



Supervised Learning AI322

Assignment 3 (CNN Report)

Name	ID
Loai Gamal Mohamed	20180206
Ahmed Kadry Abd El-Shafy	20180018

Prof. Khaled El-Sayed

O-The code:First implementation of the code.

```
# Simple CNN for the MNIST Dataset
import keras.optimizers
from keras.datasets import mnist
from keras.models import Sequential # Allows to build the architecture for neural network
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils

# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# reshape to be [samples][width][height][channels]
X_train = X_train.reshape(60000, 28, 28, 1) # it makes images in grayscale
X_test = X_test.reshape(10000, 28, 28, 1)

# one hot encode outputs
y_train = np_utils.to_categorical(y_train) # it makes label 5 = [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]

# define a simple CNN model
def baseline_model():
    # create model
    model = Sequential()
    model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))
    # Compile model
    optimizer = keras.optimizers.SGD(lr=0.05)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model
```

```

# build the model
model = baseline_model()

# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32, shuffle=True) # epochs : number of iterations when an entire data set is passed forward and backward through the neural network
model.summary()
model.count_params()
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100 - scores[1] * 100))

```

The screenshot shows a Jupyter Notebook interface with a code cell and its output. The code cell contains the following Python code:

```

model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
# Compile model
optimizer = keras.optimizers.SGD(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
baseline_model()

```

The output of the code cell shows the training progress for 10 epochs. The output is as follows:

```

Epoch 3/10
1875/1875 [=====] - 31s 16ms/step - loss: 2.3014 - accuracy: 0.1139 - val_loss: 2.3012 - val_accuracy: 0.1135
Epoch 4/10
1875/1875 [=====] - 35s 19ms/step - loss: 2.3013 - accuracy: 0.1119 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 5/10
1875/1875 [=====] - 31s 17ms/step - loss: 2.3011 - accuracy: 0.1143 - val_loss: 2.3013 - val_accuracy: 0.1135
Epoch 6/10
1875/1875 [=====] - 28s 15ms/step - loss: 2.3015 - accuracy: 0.1101 - val_loss: 2.3015 - val_accuracy: 0.1135
Epoch 7/10
1875/1875 [=====] - 27s 14ms/step - loss: 2.3010 - accuracy: 0.1120 - val_loss: 2.3012 - val_accuracy: 0.1135
Epoch 8/10
1875/1875 [=====] - 29s 15ms/step - loss: 2.3012 - accuracy: 0.1142 - val_loss: 2.3013 - val_accuracy: 0.1135
Epoch 9/10
1875/1875 [=====] - 22s 12ms/step - loss: 2.3018 - accuracy: 0.1111 - val_loss: 2.3015 - val_accuracy: 0.1135
Epoch 10/10
1875/1875 [=====] - 22s 12ms/step - loss: 2.3015 - accuracy: 0.1121 - val_loss: 2.3015 - val_accuracy: 0.1135
CNN Error: 88.65%

Process finished with exit code 0

```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

-Accuracy of the model: $100 - 88.65 = 11.35\%$

1 Epoch 1	0.1107
2 Epoch 2	0.1130
3 Epoch 3	0.1151
4 Epoch 4	0.1109
5 Epoch 5	0.1099

2-The number of parameters in the model:

Total params: 693,802

Trainable params: 693,802

Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10		
1875/1875 [=====]	- 45s	14ms/step
Epoch 2/10		
1875/1875 [=====]	- 26s	14ms/step
Epoch 3/10		
1875/1875 [=====]	- 27s	14ms/step
Epoch 4/10		
1875/1875 [=====]	- 25s	13ms/step
Epoch 5/10		
1875/1875 [=====]	- 24s	13ms/step
Epoch 6/10		
1875/1875 [=====]	- 24s	13ms/step
Epoch 7/10		
1875/1875 [=====]	- 23s	12ms/step
Epoch 8/10		
1875/1875 [=====]	- 23s	12ms/step
Epoch 9/10		
1875/1875 [=====]	- 23s	12ms/step
Epoch 10/10		
1875/1875 [=====]	- 23s	12ms/step

5-The layers of each model (including activations):

- Conv2D

- MaxPooling2D
- Flatten
- Dense
- Rule” activation “
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer :SGD
- Configuration: learing rate
- Lr:0.05

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer :SGD
- Configuration: learing rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you’ve put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- Null

0-The code: we will change the learning rate from 0.05 to 0.009:

```
optimizer = keras.optimizers.SGD(lr=0.009)
```

```

22 y_test = np_utils.to_categorical(y_test)
23 num_classes = y_test.shape[1]
24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
33     model.add(Dense(128, activation='relu'))
34     model.add(Dense(num_classes, activation='softmax'))
35     # Compile model
36     optimizer = keras.optimizers.SGD(lr=0.009)
37     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
38     return model
39
40
41 # build the model
42 model = baseline_model()
43
44 # Fit the model
45 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
46 model.summary()
47 model.count_params()
48 # Final evaluation of the model
49 scores = model.evaluate(X_test, y_test, verbose=0)
50 print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
51

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Layer	Params	Connections
Flatten (Flatten)	(None, 5488)	0
dense (Dense)	(None, 128)	692352
dense_1 (Dense)	(None, 10)	1290

Total params: 693,802
Trainable params: 693,802
Non-trainable params: 0

CNN Error: 5.48%
PS C:\Users\ahmed>

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: $100 - 5.48 = 94.52\%$

1	0.2268
2	0.8661
3	0.8894
4	0.9017
5	0.9152

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10

1875/1875 [=====] - 36s	13ms/step
Epoch 2/10	
1875/1875 [=====] - 24s	13ms/step
Epoch 3/10	
1875/1875 [=====] - 23s	13ms/step
Epoch 4/10	
1875/1875 [=====] - 23s	12ms/step
Epoch 5/10	
1875/1875 [=====] - 23s	12ms/step
Epoch 6/10	
1875/1875 [=====] - 23s	12ms/step
Epoch 7/10	
1875/1875 [=====] - 24s	13ms/step
Epoch 8/10	
1875/1875 [=====] - 23s	12ms/step
Epoch 9/10	
1875/1875 [=====] - 24s	13ms/step
Epoch 10/10	
1875/1875 [=====] - 23s	12ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Rule" activation "
- Softmax "activation"

6-The learning rate used and configuration of the optimizers:

- Optimizer :SGD
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer :SGD

- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- The CNN Error is decreased from **88.65%** to **5.48%**.

0-The code: we will change the batch size from 32 into 64 ($32 * 2$) with the same learning rate of the previous one (0.009).

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward and backward through the neural network
```



```

10: x_train = x_train.reshape(60000, 28, 28, 1) # it makes images in grayscale
11: x_test = x_test.reshape(10000, 28, 28, 1)
12:
13: # one hot encode outputs
14: y_train = np_utils.to_categorical(y_train) # it makes label 5 = [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
15: y_test = np_utils.to_categorical(y_test)
16: num_classes = y_test.shape[1]
17:
18: # define a simple CNN model
19: def baseline_model():
20:     # create model
21:     model = Sequential()
22:     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
23:     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
24:     model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
25:     model.add(Dense(128, activation='relu'))
26:     model.add(Dense(num_classes, activation='softmax'))
27:     # Compile model
28:     optimizer = keras.optimizers.SGD(lr=0.009)
29:     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
30:     return model
31:
32: # build the model
33: model = baseline_model()
34:
35: # fit the model
36: model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
37: model.summary()
38: model.count_params()
39:
40: # final evaluation of the model
41: scores = model.evaluate(x_test, y_test, verbose=0)
42: print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
43:
44:
45:
46:
47:
48:
49:
50:
51:

```

Model Summary:

Layer (type)	Output Shape	Param #	Connected to
flatten (Flatten)	(None, 5488)	0	
dense (Dense)	(None, 128)	692352	flatten (Flatten)
dense_1 (Dense)	(None, 10)	1290	dense (Dense)
Total params: 693,802			
Trainable params: 693,802			
Non-trainable params: 0			

Terminal Output:

```

CNN Error: 1.98%
PS C:\Users\ahmed>

```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy:100-1.98=98.02%

1	0.8267
2	0.9737
3	0.9848
4	0.9908
5	0.9943

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10	
938/938 [=====] - 41s	22ms/step
Epoch 2/10	

938/938 [=====] - 20s	21ms/step
Epoch 3/10	
938/938 [=====] - 21s	22ms/step
Epoch 4/10	
938/938 [=====] - 19s	21ms/step
Epoch 5/10	
938/938 [=====] - 19s	21ms/step
Epoch 6/10	
938/938 [=====] - 19s	21ms/step
Epoch 7/10	
938/938 [=====] - 20s	21ms/step
Epoch 8/10	
938/938 [=====] - 19s	21ms/step
Epoch 9/10	
938/938 [=====] - 19s	21ms/step
Epoch 10/10	
938/938 [=====] - 19s	21ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- ReLU activation “
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer :SGD
- Configuration: learning rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer :SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error decreased then when we increase the batch size from **32** to **64** with the same Learning rate the accuracy will increase, and the error will decrease.

