2018

Names	IDs
Ahmed Mohamed Abdel-Rashied	20180028
Eslam Nasser Ghoneim	20180047

Supervised Learning

ASSIGNMENT 3 (REPORT)

MNIST

1) frist model is :

```
import tensorflow as tf

import numpy as np


mnist=tf.keras.datasets.mnist


(x_train,y_train),(x_test,y_test)=mnist.load_data()

image_size = x_train.shape[1]

# resize and normalize

x_train = np.reshape(x_train,[-1, image_size, image_size, 1])

x_test = np.reshape(x_test,[-1, image_size, image_size, 1])

x_train = x_train.astype('float32') / 255

x_test = x_test.astype('float32') / 255

x_train.shape


model=tf.keras.models.Sequential([
                                        tf.keras.layers.Dense(512,activation=tf.nn.relu),

                                        tf.keras.layers.MaxPooling2D((2, 2),strides=(2, 2)),

                                        tf.keras.layers.Flatten(),

                                        #tf.keras.layers.Dropout(0.2),

                                        tf.keras.layers.Dense(10,activation=tf.nn.softmax)

])


opt = tf.keras.optimizers.SGD(lr=0.5)

model.compile(optimizer=opt,loss=tf.keras.losses.SparseCategoricalCrossentropy(),metrics=["accuracy"]
)
```

model.fit(x_train,y_train,epochs=10,batch_size=32)

model.evaluate(x_test,y_test,batch_size=32)

Final accuracy of the model :0.9013000130653381

The accuracy in the first 5 epoch :

epoch 1: 0.8386

epoch 2: 0.8983

epoch 3: 0.9005

epoch 4: 0.9046

epoch 5: 0.9031

The number of parameters in the model : 1,004,554

The average time to train in each epoch : 188.3

The average test time in each epoch :1.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers :0.5

The optimizer used with its configuration :SGD

---

2) by change epoches from 10 to 11 :

we will use the same code but different in the number of epoches in the one line:

----->>>model.fit(x_train,y_train,epochs=11,batch_size=32)

Final accuracy of the model :0.10279999673366547

The accuracy in the first 5 epoch :

epoch 1: 0.9076

epoch 2: 0.9081

epoch 3: 0.9085

epoch 4: 0.8815

epoch 5: 0.8945

The number of parameters in the model : 1,004,554

The average time to train in each epoch : 190.7s

The average test time in each epoch :1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers :0.5

The optimizer used with its configuration :SGD

---

3)by change epoches from 10 to 13 :

we will use the same code but different in the number of epoches in the one line:

----->>>model.fit(x_train,y_train,epochs=11,batch_size=32)

Final accuracy of the model : 0.11349999904632568

The accuracy in the first 5 epoch :

epoch 1: 0.8395

epoch 2: 0.8974

epoch 3: 0.8976

epoch 4: 0.8985

epoch 5: 0.8967

The number of parameters in the model : 1,004,554

The average time to train in each epoch : 188.8

The average test time in each epoch :0.85s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers :0.5

The optimizer used with its configuration :SGD

---

4)by change epoches from 10 to 15 :

we will use the same code but different in the number of epoches in the one line:

----->>>model.fit(x_train,y_train,epochs=11,batch_size=32)

Final accuracy of the model :0.8756999969482422

The accuracy in the first 5 epoch :

epoch 1: 0.8300

epoch 2: 0.9003

epoch 3: 0.8938

epoch 4: 0.8995

epoch 5: 0.9025

The number of parameters in the model : 1,004,554

The average time to train in each epoch : 202.9

The average test time in each epoch :0.8s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers :0.5

The optimizer used with its configuration :SGD

---

5) by compere from the final accuracy we see that the epoces =10 give us the best accuracy

Now we will change the learning rate from 0.5 to .0001

we will change the line : opt = tf.keras.optimizers.SGD(lr=.0001)

model.compile(optimizer=opt,loss=tf.keras.losses.SparseCategoricalCrossentropy(),metrics=["accuracy"]
)

Final accuracy of the model :0.7960000038146973

The accuracy in the first 5 epoch :

epoch 1: 0.1503

epoch 2: 0.5718

epoch 3: 0.6625

epoch 4: 0.7025

epoch 5: 0.7189

The number of parameters in the model : 1,004,554

The average time to train in each epoch : 206

The average test time in each epoch :1.2s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers :0.0001

The optimizer used with its configuration :SGD

---

6) Now we will change the learning rate from 0.5 to .0050

we will change the line : opt = tf.keras.optimizers.SGD(lr=.0050)

model.compile(optimizer=opt,loss=tf.keras.losses.SparseCategoricalCrossentropy(),metrics=["accuracy"]
)

