POOJA JAISWAL

Student ID: A20426140

CS -512 (Computer Vision)

Fall (2018)

CS 512 – Assignment 4

Report

Instructions: run cnn_test.py file in console with input image. Ex: python cnn_test.py 1.png

Press q or Q to close the images and ESC

Abstract:

In this project I have built a CNN model using Keras where I trained my custom CNN model and evaluate it in a manner that my CNN model is able to identify the numbers in the MNIST dataset as either even or odd labeled by the corresponding integer. Saving the model as **cnn_model.h5** and analyzing the accuracy, entropy loss, learning rate and other parameters as below.

Deliverables 1: Custom CNN

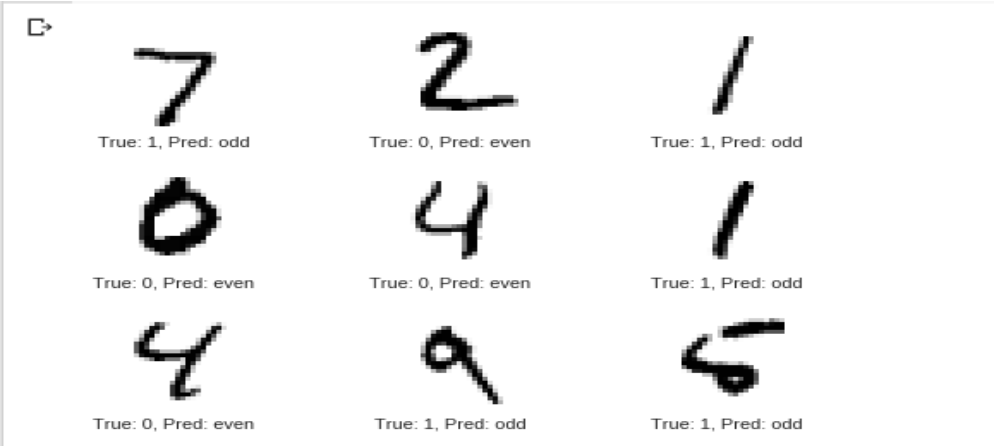
1. Train CNN using given configuration:

While creating and training our own custom CNN I am using MNIST train set: 60,000 samples with two Convolution and pooling layers of 32 filters and 64 filters of kernel size 5x5 then pooling in both layers with down sample by a factor of 2. Then preparing the model by applying cross entropy loss with dropout rate of 40% and gradient descent optimization and a learning rate of 0.001. Finally, I trained my model with 5 epochs. Below is the code snippet:

Input details:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(lr=0.001, momentum=0.8, decay=0.0, nesterov=False),
              metrics=['accuracy',recall,precision])
```

Output details of 5 epochs: As we can see the below image is using MNIST data and classify its label to a binary labeling (even/odd).



2. Report the training loss and accuracy:

a. The loss and the accuracy for each iteration as below:

Iteration1: - loss: 0.3629 - accuracy: 0.8487

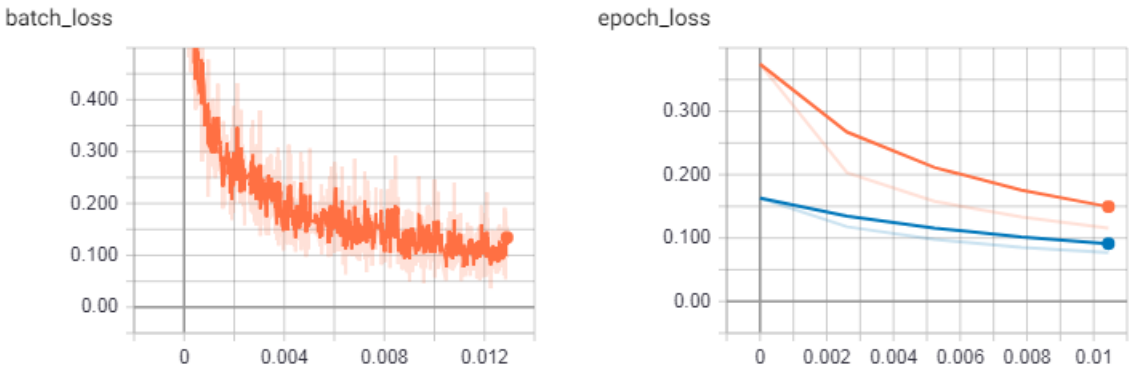
Iteration2: - loss: 0.1841 - accuracy: 0.9285

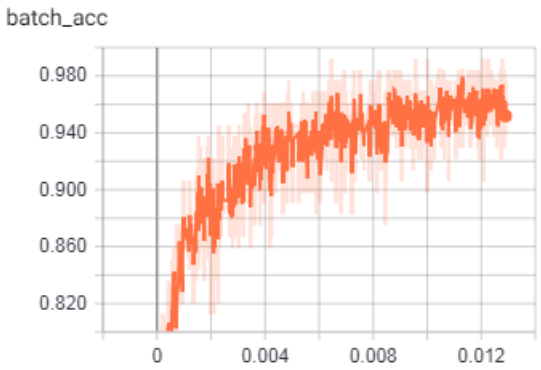
Iteration3: - loss: 0.1432 - accuracy: 0.9457