0-The code: we will change all activations from relu to sigmoid with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64)

```
model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='sigmoid'))  
# convolution layer to extract features from the input image  
model.add(Dense(128, activation='sigmoid'))
```

```

D:\>FCAL>3rd level>second>supervised learning>ass1>3.3> taskpy >...
16 X_train = X_train.reshape((60000, 28, 28, 1)) # it makes images in grayscale
17 X_test = X_test.reshape((10000, 28, 28, 1))
18
19 # one hot encode outputs
20 y_train = np_utils.to_categorical(y_train) # it makes label 5 = [ 0, 0, 0, 0, 1, 0, 0, 0, 0]
21 y_test = np_utils.to_categorical(y_test)
22 num_classes = y_test.shape[1]
23
24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (3, 3), input_shape=(28, 28, 1), activation='sigmoid')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
33     model.add(Dense(128, activation='sigmoid'))
34     model.add(Dense(num_classes, activation='softmax'))
35     # Compile model
36     optimizer = keras.optimizers.SGD(lr=0.009)
37     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
38     return model
39
40
41 # build the model
42 model = baseline_model()
43
44 # fit the model
45 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
46 model.summary()
47 model.count_params()
48 # final evaluation of the model
49 scores = model.evaluate(X_test, y_test, verbose=0)
50 print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
51

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

Flatten (Flatten) (None, 5488) 0
dense (Dense) (None, 128) 692352
dense_1 (Dense) (None, 10) 1200
Total params: 693,802
Trainable params: 693,802
Non-trainable params: 0
CNN Error: 5.62%
PS D:\FCAL\3rd level\second\supervised learning\ass1>

```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy:100-5.62=94.38%

1	0.5635
2	0.8558
3	0.8876
4	0.9046
5	0.9154

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10	
938/938 [=====] - 89s	45ms/step
Epoch 2/10	

938/938 [=====] - 40s	42ms/step
Epoch 3/10	
938/938 [=====] - 31s	33ms/step
Epoch 4/10	
938/938 [=====] - 23s	25ms/step
Epoch 5/10	
938/938 [=====] - 24s	26ms/step
Epoch 6/10	
938/938 [=====] - 22s	23ms/step
Epoch 7/10	
938/938 [=====] - 21s	22ms/step
Epoch 8/10	
938/938 [=====] - 19s	21ms/step
Epoch 9/10	
938/938 [=====] - 20s	21ms/step
Epoch 10/10	
938/938 [=====] - 19s	21ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- sigmoid" activation "
- Softmax "activation"

6-The learning rate used and configuration of the optimizers:

- Optimizer :SGD
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer :SGD

- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change all activations from **relu** to **sigmoid**.

0-The code: we will change all activations from relu to softmax with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64)

```
model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='softmax'))  
# convolution layer to extract features from the input image
```

```

model.add(Dense(128, activation='softmax'))

D:\> F:\AI> 3rd level > second > supervised learning > ass1 > 3 > task.py > baseline_model
16 X_train = X_train.reshape(60000, 28, 28, 1) # it makes images in grayscale
17 X_test = X_test.reshape(10000, 28, 28, 1)
18
19
20 # one hot encode outputs
21 y_train = np_utils.to_categorical(y_train) # it makes label 5 - [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
22 y_test = np_utils.to_categorical(y_test)
23 num_classes = y_test.shape[1]
24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='softmax')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
33     model.add(Dense(128, activation='softmax'))
34     model.add(Dense(num_classes, activation='softmax'))
35     # compile model
36     optimizer = keras.optimizers.SGD(lr=0.001)
37     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
38     return model
39
40
41 # build the model
42 model = baseline_model()
43
44 # fit the model
45 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
46 model.summary()
47 model.count_params()
48 # final evaluation of the model
49 scores = model.evaluate(X_test, y_test, verbose=0)
50 print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
51

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Layer	Params	Connections
Flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 128)	692352
dense_1 (Dense)	(None, 10)	1290

Total params: 693,802
Trainable params: 693,802
Non-trainable params: 0

CNN Error: 88.65%

PS D:\FCAI\3rd level\second\supervised learning\ass1\3>

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: $100 - 88.65 = 11.35\%$

1	0.1125
2	0.1103
3	0.1111
4	0.1137
5	0.1117

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10

938/938 [=====] -	41s 31ms/step
Epoch 2/10	
938/938 [=====] -	28s 30ms/step
Epoch 3/10	
938/938 [=====] -	29s 30ms/step
Epoch 4/10	
938/938 [=====] -	27s 28ms/step
Epoch 5/10	
938/938 [=====] -	26s 27ms/step
Epoch 6/10	
938/938 [=====] -	25s 27ms/step
Epoch 7/10	
938/938 [=====] -	25s 27ms/step
Epoch 8/10	
938/938 [=====] -	25s 27ms/step
Epoch 9/10	
938/938 [=====] -	26s 27ms/step
Epoch 10/10	
938/938 [=====] -	25s 27ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer :SGD
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer :SGD
- Configuration: learing rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change all activations from **relu** to **softmax**.

0-The code: we will change all activations from relu to selu with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64).

```
model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='selu'))
# convolution layer to extract features from the input image
model.add(Dense(128, activation='selu'))
model.add(Dense(num_classes, activation='selu'))
```

```
task.py
D:\FCAI\3rd level\second\supervised learning\ass1> task.py
16 X_train = X_train.reshape(60000, 28, 28, 1) # it makes images in grayscale
17 X_test = X_test.reshape(10000, 28, 28, 1)
18
19
20 # one hot encode outputs
21 y_train = np_utils.to_categorical(y_train) # it makes label 5 = [ 0, 0, 0, 0, 0, 1, 0, 0, 0]
22 y_test = np_utils.to_categorical(y_test)
23 num_classes = y_test.shape[1]
24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='selu')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
33     model.add(Dense(128, activation='selu'))
34     model.add(Dense(num_classes, activation='selu'))
35     # compile model
36     optimizer = keras.optimizers.SGD(lr=0.009)
37     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
38     return model
39
40
41 # build the model
42 model = baseline_model()
43
44 # Fit the model
45 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
46 model.summary()
47 model.count_params()
48 # final evaluation of the model
49 scores = model.evaluate(X_test, y_test, verbose=0)
50 print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
51
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Layer	Params	Total
Flatten (Flatten)	(None, 5400)	0
dense (Dense)	(None, 128)	692352
dense_1 (Dense)	(None, 10)	1200

Total params: 693,802
Trainable params: 693,802
Non-trainable params: 0

CNN Error: 90.18%

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: $100 - 90.18 = 9.82\%$

1	0.2243
2	0.5347
3	0.0968
4	0.0979
5	0.0970

2-The number of parameters in the model:

- Total params: 693,802

- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10	
938/938 [=====] -	39s 27ms/step
Epoch 2/10	
938/938 [=====] -	25s 26ms/step
Epoch 3/10	
938/938 [=====] -	23s 24ms/step
Epoch 4/10	
938/938 [=====] -	22s 24ms/step
Epoch 5/10	
938/938 [=====] -	22s 24ms/step
Epoch 6/10	
938/938 [=====] -	25s 27ms/step
Epoch 7/10	
938/938 [=====] -	23s 24ms/step
Epoch 8/10	
938/938 [=====] -	24s 26ms/step
Epoch 9/10	
938/938 [=====] -	22s 24ms/step
Epoch 10/10	
938/938 [=====] -	23s 24ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Selu “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer :SGD
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer :SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change all activations from **relu** to **selu**.

