EMRAH SARIBOZ

HW-8 Report

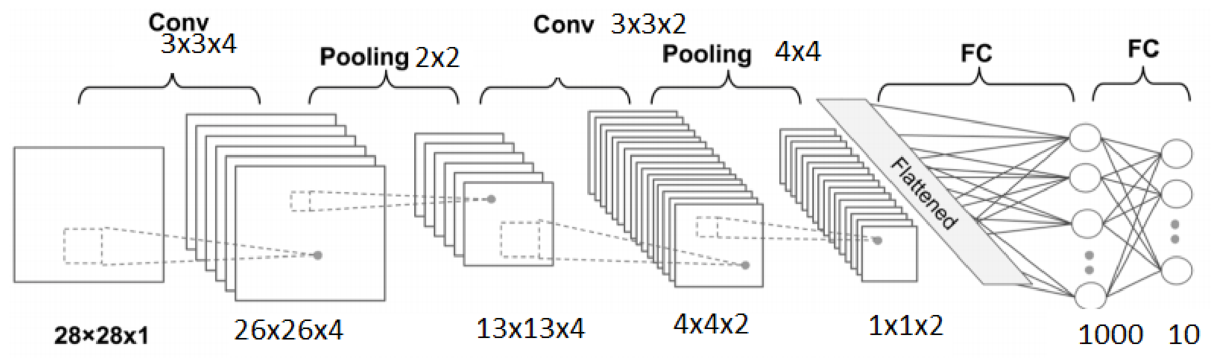
Dataset Usage and problem definition

- The goal of the homework is to apply CNN to recognize the hand-written dataset.
- In this homework, I used google COLAB.
- I downloaded the dataset from the link she posted and extracted it using python.

Question:

Show the shape of the output tensor of each layer in the CNN architecture that you design to answer Q1.

Answer:



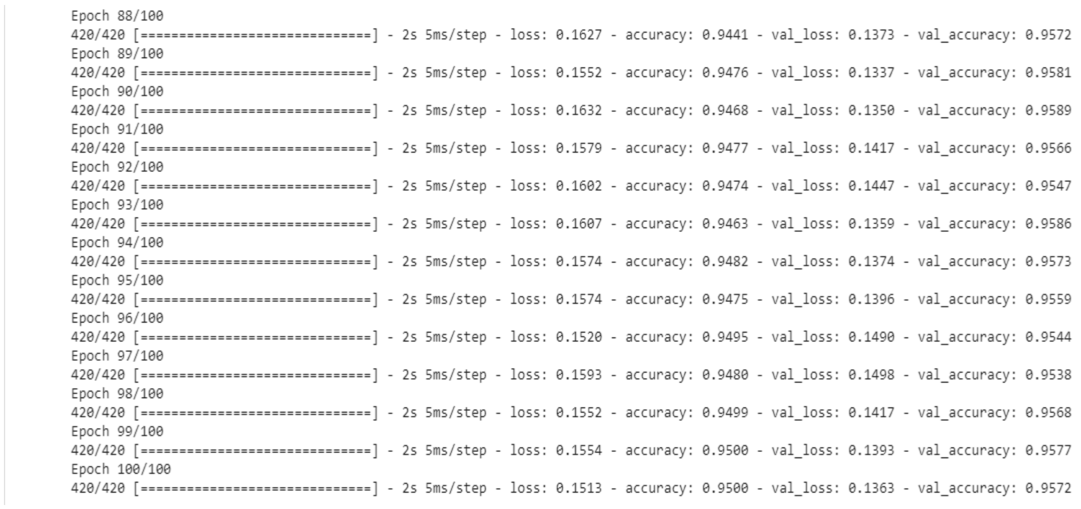
Here is a model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_41 (Conv2D)           (None, 26, 26, 4)         40
_________________________________________________________________
pool_1 (AveragePooling2D)    (None, 13, 13, 4)         0
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 4, 4, 2)           74
_________________________________________________________________
pool_2 (AveragePooling2D)    (None, 1, 1, 2)           0
_________________________________________________________________
flatten_38 (Flatten)         (None, 2)                 0
_________________________________________________________________
dropout_38 (Dropout)         (None, 2)                 0
_________________________________________________________________
flatten_39 (Flatten)         (None, 2)                 0
_________________________________________________________________
dense_38 (Dense)             (None, 1000)              3000
_________________________________________________________________
dropout_39 (Dropout)         (None, 1000)              0
_________________________________________________________________
dense_39 (Dense)             (None, 10)                10010
-----------------------------------------------------------------
```

Experiments

By running CNN model with given credentials, I got 0.45 maximum accuracy in 100 epochs. This situation gave me a hint to change output channel size for the Conv2D processes. Here are the results from that:

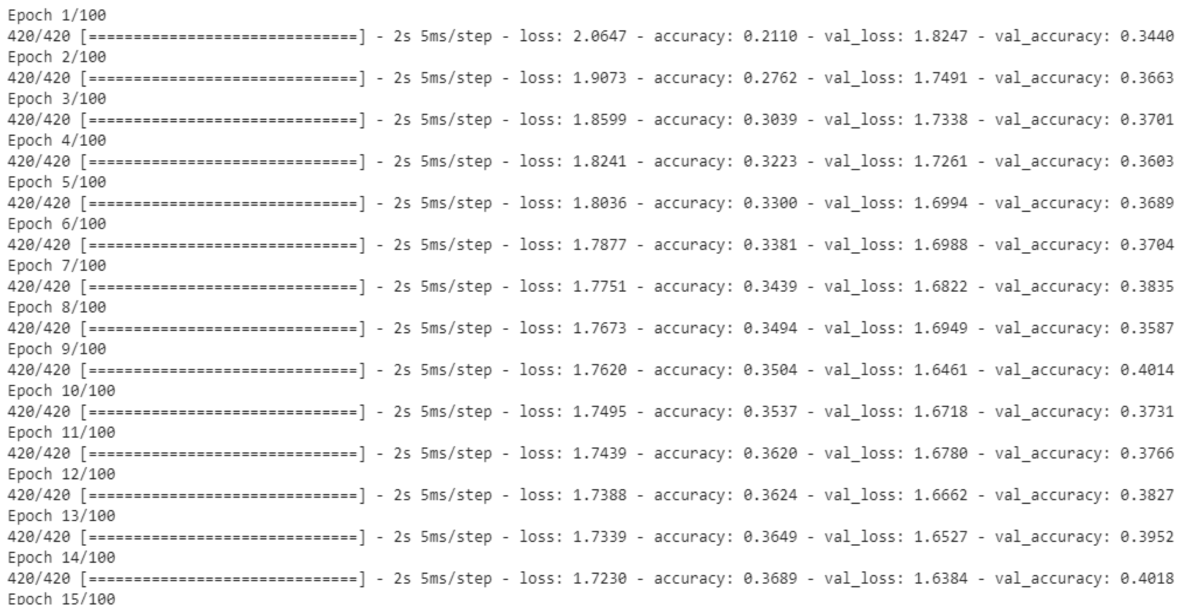
1) Just by changing the output size of first and second conv2D from 4 and 3 to 64 and 32, I was able to get 0.97 accuracy in 100 epochs. In the figure below, you can see the last 12 epochs.
   - Total time: 99 seconds.