Final accuracy of the model :0.916100025177002

The accuracy in the first 5 epoch :

epoch 1: 0.7075

epoch 2: 0.8808

epoch 3: 0.8933

epoch 4: 0.9001

epoch 5: 0.9052

The number of parameters in the model : 1,004,554

The average time to train in each epoch : 209

The average test time in each epoch :1.2s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .0050

The optimizer used with its configuration :SGD

---

7) Now we will change the learning rate from 0.5 to .001

we will change the line : opt = tf.keras.optimizers.SGD(lr=.001)

model.compile(optimizer=opt,loss=tf.keras.losses.SparseCategoricalCrossentropy(),metrics=["accuracy"]
)

Final accuracy of the model :0.9032

The accuracy in the first 5 epoch :

epoch 1: 0.5404

epoch 2: 0.7986

epoch 3: 0.8404

epoch 4: 0.8596

epoch 5: 0.8702

The number of parameters in the model : 1,004,554

The average time to train in each epoch : 204.6

The average test time in each epoch :1.2s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

8) by comper the final accuracy of the each model we see that the learning rate = .0050 and epoches =10 that give us the best model

Now we will add the new layer

```
model=tf.keras.models.Sequential([
                                        tf.keras.layers.Dense(512,activation=tf.nn.relu),
        this is add          >>>>>      tf.keras.layers.Dense(128,activation=tf.nn.relu),

                                        tf.keras.layers.MaxPooling2D((2, 2),strides=(2, 2)),
                                        tf.keras.layers.Flatten(),
                                        #tf.keras.layers.Dropout(0.2),
                                        tf.keras.layers.Dense(10,activation=tf.nn.softmax)
   ])
```

Final accuracy of the model :0.9169999957084656

The accuracy in the first 5 epoch :

epoch 1: 0.6636

epoch 2: 0.8944

epoch 3: 0.9035

epoch 4: 0.9087

epoch 5: 0.9110

The number of parameters in the model : 634,122

The average time to train in each epoch : 550.7

The average test time in each epoch :3.3s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

9) Now we will add the new layer

```
model=tf.keras.models.Sequential([
```

tf.keras.layers.Dense(512,activation=tf.nn.relu),

this is add >>>>> tf.keras.layers.Dense(256,activation=tf.nn.relu),

tf.keras.layers.MaxPooling2D((2, 2),strides=(2, 2)),

tf.keras.layers.Flatten(),

#tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(10,activation=tf.nn.softmax)

])

Final accuracy of the model :0.9169999957084656

The accuracy in the first 5 epoch :

epoch 1: 0.6636

epoch 2: 0.8944

epoch 3: 0.9035

epoch 4: 0.9087

epoch 5: 0.9110

The number of parameters in the model : 634,122

The average time to train in each epoch : 550.7

The average test time in each epoch :3.3s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

10) Now we will remove the layer

model=tf.keras.models.Sequential([

this is remove >>>>>
tf.keras.layers.Dense(512,activation=tf.nn.relu),

tf.keras.layers.MaxPooling2D((2, 2),strides=(2, 2)),

tf.keras.layers.Flatten(),

#tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(10,activation=tf.nn.softmax)

])

Final accuracy of the model :0.9489

The accuracy in the first 5 epoch :

epoch 1: 0.7160

epoch 2: 0.8947

epoch 3: 0.9078

epoch 4: 0.9152

epoch 5: 0.9236

The number of parameters in the model : 8,431,523

The average time to train in each epoch : 304

The average test time in each epoch :1.7s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

11) Now we will remove the layer

model=tf.keras.models.Sequential([

tf.keras.layers.Dense(512,activation=tf.nn.relu),

this is remove >>>>> tf.keras.layers.MaxPooling2D((2, 2),strides=(2, 2)),

tf.keras.layers.Flatten(),

#tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(10,activation=tf.nn.softmax)

])

Final accuracy of the model :0.9169999957084656

The accuracy in the first 5 epoch :

epoch 1: 0.6636

epoch 2: 0.8944

epoch 3: 0.9035

epoch 4: 0.9087

epoch 5: 0.9110

The number of parameters in the model : 634,122

The average time to train in each epoch : 550.7

The average test time in each epoch :3.3s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

12) by comper the final accuracy we see that when remove tf.keras.layers.Dense(512,activation=tf.nn.relu), that we have the best model

Now we will change the batch size fron 32 to 64

this code that will change:-- model.fit(x_train,y_train,epochs=10,batch_size=64)