0-The code: we will change the optimizer from SGD to Adam with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64, activation = relu).

```
model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) #  
convolution layer to extract features from the input image
```

```

model.add(Dense(128, activation='relu'))
optimizer = keras.optimizers.Adam(lr=0.009)

# ... (rest of the script) ...

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("On Error: %.3f" % (100 - scores[1] * 100))

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Layer	Params	Trainable
flatten (Flatten)	(None, 5400)	0
dense (Dense)	(None, 128)	692352
dense_1 (Dense)	(None, 10)	1200
Total params:	693,802	
Trainable params:	693,802	
Non-trainable params:	0	

On Error: 4.48%

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: $100 - 4.48 = 95.52\%$

1	0.8572
2	0.9656
3	0.9726
4	0.9730
5	0.9715

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10

938/938 [=====] - 37s	25ms/step
Epoch 2/10	
938/938 [=====] - 22s	24ms/step
Epoch 3/10	
938/938 [=====] - 21s	22ms/step
Epoch 4/10	
938/938 [=====] - 24s	26ms/step
Epoch 5/10	
938/938 [=====] - 23s	25ms/step
Epoch 6/10	
938/938 [=====] - 21s	23ms/step
Epoch 7/10	
938/938 [=====] - 22s	24ms/step
Epoch 8/10	
938/938 [=====] - 21s	22ms/step
Epoch 9/10	
938/938 [=====] - 21s	23ms/step
Epoch 10/10	
938/938 [=====] - 22s	24ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer :Adem
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer :Adem
- Configuration: learing rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

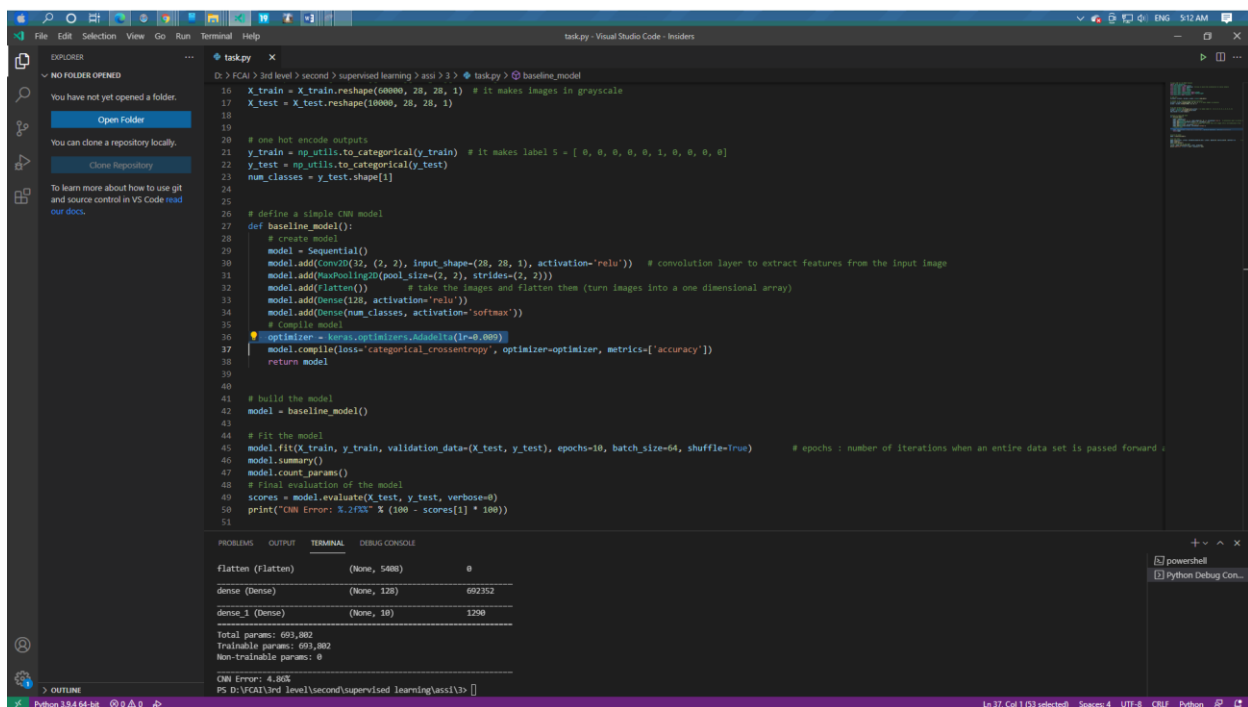
- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change the optimizer from **SGD** to **Adam**.

0-The code: we will change the optimizer from SGD to Adadelata with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64, activation = relu).

```
optimizer = keras.optimizers.Adadelta(lr=0.009)
```



```
D:\FCAT\3rd level\supervised learning\ass1\3> taskpy > baseline_model
16 X_train = X_train.reshape(60000, 28, 28, 1) # it makes images in grayscale
17 X_test = X_test.reshape(10000, 28, 28, 1)
18
19
20 # one hot encode outputs
21 y_train = np_utils.to_categorical(y_train) # it makes label 5 = [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
22 y_test = np_utils.to_categorical(y_test)
23 num_classes = y_test.shape[1]
24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
33     model.add(Dense(128, activation='relu'))
34     model.add(Dense(num_classes, activation='softmax'))
35     # compile model
36     optimizer = keras.optimizers.Adadelta(lr=0.009)
37     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
38     return model
39
40
41 # build the model
42 model = baseline_model()
43
44 # Fit the model
45 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
46 model.summary()
47 model.count_params()
48 # Final evaluation of the model
49 scores = model.evaluate(X_test, y_test, verbose=0)
50 print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
51
```

Layer	Params	Trainable	Size
Flatten (Flatten)	(None, 5408)	0	
dense (Dense)	(None, 128)	69232	
dense_1 (Dense)	(None, 10)	1200	

Total params: 693,882
Trainable params: 693,882
Non-trainable params: 0

CNN Error: 4.86%

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: $100 - 4.86 = 95.14\%$

1	0.5077
2	0.8715
3	0.9053
4	0.9222
5	0.9357

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch & The average test time in each epoch:

Epoch 1/10	
938/938 [=====] - 37s	26ms/step
Epoch 2/10	
938/938 [=====] - 26s	28ms/step
Epoch 3/10	
938/938 [=====] - 25s	26ms/step
Epoch 4/10	
938/938 [=====] - 23s	25ms/step
Epoch 5/10	
938/938 [=====] - 22s	24ms/step
Epoch 6/10	
938/938 [=====] - 22s	23ms/step
Epoch 7/10	
938/938 [=====] - 23s	25ms/step
Epoch 8/10	
938/938 [=====] - 22s	24ms/step
Epoch 9/10	
938/938 [=====] - 21s	23ms/step
Epoch 10/10	
938/938 [=====] - 22s	24ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : Adadelata
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : Adadelata
- Configuration: learing rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you’ve put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change the optimizer from **SGD** to **Adadelata**.

0-The code: we will add dropout layer with dropout rate = 0.2 in the model after max pool layer with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64, activation = relu, optimizer = SGD).

```
model.add(Dropout(0.2))  
optimizer = keras.optimizers.SGD(lr=0.009)
```

```

19
20 # one hot encode outputs
21 y_train = np_utils.to_categorical(y_train) # It makes label 5 = [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
22 y_test = np_utils.to_categorical(y_test)
23 num_classes = y_test.shape[1]
24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Dropout(0.2))
33     model.add(Flatten()) # Take the images and Flatten them (turn images into a one dimensional array)
34     model.add(Dense(128, activation='relu'))
35     model.add(Dense(num_classes, activation='softmax'))
36     # Compile model
37     optimizer = keras.optimizers.Adam(lr=0.0005)
38     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
39     return model
40
41
42 # build the model
43 model = baseline_model()
44
45 # fit the model
46 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
47 model.summary()
48 # Final evaluation of the model
49 scores = model.evaluate(X_test, y_test, verbose=0)
50 print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
51

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Layer	Output Shape	Param #
Flatten (Flatten)	(None, 5408)	0
dense_1 (Dense)	(None, 128)	692352
dense_2 (Dense)	(None, 10)	1200

Total params: 693,802
Trainable params: 693,802
Non-trainable params: 0

CNN Error: 1.69%

PS D:\FCAT\3rd level\second\supervised learning\ass1>

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy : $100-1.69=98.31\%$

1	0.7953
2	0.9589
3	0.9688
4	0.9749
5	0.9789

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10	
938/938 [=====] - 42s	29ms/step
Epoch 2/10	

938/938 [=====] - 26s	27ms/step
Epoch 3/10	
938/938 [=====] - 25s	27ms/step
Epoch 4/10	
938/938 [=====] - 26s	28ms/step
Epoch 5/10	
938/938 [=====] - 25s	27ms/step
Epoch 6/10	
938/938 [=====] - 24s	26ms/step
Epoch 7/10	
938/938 [=====] - 25s	27ms/step
Epoch 8/10	
938/938 [=====] - 26s	27ms/step
Epoch 9/10	
938/938 [=====] - 25s	27ms/step
Epoch 10/10	
938/938 [=====] - 25s	26ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learning rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- add dropout layer with dropout rate = 0.2 in the model after max pool layer

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error decreased from 2.04% to 1.69% when we add dropout layer after max pool layer with dropout rate = 0.2.

0-The code: we will add dropout layer with dropout rate = 0.5 in the model after max pool layer with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64, activation = relu, optimizer = SGD).

```
model.add(Dropout(0.5))
```

```

19
20 # one hot encode outputs
21 y_train = np_utils.to_categorical(y_train) # it makes label 5 = [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
22 y_test = np_utils.to_categorical(y_test)
23 num_classes = y_test.shape[1]
24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Flatten()) # Take the images and flatten them (turn images into a one dimensional array)
33     model.add(Dense(128, activation='relu'))
34     model.add(Dense(num_classes, activation='softmax'))
35     # Compile model
36     optimizer = keras.optimizers.SGD(lr=0.009)
37     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
38     return model
39
40
41
42 # build the model
43 model = baseline_model()
44
45 # Fit the model
46 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
47 model.summary()
48 # final evaluation of the model
49 scores = model.evaluate(X_test, y_test, verbose=0)
50 print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
51

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

Flatten (Flatten) (None, 5408) 0
dense (Dense) (None, 128) 692352
dense_1 (Dense) (None, 10) 1290
Total params: 693,802
Trainable params: 693,802
Non-trainable params: 0
CNN Error: 11.40%
PS D:\VCAT\3rd level\second\supervised learning\3>

```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: $100 - 11.40 = 88.60\%$

1	0.1264
2	0.5525
3	0.6747
4	0.6850
5	0.6908

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch & The average test time in each epoch:

Epoch 1/10 938/938 [=====] - 39s	28ms/step
-------------------------------------	-----------

Epoch 2/10		
938/938 [=====]	- 27s	29ms/step
Epoch 3/10		
938/938 [=====]	- 24s	26ms/step
Epoch 4/10		
938/938 [=====]	- 25s	27ms/step
Epoch 5/10		
938/938 [=====]	- 27s	29ms/step
Epoch 6/10		
938/938 [=====]	- 24s	26ms/step
Epoch 7/10		
938/938 [=====]	- 24s	26ms/step
Epoch 8/10		
938/938 [=====]	- 25s	26ms/step
Epoch 9/10		
938/938 [=====]	- 26s	28ms/step
Epoch 10/10		
938/938 [=====]	- 26s	28ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- add dropout layer with dropout rate = 0.5 in the model after max pool layer

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased from 2.04% to 11.40% when we add dropout layer after max pool layer with dropout rate = 0.5.