```
Epoch 88/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1627 - accuracy: 0.9441 - val_loss: 0.1373 - val_accuracy: 0.9572
Epoch 89/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1552 - accuracy: 0.9476 - val_loss: 0.1337 - val_accuracy: 0.9581
Epoch 90/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1632 - accuracy: 0.9468 - val_loss: 0.1350 - val_accuracy: 0.9589
Epoch 91/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1579 - accuracy: 0.9477 - val_loss: 0.1417 - val_accuracy: 0.9566
Epoch 92/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1602 - accuracy: 0.9474 - val_loss: 0.1447 - val_accuracy: 0.9547
Epoch 93/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1607 - accuracy: 0.9463 - val_loss: 0.1359 - val_accuracy: 0.9586
Epoch 94/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1574 - accuracy: 0.9482 - val_loss: 0.1374 - val_accuracy: 0.9573
Epoch 95/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1574 - accuracy: 0.9475 - val_loss: 0.1396 - val_accuracy: 0.9559
Epoch 96/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1520 - accuracy: 0.9495 - val_loss: 0.1490 - val_accuracy: 0.9544
Epoch 97/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1593 - accuracy: 0.9480 - val_loss: 0.1498 - val_accuracy: 0.9538
Epoch 98/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1552 - accuracy: 0.9499 - val_loss: 0.1417 - val_accuracy: 0.9568
Epoch 99/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1554 - accuracy: 0.9500 - val_loss: 0.1393 - val_accuracy: 0.9577
Epoch 100/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1513 - accuracy: 0.9500 - val_loss: 0.1363 - val_accuracy: 0.9572
```

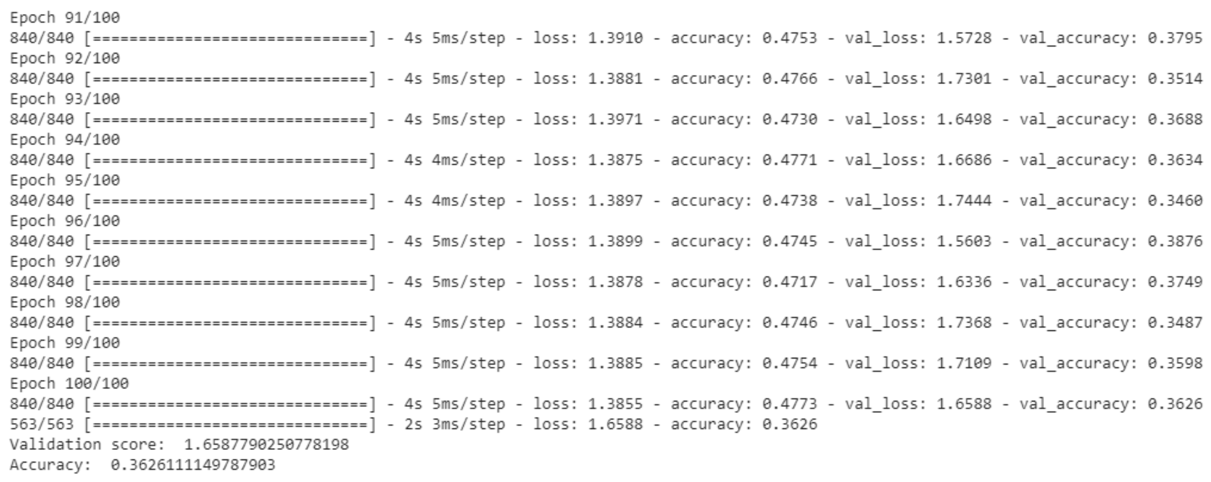
Figure 1: CNN epochs.

2) Instead of increasing the output channel, I increased the kernel size from 3x3 to 4x4 for both con2D operation. However, I got low accuracy. It did not make any changes.
   - Total time: 90 seconds.



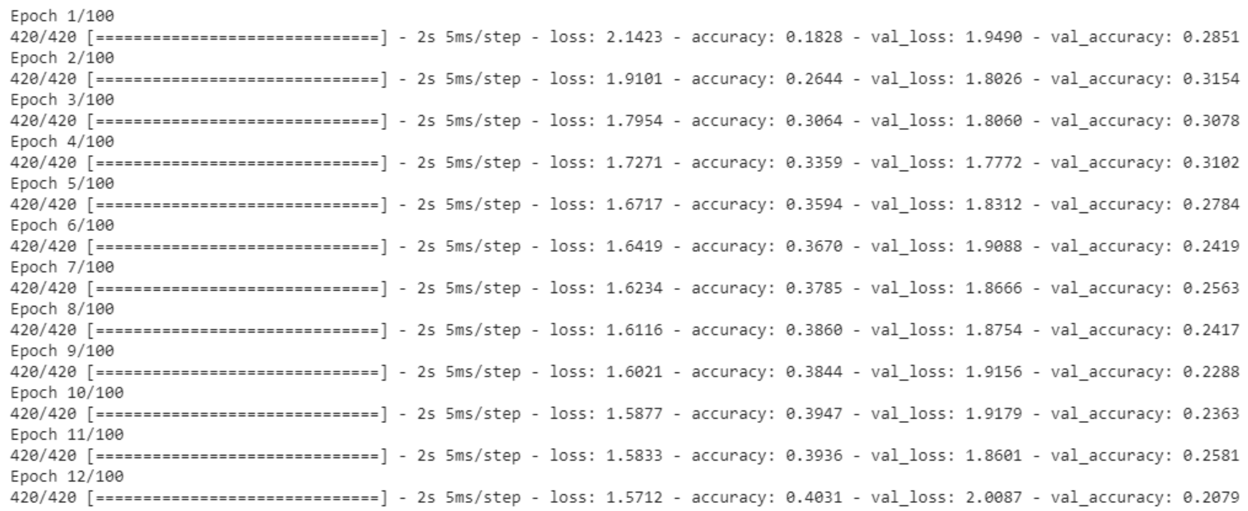
```
Epoch 1/100
