Mnist Case Study

Name: Youseef Osama Ahmed ID:20190629
Name: Mohamed Al-Amin aba-Alziz ID:20190720

In this case study we loaded the popular mnist dataset and tried different techniques, archs, activations …etc. in order 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)

Base Model:-

Model1:
final train: 0.9217 / Val: 0.9180
epoch 1: 0.4910 0.7171
epoch 2: 0.7866 0.8358
epoch 3: 0.8531 0.8727
epoch 4: 0.8781 0.8893
epoch 5: 0.8904 0.8981
#params 149834
Avg 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'
Optimizers: SGD, lr 0.0001, momentum 0.9, epochs 10, batch size 32

Trying different number of epochs

Model 2:
final train: 0.9325 0.9370
epoch 1: 0.4889 0.6504
epoch 2: 0.7332 0.8063
epoch 3: 0.8338 0.8562
epoch 4: 0.8679 0.8814
epoch 5: 0.8851 0.8920
#params 149834

Avg epoch time 9s ~ 10s
model Arch: Same as Model 1
Optimizers: epochs 15, ... SAME
Increasing the epochs led to increase in the accuracy such that the model took extra time and training steps.

Model 3
final train: 0.9412 0.9444
epoch 1: 0.4906 0.7059
epoch 2: 0.7799 0.8276
epoch 3: 0.8487 0.8644
epoch 4: 0.8740 0.8849
epoch 5: 0.8871 0.8941
#params 149834
Avg epoch time 9s ~ 10s
model Arch Same as Model 1
Optimizers epochs 20, ...SAME
also increasing it more lead to more improvement but we prefer to step here in order to stop the model from overfitting

Trying different learning_rate

model 4:
final train: 0.9959 0.9848
epoch 1: 0.9480 0.9790
epoch 2: 0.9821 0.9837
epoch 3: 0.9864 0.9844
epoch 4: 0.9898 0.9856
epoch 5: 0.9921 0.9863
#params 149834
Avg epoch time 9s ~ 10s
model Arch Same as Mode1
Optimizers lr 0.05, ...SAME
#Increasing the LR helped the model to converge faster
model 5:
final train: 0.9893 0.9861
epoch 1: 0.8103 0.9116
epoch 2: 0.9230 0.9420
epoch 3: 0.9458 0.9568
epoch 4: 0.9580 0.9644
epoch 5: 0.9657 0.9699
#params 149834
Avg epoch time 9s ~ 10s

model Arch Same as Model1
Optimizers lr 0.001, ...SAME
the model improved slower than model 4
model 6:
final train: 0.9999 0.9906
epoch 1: 0.9247 0.9671
epoch 2: 0.9778 0.9822
epoch 3: 0.9842 0.9839
epoch 4: 0.9874 0.9863
epoch 5: 0.9910 0.9881
#params 149834
Avg epoch time 9s ~ 10s
model Arch Same as Model1
Optimizers lr 0.01, ...SAME
#The model improvement is in between.

Trying different architectures

model 7:
final train: 0.1000 0.9912
epoch 1: 0.9461 0.9791
epoch 2: 0.9830 0.9864
epoch 3: 0.9885 0.9872
epoch 4: 0.9908 0.9877
epoch 5: 0.9927 0.9884
#params 63242
Avg epoch time 27s ~ 29s
model Arch
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
Optimizers SAME
#adding extra Conv layer improved the model since the model now captuer extra patterns
model 8:
final train: 0.9837 0.9710
epoch 1: 0.8571 0.9232
epoch 2: 0.9439 0.9447
epoch 3: 0.9549 0.9601
epoch 4: 0.9627 0.9572

epoch 5: 0.9668 0.9563
#params 9322
Avg epoch time 5s ~ 6s
model Arch:
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
Optimizers SAME
#decreasing the number of filters in the first Conv layer didn't hurt the model that much since it just learning basic features