0-The code: we will add dropout layer with dropout rate = 0.2 in the model after flatten layer with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64, activation = relu, optimizer = SGD).

```
model.add(Flatten())  
model.add(Dropout(0.2))
```



```

24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
33     model.add(Dropout(0.2))
34
35     model.add(Dense(128, activation='relu'))
36     model.add(Dense(num_classes, activation='softmax'))
37     # compile model
38     optimizer = keras.optimizers.Adam(lr=0.0001)
39     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
40     return model
41
42
43 # build the model
44 model = baseline_model()
45
46 # fit the model
47 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
48 model.summary()
49 # Final evaluation of the model
50 scores = model.evaluate(X_test, y_test, verbose=0)
51 print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
52

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

dropout (Dropout) (None, 5408) 0
dense (Dense) (None, 128) 692352
dense_1 (Dense) (None, 10) 1200
Total params: 693,802
Trainable params: 693,802
Non-trainable params: 0
CNN Error: 88.65%
PS C:\Users\ahmed>

```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: $100 - 88.65\% = 11.35\%$

1	0.1065
2	0.1120
3	0.1118
4	0.1108
5	0.1133

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10 938/938 [=====] - 36s	25ms/step
-------------------------------------	-----------

Epoch 2/10		
938/938 [=====]	- 24s	25ms/step
Epoch 3/10		
938/938 [=====]	- 28s	30ms/step
Epoch 4/10		
938/938 [=====]	- 27s	28ms/step
Epoch 5/10		
938/938 [=====]	- 26s	28ms/step
Epoch 6/10		
938/938 [=====]	- 25s	27ms/step
Epoch 7/10		
938/938 [=====]	- 25s	26ms/step
Epoch 8/10		
938/938 [=====]	- 26s	27ms/step
Epoch 9/10		
938/938 [=====]	- 26s	27ms/step
Epoch 10/10		
938/938 [=====]	- 26s	28ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- add dropout layer with dropout rate = 0.2 in the model after flatten layer

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased from 2.04% to 88.65% when we add dropout layer after flatten layer with dropout rate = 0.2.

0-The code: we will add dropout layer with dropout rate = 0.5 in the model after flatten layer with the same previous parameters (Learning rate = 0.009, Epochs = 10, Batch size = 64, activation = relu, optimizer = SGD).

```
model.add(Flatten())  
model.add(Dropout(0.5))
```

```

24
25
26 # define a simple CNN model
27 def baseline_model():
28     # create model
29     model = Sequential()
30     model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu')) # convolution layer to extract features from the input image
31     model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
32     model.add(Flatten()) # take the images and flatten them (turn images into a one dimensional array)
33     model.add(Dropout(0.5))
34
35     model.add(Dense(128, activation='relu'))
36     model.add(Dense(num_classes, activation='softmax'))
37     # Compile model
38     optimizer = keras.optimizers.Adam(lr=0.001)
39     model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
40     return model
41
42
43 # build the model
44 model = baseline_model()
45
46 # fit the model
47 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
48 model.summary()
49 # Final evaluation of the model
50 scores = model.evaluate(X_test, y_test, verbose=0)
51 print("CNN Error: %.3f%%" % (100 - scores[1] * 100))
52

```

2021-05-17 05:59:52.174459: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
 To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 C:\Users\sham\AppData\Local\Programs\Python\Python38\lib\site-packages\tensorflow\python\keras\optimizer_v2\optimizer_v2.py:174: UserWarning: the 'lr' argument is deprecated, use 'learning_rate' instead.
 warnings.warn(
 2021-05-17 05:59:52.844884: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of the MLIR Optimization Passes are enabled (registered 2)
 Epoch 1/10
 938/938 [=====] - 40s 29ms/step - loss: 2.1380 - accuracy: 0.6988 - val_loss: 0.1616 - val_accuracy: 0.9409
 Epoch 2/10
 938/938 [=====] - 27s 29ms/step - loss: 0.3130 - accuracy: 0.9016 - val_loss: 0.1285 - val_accuracy: 0.9615
 Epoch 3/10

1-Final accuracy of the model and the accuracy in the first 5 epoch:

1	0.6988
2	0.9016
3	0.9156
4	0.9263
5	0.9333

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10	
938/938 [=====] - 40s	29ms/step
Epoch 2/10	
938/938 [=====] - 27s	29ms/step

Epoch 3/10		
938/938 [=====]	- 26s	28ms/step
Epoch 4/10		
938/938 [=====]	- 25s	26ms/step
Epoch 5/10		
938/938 [=====]	- 26s	27ms/step
Epoch 6/10		
938/938 [=====]	- 27s	29ms/step
Epoch 7/10		
938/938 [=====]	- 27s	29ms/step
Epoch 8/10		
938/938 [=====]	- 27s	29ms/step
Epoch 9/10		
938/938 [=====]	- 26s	28ms/step
Epoch 10/10		
938/938 [=====]	- 26s	28ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learing rate
- Lr:0.009

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- add dropout layer with dropout rate = 0.5 in the model after flatten layer

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased from 2.04% to 2.25% when we add dropout layer after flatten layer with dropout rate = 0.5.

0-The code: we will change the learning rate from 0.009 to 0.005 with the same previous parameters (Epochs = 10, Batch size = 64, activation = relu, optimizer = SGD).

```
optimizer = keras.optimizers.SGD(learning_rate=0.005)
```

```

37 # Compile model
38 optimizer = keras.optimizers.SGD(learning_rate=0.005)
39 model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
40 print(model.summary(), '\n')
41 baseline_model()

```

2021-05-17 10:32:46.780524: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of the MLIR Optimization Passes are enabled (registered 2)

Epoch 1/10
938/938 [=====] - 51s 26ms/step - loss: 2.9775 - accuracy: 0.6374 - val_loss: 0.1799 - val_accuracy: 0.9439
Epoch 2/10
938/938 [=====] - 22s 24ms/step - loss: 0.3140 - accuracy: 0.9010 - val_loss: 0.1364 - val_accuracy: 0.9576
Epoch 3/10
938/938 [=====] - 23s 24ms/step - loss: 0.2522 - accuracy: 0.9195 - val_loss: 0.1152 - val_accuracy: 0.9635
Epoch 4/10
938/938 [=====] - 23s 24ms/step - loss: 0.2063 - accuracy: 0.9353 - val_loss: 0.1078 - val_accuracy: 0.9650
Epoch 5/10
938/938 [=====] - 23s 24ms/step - loss: 0.1960 - accuracy: 0.9372 - val_loss: 0.0927 - val_accuracy: 0.9704
Epoch 6/10
938/938 [=====] - 23s 25ms/step - loss: 0.1797 - accuracy: 0.9437 - val_loss: 0.0841 - val_accuracy: 0.9750
Epoch 7/10
938/938 [=====] - 25s 26ms/step - loss: 0.1605 - accuracy: 0.9500 - val_loss: 0.0740 - val_accuracy: 0.9765
Epoch 8/10
938/938 [=====] - 23s 24ms/step - loss: 0.1522 - accuracy: 0.9528 - val_loss: 0.0769 - val_accuracy: 0.9745
Epoch 9/10
938/938 [=====] - 24s 26ms/step - loss: 0.1446 - accuracy: 0.9534 - val_loss: 0.0759 - val_accuracy: 0.9738
Epoch 10/10
938/938 [=====] - 27s 28ms/step - loss: 0.1355 - accuracy: 0.9561 - val_loss: 0.0815 - val_accuracy: 0.9717
CNN Error: 2.83%

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: $100 - 2.83\% = 97.17\%$

1	0.6374
2	0.9010
3	0.9195
4	0.9353
5	0.9372

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10		
938/938 [=====]	- 51s	26ms/step
Epoch 2/10		
938/938 [=====]	- 22s	24ms/step
Epoch 3/10		
938/938 [=====]	- 23s	24ms/step
Epoch 4/10		
938/938 [=====]	- 23s	24ms/step
Epoch 5/10		
938/938 [=====]	- 23s	24ms/step
Epoch 6/10		
938/938 [=====]	- 23s	25ms/step
Epoch 7/10		
938/938 [=====]	- 25s	26ms/step
Epoch 8/10		
938/938 [=====]	- 23s	24ms/step
Epoch 9/10		
938/938 [=====]	- 24s	26ms/step
Epoch 10/10		
938/938 [=====]	- 27s	28ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learing rate
- Lr:0.005

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error decreased when we change the learning rate from **0.009** to **0.005**

0-The code: we will change the learning rate from 0.005 to 0.001 with the same previous parameters (Epochs = 10, Batch size = 64, activation = relu, optimizer = SGD).

```
optimizer = keras.optimizers.SGD(learning_rate=0.001)
```

```

36 model.add(Dense(num_classes, activation='softmax'))
37 # Compile model
38 optimizer = keras.optimizers.SGD(learning_rate=0.001)
39 model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
40 print(model.summary(), '\n')
41 return model
42