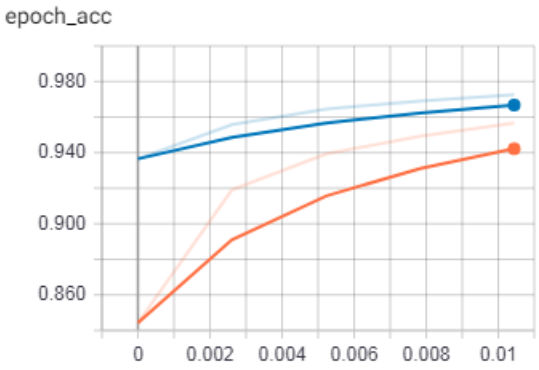
Iteration4: - loss: 0.1233 - accuracy: 0.9533

Iteration5: - loss: 0.1084 - accuracy: 0.9607

Below is the plotted graph where Orange lines are training set and blue is validation representation for each Iterations of loss and accuracy:







b. The loss and accuracy value of the final training step:

loss: 0.1084
accuracy: 0.9607

3. Report the evaluation loss, accuracy, precision, and recall using MNIST test set (10,000 samples)

a. for each epoch trained, compute and plot the curve:

Validate on 10000 samples for 5 epochs
Iteration1: - val_loss: 0.1474 - val_acc: 0.9453 - val_recall: 0.9453 - val_precision: 0.9453

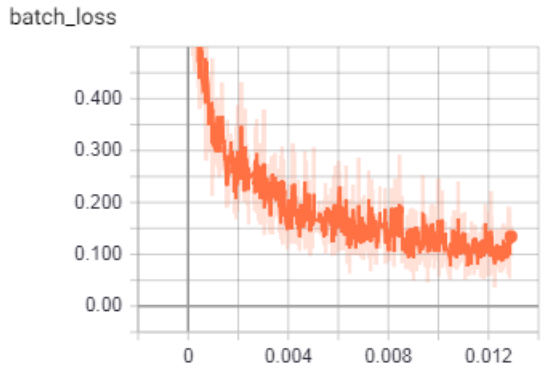
Iteration2: - val_loss: 0.1057 - val_acc: 0.9639 - val_recall: 0.9639 - val_precision: 0.9639

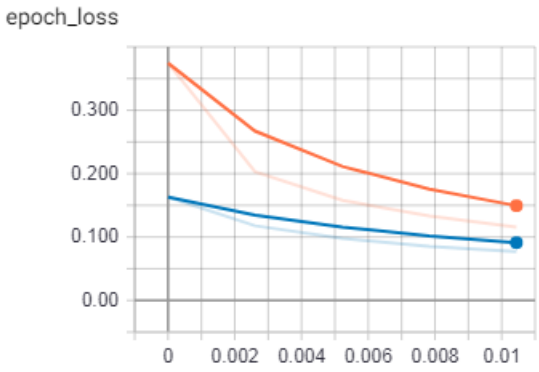
Iteration3: - val_loss: 0.0884 - val_acc: 0.9693 - val_recall: 0.9693 - val_precision: 0.9693

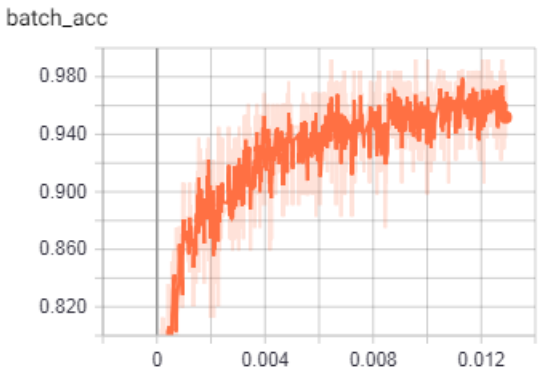
Iteration4: - val_loss: 0.0775 - val_acc: 0.9734 - val_recall: 0.9734 - val_precision: 0.9734

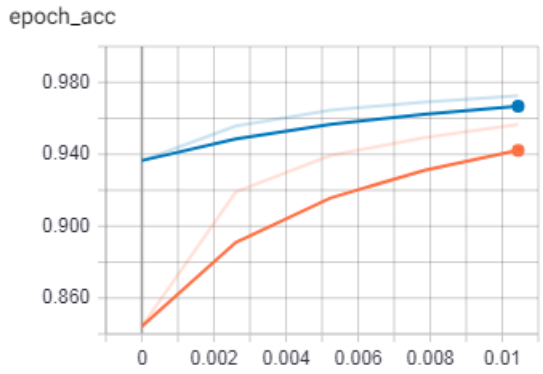
Iteration5: - val_loss: 0.0707 - val_acc: 0.9751 - val_recall: 0.9751 - val_precision: 0.9751

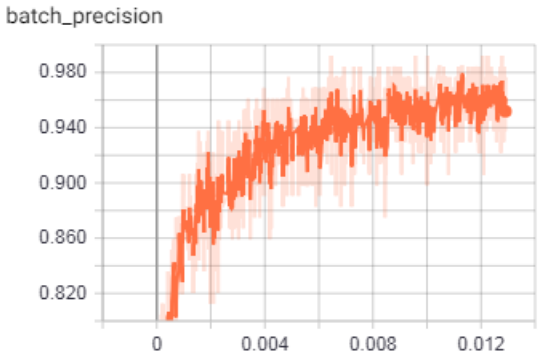
Below is the plotted graph where Orange lines are training set and blue is validation representation for each Iterations of loss and accuracy:

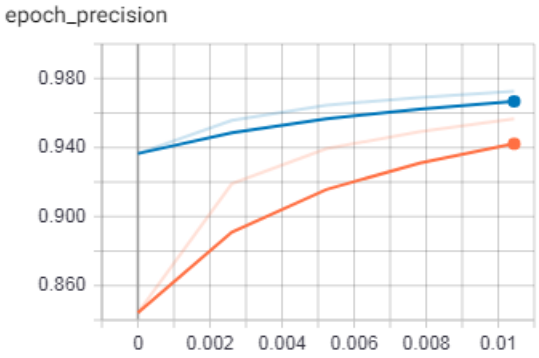


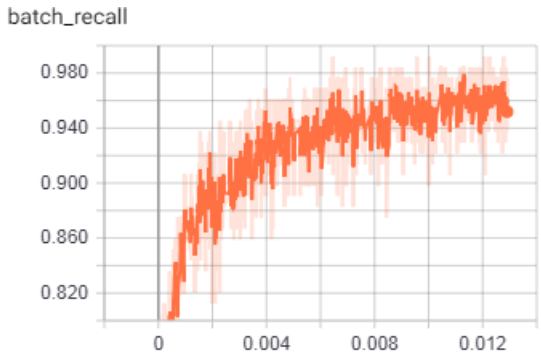


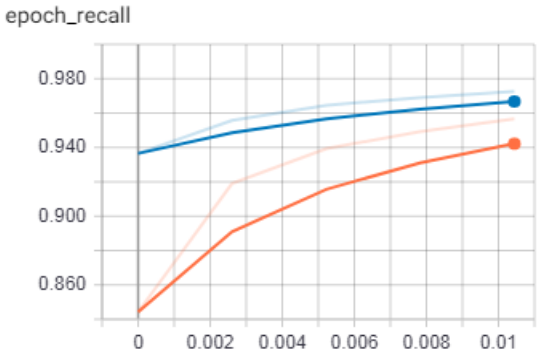






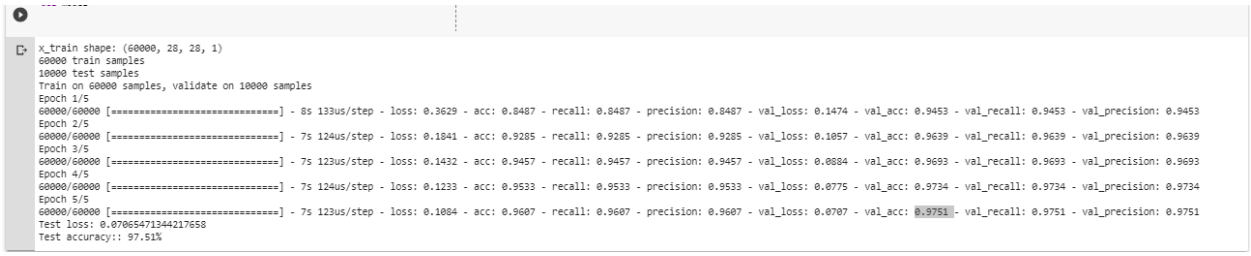






b. The loss, accuracy, recall and precision value of the last epoch as:

loss: 0.0707
accuracy: 0.9751
recall: 0.9751
precision: 0.9751



