Kubra Iqbal
CSC -578
Kaggle username : kubraiqbal11@gmail.com
Leaderboard : 28

To Build the model – these are all the steps I did and what is the theory behind them.
Building the neural network requires configuring the layers of the model and then further compiling the model. The basic thing of building an NN is the layer. The layer further extracts representations from the data to feed into them. Most of the deep learning consists of chaining together simple layers.

To start with the what the model, according to what was given in the homework question, I adjusted the code with the requirements.

```
#submission 1
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (5, 5), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

The Sequential was added in the top because for most deep learning networks that you build, the Sequential model is likely what you will use. It will allow you to easily stack sequential layers and of the network in order from input to output.
The first line declares the model types as Sequential ().
Next step is that we add a 2D Conventional layer to process the 2D MNIST input images. In this case we have 32 output channels. The next input is kernel_size which is the case we have chosen to be 5x5 moving window flowed by the strides (1,1).
Next the activation function is. Linear unit and supplies the model with the input layer.
Declaring the input shape is only required for the first layer – Keras is good enough to work out the seize.

Next, we add a 2D Max pooling layer – we add the pooling layer and and the x and y directions.

Next, we add another convolution _max pooling layer.

Now that we have built our convolutional layers in the keras tutorial – we want to flatten the output from these to enter the layers. In TensorFLow – we had to figure out what will be the size out the output tensor from the layers to flatten it. The next two lines declare our fully connected layers – using the Dense() layer in Keras. The second from that is our soft-max classification which Is the size of the number of our classes – 10 in our case.

The main idea is that since we have now developed the architecture of the CNN in Keras – we have not specified the loss function. I used this to compile the model (I did make changes to it when I had to submit my code a few times, I will explain each one of those in detail in the end when I am discussing the results).

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(),
              metrics=['accuracy'])
```

The next step is to train the model – which can be done by running this. Since the Question asked us to run 30 epochs, so we specified that when we are training the model.

```
model.fit(x_train,y_train_bin, validation_data= (x_valid, y_valid_bin), ep
ochs =30)
```

Furthermore, run this line of code to predict the model:

```
results_y = model.predict(x_test)
```

And lastly – saved the results and uploaded them on Kaggle.

I worked on 11 models – that I will be explaining each one of them, showing what I changed. My accuracy did not go as low as it should have been, I was also very low on the Leaderboard – I didn't have more time to work on more submissions because they finished for me since it was going to close today. It was interesting to see how everything changed, I feel like I will need more tries and some more learning to do before I can understand how to bring my accuracy down. Below are the 10 models with explanation:

Model 1

For Model 1 – I ran the base model that was already given as I have mentioned above as well. I did not change anything and the accuracy was low as compared to what other models gave me. The accuracy was 0.642 for this model which I thought was higher – and my goal was to make it lower.

```
#submission 1
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (5, 5), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 66s 2ms/step - loss: 0.3301
- acc: 0.8829 - val_loss: 1.6104 - val_acc: 0.6402
```

Model 2

For model 2 – I changed the number to 62 from 32 and ran it. The model did a little better and the accuracy was low – which was something beneficial for this model. Other than changing number 62, there was nothing else that I changed. For this model the accuracy decreased a little bit and went down to 0.6243.

```
#submission 2
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(62, (5, 5), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
```

```
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 85s 2ms/step - loss: 0.1585 -
acc: 0.9446 - val_loss: 2.2150 - val_acc: 0.6243
```

Model 3

For Model 3 – I changed the number from 62 to 32 and increased the numbers to (7,7). Increased them to (7,7), increased my acracy results. Which I think was the reason for the increase.

```
#submission 3
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (7, 7), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 76s 2ms/step - loss: 0.3723
- acc: 0.8702 - val_loss: 1.3402 - val_acc: 0.6387
```

Model 4

For Model 4 – I didn't want to change the number 32 and wanted it to stay the same as it was in the base model but changed the other numbers to (2,2) as compared to (3,3) which were given in the base model. This did not help the model as well as the accuracy increased. The accuracy increased to 0.100

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (2, 2), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 51s 1ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

Model 5

For Model 5 – I increased number from (4,4) to (5,5) – wanted to see if that would help increasing the accuracy. But that didn't really help and the accuracy remained the same.

```
#submission 5
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
```

```
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (4, 4), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 64s 2ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

Model 6

For model 6, I wanted to try something different since my accuracy was not moving at all to any less. I tried increasing the number to 128 which I think was a mistake since it was to high and I changed the numbers from (3,3) to (2,2) – which I also later figured out was too small and I should have not done that. After running the model, the accuracy was still the same as the previous model.