```

```

Run: main
2021-09-17 10:30:00.222110: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:210] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/10
938/938 [=====] - 45s 24ms/step - loss: 2.9793 - accuracy: 0.6732 - val_loss: 0.2490 - val_accuracy: 0.9266
Epoch 2/10
938/938 [=====] - 23s 24ms/step - loss: 0.3774 - accuracy: 0.8842 - val_loss: 0.1738 - val_accuracy: 0.9584
Epoch 3/10
938/938 [=====] - 25s 26ms/step - loss: 0.2890 - accuracy: 0.9113 - val_loss: 0.1378 - val_accuracy: 0.9599
Epoch 4/10
938/938 [=====] - 22s 23ms/step - loss: 0.2439 - accuracy: 0.9259 - val_loss: 0.1226 - val_accuracy: 0.9644
Epoch 5/10
938/938 [=====] - 23s 25ms/step - loss: 0.2033 - accuracy: 0.9366 - val_loss: 0.1128 - val_accuracy: 0.9673
Epoch 6/10
938/938 [=====] - 27s 28ms/step - loss: 0.1912 - accuracy: 0.9395 - val_loss: 0.1020 - val_accuracy: 0.9699
Epoch 7/10
938/938 [=====] - 28s 30ms/step - loss: 0.1845 - accuracy: 0.9435 - val_loss: 0.0950 - val_accuracy: 0.9718
Epoch 8/10
938/938 [=====] - 25s 27ms/step - loss: 0.1629 - accuracy: 0.9492 - val_loss: 0.0932 - val_accuracy: 0.9730
Epoch 9/10
938/938 [=====] - 26s 28ms/step - loss: 0.1618 - accuracy: 0.9492 - val_loss: 0.0862 - val_accuracy: 0.9751
Epoch 10/10
938/938 [=====] - 22s 24ms/step - loss: 0.1440 - accuracy: 0.9543 - val_loss: 0.0838 - val_accuracy: 0.9757
CNN Accuracy: 97.57%

```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: 97.57%

1	0.6732
2	0.8842
3	0.9113
4	0.9259
5	0.9366

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10		
938/938 [=====]	- 45s	24ms/step
Epoch 2/10		
938/938 [=====]	- 23s	24ms/step
Epoch 3/10		
938/938 [=====]	- 25s	26ms/step
Epoch 4/10		
938/938 [=====]	- 22s	23ms/step
Epoch 5/10		
938/938 [=====]	- 23s	25ms/step
Epoch 6/10		
938/938 [=====]	- 27s	28ms/step
Epoch 7/10		
938/938 [=====]	- 28s	30ms/step
Epoch 8/10		
938/938 [=====]	- 25s	27ms/step
Epoch 9/10		
938/938 [=====]	- 26s	28ms/step
Epoch 10/10		
938/938 [=====]	- 22s	24ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learing rate
- Lr:0.001

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error decreased when we change the learning rate from **0.005** to **0.001**

0-The code: we will change the learning rate from 0.001 to 0.0001 with the same previous parameters (Epochs = 10, Batch size = 64, activation = relu, optimizer = SGD).

```
optimizer = keras.optimizers.SGD(learning_rate=0.0001)
```

```
Project main.py x
36 model.add(Dense(num_classes, activation='softmax'))
37 # Compile model
38 optimizer = keras.optimizers.SGD(learning_rate=0.0001)
39 model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
40 print(model.summary(), '\n')
41 return model
42
baseline_model()

Runs: main x
Epoch 1/10
938/938 [=====] - 49s 26ms/step - loss: 9.7372 - accuracy: 0.5328 - val_loss: 0.5887 - val_accuracy: 0.8183
Epoch 2/10
938/938 [=====] - 24s 26ms/step - loss: 1.2602 - accuracy: 0.7002 - val_loss: 0.4993 - val_accuracy: 0.8553
Epoch 3/10
938/938 [=====] - 22s 24ms/step - loss: 0.8913 - accuracy: 0.7467 - val_loss: 0.4294 - val_accuracy: 0.8766
Epoch 4/10
938/938 [=====] - 25s 26ms/step - loss: 0.7301 - accuracy: 0.7832 - val_loss: 0.3836 - val_accuracy: 0.8893
Epoch 5/10
938/938 [=====] - 23s 24ms/step - loss: 0.6307 - accuracy: 0.8067 - val_loss: 0.3547 - val_accuracy: 0.8959
Epoch 6/10
938/938 [=====] - 25s 26ms/step - loss: 0.5614 - accuracy: 0.8262 - val_loss: 0.3274 - val_accuracy: 0.9042
Epoch 7/10
938/938 [=====] - 24s 25ms/step - loss: 0.5116 - accuracy: 0.8414 - val_loss: 0.3103 - val_accuracy: 0.9089
Epoch 8/10
938/938 [=====] - 24s 26ms/step - loss: 0.4885 - accuracy: 0.8488 - val_loss: 0.2958 - val_accuracy: 0.9125
Epoch 9/10
938/938 [=====] - 23s 24ms/step - loss: 0.4614 - accuracy: 0.8571 - val_loss: 0.2801 - val_accuracy: 0.9184
Epoch 10/10
938/938 [=====] - 23s 25ms/step - loss: 0.4459 - accuracy: 0.8631 - val_loss: 0.2659 - val_accuracy: 0.9239
CNN Accuracy: 92.30%
```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: 92.30%

1	0.5328
2	0.7002
3	0.7467
4	0.7832
5	0.8067

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/10		
938/938 [=====]	- 49s	26ms/step
Epoch 2/10		
938/938 [=====]	- 24s	26ms/step
Epoch 3/10		
938/938 [=====]	- 22s	24ms/step
Epoch 4/10		
938/938 [=====]	- 25s	26ms/step
Epoch 5/10		
938/938 [=====]	- 23s	24ms/step
Epoch 6/10		
938/938 [=====]	- 25s	26ms/step
Epoch 7/10		
938/938 [=====]	- 24s	25ms/step
Epoch 8/10		
938/938 [=====]	- 24s	26ms/step
Epoch 9/10		
938/938 [=====]	- 23s	24ms/step
Epoch 10/10		
938/938 [=====]	- 23s	25ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learing rate
- Lr:0.0001

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change the learning rate from **0.001** to **0.0001**

0-The code: we will change the number of Epochs from 10 to 13 with the same previous parameters (Learning Rate = 0.001, Batch size = 64, activation = relu, optimizer = SGD).

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=13, batch_size=64, shuffle=True)
```

```

42
43
44 # build the model
45 model = baseline_model()
46
47 # Fit the model
48 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=13, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
49

```

```

Run: main
Epoch 4/13
938/938 [=====] - 25s 26ms/step - loss: 0.2455 - accuracy: 0.9255 - val_loss: 0.1384 - val_accuracy: 0.9587
Epoch 5/13
938/938 [=====] - 23s 24ms/step - loss: 0.2163 - accuracy: 0.9313 - val_loss: 0.1171 - val_accuracy: 0.9643
Epoch 6/13
938/938 [=====] - 26s 28ms/step - loss: 0.1977 - accuracy: 0.9382 - val_loss: 0.1100 - val_accuracy: 0.9686
Epoch 7/13
938/938 [=====] - 27s 29ms/step - loss: 0.1871 - accuracy: 0.9416 - val_loss: 0.0990 - val_accuracy: 0.9722
Epoch 8/13
938/938 [=====] - 27s 29ms/step - loss: 0.1667 - accuracy: 0.9482 - val_loss: 0.0944 - val_accuracy: 0.9711
Epoch 9/13
938/938 [=====] - 27s 29ms/step - loss: 0.1575 - accuracy: 0.9510 - val_loss: 0.0903 - val_accuracy: 0.9721
Epoch 10/13
938/938 [=====] - 30s 32ms/step - loss: 0.1516 - accuracy: 0.9536 - val_loss: 0.0833 - val_accuracy: 0.9741
Epoch 11/13
938/938 [=====] - 25s 27ms/step - loss: 0.1437 - accuracy: 0.9559 - val_loss: 0.0868 - val_accuracy: 0.9728
Epoch 12/13
938/938 [=====] - 23s 24ms/step - loss: 0.1374 - accuracy: 0.9567 - val_loss: 0.0797 - val_accuracy: 0.9756
Epoch 13/13
938/938 [=====] - 24s 26ms/step - loss: 0.1290 - accuracy: 0.9597 - val_loss: 0.0785 - val_accuracy: 0.9750
CNN Accuracy: 97.50%

```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: 97.50%

1	0.6702
2	0.8743
3	0.9105
4	0.9255
5	0.9313

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/13		
938/938 [=====]	- 55s	24ms/step
Epoch 2/13		
938/938 [=====]	- 22s	24ms/step
Epoch 3/13		
938/938 [=====]	- 23s	26ms/step
Epoch 4/13		
938/938 [=====]	- 25s	23ms/step
Epoch 5/13		
938/938 [=====]	- 23s	25ms/step
Epoch 6/13		
938/938 [=====]	- 26s	28ms/step
Epoch 7/13		
938/938 [=====]	- 27s	30ms/step
Epoch 8/13		
938/938 [=====]	- 27s	27ms/step
Epoch 9/13		
938/938 [=====]	- 27s	28ms/step
Epoch 10/13		
938/938 [=====]	- 30s	24ms/step
Epoch 11/13		
938/938 [=====]	- 25s	24ms/step
Epoch 12/13		
938/938 [=====]	- 23s	24ms/step
Epoch 13/13		
938/938 [=====]	- 24s	24ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learning rate
- Lr:0.001

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you've put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change the number of Epochs from **10** to **13**

0-The code: we will change the number of Epochs from 10 to 15 with the same previous parameters (Learning Rate = 0.001, Batch size = 64, activation = relu, optimizer = SGD).

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=64, shuffle=True)
```

```

39 optimizer = keras.optimizers.Adam(learning_rate=0.001)
40 model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
41 print(model.summary(), '\n')
42 return model
43
44 # build the model
45 model = baseline_model()
46
47 # Fit the model
48 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=64, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
49

