Cairo University
Faculty of Computers and Information

Assignment 3

Supervised learning

18th May 2021

Marina Moheb Nafee 20180208

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(2,
2),strides=(2,2),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(32, (3, 3), activation='relu'))#
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

()model.summary

opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
,history = model.fit(x_train, y_train
,batch_size=batch_size
,epochs=epochs,shuffle=True
,verbose=1
(validation_data=(x_test, y_test)
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 14, 14, 32)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
dense (Dense)                (None, 128)               200832
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 202,282
Trainable params: 202,282
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
938/938 [==============================] - 36s 39ms/step - loss: 0.4835 -
accuracy: 0.8524 - val_loss: 0.2346 - val_accuracy: 0.9246
Epoch 2/10
938/938 [==============================] - 33s 35ms/step - loss: 0.2169 -
accuracy: 0.9321 - val_loss: 0.1592 - val_accuracy: 0.9499
Epoch 3/10
```

938/938 [==============================] - 32s 35ms/step - loss: 0.1531 - accuracy: 0.9526 - val_loss: 0.1299 - val_accuracy: 0.9574
Epoch 4/10
938/938 [==============================] - 32s 34ms/step - loss: 0.1183 - accuracy: 0.9636 - val_loss: 0.1237 - val_accuracy: 0.9597
Epoch 5/10
938/938 [==============================] - 33s 35ms/step - loss: 0.0971 - accuracy: 0.9704 - val_loss: 0.0994 - val_accuracy: 0.9669
Test loss: 0.06670215725898743
Test accuracy: 0.9790999889373779

Point 2

```
()model = Sequential
model.add(Conv2D(32, kernel_size=(2,
2),strides=(2,2),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

()model.summary

opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
,history = model.fit(x_train, y_train
,batch_size=batch_size
,epochs=epochs ,shuffle=True
,verbose=1
(validation_data=(x_test, y_test)
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 14, 14, 32)	160
max_pooling2d_1 (MaxPooling2	(None, 7, 7, 32)	0
flatten_1 (Flatten)	(None, 1568)	0
dense_2 (Dense)	(None, 128)	200832
dense_3 (Dense)	(None, 10)	1290

```
=================================================================
Total params: 202,282
Trainable params: 202,282
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
938/938 [==============================] - 35s 38ms/step - loss: 0.4769 -
accuracy: 0.8533 - val_loss: 0.2704 - val_accuracy: 0.9153
Epoch 2/10
938/938 [==============================] - 34s 36ms/step - loss: 0.2125 -
accuracy: 0.9339 - val_loss: 0.1702 - val_accuracy: 0.9460
Epoch 3/10
938/938 [==============================] - 33s 35ms/step - loss: 0.1491 -
accuracy: 0.9527 - val_loss: 0.1289 - val_accuracy: 0.9594
Epoch 4/10
938/938 [==============================] - 30s 32ms/step - loss: 0.1127 -
accuracy: 0.9652 - val_loss: 0.1022 - val_accuracy: 0.9673
Epoch 5/10
938/938 [==============================] - 31s 33ms/step - loss: 0.0936 -
accuracy: 0.9715 - val_loss: 0.0887 - val_accuracy: 0.9704
Test loss: 0.08163710683584213
Test accuracy: 0.9728000164031982
The layers of model 0: Convolution layer,pooling layer, Flattening layer, Full
Connection layer, Output Layer
The learning rate used:0.01 and  optimizers:SGD
```

Test different epochs

Model 1

```
()model = Sequential
model.add(Conv2D(32, kernel_size=(2,
2),strides=(2,2),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

()model.summary

opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
,history = model.fit(x_train, y_train
,batch_size=batch_size
,epochs=5 ,shuffle=True
,verbose=1
(validation_data=(x_test, y_test)
score = model.evaluate(x_test, y_test, verbose=0)
```

```
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 14, 14, 32)        160
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               200832
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 202,282
Trainable params: 202,282
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
938/938 [==============================] - 38s 40ms/step - loss: 0.4675 -
accuracy: 0.8582 - val_loss: 0.2495 - val_accuracy: 0.9240
Epoch 2/5
938/938 [==============================] - 34s 36ms/step - loss: 0.2052 -
accuracy: 0.9352 - val_loss: 0.1467 - val_accuracy: 0.9515
Epoch 3/5
938/938 [==============================] - 35s 37ms/step - loss: 0.1431 -
accuracy: 0.9559 - val_loss: 0.1161 - val_accuracy: 0.9632
Epoch 4/5
938/938 [==============================] - 34s 36ms/step - loss: 0.1120 -
accuracy: 0.9657 - val_loss: 0.0974 - val_accuracy: 0.9679
Epoch 5/5
938/938 [==============================] - 34s 36ms/step - loss: 0.0924 -
accuracy: 0.9719 - val_loss: 0.0810 - val_accuracy: 0.9736
Test loss: 0.08101462572813034
Test accuracy: 0.9735999703407288
The layers of model 1: Convolution layer,pooling layer,  Flattening layer, Full
Connection layer, Output Layer
The learning rate used:0.01 and  optimizers:SGD
```