```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 8s 133us/step - loss: 0.3629 - acc: 0.8487 - recall: 0.8487 - precision: 0.8487 - val_loss: 0.1474 - val_acc: 0.9453 - val_recall: 0.9453 - val_precision: 0.9453
Epoch 2/5
60000/60000 [==============================] - 7s 124us/step - loss: 0.1841 - acc: 0.9285 - recall: 0.9285 - precision: 0.9285 - val_loss: 0.1057 - val_acc: 0.9639 - val_recall: 0.9639 - val_precision: 0.9639
Epoch 3/5
60000/60000 [==============================] - 7s 123us/step - loss: 0.1432 - acc: 0.9457 - recall: 0.9457 - precision: 0.9457 - val_loss: 0.0884 - val_acc: 0.9693 - val_recall: 0.9693 - val_precision: 0.9693
Epoch 4/5
60000/60000 [==============================] - 7s 124us/step - loss: 0.1233 - acc: 0.9533 - recall: 0.9533 - precision: 0.9533 - val_loss: 0.0775 - val_acc: 0.9734 - val_recall: 0.9734 - val_precision: 0.9734
Epoch 5/5
60000/60000 [==============================] - 7s 123us/step - loss: 0.1084 - acc: 0.9607 - recall: 0.9607 - precision: 0.9607 - val_loss: 0.0707 - val_acc: 0.9751 - val_recall: 0.9751 - val_precision: 0.9751
Test loss: 0.07065471344217658
Test accuracy:: 97.51%
```

Deliverables 2: Parameter Tuning

Experiment1: Changed the network architecture

Model's parameter details: Change in organization of layers

Model1: Epoch Trained - 5, Dropout Rate- 40%, learn rate- 0.001, number of convolution layers- 2, kernel size 5x5, **filters size- 32 and 64**

Model2: Epoch Trained - 5, Dropout Rate- 40%, learn rate- 0.001, number of convolution layers- 2, kernel size 5x5, **filters size- 64 and 64**

Observations:

Model1 Observation: The observation of the model 1 is mentioned below:

Epoch 1/5: - loss: 0.3859 - acc: 0.8403 - recall: 0.8403 - precision: 0.8403 - val_loss: 0.1532 - val_acc: 0.9417 - val_recall: 0.9417 - val_precision: 0.9417
Epoch 2/5: - loss: 0.1920 - acc: 0.9248 - recall: 0.9248 - precision: 0.9248 - val_loss: 0.1088 - val_acc: 0.9610 - val_recall: 0.9610 - val_precision: 0.9610