model 9:
final train: 1.0000 0.9914
epoch 1: 0.9472 0.9806
epoch 2: 0.9829 0.9866
epoch 3: 0.9886 0.9822
epoch 4: 0.9910 0.9882
epoch 5: 0.9928 0.9883
#params 42698
Avg epoch time 30s ~ 31s
model Arch:
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
Optimizers SAME
#increasing the strides improved the model since it now look at bigger area helping him to capture extra features

model 10:
final train: 1.0000 0.9881
epoch 1: 0.9282 0.9733
epoch 2: 0.9785 0.9754
epoch 3: 0.9852 0.9846
epoch 4: 0.9892 0.9847
epoch 5: 0.9928 0.9844
#params 693866
Avg epoch time 38s ~ 41s
model Arch:
Conv2D: 128, each(3, 3), relu

MaxPooling2D: pool_size(2, 2), strides(2, 2)
Dense: 32, relu
Dense: 10, softmax

Optimizers: Same

#The model not as model 9 but 9 is better since 9 in just adding extra feature maps in the first Conv layers which doesn't help him but learning extra basic features not the complex ones

Trying different batch sizes

model 11:

final train: 0.9992 0.9902
epoch 1: 0.9581 0.9838
epoch 2: 0.9843 0.9880
epoch 3: 0.9893 0.9864
epoch 4: 0.9916 0.9888
epoch 5: 0.9935 0.9899
#params 42698
Avg epoch time 28s ~ 29s
model Arch SAME as Model9
Optimizers Batch 16, ...Same

#Same no improvements

model 12:

final train: 0.9993 0.9897
epoch 1: 0.9231 0.9736
epoch 2: 0.9766 0.9814
epoch 3: 0.9835 0.9842
epoch 4: 0.9870 0.9861
epoch 5: 0.9887 0.9859
#params 42698
Avg epoch time 20s ~ 21s
model Arch Same as Model 9
Optimizers Batch 64, ...SAME

#Same no improvements

Trying different activation functions

model 13:

final train: 0.9990 0.9906
epoch 1: 0.9434 0.9766
epoch 2: 0.9820 0.9850
epoch 3: 0.9880 0.9846
epoch 4: 0.9911 0.9860
epoch 5: 0.9934 0.9882

#params 63,242
Avg epoch time 19s ~ 20s
model Arch:
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
Optimizers: Same opt3
#Same no improvements

model 14:
final train:
epoch 1: 0.9560 0.9771
epoch 2: 0.9835 0.9836
epoch 3: 0.9883 0.9876
epoch 4: 0.9911 0.9847
epoch 5: 0.9932 0.9879
#params 63,242
Avg epoch time 20s ~ 21s
model Arch: Same as Model 13 but wit SELU
Optimizers: Same opt3
selu hurt the training it is always better to use the tan or relu in Conv
model 15:
final train: 0.9977 0.9887
epoch 1: 0.9432 0.9802
epoch 2: 0.9843 0.9853
epoch 3: 0.9889 0.9880
epoch 4: 0.9915 0.9878
epoch 5: 0.9932 0.9897
#params 63,242
Avg epoch time 20s ~ 22s
model Arch Same as model 13 but with LeakyRelu
Optimizers Same opt3
#Same no improvements

Trying different optimizers

model 16:
final train: 0.9822 0.9749
epoch 1: 0.9517 0.9714
epoch 2: 0.9734 0.9708

epoch 3: 0.9768 0.9772
epoch 4: 0.9788 0.9802
epoch 5: 0.9810 0.9790
#params 42698
Avg epoch time 23s ~ 24s
model Arch Same as Model 9
Optimizers opt4

model 17:
final train: 0.9812 0.9702
epoch 1: 0.9460 0.9650
epoch 2: 0.9742 0.9722
epoch 3: 0.9768 0.9730
epoch 4: 0.9781 0.9790
epoch 5: 0.9776 0.9804
#params 42698
Avg epoch time 23s ~ 24s
model Arch Same as model 9
Optimizers opt5