Model2

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(2,
2),strides=(2,2),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Flatten())
```

```
model.add(Dense(units=128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

opt = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=8, ,shuffle=True
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 14, 14, 32)        160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               200832
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
=================================================================
Total params: 202,282
Trainable params: 202,282
Non-trainable params: 0
_________________________________________________________________
Epoch 1/8
938/938 [==============================] - 37s 40ms/step - loss: 0.4692 -
accuracy: 0.8580 - val_loss: 0.2519 - val_accuracy: 0.9202
Epoch 2/8
938/938 [==============================] - 35s 37ms/step - loss: 0.1956 -
accuracy: 0.9392 - val_loss: 0.1482 - val_accuracy: 0.9524
Epoch 3/8
938/938 [==============================] - 35s 38ms/step - loss: 0.1386 -
accuracy: 0.9571 - val_loss: 0.1189 - val_accuracy: 0.9614
Epoch 4/8
938/938 [==============================] - 37s 40ms/step - loss: 0.1098 -
accuracy: 0.9659 - val_loss: 0.0973 - val_accuracy: 0.9686
Epoch 5/8
```

938/938 [==============================] - 37s 39ms/step - loss: 0.0914 - accuracy: 0.9723 - val_loss: 0.0836 - val_accuracy: 0.9709
Test loss: 0.07921337336301804
Test accuracy: 0.9731000065803528

The layers of model 2: Convolution layer,pooling layer ,Flattening layer, Full Connection layer, Output Layer
The learning rate used:0.01 and optimizers:SGD
When the epoch is 10 the speed is faster and and loss decrease and accuracy increase because it take time to learn from data

Test different LEARNING RATE

Model 1

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(2,
2),strides=(2,2),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
#change learning rate
opt = SGD(lr=0.05, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=epochs, ,shuffle=True,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 14, 14, 32)	160
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 32)	0
flatten_4 (Flatten)	(None, 1568)	0
dense_8 (Dense)	(None, 128)	200832
dense_9 (Dense)	(None, 10)	1290

Total params: 202,282
Trainable params: 202,282
Non-trainable params: 0

_____

Epoch 1/10
938/938 [==============================] - 44s 47ms/step - loss: 0.3003 - accuracy: 0.9061 - val_loss: 0.1319 - val_accuracy: 0.9547
Epoch 2/10
938/938 [==============================] - 32s 34ms/step - loss: 0.1044 - accuracy: 0.9678 - val_loss: 0.0911 - val_accuracy: 0.9695
Epoch 3/10
938/938 [==============================] - 29s 30ms/step - loss: 0.0757 - accuracy: 0.9768 - val_loss: 0.0752 - val_accuracy: 0.9742
Epoch 4/10
938/938 [==============================] - 28s 30ms/step - loss: 0.0593 - accuracy: 0.9813 - val_loss: 0.0736 - val_accuracy: 0.9752
Epoch 5/10
938/938 [==============================] - 28s 29ms/step - loss: 0.0487 - accuracy: 0.9843 - val_loss: 0.0741 - val_accuracy: 0.9766
Test loss: 0.08227087557315826
Test accuracy: 0.9767000079154968

The layers of model 1: Convolution layer,pooling layer, Flattening layer, Full Connection layer, Output Layer
The learning rate used:0.05 and optimizers:SGD

Model2

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(2,
2),strides=(2,2),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()#change learning rate
opt = SGD(lr=0.5, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=epochs, ,shuffle=True
       , verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
```

```
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 14, 14, 32)        160
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               200832
_________________________________________________________________
dense_13 (Dense)             (None, 10)                1290
=================================================================
Total params: 202,282
Trainable params: 202,282
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
938/938 [==============================] - 25s 27ms/step - loss: 2.2530 -
accuracy: 0.1437 - val_loss: 2.3279 - val_accuracy: 0.1135
Epoch 2/10
938/938 [==============================] - 22s 23ms/step - loss: 2.3189 -
accuracy: 0.1019 - val_loss: 2.3309 - val_accuracy: 0.0892
Epoch 3/10
938/938 [==============================] - 20s 21ms/step - loss: 2.3203 -
accuracy: 0.1021 - val_loss: 2.3122 - val_accuracy: 0.1032
Epoch 4/10
938/938 [==============================] - 23s 25ms/step - loss: 2.3199 -
accuracy: 0.1035 - val_loss: 2.3194 - val_accuracy: 0.0892
Epoch 5/10
938/938 [==============================] - 24s 25ms/step - loss: 2.3196 -
accuracy: 0.1035 - val_loss: 2.3240 - val_accuracy: 0.0982
Test loss: 2.3139781951904297
Test accuracy: 0.11349999904632568
```

The layers of model 2: Convolution layer,pooling layer, Flattening layer, Full Connection layer, Output(activation) Layer
The learning rate used:0.5 and optimizers:SGD

Model3

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(2,
2),strides=(2,2),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Flatten())
```