420/420 [==============================] - 2s 5ms/step - loss: 2.0647 - accuracy: 0.2110 - val_loss: 1.8247 - val_accuracy: 0.3440
Epoch 2/100
420/420 [==============================] - 2s 5ms/step - loss: 1.9073 - accuracy: 0.2762 - val_loss: 1.7491 - val_accuracy: 0.3663
Epoch 3/100
420/420 [==============================] - 2s 5ms/step - loss: 1.8599 - accuracy: 0.3039 - val_loss: 1.7338 - val_accuracy: 0.3701
Epoch 4/100
420/420 [==============================] - 2s 5ms/step - loss: 1.8241 - accuracy: 0.3223 - val_loss: 1.7261 - val_accuracy: 0.3603
Epoch 5/100
420/420 [==============================] - 2s 5ms/step - loss: 1.8036 - accuracy: 0.3300 - val_loss: 1.6994 - val_accuracy: 0.3689
Epoch 6/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7877 - accuracy: 0.3381 - val_loss: 1.6988 - val_accuracy: 0.3704
Epoch 7/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7751 - accuracy: 0.3439 - val_loss: 1.6822 - val_accuracy: 0.3835
Epoch 8/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7673 - accuracy: 0.3494 - val_loss: 1.6949 - val_accuracy: 0.3587
Epoch 9/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7620 - accuracy: 0.3504 - val_loss: 1.6461 - val_accuracy: 0.4014
Epoch 10/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7495 - accuracy: 0.3537 - val_loss: 1.6718 - val_accuracy: 0.3731
Epoch 11/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7439 - accuracy: 0.3620 - val_loss: 1.6780 - val_accuracy: 0.3766
Epoch 12/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7388 - accuracy: 0.3624 - val_loss: 1.6662 - val_accuracy: 0.3827
Epoch 13/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7339 - accuracy: 0.3649 - val_loss: 1.6527 - val_accuracy: 0.3952
Epoch 14/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7230 - accuracy: 0.3689 - val_loss: 1.6384 - val_accuracy: 0.4018
Epoch 15/100
```

3) For this experiment, I changed the batch size from 100 to 50 to see the performance. This only reduced the run time of the program. Maximum accuracy I received was 0.40.
    - Total time: 85 seconds.