```

Run: main

```

938/938 [=====] - 21s 22ms/step - loss: 0.1833 - accuracy: 0.9424 - val_loss: 0.1068 - val_accuracy: 0.9681
Epoch 8/15
938/938 [=====] - 20s 21ms/step - loss: 0.1757 - accuracy: 0.9457 - val_loss: 0.0961 - val_accuracy: 0.9708
Epoch 9/15
938/938 [=====] - 21s 23ms/step - loss: 0.1685 - accuracy: 0.9495 - val_loss: 0.0924 - val_accuracy: 0.9729
Epoch 10/15
938/938 [=====] - 20s 21ms/step - loss: 0.1574 - accuracy: 0.9508 - val_loss: 0.0899 - val_accuracy: 0.9735
Epoch 11/15
938/938 [=====] - 22s 23ms/step - loss: 0.1482 - accuracy: 0.9541 - val_loss: 0.0991 - val_accuracy: 0.9697
Epoch 12/15
938/938 [=====] - 19s 20ms/step - loss: 0.1466 - accuracy: 0.9543 - val_loss: 0.0832 - val_accuracy: 0.9741
Epoch 13/15
938/938 [=====] - 19s 21ms/step - loss: 0.1373 - accuracy: 0.9568 - val_loss: 0.0804 - val_accuracy: 0.9751
Epoch 14/15
938/938 [=====] - 20s 21ms/step - loss: 0.1322 - accuracy: 0.9587 - val_loss: 0.0848 - val_accuracy: 0.9732
Epoch 15/15
938/938 [=====] - 20s 21ms/step - loss: 0.1279 - accuracy: 0.9597 - val_loss: 0.0771 - val_accuracy: 0.9768
CNN Accuracy: 97.68%

```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: 97.68%

1	0.6459
2	0.8718
3	0.9057
4	0.9236
5	0.9327

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/15		
938/938 [=====]	- 34s	24ms/step
Epoch 2/15		
938/938 [=====]	- 20s	21ms/step
Epoch 3/15		
938/938 [=====]	- 19s	21ms/step
Epoch 4/15		
938/938 [=====]	- 20s	22ms/step
Epoch 5/15		
938/938 [=====]	- 21s	22ms/step
Epoch 6/15		
938/938 [=====]	- 21s	22ms/step
Epoch 7/15		
938/938 [=====]	- 21s	22ms/step
Epoch 8/15		
938/938 [=====]	- 20s	21ms/step
Epoch 9/15		
938/938 [=====]	- 21s	23ms/step
Epoch 10/15		
938/938 [=====]	- 20s	21ms/step
Epoch 11/15		
938/938 [=====]	- 22s	23ms/step
Epoch 12/15		
938/938 [=====]	- 19s	20ms/step
Epoch 13/15		
938/938 [=====]	- 19s	21ms/step
Epoch 14/15		
938/938 [=====]	- 20s	21ms/step
Epoch 15/15		
938/938 [=====]	- 20s	21ms/step

5-The layers of each model (including activations):

- Conv2D
- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”

- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learning rate
- Lr:0.001

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you’ve put it):

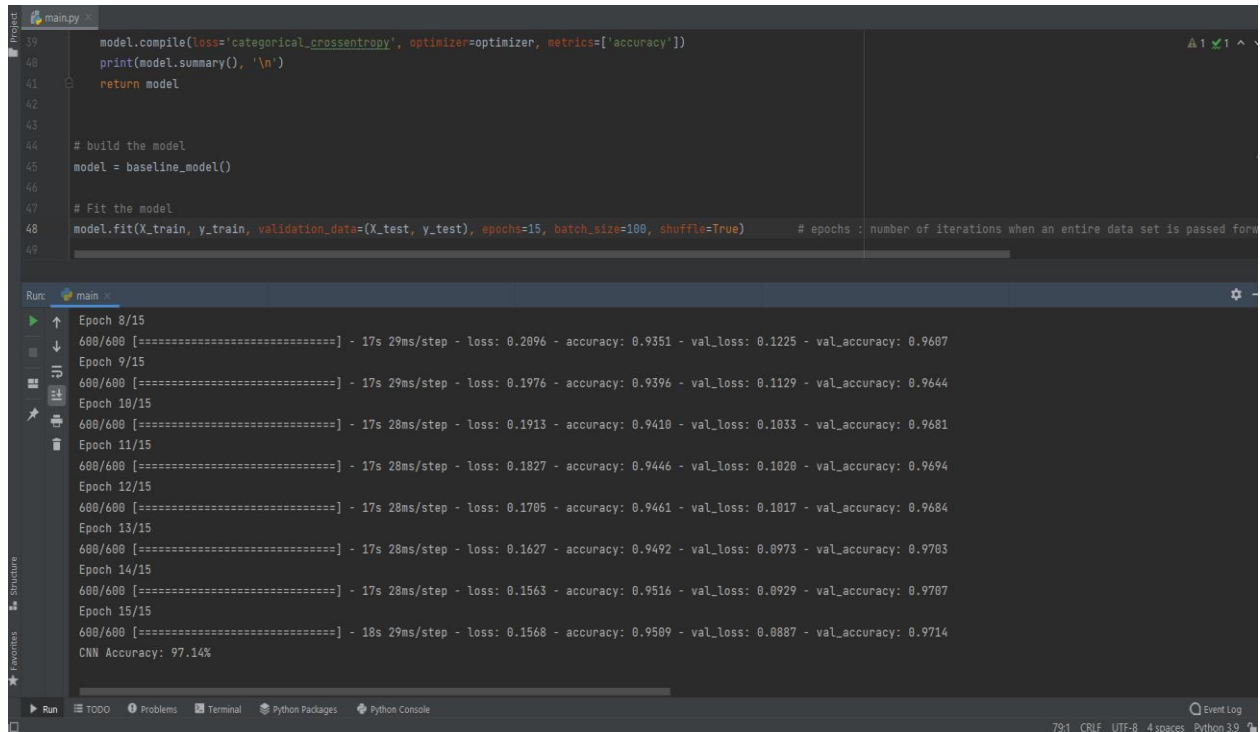
- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error decreased when we change the number of Epochs from **10** to **15**

0-The code: we will change the number of Batch Size from 64 to 100 with the same previous parameters (Learning Rate = 0.001, Epochs = 15, activation = relu, optimizer = SGD).

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=100, shuffle=True)
```



The screenshot shows an IDE with a Python file named 'main.py' and a terminal window. The code in 'main.py' includes:
39 model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
40 print(model.summary(), '\n')
41 return model
42
43
44 # build the model
45 model = baseline_model()
46
47 # Fit the model
48 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=100, shuffle=True)
49
The terminal output shows the progress of the model training over 15 epochs. The output for each epoch includes the number of samples processed (600/600), the time taken per step, the loss, accuracy, validation loss, and validation accuracy. The final output is 'CNN Accuracy: 97.14%'.

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: 97.14%

1	0.6117
2	0.8522
3	0.8883
4	0.9080
5	0.9157

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/15		
600/600[=====] - 29s		30ms/step
Epoch 2/15		
600/600 [=====] - 18s		31ms/step
Epoch 3/15		
600/600[=====] - 17s		29ms/step
Epoch 4/15		
600/600 [=====] - 17s		29ms/step
Epoch 5/15		
600/600[=====] - 17s		28ms/step
Epoch 6/15		
600/600 [=====] - 17s		29ms/step
Epoch 7/15		
600/600[=====] - 17s		29ms/step
Epoch 8/15		
600/600[=====] - 17s		29ms/step
Epoch 9/15		
600/600[=====] - 17s		28ms/step
Epoch 10/15		
600/600[=====] - 17s		28ms/step
Epoch 11/15		
600/600 [=====] - 17s		28ms/step
Epoch 12/15		
600/600[=====] – 17s		28ms/step
Epoch 13/15		
600/600[=====] - 17s		28ms/step
Epoch 14/15		
600/600 [=====] - 17s		28ms/step
Epoch 15/15		
600/600[=====] - 17s		29ms/step

5-The layers of each model (including activations):

- Conv2D

- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learning rate
- Lr:0.001

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you’ve put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change the number of Batch Size from **64** to **100**

0-The code: we will change the number of Batch Size from 64 to 150 with the same previous parameters (Learning Rate = 0.001, Epochs = 15, activation = relu, optimizer = SGD).

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=150, shuffle=True)
```

```
45 model = baseline_model()
46
47 # Fit the model
48 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=150, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
49
```