Epoch 3/5: - loss: 0.1484 - acc: 0.9441 - recall: 0.9441 - precision: 0.9441 - val_loss: 0.0898 - val_acc: 0.9677 - val_recall: 0.9677 - val_precision: 0.9677
Epoch 4/5: - loss: 0.1255 - acc: 0.9533 - recall: 0.9533 - precision: 0.9533 - val_loss: 0.0794 - val_acc: 0.9722 - val_recall: 0.9722 - val_precision: 0.9722
Epoch 5/5: - loss: 0.1091 - acc: 0.9607 - recall: 0.9607 - precision: 0.9607 - val_loss: 0.0733 - val_acc: 0.9745 - val_recall: 0.9745 - val_precision: 0.9745
Test loss: 0.07326604500412941
Test accuracy: 97.45%

Model2 Observation: The observation of the model 2 is mentioned below:

Epoch 1/5: - loss: 0.3266 - acc: 0.8680 - recall: 0.8680 - precision: 0.8680 - val_loss: 0.1219 - val_acc: 0.9547 - val_recall: 0.9547 - val_precision: 0.9547
Epoch 2/5: - loss: 0.1609 - acc: 0.9381 - recall: 0.9381 - precision: 0.9381 - val_loss: 0.0879 - val_acc: 0.9665 - val_recall: 0.9665 - val_precision: 0.9665
Epoch 3/5: - loss: 0.1254 - acc: 0.9540 - recall: 0.9540 - precision: 0.9540 - val_loss: 0.0742 - val_acc: 0.9737 - val_recall: 0.9737 - val_precision: 0.9737
Epoch 4/5: - loss: 0.1054 - acc: 0.9610 - recall: 0.9610 - precision: 0.9610 - val_loss: 0.0653 - val_acc: 0.9772 - val_recall: 0.9772 - val_precision: 0.9772
Epoch 5/5: - loss: 0.0938 - acc: 0.9660 - recall: 0.9660 - precision: 0.9660 - val_loss: 0.0602 - val_acc: 0.9793 - val_recall: 0.9793 - val_precision: 0.9793
Test loss: 0.060198101262748244
Test accuracy: 97.93%

Result: From above observation it can be deduced that accuracy in model 2 is more than the accuracy reported in model 1. Similarly, the loss of the model 1 is more than the model 2. So, when we increase the convolution filter size or the number of similar convolution models then the accuracy increases.