```
model.add(Dense(units=128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()#change learning rate
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=epochs,shuffle=True
        ,verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Model: "sequential_7"

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 14, 14, 32)	160
max_pooling2d_7 (MaxPooling2	(None, 7, 7, 32)	0
flatten_7 (Flatten)	(None, 1568)	0
dense_14 (Dense)	(None, 128)	200832
dense_15 (Dense)	(None, 10)	1290

Total params: 202,282
Trainable params: 202,282
Non-trainable params: 0

```
Epoch 1/10
938/938 [==============================] - 24s 26ms/step - loss: 0.2591 -
accuracy: 0.9187 - val_loss: 0.1136 - val_accuracy: 0.9635
Epoch 2/10
938/938 [==============================] - 25s 27ms/step - loss: 0.1083 -
accuracy: 0.9654 - val_loss: 0.1063 - val_accuracy: 0.9634
Epoch 3/10
938/938 [==============================] - 23s 25ms/step - loss: 0.0776 -
accuracy: 0.9748 - val_loss: 0.0887 - val_accuracy: 0.9704
Epoch 4/10
938/938 [==============================] - 23s 25ms/step - loss: 0.0595 -
accuracy: 0.9807 - val_loss: 0.0864 - val_accuracy: 0.9715
Epoch 5/10
```

938/938 [==============================] - 24s 26ms/step - loss: 0.0490 - accuracy: 0.9837 - val_loss: 0.0961 - val_accuracy: 0.9694
Test loss: 0.09202394634485245
Test accuracy: 0.9785000085830688
The layers of model 3: Convolution layer,pooling layer, Flattening layer , Full Connection layer, Output Layer
The learning rate used:0.09 and optimizers:SGD
The best learning rate is 0.09 as the learning rate increase the accuracy of the model increase and time of execution decreases

Four models of adding or removing parameter and layers

Model 1 by adding another convolution layer

#model1

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),strides=(2,
2),activation='relu',input_shape=input_shape))
model.add(Conv2D(128, (3, 3),padding="valid", activation='relu'))
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=epochs,
       ,shuffle=True, verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Model: "sequential_8"

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 13, 13, 32)	320
conv2d_9 (Conv2D)	(None, 11, 11, 128)	36992
flatten_8 (Flatten)	(None, 15488)	0
dense_16 (Dense)	(None, 128)	1982592
dense_17 (Dense)	(None, 10)	1290

Total params: 2,021,194

Trainable params: 2,021,194
Non-trainable params: 0
_____
Epoch 1/10
938/938 [==============================] - 152s 162ms/step - loss: 0.1754 - accuracy: 0.9488 - val_loss: 0.0646 - val_accuracy: 0.9787
Epoch 2/10
938/938 [==============================] - 149s 159ms/step - loss: 0.0527 - accuracy: 0.9840 - val_loss: 0.0455 - val_accuracy: 0.9858
Epoch 3/10
938/938 [==============================] - 144s 154ms/step - loss: 0.0307 - accuracy: 0.9905 - val_loss: 0.0452 - val_accuracy: 0.9855
Epoch 4/10
938/938 [==============================] - 147s 157ms/step - loss: 0.0223 - accuracy: 0.9933 - val_loss: 0.0468 - val_accuracy: 0.9873
Epoch 5/10
938/938 [==============================] - 148s 157ms/step - loss: 0.0185 - accuracy: 0.9942 - val_loss: 0.0548 - val_accuracy: 0.9841
Test loss: 0.0600612498819828
Test accuracy: 0.9879999756813049
The layers of model 1: Convolution layer,convolution layer, Flattening layer, Full Connection layer, Output Layer
The learning rate used:0.09 and optimizers:SGD
Althought it has higher accuracy the time of excutetion increase double and the parameter increase

model2 by change kernel size

#model2

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Flatten())
model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(10, activation='softmax'))

model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy, metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=epochs,
        ,shuffle=True ,verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
```