Run: main

400/400 [=====] - 15s 37ms/step - loss: 0.3030 - accuracy: 0.8807 - val_loss: 0.1001 - val_accuracy: 0.7430
Epoch 5/15
400/400 [=====] - 15s 37ms/step - loss: 0.3221 - accuracy: 0.9005 - val_loss: 0.1755 - val_accuracy: 0.9474
Epoch 6/15
400/400 [=====] - 15s 39ms/step - loss: 0.2830 - accuracy: 0.9131 - val_loss: 0.1616 - val_accuracy: 0.9513
Epoch 7/15
400/400 [=====] - 15s 38ms/step - loss: 0.2672 - accuracy: 0.9171 - val_loss: 0.1435 - val_accuracy: 0.9575
Epoch 8/15
400/400 [=====] - 15s 38ms/step - loss: 0.2429 - accuracy: 0.9244 - val_loss: 0.1334 - val_accuracy: 0.9605
Epoch 9/15
400/400 [=====] - 15s 37ms/step - loss: 0.2300 - accuracy: 0.9286 - val_loss: 0.1278 - val_accuracy: 0.9618
Epoch 10/15
400/400 [=====] - 15s 37ms/step - loss: 0.2129 - accuracy: 0.9340 - val_loss: 0.1234 - val_accuracy: 0.9637
Epoch 11/15
400/400 [=====] - 15s 38ms/step - loss: 0.2087 - accuracy: 0.9353 - val_loss: 0.1186 - val_accuracy: 0.9639
Epoch 12/15
400/400 [=====] - 15s 37ms/step - loss: 0.2010 - accuracy: 0.9372 - val_loss: 0.1141 - val_accuracy: 0.9663
Epoch 13/15
400/400 [=====] - 15s 38ms/step - loss: 0.1974 - accuracy: 0.9396 - val_loss: 0.1087 - val_accuracy: 0.9661
Epoch 14/15
400/400 [=====] - 15s 38ms/step - loss: 0.1889 - accuracy: 0.9416 - val_loss: 0.1057 - val_accuracy: 0.9686
Epoch 15/15
400/400 [=====] - 15s 37ms/step - loss: 0.1789 - accuracy: 0.9446 - val_loss: 0.1031 - val_accuracy: 0.9688
CNN Accuracy: 96.88%

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: 96.88%

1	0.5938
2	0.8203
3	0.8633
4	0.8854
5	0.8966

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/15		
400/400[=====] - 29s		43ms/step
Epoch 2/15		
400/400[=====] - 18s		44ms/step
Epoch 3/15		
400/400[=====] - 17s		43ms/step
Epoch 4/15		
400/400[=====] - 16s		41ms/step
Epoch 5/15		
400/400[=====] - 17s		44ms/step
Epoch 6/15		
400/400 [=====] - 18s		44ms/step
Epoch 7/15		
400/400[=====] – 18s		46ms/step
Epoch 8/15		
400/400[=====] - 17s		42ms/step
Epoch 9/15		
400/400[=====] - 16s		42ms/step
Epoch 10/15		
400/400[=====] - 16s		41ms/step
Epoch 11/15		
400/400 [=====] - 16s		40ms/step
Epoch 12/15		
400/400[=====] – 16s		41ms/step
Epoch 13/15		
400/400[=====] - 16s		41ms/step
Epoch 14/15		
400/400 [=====] - 16s		41ms/step
Epoch 15/15		
400/400[=====] - 16s		41ms/step

5-The layers of each model (including activations):

- Conv2D

- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learning rate
- Lr:0.001

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you’ve put it):

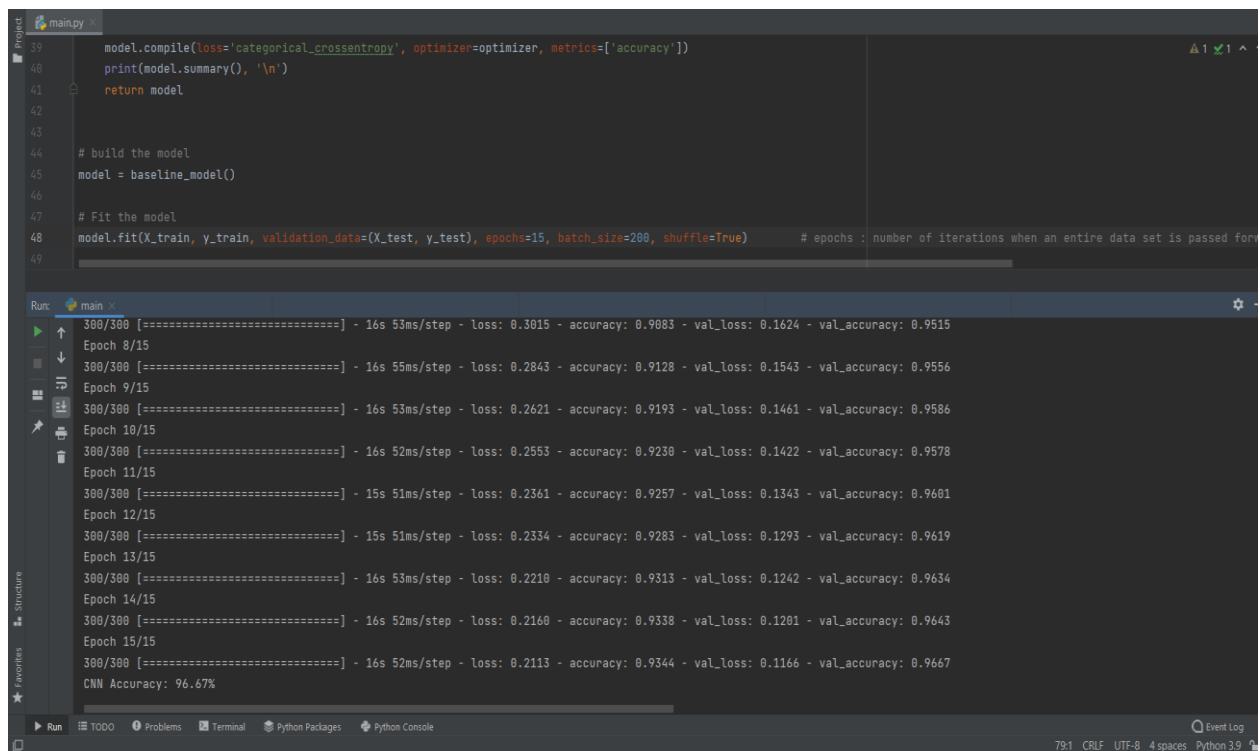
- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change the number of Batch Size from **64** to **150**

0-The code: we will change the number of Batch Size from 64 to 200 with the same previous parameters (Learning Rate = 0.001, Epochs = 15, activation = relu, optimizer = SGD).

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=200, shuffle=True)
```



```
39 model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
40 print(model.summary(), '\n')
41 return model
42
43
44 # build the model
45 model = baseline_model()
46
47 # Fit the model
48 model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=200, shuffle=True) # epochs : number of iterations when an entire data set is passed forward
49
```

Run: main x