```
Epoch 91/100
840/840 [==============================] - 4s 5ms/step - loss: 1.3910 - accuracy: 0.4753 - val_loss: 1.5728 - val_accuracy: 0.3795
Epoch 92/100
840/840 [==============================] - 4s 5ms/step - loss: 1.3881 - accuracy: 0.4766 - val_loss: 1.7301 - val_accuracy: 0.3514
Epoch 93/100
840/840 [==============================] - 4s 5ms/step - loss: 1.3971 - accuracy: 0.4730 - val_loss: 1.6498 - val_accuracy: 0.3688
Epoch 94/100
840/840 [==============================] - 4s 4ms/step - loss: 1.3875 - accuracy: 0.4771 - val_loss: 1.6686 - val_accuracy: 0.3634
Epoch 95/100
840/840 [==============================] - 4s 4ms/step - loss: 1.3897 - accuracy: 0.4738 - val_loss: 1.7444 - val_accuracy: 0.3460
Epoch 96/100
840/840 [==============================] - 4s 5ms/step - loss: 1.3899 - accuracy: 0.4745 - val_loss: 1.5603 - val_accuracy: 0.3876
Epoch 97/100
840/840 [==============================] - 4s 5ms/step - loss: 1.3878 - accuracy: 0.4717 - val_loss: 1.6336 - val_accuracy: 0.3749
Epoch 98/100
840/840 [==============================] - 4s 5ms/step - loss: 1.3884 - accuracy: 0.4746 - val_loss: 1.7368 - val_accuracy: 0.3487
Epoch 99/100
840/840 [==============================] - 4s 5ms/step - loss: 1.3885 - accuracy: 0.4754 - val_loss: 1.7109 - val_accuracy: 0.3598
Epoch 100/100
840/840 [==============================] - 4s 5ms/step - loss: 1.3855 - accuracy: 0.4773 - val_loss: 1.6588 - val_accuracy: 0.3626
563/563 [==============================] - 2s 3ms/step - loss: 1.6588 - accuracy: 0.3626
Validation score:  1.6587790250778198
Accuracy:  0.3626111149787903
```

4) In third experiment, I changed the pooling function from max to average. As before, my accuracy did not go beyond 0.40 with the output sizes 4 and 2.
    - Total time: 102 seconds.

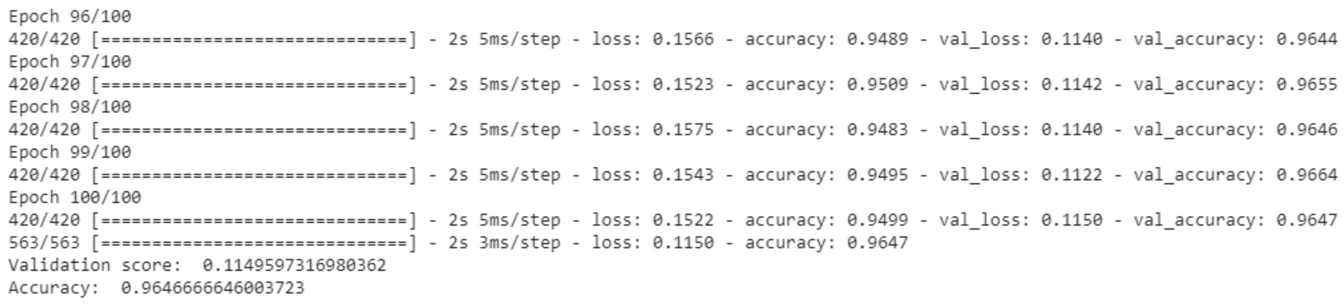


