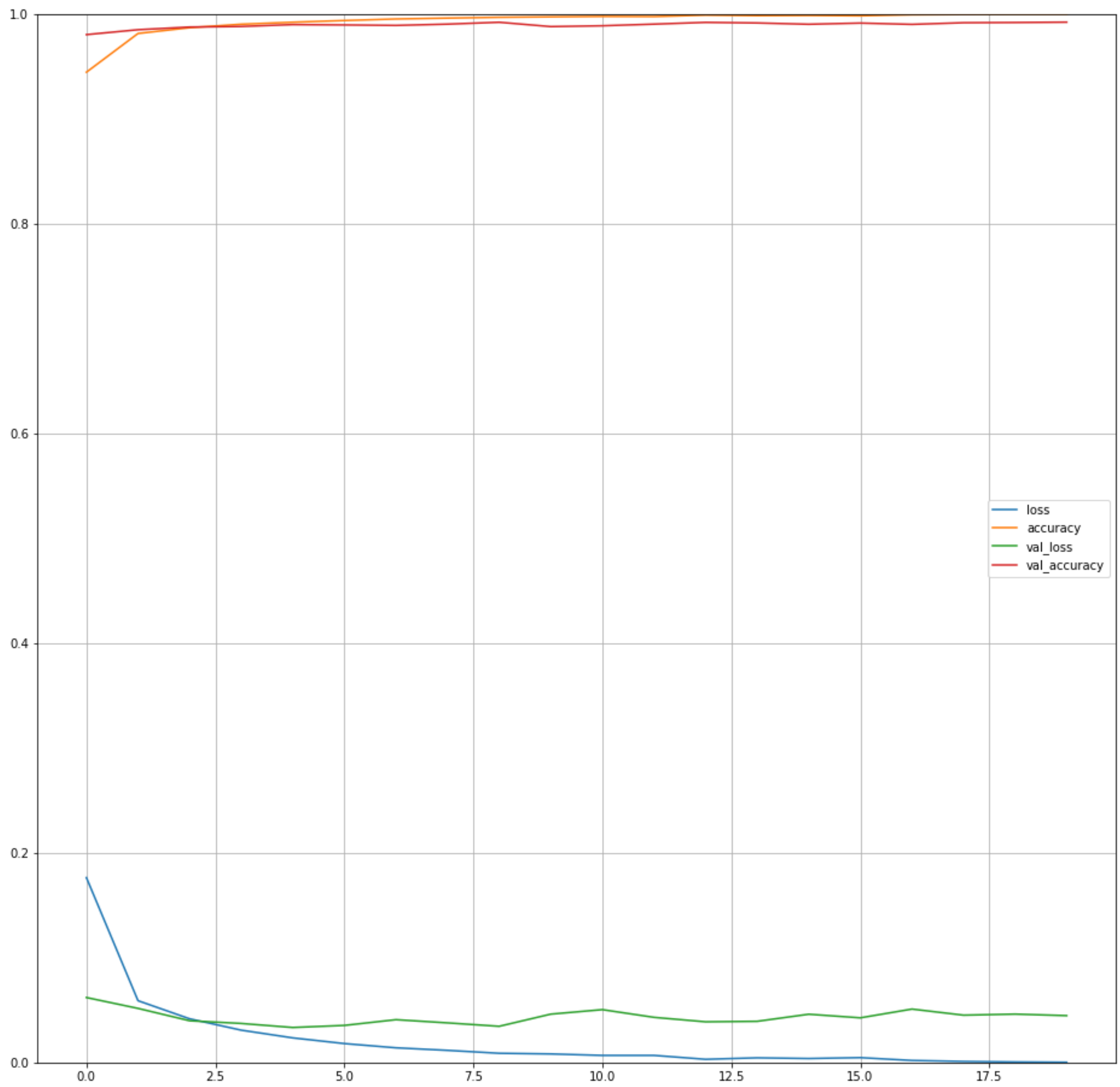
We used ReLU as our activation function in all the layers except for the output layer where we used a softmax activation function

Our architecture is as follows:

our first convolutional layer was 32 filters of size 3x3 followed by a max pool layer

the second convolutional layer is 32 filters of size 5x5 also followed by a max pool layer

We chose 1 FC layer consisting of 32 neurons followed by an output layer.



Loss and Accuracy of our best model both on the training and validation sets