```
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
flatten_9 (Flatten)          (None, 5408)              0
_________________________________________________________________
dense_18 (Dense)             (None, 100)               540900
_________________________________________________________________
dense_19 (Dense)             (None, 10)                1010
=================================================================
Total params: 542,230
Trainable params: 542,230
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
938/938 [==============================] - 76s 81ms/step - loss: 0.1806 -
accuracy: 0.9470 - val_loss: 0.0846 - val_accuracy: 0.9726
Epoch 2/10
938/938 [==============================] - 75s 80ms/step - loss: 0.0145 -
accuracy: 0.9949 - val_loss: 0.0667 - val_accuracy: 0.9833
Epoch 8/10
938/938 [==============================] - 73s 78ms/step - loss: 0.0101 -
accuracy: 0.9964 - val_loss: 0.0783 - val_accuracy: 0.9840
Epoch 9/10
938/938 [==============================] - 76s 81ms/step - loss: 0.0110 -
accuracy: 0.9966 - val_loss: 0.0942 - val_accuracy: 0.9813
Epoch 10/10
938/938 [==============================] - 83s 88ms/step - loss: 0.0088 -
accuracy: 0.9970 - val_loss: 0.0771 - val_accuracy: 0.9838
Test loss: 0.0770789086818695
Test accuracy: 0.9837999939918518
The layers of model 1: Convolution layer,pooling layer, Flattening layer, Full
Connection layer, Output Layer
The learning rate used:0.09 and  optimizers:SGD
Faster than model 2 but less accuracy than model1 and lesser parameter
```

model3 by adding 2 cnn layers and change of momentum

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
```

```
            activation='relu',
            input_shape=input_shape))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))

model.summary()
opt = SGD(lr=0.09, momentum=0.7)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=epochs,shuffle=True
        .verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 24, 24, 128)       36992
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 22, 22, 64)        73792
_________________________________________________________________
flatten_10 (Flatten)         (None, 30976)             0
_________________________________________________________________
dense_20 (Dense)             (None, 10)                309770
=================================================================
Total params: 420,874
Trainable params: 420,874
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
938/938 [==============================] - 940s 1s/step - loss: 2.3047 - accuracy:
0.1189 - val_loss: 2.3021 - val_accuracy: 0.1135
Epoch 2/10
938/938 [==============================] - 889s 947ms/step - loss: 2.3024 - accuracy:
0.1098 - val_loss: 2.3030 - val_accuracy: 0.1028
Epoch 3/10
938/938 [==============================] - 1234s 1s/step - loss: 2.3022 - accuracy:
0.1089 - val_loss: 2.3016 - val_accuracy: 0.1135
Epoch 4/10
938/938 [==============================] - 1309s 1s/step - loss: 2.3024 - accuracy:
0.1086 - val_loss: 2.3019 - val_accuracy: 0.0982
Epoch 5/10
938/938 [==============================] - 1243s 1s/step - loss: 2.3022 - accuracy:
0.1086 - val_loss: 2.3026 - val_accuracy: 0.1135
```

The layers of model 3: Convolution layer,convolution layer, convolution layer, Flattening layer, Output Layer
The learning rate used:0.09 and optimizers:SGD
It is very slow and had a very bad accuracy as number of convolution layer increase accuracy decrease it is very hard to processe

Model4

```
()model = Sequential
model.add(Conv2D(32, kernel_size=(5,
5),activation='relu',input_shape=input_shape))
model.add(MaxPooL2D(pool_size=2, strides=2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))


()model.summary
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
,history = model.fit(x_train, y_train
,batch_size=batch_size
,epochs=epochs,shuffle=True
,verbose=1
(validation_data=(x_test, y_test)
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Model: "sequential_12"

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 24, 24, 32)	832
max_pooling2d_9 (MaxPooling2	(None, 12, 12, 32)	0
conv2d_16 (Conv2D)	(None, 10, 10, 128)	36992
max_pooling2d_10 (MaxPooling	(None, 3, 3, 128)	0
flatten_11 (Flatten)	(None, 1152)	0
dense_21 (Dense)	(None, 10)	11530

Total params: 49,354
Trainable params: 49,354
Non-trainable params: 0

```
_________________________________________________________________
Epoch 1/10
938/938 [==============================] - 145s 155ms/step - loss: 0.1784 -
accuracy: 0.9467 - val_loss: 0.0529 - val_accuracy: 0.9837
Epoch 2/10
938/938 [==============================] - 141s 151ms/step - loss: 0.0601 -
accuracy: 0.9812 - val_loss: 0.0530 - val_accuracy: 0.9825
Epoch 3/10
938/938 [==============================] - 136s 145ms/step - loss: 0.0451 -
accuracy: 0.9862 - val_loss: 0.0567 - val_accuracy: 0.9818
Epoch 4/10
938/938 [==============================] - 134s 143ms/step - loss: 0.0378 -
accuracy: 0.9879 - val_loss: 0.0426 - val_accuracy: 0.9880
Epoch 5/10
938/938 [==============================] - 139s 148ms/step - loss: 0.0317 -
accuracy: 0.9900 - val_loss: 0.0447 - val_accuracy: 0.9859
Test loss: 0.0466243177652359
Test accuracy: 0.9883000254631042
```

The layers of model 4: Convolution layer,pooling layer,convolution layer, pooling layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers:SGD
It is faster and has less loss and increase in accuracy

Test two different batch size

Mode1

#diferent batch size model1

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5,
5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))
model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_13"
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 10, 10, 128)       36992
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_12 (Flatten)         (None, 1152)              0
_________________________________________________________________
dense_22 (Dense)             (None, 10)                11530
=================================================================
Total params: 49,354
Trainable params: 49,354
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
469/469 [==============================] - 127s 270ms/step - loss: 0.2023 -
accuracy: 0.9406 - val_loss: 0.0587 - val_accuracy: 0.9820
Epoch 2/10
469/469 [==============================] - 127s 270ms/step - loss: 0.0529 -
accuracy: 0.9838 - val_loss: 0.0401 - val_accuracy: 0.9861
Epoch 3/10
469/469 [==============================] - 130s 277ms/step - loss: 0.0383 -
accuracy: 0.9880 - val_loss: 0.0355 - val_accuracy: 0.9877
Epoch 4/10
469/469 [==============================] - 86s 184ms/step - loss: 0.0325 -
accuracy: 0.9897 - val_loss: 0.0377 - val_accuracy: 0.9889
Epoch 5/10
469/469 [==============================] - 71s 152ms/step - loss: 0.0259 -
accuracy: 0.9920 - val_loss: 0.0362 - val_accuracy: 0.9873
Test loss: 0.03951968252658844
Test accuracy: 0.9901000261306763
```