Experiment2: Changed the receptive field and stride parameters

Model's parameter details: Change in kernel size

Model1: Epoch Trained - 5, Dropout Rate- 40%, learn rate- 0.001, number of convolution layers- 2, **kernel size 5x5**, filters size- 32 and 64

Model2: Epoch Trained - 5, Dropout Rate- 40%, learn rate- 0.001, number of convolution layers- 2, **kernel size 3x3**, filters size- 32 and 64

Observations:

Model1 Observation: The observation of the model 1 is mentioned below:

Epoch 1/5: - loss: 0.3859 - acc: 0.8403 - recall: 0.8403 - precision: 0.8403 - val_loss: 0.1532 - val_acc: 0.9417 - val_recall: 0.9417 - val_precision: 0.9417
Epoch 2/5: - loss: 0.1920 - acc: 0.9248 - recall: 0.9248 - precision: 0.9248 - val_loss: 0.1088 - val_acc: 0.9610 - val_recall: 0.9610 - val_precision: 0.9610
Epoch 3/5: - loss: 0.1484 - acc: 0.9441 - recall: 0.9441 - precision: 0.9441 - val_loss: 0.0898 - val_acc: 0.9677 - val_recall: 0.9677 - val_precision: 0.9677
Epoch 4/5: - loss: 0.1255 - acc: 0.9533 - recall: 0.9533 - precision: 0.9533 - val_loss: 0.0794 - val_acc: 0.9722 - val_recall: 0.9722 - val_precision: 0.9722
Epoch 5/5: - loss: 0.1091 - acc: 0.9607 - recall: 0.9607 - precision: 0.9607 - val_loss: 0.0733 - val_acc: 0.9745 - val_recall: 0.9745 - val_precision: 0.9745
Test loss: 0.07326604500412941
Test accuracy: 97.45%

Model2 Observation:

Epoch 1/5: - loss: 0.3693 - acc: 0.8471 - recall: 0.8471 - precision: 0.8471 - val_loss: 0.1504 - val_acc: 0.9460 - val_recall: 0.9460 - val_precision: 0.9460
Epoch 2/5: - loss: 0.1921 - acc: 0.9265 - recall: 0.9265 - precision: 0.9265 - val_loss: 0.1085 - val_acc: 0.9629 - val_recall: 0.9629 - val_precision: 0.9629
Epoch 3/5: - loss: 0.1505 - acc: 0.9432 - recall: 0.9432 - precision: 0.9432 - val_loss: 0.0917 - val_acc: 0.9683 - val_recall: 0.9683 - val_precision: 0.9683
Epoch 4/5: - loss: 0.1273 - acc: 0.9533 - recall: 0.9533 - precision: 0.9533 - val_loss: 0.0812 - val_acc: 0.9719 - val_recall: 0.9719 - val_precision: 0.9719
Epoch 5/5: - loss: 0.1129 - acc: 0.9592 - recall: 0.9592 - precision: 0.9592 - val_loss: 0.0747 - val_acc: 0.9739 - val_recall: 0.9739 - val_precision: 0.9739
Test loss: 0.07468675887212158
Test accuracy: 97.39%

Result:

From above observation it can be deduced that there is slight decrease in accuracy and increase in loss when the size of kernel decrease.