#No improvement opt5 is not better the extra Adam us helping

Trying different dropout rates

model 18:
final train: 0.9973 0.9908
epoch 1: 0.9424 0.9779
epoch 2: 0.9828 0.9860
epoch 3: 0.9874 0.9876
epoch 4: 0.9905 0.9868
epoch 5: 0.9925 0.9904
#params 63242
Avg epoch time 19s ~ 20s
model Arch:
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
Optimizers: SGD, lr 0.01, momentum 0.9

#Same no improvements

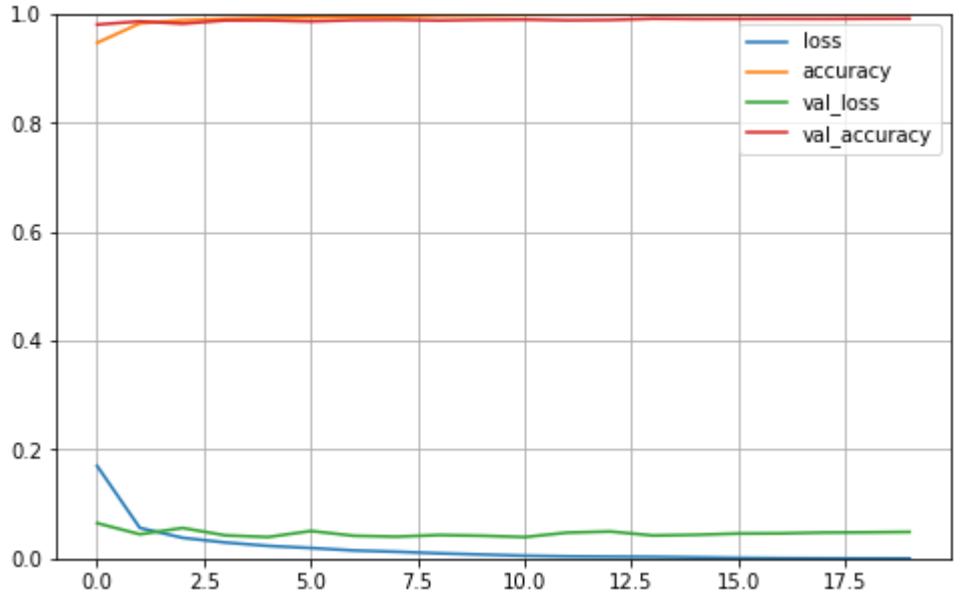
model 19:
final train: 0.9600 0.9851
epoch 1: 0.7766 0.9693
epoch 2: 0.9139 0.9766
epoch 3: 0.9340 0.9786
epoch 4: 0.9419 0.9846
epoch 5: 0.9482 0.9843
#params 63242
Avg epoch time 21s ~ 20s
model Arch:
Conv2D: 32, each(3, 3), relu
MaxPooling2D: pool_size=(2, 2), strides=(2, 2)
Conv2D: 32, each(3, 3), 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
Optimizers same as Model 18
#Same no improvements
model 20:
final train: 0.1120 0.1101
epoch 1: 0.1112 0.1101
epoch 2: 0.1110 0.1101
epoch 3: 0.1117 0.1101
epoch 4: 0.1111 0.1101
epoch 5: 0.1110 0.1101
#params 63242
Avg epoch time 21s ~ 20s
model Arch:
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
Optimizers same as Model 18
model 21:
final train: 0.4919 0.7359

epoch 1: 0.1491 0.2524
epoch 2: 0.2285 0.3947
epoch 3: 0.3184 0.5078
epoch 4: 0.3900 0.6452
epoch 5: 0.4278 0.6483
#params 63242
Avg epoch time 19s ~ 20s
model Arch:
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
Optimizers same as Model 18
#Increasing the dropout prevent the model from learning since most of the neurons will be now off

---

Summary of our best model, model 9:-

number of epochs is 20
size of batch is 32
stochastic SGD 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 follow:
our first conc layer was 32 filters of size 3x3 followed by a max pool layer
the second conv 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.



For extra info: https://github.com/Al-ameen007/CNN/blob/main/CNN.ipynb