The layers of model 2: Convolution layer,pooling layer,convolution layer, pooling layer, Flattening layer Output Layer

The learning rate used:0.09 and optimizers:SGD

As batch size =64*2 it gets faster and it increase its accuracy

Model2

diferent batch size model2##

```
()model = Sequential
model.add(Conv2D(32, kernel_size=(5,
5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(128, (3, 3), activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))


()model.summary
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
,history = model.fit(x_train, y_train
,batch_size=192,shuffle=True
,epochs=epochs
,verbose=1
(validation_data=(x_test, y_test)
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 10, 10, 128)       36992
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_13 (Flatten)         (None, 1152)              0
_________________________________________________________________
dense_23 (Dense)             (None, 10)                11530
=================================================================
Total params: 49,354
Trainable params: 49,354
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
313/313 [==============================] - 72s 230ms/step - loss: 0.2921 -
accuracy: 0.9190 - val_loss: 0.0582 - val_accuracy: 0.9818
Epoch 2/10
313/313 [==============================] - 72s 230ms/step - loss: 0.0626 -
accuracy: 0.9801 - val_loss: 0.0470 - val_accuracy: 0.9849
Epoch 3/10
313/313 [==============================] - 74s 236ms/step - loss: 0.0455 -
accuracy: 0.9858 - val_loss: 0.0478 - val_accuracy: 0.9853
Epoch 4/10
```

313/313 [==============================] - 68s 219ms/step - loss: 0.0381 - accuracy: 0.9879 - val_loss: 0.0357 - val_accuracy: 0.9881
Epoch 5/10
313/313 [==============================] - 67s 214ms/step - loss: 0.0337 - accuracy: 0.9893 - val_loss: 0.0537 - val_accuracy: 0.9837
Test loss: 0.032638829201459885
Test accuracy: 0.989799976348877
The layers of model 2: Convolution layer,pooling layer,convolution layer, pooling layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers:SGD

As batch size =64*3 it gets faster and it increase its accuracy but not as faster or acurate as when batch size =128

Test 3 different activation function

Mode1

#diferent activation function model1

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5,
5),activation='sigmoid',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(128, (3, 3), activation='sigmoid'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='sigmoid'))


model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_21 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 12, 12, 32)        0
```

```
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 10, 10, 128)       36992
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_14 (Flatten)         (None, 1152)              0
_________________________________________________________________
dense_24 (Dense)             (None, 10)                11530
=================================================================
Total params: 49,354
Trainable params: 49,354
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
469/469 [==============================] - 77s 164ms/step - loss: 2.3035 - acc
uracy: 0.1116 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 2/10
469/469 [==============================] - 79s 169ms/step - loss: 2.3026 - acc
uracy: 0.1124 - val_loss: 2.3024 - val_accuracy: 0.1135
Epoch 3/10
469/469 [==============================] - 77s 163ms/step - loss: 2.3025 - acc
uracy: 0.1124 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 4/10
469/469 [==============================] - 89s 189ms/step - loss: 2.3025 - acc
uracy: 0.1124 - val_loss: 2.3023 - val_accuracy: 0.1135
Epoch 5/10
469/469 [==============================] - 86s 183ms/step - loss: 2.3025 - acc
uracy: 0.1124 - val_loss: 2.3023 - val_accuracy: 0.1135
Test loss: 1.6922441720962524
Test accuracy: 0.12700000405311584
```

The layers of model 2: Convolution layer,pooling layer,convolution layer, pooling layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers:SGD
Activation sigmoid function
Very bad accuracy and it is faster but has bad loss and bad accuracy

Model2

#diferent activation function model2

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='tanh',input_shape=input_shap
e))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(128, (3, 3), activation='tanh'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='tanh'))
```

```
model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy, metrics=['
accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 10, 10, 128)       36992
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_15 (Flatten)         (None, 1152)              0
_________________________________________________________________
dense_25 (Dense)             (None, 10)                11530
=================================================================
Total params: 49,354
Trainable params: 49,354
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
469/469 [==============================] - 70s 150ms/step - loss: 6.6516 - acc
uracy: 0.0994 - val_loss: 6.7197 - val_accuracy: 0.1032
Epoch 2/10
469/469 [==============================] - 69s 148ms/step - loss: 6.6521 - acc
uracy: 0.0993 - val_loss: 6.7197 - val_accuracy: 0.1032
Epoch 3/10
469/469 [==============================] - 69s 147ms/step - loss: 6.6521 - acc
uracy: 0.0993 - val_loss: 6.7197 - val_accuracy: 0.1032
Epoch 4/10
469/469 [==============================] - 69s 146ms/step - loss: 6.6521 - acc
uracy: 0.0993 - val_loss: 6.7197 - val_accuracy: 0.1032
Epoch 5/10
469/469 [==============================] - 69s 147ms/step - loss: 6.6521 - acc
uracy: 0.0993 - val_loss: 6.7197 - val_accuracy: 0.1032
```