Experiment3: Changed optimizer and loss function

Model's parameter details: Change in optimizer and loss function

Model1: Epoch Trained - 5, Dropout Rate- 40%, learn rate- 0.001, number of convolution layers- 2, kernel size 5x5, filters size- 32 and 64, **pooling factor by 2,** **keras.optimizers.SGD(lr=0.001, momentum=0., decay=0., nesterov=False)**

Model2: Epoch Trained - 2, Dropout Rate- 30%, learn rate- 0.01, number of convolution layers- 2, kernel size 5x5, filters size- 32 and 64, **pooling factor by 3,** **keras.optimizers.Adagrad(lr=0.01, epsilon=1e-6) OR keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=1e-6)**

Observations:

Model1 Observation: The observation of the model 1 is mentioned below:

Epoch 1/5: - loss: 0.3859 - acc: 0.8403 - recall: 0.8403 - precision: 0.8403 - val_loss: 0.1532 - val_acc: 0.9417 - val_recall: 0.9417 - val_precision: 0.9417
Epoch 2/5: - loss: 0.1920 - acc: 0.9248 - recall: 0.9248 - precision: 0.9248 - val_loss: 0.1088 - val_acc: 0.9610 - val_recall: 0.9610 - val_precision: 0.9610
Epoch 3/5: - loss: 0.1484 - acc: 0.9441 - recall: 0.9441 - precision: 0.9441 - val_loss: 0.0898 - val_acc: 0.9677 - val_recall: 0.9677 - val_precision: 0.9677
Epoch 4/5: - loss: 0.1255 - acc: 0.9533 - recall: 0.9533 - precision: 0.9533 - val_loss: 0.0794 - val_acc: 0.9722 - val_recall: 0.9722 - val_precision: 0.9722

Epoch 5/5: - loss: 0.1091 - acc: 0.9607 - recall: 0.9607 - precision: 0.9607 - val_loss: 0.0733 - val_acc: 0.9745 - val_recall: 0.9745 - val_precision: 0.9745
Test loss: 0.07326604500412941
Test accuracy: 97.45%

Model2 Observation:

Epoch 1/5: - loss: 0.4713 - acc: 0.8197 - recall: 0.8197 - precision: 0.8197 - val_loss: 0.1697 - val_acc: 0.9346 - val_recall: 0.9346 - val_precision: 0.9346
Epoch 2/5: - loss: 0.2123 - acc: 0.9200 - recall: 0.9200 - precision: 0.9200 - val_loss: 0.1163 - val_acc: 0.9574 - val_recall: 0.9574 - val_precision: 0.9574
Epoch 3/5: - loss: 0.1606 - acc: 0.9413 - recall: 0.9413 - precision: 0.9413 - val_loss: 0.0949 - val_acc: 0.9650 - val_recall: 0.9650 - val_precision: 0.9650
Epoch 4/5: - loss: 0.1367 - acc: 0.9504 - recall: 0.9504 - precision: 0.9504 - val_loss: 0.0835 - val_acc: 0.9686 - val_recall: 0.9686 - val_precision: 0.9686
Epoch 5/5: - loss: 0.1186 - acc: 0.9572 - recall: 0.9572 - precision: 0.9572 - val_loss: 0.0766 - val_acc: 0.9715 - val_recall: 0.9715 - val_precision: 0.9715
Test loss: 0.07662434375472367
Test accuracy: 97.15%

Result:

From above observation it can be deduced that there is slight decrease in accuracy and increase in loss when the pooling factor increase.

Experiment4: Changed various parameters

Model's parameter details: Change in dropout, learning rate, number of epochs

Model1: Epoch Trained - 5, Dropout Rate- 40%, learn rate- 0.001, number of convolution layers- 2, kernel size 5x5, filters size- 32 and 64

Model2: Epoch Trained - 2, Dropout Rate- 30%, **learn rate- 0.01**, number of convolution layers- 2, kernel size 5x5, filters size- 32 and 64

Observations:

Model1 Observation: The observation of the model 1 is mentioned below:

Epoch 1/5: - loss: 0.3859 - acc: 0.8403 - recall: 0.8403 - precision: 0.8403 - val_loss: 0.1532 - val_acc: 0.9417 - val_recall: 0.9417 - val_precision: 0.9417
Epoch 2/5: - loss: 0.1920 - acc: 0.9248 - recall: 0.9248 - precision: 0.9248 - val_loss: 0.1088 - val_acc: 0.9610 - val_recall: 0.9610 - val_precision: 0.9610
Epoch 3/5: - loss: 0.1484 - acc: 0.9441 - recall: 0.9441 - precision: 0.9441 - val_loss: 0.0898 - val_acc: 0.9677 - val_recall: 0.9677 - val_precision: 0.9677
Epoch 4/5: - loss: 0.1255 - acc: 0.9533 - recall: 0.9533 - precision: 0.9533 - val_loss: 0.0794 - val_acc: 0.9722 - val_recall: 0.9722 - val_precision: 0.9722
Epoch 5/5: - loss: 0.1091 - acc: 0.9607 - recall: 0.9607 - precision: 0.9607 - val_loss: 0.0733 - val_acc: 0.9745 - val_recall: 0.9745 - val_precision: 0.9745
Test loss: 0.07326604500412941
Test accuracy: 97.45%

Model2 Observation:

Epoch 1/5: - loss: 0.1526 - acc: 0.9468 - recall: 0.9468 - precision: 0.9468 - val_loss: 0.0589 - val_acc: 0.9811 - val_recall: 0.9811 - val_precision: 0.9811
Epoch 2/5: - loss: 0.0647 - acc: 0.9769 - recall: 0.9769 - precision: 0.9769 - val_loss: 0.0418 - val_acc: 0.9865 - val_recall: 0.9865 - val_precision: 0.9865
Epoch 3/5: - loss: 0.0486 - acc: 0.9828 - recall: 0.9828 - precision: 0.9828 - val_loss: 0.0354 - val_acc: 0.9882 - val_recall: 0.9882 - val_precision: 0.9882

Epoch 4/5: - loss: 0.0411 - acc: 0.9858 - recall: 0.9858 - precision: 0.9858 - val_loss: 0.0309 - val_acc: 0.9904 - val_recall: 0.9904 - val_precision: 0.9904
Epoch 5/5: - loss: 0.0342 - acc: 0.9883 - recall: 0.9883 - precision: 0.9883 - val_loss: 0.0291 - val_acc: 0.9912 - val_recall: 0.9912 - val_precision: 0.9912
Test loss: 0.02909377718269825
Test accuracy: 99.12%

Result: From above observation the accuracy will increase, and the loss will decrease if you increase the Learn rate value and decrease the Drop rate value.