```
300/300 [=====] - 16s 53ms/step - loss: 0.3015 - accuracy: 0.9083 - val_loss: 0.1624 - val_accuracy: 0.9515
Epoch 8/15
300/300 [=====] - 16s 55ms/step - loss: 0.2843 - accuracy: 0.9128 - val_loss: 0.1543 - val_accuracy: 0.9556
Epoch 9/15
300/300 [=====] - 16s 53ms/step - loss: 0.2621 - accuracy: 0.9193 - val_loss: 0.1461 - val_accuracy: 0.9586
Epoch 10/15
300/300 [=====] - 16s 52ms/step - loss: 0.2553 - accuracy: 0.9230 - val_loss: 0.1422 - val_accuracy: 0.9578
Epoch 11/15
300/300 [=====] - 15s 51ms/step - loss: 0.2361 - accuracy: 0.9257 - val_loss: 0.1343 - val_accuracy: 0.9601
Epoch 12/15
300/300 [=====] - 15s 51ms/step - loss: 0.2334 - accuracy: 0.9283 - val_loss: 0.1293 - val_accuracy: 0.9619
Epoch 13/15
300/300 [=====] - 16s 53ms/step - loss: 0.2210 - accuracy: 0.9313 - val_loss: 0.1242 - val_accuracy: 0.9634
Epoch 14/15
300/300 [=====] - 16s 52ms/step - loss: 0.2160 - accuracy: 0.9338 - val_loss: 0.1201 - val_accuracy: 0.9643
Epoch 15/15
300/300 [=====] - 16s 52ms/step - loss: 0.2113 - accuracy: 0.9344 - val_loss: 0.1166 - val_accuracy: 0.9667
CNN Accuracy: 96.67%
```

1-Final accuracy of the model and the accuracy in the first 5 epoch:

- Accuracy: 96.67%

1	0.6061
2	0.8093
3	0.8548
4	0.8773
5	0.8895

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/15		
300/300[=====] - 30s		59ms/step
Epoch 2/15		
300/300[=====] - 16s		53ms/step
Epoch 3/15		
300/300[=====] - 17s		56ms/step
Epoch 4/15		
300/300[=====] - 16s		54ms/step
Epoch 5/15		
300/300[=====] - 15s		49ms/step
Epoch 6/15		
300/300 [=====] - 16s		52ms/step
Epoch 7/15		
300/300[=====] – 16s		53ms/step
Epoch 8/15		
300/300[=====] - 16s		55ms/step
Epoch 9/15		
300/300[=====] - 16s		53ms/step
Epoch 10/15		
300/300[=====] - 16s		52ms/step
Epoch 11/15		
300/300 [=====] - 15s		51ms/step
Epoch 12/15		
300/300[=====] – 15s		51ms/step
Epoch 13/15		
300/300[=====] - 16s		53ms/step
Epoch 14/15		
300/300 [=====] - 16s		52ms/step
Epoch 15/15		
300/300[=====] - 16s		52ms/step

5-The layers of each model (including activations):

- Conv2D

- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learing rate
- Lr:0.001

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learing rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you’ve put it):

- Null

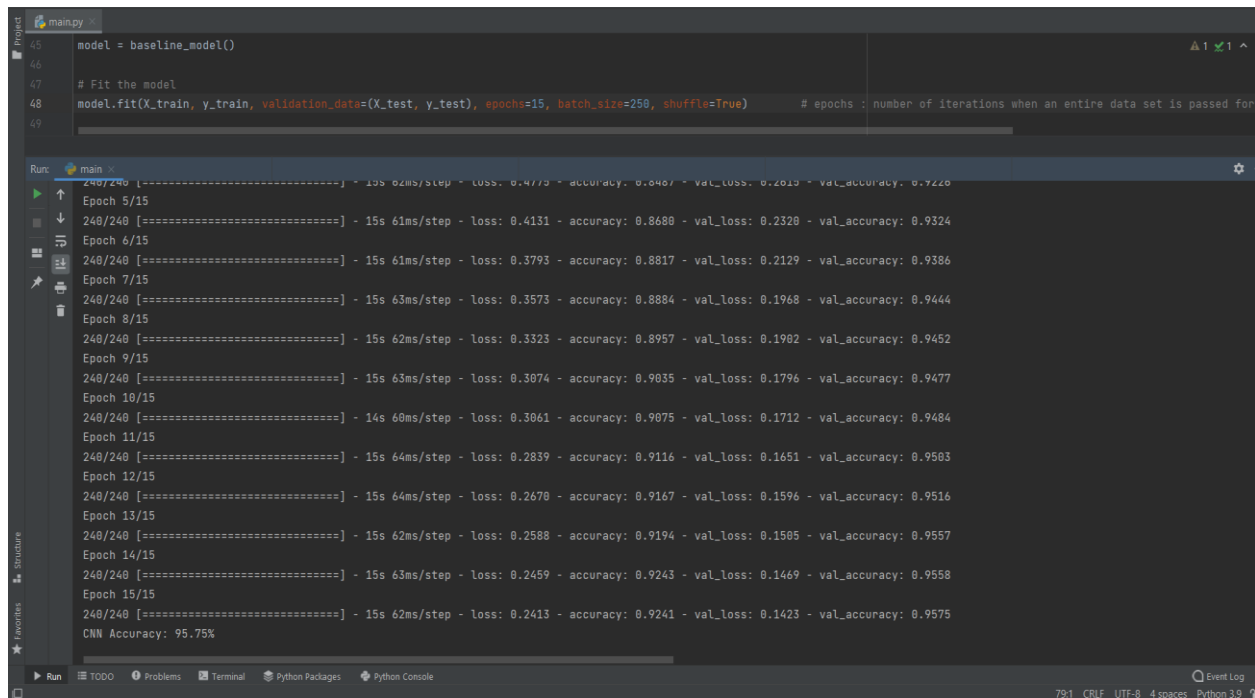
10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change the number of Batch Size from **64** to **200**

0-The code: we will change the number of Batch Size from 64 to 250 with the same previous parameters (Learning Rate = 0.001, Epochs = 15, activation = relu, optimizer = SGD).

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=250, shuffle=True)
```

1-Final accuracy of the model and the accuracy in the first 5 epoch:



```
model = baseline_model()
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15, batch_size=250, shuffle=True) # epochs : number of iterations when an entire data set is passed for
```

```
240/240 [=====] - 15s 62ms/step - loss: 0.4773 - accuracy: 0.8467 - val_loss: 0.2019 - val_accuracy: 0.9420
Epoch 5/15
240/240 [=====] - 15s 61ms/step - loss: 0.4131 - accuracy: 0.8680 - val_loss: 0.2320 - val_accuracy: 0.9324
Epoch 6/15
240/240 [=====] - 15s 61ms/step - loss: 0.3793 - accuracy: 0.8817 - val_loss: 0.2129 - val_accuracy: 0.9386
Epoch 7/15
240/240 [=====] - 15s 63ms/step - loss: 0.3573 - accuracy: 0.8884 - val_loss: 0.1968 - val_accuracy: 0.9444
Epoch 8/15
240/240 [=====] - 15s 62ms/step - loss: 0.3323 - accuracy: 0.8957 - val_loss: 0.1902 - val_accuracy: 0.9452
Epoch 9/15
240/240 [=====] - 15s 63ms/step - loss: 0.3074 - accuracy: 0.9035 - val_loss: 0.1796 - val_accuracy: 0.9477
Epoch 10/15
240/240 [=====] - 14s 60ms/step - loss: 0.3061 - accuracy: 0.9075 - val_loss: 0.1712 - val_accuracy: 0.9484
Epoch 11/15
240/240 [=====] - 15s 64ms/step - loss: 0.2839 - accuracy: 0.9116 - val_loss: 0.1651 - val_accuracy: 0.9503
Epoch 12/15
240/240 [=====] - 15s 64ms/step - loss: 0.2670 - accuracy: 0.9167 - val_loss: 0.1596 - val_accuracy: 0.9516
Epoch 13/15
240/240 [=====] - 15s 62ms/step - loss: 0.2588 - accuracy: 0.9194 - val_loss: 0.1505 - val_accuracy: 0.9557
Epoch 14/15
240/240 [=====] - 15s 63ms/step - loss: 0.2459 - accuracy: 0.9243 - val_loss: 0.1469 - val_accuracy: 0.9558
Epoch 15/15
240/240 [=====] - 15s 62ms/step - loss: 0.2413 - accuracy: 0.9241 - val_loss: 0.1423 - val_accuracy: 0.9575
CNN Accuracy: 95.75%
```

- Accuracy: 95.75%

1	0.4920
2	0.7563
3	0.8193
4	0.8487
5	0.8680

2-The number of parameters in the model:

- Total params: 693,802
- Trainable params: 693,802
- Non-trainable params: 0

3&4- The average time to train in each epoch &The average test time in each epoch:

Epoch 1/15		
240/240[=====]	- 26s	64ms/step
Epoch 2/15		
240/240 [=====]	- 15s	62ms/step
Epoch 3/15		
240/240 [=====]	- 15s	62ms/step
Epoch 4/15		
240/240 [=====]	- 15s	62ms/step
Epoch 5/15		
240/240 [=====]	- 15s	61ms/step
Epoch 6/15		
240/240 [=====]	- 15s	61ms/step
Epoch 7/15		
240/240 [=====]	- 15s	63ms/step
Epoch 8/15		
240/240 [=====]	- 15s	62ms/step
Epoch 9/15		
240/240 [=====]	- 15s	63ms/step
Epoch 10/15		
240/240 [=====]	- 14s	60ms/step
Epoch 11/15		
240/240 [=====]	- 15s	64ms/step
Epoch 12/15		
240/240 [=====]	- 15s	64ms/step
Epoch 13/15		
240/240 [=====]	- 15s	62ms/step
Epoch 14/15		
240/240 [=====]	- 15s	63ms/step
Epoch 15/15		
240/240 [=====]	- 15s	62ms/step

5-The layers of each model (including activations):

- Conv2D

- MaxPooling2D
- Flatten
- Dense
- Dropout
- Relu “activation”
- Softmax “activation”

6-The learning rate used and configuration of the optimizers:

- Optimizer : SGD
- Configuration: learning rate
- Lr:0.001

7-You are not required to test learning decays, if you wish to include it, then include it:

- Null

8-The optimizer used with its configuration:

- Optimizer : SGD
- Configuration: learning rate

9-If you used a dropout layer (you will use it in #7), write where you put it, what dropout rate used and why (your perspective on the location you’ve put it):

- Null

10-Write what you observed has changed due to the parameter you changed (according to the above part):

- As you see that the CNN Error increased when we change the number of Batch Size from **64** to **250**

So, the best model we made is:

```
# Simple CNN for the MNIST Dataset
import keras.optimizers
from keras.datasets import mnist
from keras.models import Sequential # Allows to build the architecture for
neural network
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils.np_utils import to_categorical

# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# reshape to be [samples][width][height][channels]
X_train = X_train.reshape(60000, 28, 28, 1) # it makes images in grayscale
X_test = X_test.reshape(10000, 28, 28, 1)

# one hot encode outputs
y_train = to_categorical(y_train) # it makes label 5 = [ 0, 0, 0, 0, 0, 1,
0, 0, 0, 0]
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# define a simple CNN model
def baseline_model():
    # create model
    model = Sequential()
    model.add(Conv2D(32, (2, 2), input_shape=(28, 28, 1), activation='relu'))
# convolution layer to extract features from the input image
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Flatten()) # take the images and flatten them (turn
images into a one dimensional array)
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))
    # Compile model
    optimizer = keras.optimizers.SGD(learning_rate=0.001)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])
    print(model.summary(), '\n')
    return model

# build the model
model = baseline_model()
```

```
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=15,
batch_size=64, shuffle=True)      # epochs : number of iterations when an
entire data set is passed forward and backward through the neural network

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN Accuracy: %.2f%%" % (scores[1] * 100), '\n')
```