Test loss: 6.719670295715332
Test accuracy: 0.10320000350475311
The layers of model 2: Convolution layer,pooling layer,convolution layer, pooling layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers:SGD
Activation tanh function
Very bad accuracy and increase in loss by huge rate but it is faster than using sigmoi d function in training

Model3

#diferent activation function model3

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='selu',input_shape=input_shape
))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(128, (3, 3), activation='selu'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='selu'))


model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy, metrics=['
accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Model: "sequential_17"

Layer (type)	Output Shape	Param #
conv2d_25 (Conv2D)	(None, 24, 24, 32)	832
max_pooling2d_19 (MaxPooling	(None, 12, 12, 32)	0
conv2d_26 (Conv2D)	(None, 10, 10, 128)	36992
max_pooling2d_20 (MaxPooling	(None, 3, 3, 128)	0
flatten_16 (Flatten)	(None, 1152)	0

dense_26 (Dense) (None, 10) 11530
===
Total params: 49,354
Trainable params: 49,354
Non-trainable params: 0
_____
Epoch 1/10
469/469 [==============================] - 67s 142ms/step - loss: 4.8479 - acc
uracy: 0.1820 - val_loss: 4.7651 - val_accuracy: 0.1009
Epoch 2/10
469/469 [==============================] - 66s 142ms/step - loss: 4.7178 - acc
uracy: 0.2975 - val_loss: 5.9674 - val_accuracy: 0.0894
Epoch 3/10
469/469 [==============================] - 67s 142ms/step - loss: 5.5606 - acc
uracy: 0.4200 - val_loss: 5.3618 - val_accuracy: 0.6093
Epoch 4/10
469/469 [==============================] - 67s 143ms/step - loss: 4.7846 - acc
uracy: 0.5796 - val_loss: 3.6105 - val_accuracy: 0.7220
Epoch 5/10
469/469 [==============================] - 67s 144ms/step - loss: 3.5755 - acc
uracy: 0.7251 - val_loss: 3.5097 - val_accuracy: 0.7463
Test loss: 3.723034381866455
Test accuracy: 0.10090000182390213
The layers of model 2: Convolution layer,pooling layer,convolution layer, pooling layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers:SGD
Activation selu function
Very bad accuracy and increase in loss by huge rate but it is faster than using sigmoi
d function in training and tanh function

Test two different optimazation function

Mode1

#diferent optimization function model1

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))


model.summary()
keras.optimizers.RMSprop(lr=0.09, momentum=0.9)
```

```
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
print('Test accuracy:', score[1])
```

Model: "sequential_22"

Layer (type)	Output Shape	Param #
conv2d_35 (Conv2D)	(None, 24, 24, 32)	832
max_pooling2d_29 (MaxPooling	(None, 12, 12, 32)	0
conv2d_36 (Conv2D)	(None, 10, 10, 128)	36992
max_pooling2d_30 (MaxPooling	(None, 3, 3, 128)	0
flatten_21 (Flatten)	(None, 1152)	0
dense_31 (Dense)	(None, 10)	11530

Total params: 49,354
Trainable params: 49,354
Non-trainable params: 0

```
Epoch 1/10
469/469 [==============================] - 70s 150ms/step - loss: 0.1922 - acc
uracy: 0.9394 - val_loss: 0.0627 - val_accuracy: 0.9794
Epoch 2/10
469/469 [==============================] - 69s 146ms/step - loss: 0.0524 - acc
uracy: 0.9841 - val_loss: 0.0442 - val_accuracy: 0.9843
Epoch 3/10
469/469 [==============================] - 68s 145ms/step - loss: 0.0393 - acc
uracy: 0.9876 - val_loss: 0.0462 - val_accuracy: 0.9859
Epoch 4/10
469/469 [==============================] - 68s 144ms/step - loss: 0.0304 - acc
uracy: 0.9904 - val_loss: 0.0382 - val_accuracy: 0.9880
Epoch 5/10
469/469 [==============================] - 68s 145ms/step - loss: 0.0259 - acc
uracy: 0.9913 - val_loss: 0.0394 - val_accuracy: 0.9880
Test loss: 0.053791578859090805
Test accuracy: 0.9876999855041504
```

The layers of model 1: Convolution layer,pooling layer,convolution layer, pooling layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers: RMSprop
Faster and have a very good accuracy but SGD is better

Model2

#diferent optimization function model2

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))


model.summary()
keras.optimizers.Adam(lr=0.09)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_23"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_37 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 10, 10, 128)       36992
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_22 (Flatten)         (None, 1152)              0
_________________________________________________________________
dense_32 (Dense)             (None, 10)                11530
=================================================================
Total params: 49,354
Trainable params: 49,354
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
469/469 [==============================] - 63s 135ms/step - loss: 0.2291 - accuracy:
0.9325 - val_loss: 0.0584 - val_accuracy: 0.9798
```

```
Epoch 2/10
469/469 [==============================] - 63s 134ms/step - loss: 0.0578 - accuracy:
0.9822 - val_loss: 0.0520 - val_accuracy: 0.9830
Epoch 3/10
469/469 [==============================] - 64s 136ms/step - loss: 0.0444 - accuracy:
0.9863 - val_loss: 0.0421 - val_accuracy: 0.9857
Epoch 4/10
469/469 [==============================] - 64s 137ms/step - loss: 0.0369 - accuracy:
0.9888 - val_loss: 0.0393 - val_accuracy: 0.9861
Epoch 5/10
469/469 [==============================] - 65s 138ms/step - loss: 0.0325 - accuracy:
0.9897 - val_loss: 0.0389 - val_accuracy: 0.9869
Test loss: 0.047830529510974884
Test accuracy: 0.9868000149726868
```