Experiment5: Adding batch and layer normalization

Model's parameter details

Model1: With batch and layer normalization

Model2: Without batch and layer normalization

Observations:

Model1 Observation: The observation of the model 1 is mentioned below:

Epoch 1/5: - loss: 0.3859 - acc: 0.8403 - recall: 0.8403 - precision: 0.8403 - val_loss: 0.1532 - val_acc: 0.9417 - val_recall: 0.9417 - val_precision: 0.9417
Epoch 2/5: - loss: 0.1920 - acc: 0.9248 - recall: 0.9248 - precision: 0.9248 - val_loss: 0.1088 - val_acc: 0.9610 - val_recall: 0.9610 - val_precision: 0.9610
Epoch 3/5: - loss: 0.1484 - acc: 0.9441 - recall: 0.9441 - precision: 0.9441 - val_loss: 0.0898 - val_acc: 0.9677 - val_recall: 0.9677 - val_precision: 0.9677
Epoch 4/5: - loss: 0.1255 - acc: 0.9533 - recall: 0.9533 - precision: 0.9533 - val_loss: 0.0794 - val_acc: 0.9722 - val_recall: 0.9722 - val_precision: 0.9722
Epoch 5/5: - loss: 0.1091 - acc: 0.9607 - recall: 0.9607 - precision: 0.9607 - val_loss: 0.0733 - val_acc: 0.9745 - val_recall: 0.9745 - val_precision: 0.9745
Test loss: 0.07326604500412941
Test accuracy: 97.45%

Model2 Observation:

Epoch 1/5: - loss: 0.5103 - acc: 0.7464 - recall: 0.7464 - precision: 0.7464 - val_loss: 0.3591 - val_acc: 0.8566 - val_recall: 0.8566 - val_precision: 0.8566
Epoch 2/5: - loss: 0.3620 - acc: 0.8444 - recall: 0.8444 - precision: 0.8444 - val_loss: 0.2924 - val_acc: 0.8872 - val_recall: 0.8872 - val_precision: 0.8872
Epoch 3/5: - loss: 0.3074 - acc: 0.8724 - recall: 0.8724 - precision: 0.8724 - val_loss: 0.2515 - val_acc: 0.9039 - val_recall: 0.9039 - val_precision: 0.9039
Epoch 4/5: - loss: 0.2705 - acc: 0.8902 - recall: 0.8902 - precision: 0.8902 - val_loss: 0.2212 - val_acc: 0.9197 - val_recall: 0.9197 - val_precision: 0.9197
Epoch 5/5: - loss: 0.2404 - acc: 0.9061 - recall: 0.9061 - precision: 0.9061 - val_loss: 0.1982 - val_acc: 0.9310 - val_recall: 0.9310 - val_precision: 0.9310
Test loss: 0.1981606138586998
Test accuracy: 93.10%

Result:

From above observation we conclude that the accuracy will decrease, and loss will increase without batch normalization

Experiment6: Using different weight initializers

Model's parameter details: Change in number of epochs

Model1: epoch = 5

Model2: epoch = 1

Model2: epoch = 10

Observations:

Model1 Observation: The observation of the model 1 is mentioned below:

Epoch 1/5: - loss: 0.3859 - acc: 0.8403 - recall: 0.8403 - precision: 0.8403 - val_loss: 0.1532 - val_acc: 0.9417 - val_recall: 0.9417 - val_precision: 0.9417
Epoch 2/5: - loss: 0.1920 - acc: 0.9248 - recall: 0.9248 - precision: 0.9248 - val_loss: 0.1088 - val_acc: 0.9610 - val_recall: 0.9610 - val_precision: 0.9610
Epoch 3/5: - loss: 0.1484 - acc: 0.9441 - recall: 0.9441 - precision: 0.9441 - val_loss: 0.0898 - val_acc: 0.9677 - val_recall: 0.9677 - val_precision: 0.9677
Epoch 4/5: - loss: 0.1255 - acc: 0.9533 - recall: 0.9533 - precision: 0.9533 - val_loss: 0.0794 - val_acc: 0.9722 - val_recall: 0.9722 - val_precision: 0.9722
Epoch 5/5: - loss: 0.1091 - acc: 0.9607 - recall: 0.9607 - precision: 0.9607 - val_loss: 0.0733 - val_acc: 0.9745 - val_recall: 0.9745 - val_precision: 0.9745