```
Epoch 1/100
420/420 [==============================] - 2s 5ms/step - loss: 2.1423 - accuracy: 0.1828 - val_loss: 1.9490 - val_accuracy: 0.2851
Epoch 2/100
420/420 [==============================] - 2s 5ms/step - loss: 1.9101 - accuracy: 0.2644 - val_loss: 1.8026 - val_accuracy: 0.3154
Epoch 3/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7954 - accuracy: 0.3064 - val_loss: 1.8060 - val_accuracy: 0.3078
Epoch 4/100
420/420 [==============================] - 2s 5ms/step - loss: 1.7271 - accuracy: 0.3359 - val_loss: 1.7772 - val_accuracy: 0.3102
Epoch 5/100
420/420 [==============================] - 2s 5ms/step - loss: 1.6717 - accuracy: 0.3594 - val_loss: 1.8312 - val_accuracy: 0.2784
Epoch 6/100
420/420 [==============================] - 2s 5ms/step - loss: 1.6419 - accuracy: 0.3670 - val_loss: 1.9088 - val_accuracy: 0.2419
Epoch 7/100
420/420 [==============================] - 2s 5ms/step - loss: 1.6234 - accuracy: 0.3785 - val_loss: 1.8666 - val_accuracy: 0.2563
Epoch 8/100
420/420 [==============================] - 2s 5ms/step - loss: 1.6116 - accuracy: 0.3860 - val_loss: 1.8754 - val_accuracy: 0.2417
Epoch 9/100
420/420 [==============================] - 2s 5ms/step - loss: 1.6021 - accuracy: 0.3844 - val_loss: 1.9156 - val_accuracy: 0.2288
Epoch 10/100
420/420 [==============================] - 2s 5ms/step - loss: 1.5877 - accuracy: 0.3947 - val_loss: 1.9179 - val_accuracy: 0.2363
Epoch 11/100
420/420 [==============================] - 2s 5ms/step - loss: 1.5833 - accuracy: 0.3936 - val_loss: 1.8601 - val_accuracy: 0.2581
Epoch 12/100
420/420 [==============================] - 2s 5ms/step - loss: 1.5712 - accuracy: 0.4031 - val_loss: 2.0087 - val_accuracy: 0.2079
```

4) Model Evaluation

- To make sure the models generalizes well on unseen data rather than memorizing it, I used keras.model_evaluate() function to see my loss and accuracy on testing dataset.
- This function returns two values: loss score and accuracy.

Loss-score is a difference between the predicted value and true value. The lower loss-score, the better it generalizes. For this experiment, I changed my output size from 4,2 to 64 and 32, respectively. With the 100 epochs, I got the following values.



```
Epoch 96/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1566 - accuracy: 0.9489 - val_loss: 0.1140 - val_accuracy: 0.9644
Epoch 97/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1523 - accuracy: 0.9509 - val_loss: 0.1142 - val_accuracy: 0.9655
Epoch 98/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1575 - accuracy: 0.9483 - val_loss: 0.1140 - val_accuracy: 0.9646
Epoch 99/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1543 - accuracy: 0.9495 - val_loss: 0.1122 - val_accuracy: 0.9664
Epoch 100/100
420/420 [==============================] - 2s 5ms/step - loss: 0.1522 - accuracy: 0.9499 - val_loss: 0.1150 - val_accuracy: 0.9647
563/563 [==============================] - 2s 3ms/step - loss: 0.1150 - accuracy: 0.9647
Validation score:  0.1149597316980362
Accuracy:  0.9646666646003723
```

- Here, as it can be seen from the figure above, the validation score is 0.11 and accuracy is 0.96.
- Getting this accuracy gives a hint the non-existence of the overfitting issue.