The layers of model 2: Convolution layer,pooling layer,convolution layer, pooling layer, Flattening layer Output Layer

The learning rate used:0.09 and optimizers: Adam

Faster and have a very good accuracy but SGD is better but it is better than model1 in accuracy and in speed

Put a dropout layer in the model, anywhere you see fit, try 2 places and test at least 2 different dropout rates

Model1

#diferent dropout rate model1

```
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))


model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Model: "sequential_24"

Layer (type)	Output Shape	Param #
conv2d_39 (Conv2D)	(None, 24, 24, 32)	832
max_pooling2d_33 (MaxPooling	(None, 12, 12, 32)	0
batch_normalization (BatchNo	(None, 12, 12, 32)	128
dropout (Dropout)	(None, 12, 12, 32)	0
conv2d_40 (Conv2D)	(None, 10, 10, 128)	36992
max_pooling2d_34 (MaxPooling	(None, 3, 3, 128)	0
flatten_23 (Flatten)	(None, 1152)	0
dense_33 (Dense)	(None, 10)	11530

Total params: 49,482
Trainable params: 49,418
Non-trainable params: 64

```
Epoch 1/10
469/469 [==============================] - 103s 220ms/step - loss: 0.5724 - accuracy:
0.8256 - val_loss: 0.1601 - val_accuracy: 0.9517
Epoch 2/10
469/469 [==============================] - 89s 189ms/step - loss: 0.2590 - accuracy:
0.9203 - val_loss: 0.1001 - val_accuracy: 0.9701
Epoch 3/10
469/469 [==============================] - 94s 201ms/step - loss: 0.2206 - accuracy:
0.9329 - val_loss: 0.0946 - val_accuracy: 0.9695
Epoch 4/10
469/469 [==============================] - 94s 200ms/step - loss: 0.2048 - accuracy:
0.9371 - val_loss: 0.0956 - val_accuracy: 0.9710
Epoch 5/10
469/469 [==============================] - 91s 193ms/step - loss: 0.1964 - accuracy:
0.9404 - val_loss: 0.0909 - val_accuracy: 0.9670
Test loss: 0.0766831710934639
Test accuracy: 0.9761999845504761
```

The layers of model 1: Convolution layer,pooling layer, BatchNormalization layer,dropoutlayer, convolution layer, pooling layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers: SGD
Dropout rate=0.5
It has 64 sample not train as for dropout layer some of sample is not train
So it reduce accuracy and increase loss

Model2

```
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
model = Sequential()
model.add(Conv2D(32, kernel_size=(5,
5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))


model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_25"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_41 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_35 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 12, 12, 32)        128
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 10, 10, 128)       36992
_________________________________________________________________
max_pooling2d_36 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 3, 3, 128)         512
_________________________________________________________________
dropout_2 (Dropout)          (None, 3, 3, 128)         0
```

```
_________________________________________________________________
flatten_24 (Flatten)         (None, 1152)              0
_________________________________________________________________
dense_34 (Dense)             (None, 10)                11530
=================================================================
Total params: 49,994
Trainable params: 49,674
Non-trainable params: 320
_________________________________________________________________
Epoch 1/10
469/469 [==============================] - 91s 194ms/step - loss: 0.2996 - acc
uracy: 0.9282 - val_loss: 0.0718 - val_accuracy: 0.9761
Epoch 2/10
469/469 [==============================] - 97s 207ms/step - loss: 0.1228 - acc
uracy: 0.9619 - val_loss: 0.0570 - val_accuracy: 0.9816
Epoch 3/10
469/469 [==============================] - 90s 192ms/step - loss: 0.1068 - acc
uracy: 0.9664 - val_loss: 0.0486 - val_accuracy: 0.9844
Epoch 4/10
469/469 [==============================] - 96s 205ms/step - loss: 0.0972 - acc
uracy: 0.9697 - val_loss: 0.0484 - val_accuracy: 0.9836
Epoch 5/10
469/469 [==============================] - 91s 195ms/step - loss: 0.0885 - acc
uracy: 0.9723 - val_loss: 0.0414 - val_accuracy: 0.9878
Test loss: 0.03855862095952034
Test accuracy: 0.9866999983787537
```

The layers of model 2: Convolution layer,pooling layer, BatchNormalization layer,dropout layer, convolution layer, pooling layer, BatchNormalization layer,dropout layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers: SGD
Dropout rate=0.5
Increase of non trained sample as dropout layer increase and it counsumed more time to excute but notice increase of accuracy

Model3

#diferent dropout rate model3