Test loss: 0.07326604500412941
Test accuracy: 97.45%

Model2 Observation:

Epoch 1/1: - loss: 0.3811 - acc: 0.8403 - recall: 0.8403 - precision: 0.8403 - val_loss: 0.1505 - val_acc: 0.9430 - val_recall: 0.9430 - val_precision: 0.9430

Test loss: 0.15047877359986306
Test accuracy: 94.30%

Model3 Observation:

Epoch 1/10: - loss: 0.3663 - acc: 0.8475 - recall: 0.8475 - precision: 0.8475 - val_loss: 0.1422 - val_acc: 0.9473 - val_recall: 0.9473 - val_precision: 0.9473
Epoch 2/10: - loss: 0.1866 - acc: 0.9284 - recall: 0.9284 - precision: 0.9284 - val_loss: 0.1032 - val_acc: 0.9625 - val_recall: 0.9625 - val_precision: 0.9625
Epoch 3/10: - loss: 0.1431 - acc: 0.9460 - recall: 0.9460 - precision: 0.9460 - val_loss: 0.0858 - val_acc: 0.9702 - val_recall: 0.9702 - val_precision: 0.9702
Epoch 4/10: - loss: 0.1210 - acc: 0.9547 - recall: 0.9547 - precision: 0.9547 - val_loss: 0.0756 - val_acc: 0.9732 - val_recall: 0.9732 - val_precision: 0.9732
Epoch 5/10: - loss: 0.1090 - acc: 0.9594 - recall: 0.9594 - precision: 0.9594 - val_loss: 0.0691 - val_acc: 0.9749 - val_recall: 0.9749 - val_precision: 0.9749
Epoch 6/10: - loss: 0.0979 - acc: 0.9643 - recall: 0.9643 - precision: 0.9643 - val_loss: 0.0640 - val_acc: 0.9776 - val_recall: 0.9776 - val_precision: 0.9776
Epoch 7/10: - loss: 0.0899 - acc: 0.9675 - recall: 0.9675 - precision: 0.9675 - val_loss: 0.0602 - val_acc: 0.9794 - val_recall: 0.9794 - val_precision: 0.9794
Epoch 8/10: - loss: 0.0848 - acc: 0.9691 - recall: 0.9691 - precision: 0.9691 - val_loss: 0.0570 - val_acc: 0.9799 - val_recall: 0.9799 - val_precision: 0.9799
Epoch 9/10: - loss: 0.0802 - acc: 0.9714 - recall: 0.9714 - precision: 0.9714 - val_loss: 0.0545 - val_acc: 0.9812 - val_recall: 0.9812 - val_precision: 0.9812
Epoch 10/10: - loss: 0.0778 - acc: 0.9723 - recall: 0.9723 - precision: 0.9723 - val_loss: 0.0521 - val_acc: 0.9823 - val_recall: 0.9823 - val_precision: 0.9823
Test loss: 0.052122088477015494
Test accuracy: 98.23%

Result: In this experiment we observed that on increase number of epochs the accuracy rate increase and on decrease number of epoch accuracy rate decrease.

Deliverable 3: Application

Steps of Implementation:

Step1: create a python file name cnn_test.py and import all keras API to run the model

Step2: load the cnn.model which was generated using cnn.py code

Step3: extract the image path from console

Step4: resize, gray out, apply threshold and convert the loaded image into binary.

Step5: reshape the image to the shape and dimension of model image for compatibility.

Step6: image is compatible with cnn model, use predict function to check even and odd number

Step7: display the original image and binary image

Conclusion: The model created and implemented has excellent accuracy rate that's mean it has low loss rate while predicting image as even/odd from MNIST dataset. But while importing custom image to cnn.model accuracy rate is low. The model needs more enhancement for the accurate custom image prediction.