```
#submission 6
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(128, (2, 2), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 95s 2ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

Model 7

For model 7 – I decreased the number from 128 to 32, which was not a huge help and the accuracy again remained the same.

```
#submission 7
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (2, 2), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 54s 1ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

Model 8

For Model 8 – I changed back to 32 and (2,2) remained the same. That did not really help with the accuracy again.

```
#submission 8
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (2, 2), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 55s 1ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

Model 9

For model 9 – I changed the compile function, which has been attached below. I further increased the values from (2,2) to (3,3) and changed the number to 32, hoping it would help. But that didn't help either.

```
#submission 9
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(32, (3, 3), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))


sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])
```

```
Epoch 30/30
40000/40000 [==============================] - 61s 2ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

Model 10

Before doing my model 10, I did some research and found some articles that I thought would be useful in improving the articles. I decided to add the Dropout – I added the dropout thrice, in three multiple locations – that didn't help much either. I used the same compile function.

```
#submission 10
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(32, (3, 3), strides =(1,1),activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 56s 1ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

Model 11

For model 11 – I changed the dropout values to 0.25 instead of 0.5. There was nothing else I changed in this other than this change – that didn't help my model either.

```
#submission 11
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu', input_shape = x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), strides =(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides = (2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Epoch 30/30
40000/40000 [==============================] - 74s 2ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

Model 12

I ran out of chances to post on Kaggle – but I wanted to try something to make my model better. Even though the submission couldn't go up on Kaggle – I wanted to put it here. The accuracy didn't really improve but I made some changes to the model. I added more layers, changed the drop out value and made some other minor changes which I thought would have helped the accuracy get a little better.

```
Epoch 30/30
40000/40000 [==============================] - 151s 4ms/step - loss:
14.5063 - acc: 0.1000 - val_loss: 14.5063 - val_acc: 0.1000
```

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), activation='relu'
, input_shape=x_train.shape[1:]))
```

```
model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), activation='relu'
, input_shape=x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.27))
model.add(Conv2D(64, kernel_size=(3, 3), strides=(1, 1), activation='relu'
, input_shape=x_train.shape[1:]))
model.add(Conv2D(64, kernel_size=(3, 3), strides=(1, 1), activation='relu'
, input_shape=x_train.shape[1:]))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.27))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
```

In the end I would like to say – that Model 2 worked the best for me. Even though I tried so many different things for the model, the accuracy remained the same for me. Adding different numbers, layers and output didn't really help change the accuracy and make it low which would have been the ideal situation. A little more understanding would help me understand how to make the accuracy low.

Reflection :

This homework was very interesting – I really enjoyed the Kaggle project. I have been trying to do these from a long time but never find the time. I think it was really interesting that the homework assignment was to work on it, I got a hang of how it works and will definitely work more on Kaggle now as it also helps to build your portfolio.
The coding part and understanding how to build a model was not that difficult, the lecture and all the articles and other notes in the class helped with that part. I struggled a lot with bringing my accuracy low – that was really the biggest challenge for me.
Even though I was stuck on one part and my accuracy really didn't go as low as it should have been, I did get to learn a lot more by reading so many articles and trying different things about how to bring the accuracy down.
It was also very interesting to compare my results with the others, just to think that how can I bring it lower and understanding or seeing how lower it can go by looking at others results.
All the homework's in this class have been pretty interesting, its like learning a new concept each week and aren't repeating a similar pattern which I think has really helped and made me enjoy the course a lot more.
My biggest challenge was I couldn't bring the accuracy down, I will need to do more research about getting this to work. My model 2 worked the best with the lowest accuracy.