```
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
model = Sequential()
model.add(Conv2D(32, kernel_size=(5,
5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(BatchNormalization())
model.add(Dropout(0.4))
```

```
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))


model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1]))
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_43 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_37 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 32)        128
_________________________________________________________________
dropout_3 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 10, 10, 128)       36992
_________________________________________________________________
max_pooling2d_38 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_25 (Flatten)         (None, 1152)              0
_________________________________________________________________
dense_35 (Dense)             (None, 10)                11530
=================================================================
Total params: 49,482
Trainable params: 49,418
Non-trainable params: 64
_________________________________________________________________
Epoch 1/10
469/469 [==============================] - 108s 230ms/step - loss: 0.2708 - ac
curacy: 0.9236 - val_loss: 0.0830 - val_accuracy: 0.9736
Epoch 2/10
```

```
469/469 [==============================] - 102s 217ms/step - loss: 0.1141 - ac
curacy: 0.9651 - val_loss: 0.0607 - val_accuracy: 0.9794
Epoch 3/10
469/469 [==============================] - 94s 199ms/step - loss: 0.0947 - acc
uracy: 0.9708 - val_loss: 0.0439 - val_accuracy: 0.9864
Epoch 4/10
469/469 [==============================] - 87s 186ms/step - loss: 0.0848 - acc
uracy: 0.9739 - val_loss: 0.0414 - val_accuracy: 0.9871
Epoch 5/10
469/469 [==============================] - 91s 193ms/step - loss: 0.0822 - acc
uracy: 0.9751 - val_loss: 0.0478 - val_accuracy: 0.9856
Test loss: 0.04655031859874725
Test accuracy: 0.9865999817848206
```

The layers of model 3: Convolution layer,pooling layer, BatchNormalization layer,dropoutlayer, convolution layer, pooling layer, Flattening layer Output Layer
The learning rate used:0.09 and optimizers: SGD
Dropout rate=0.4
As dropout rate decrease it increase in time to train but increase the accuracy

Model4

#diferent dropout rate model4

```
from keras.layers.normalization import BatchNormalization
from keras.layers import Dropout
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=3))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))


model.summary()
opt = SGD(lr=0.09, momentum=0.9)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])
history = model.fit(x_train, y_train,
        batch_size=128,shuffle=True,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
```

```
print('Test loss:', score[0])
print('Test accuracy:', score[1])
Model: "sequential_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_45 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_39 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
batch_normalization_4 (Batch (None, 12, 12, 32)        128
_________________________________________________________________
dropout_4 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 10, 10, 128)       36992
_________________________________________________________________
max_pooling2d_40 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
batch_normalization_5 (Batch (None, 3, 3, 128)         512
_________________________________________________________________
dropout_5 (Dropout)          (None, 3, 3, 128)         0
_________________________________________________________________
flatten_26 (Flatten)         (None, 1152)              0
_________________________________________________________________
dense_36 (Dense)             (None, 10)                11530
=================================================================
Total params: 49,994
Trainable params: 49,674
Non-trainable params: 320
_________________________________________________________________
Epoch 1/10
469/469 [==============================] - 101s 216ms/step - loss: 0.2448 - accuracy: 0.
9414 - val_loss: 0.0544 - val_accuracy: 0.9802
Epoch 2/10
469/469 [==============================] - 100s 213ms/step - loss: 0.0904 - accuracy: 0.
9715 - val_loss: 0.0510 - val_accuracy: 0.9833
Epoch 3/10
469/469 [==============================] - 99s 211ms/step - loss: 0.0789 - accuracy: 0.9
750 - val_loss: 0.0404 - val_accuracy: 0.9871
Epoch 4/10
469/469 [==============================] - 93s 199ms/step - loss: 0.0709 - accuracy: 0.9
774 - val_loss: 0.0448 - val_accuracy: 0.9857
Epoch 5/10
469/469 [==============================] - 87s 187ms/step - loss: 0.0673 - accuracy: 0.9
792 - val_loss: 0.0377 - val_accuracy: 0.9870
Test loss: 0.03039797581732273
Test accuracy: 0.989300012588501
```

The layers of model 4: Convolution layer,pooling layer, BatchNormalization layer,dropout layer, convolution layer, pooling layer, BatchNormalization layer,dropout layer, Flattening layer Output Layer

The learning rate used:0.09 and optimizers: SGD

Dropout rate=0.4

As dropout layer decrease in both layer

Accuracy increase and loss decrease but it counsumed more time than the higher rate