Final accuracy of the model : 0.8772000074386597

The accuracy in the first 5 epoch :

epoch 1: 0.3037

epoch 2: 0.7457

epoch 3: 0.8068

epoch 4: 0.8263

epoch 5: 0.8397

The number of parameters in the model : 1,970

The average time to train in each epoch : 2

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

13) Now we will change the batch size fron 32 to 128

this code that will change:-- model.fit(x_train,y_train,epochs=10,batch_size=64)

Final accuracy of the model : 0.8593000173568726

The accuracy in the first 5 epoch :

epoch 1: 0.2528

epoch 2: 0.6704

epoch 3: 0.7626

epoch 4: 0.7966

epoch 5: 0.8156

The number of parameters in the model : 1,970

The average time to train in each epoch : 2

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

14) by comper the final accuracy we see that the batch size = 32 is the best model
Now we will change the activation function

we will change this line from tf.keras.layers.Dense(10,activation=tf.nn.softmax)

to tf.keras.layers.Dense(10,activation=tf.nn.sigmoid)

Final accuracy of the model 0.8891000151634216

The accuracy in the first 5 epoch :

epoch 1: 0.2823

epoch 2: 0.7554

epoch 3: 0.8256

epoch 4: 0.8455

epoch 5: 0.8532

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.1

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

15) we will change this line from tf.keras.layers.Dense(10,activation=tf.nn.softmax)

to tf.keras.layers.Dense(10,activation=tf.nn.tanh)

Final accuracy of the model : 0.09769999980926514

The accuracy in the first 5 epoch :

epoch 1: 0.0958

epoch 2: 0.0971

epoch 3: 0.0942

epoch 4: 0.0969

epoch 5: 0.0969

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.1

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

16) we will change this line from tf.keras.layers.Dense(10,activation=tf.nn.softmax)

to tf.keras.layers.Dense(10,activation=tf.nn.relu)

Final accuracy of the model : 0.10859999805688858

The accuracy in the first 5 epoch :

epoch 1: 0.2540

epoch 2: 0.0860

epoch 3: 0.0860

epoch 4: 0.1002

epoch 5: 0.1081

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :SGD

---

17) by comper the final accuracy we see that the activation ='softmax' give the best model

Now we change the optimize from SGD to Adam

opt = tf.keras.optimizers.Adam(lr=.0050)

Final accuracy of the model : 0.9165999889373779

The accuracy in the first 5 epoch :

epoch 1: 0.8271

epoch 2: 0.9067

epoch 3: 0.9098

epoch 4: 0.9163

epoch 5: 0.9163

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.1

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :Adam

---

18) Now we change the optimize from SGD to Nadam

opt = tf.keras.optimizers.Adam(lr=.0050)

Final accuracy of the model : 0.9187999963760376

The accuracy in the first 5 epoch :

epoch 1: 0.8229

epoch 2: 0.9083

epoch 3: 0.9119

epoch 4: 0.9136

epoch 5 : 0.9166

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.1

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :Nadam

---

19) by comper the final accuracy we finf=d that the optemizer Nadam give the best model

Now we make the dropout

we will add this line tf.keras.layers.Dropout(0.2),

Final accuracy of the model : 0.9068999886512756

The accuracy in the first 5 epoch :

epoch 1: 0.7759

epoch 2: 0.8616

epoch 3: 0.8638

epoch 4: 0.8640

epoch 5 : 0.8668

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.1

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :Nadam

---

20) we will add this line tf.keras.layers.Dropout(0.4),

Final accuracy of the model : 0.8996999859809875

The accuracy in the first 5 epoch :

epoch 1: 0.7189

epoch 2: 0.8077

epoch 3: 0.8103

epoch 4: 0.8135

epoch 5 : 0.8107

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.1

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :Nadam

---

21) we will add this line tf.keras.layers.Dropout(0.6),

Final accuracy of the model : 0.8808000087738037

The accuracy in the first 5 epoch :

epoch 1: 0.6394

epoch 2: 0.7267

epoch 3: 0.7273

epoch 4: 0.7283

epoch 5 : 0.7305

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.1

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :Nadam

---

22)) we will add this line tf.keras.layers.Dropout(0.8),

Final accuracy of the model : 0.8808000087738037

The accuracy in the first 5 epoch :

epoch 1: 0.4830

epoch 2: 0.5648

epoch 3: 0.5688

epoch 4: 0.5602

epoch 5 : 0.5653

The number of parameters in the model : 1,970

The average time to train in each epoch : 3.1

The average test time in each epoch :0.1s

The layers in the model : dense,max_pooling2d,flatten,

The learning rate used and configuration of the optimizers : .005

The optimizer used with its configuration :Nadam

By comper the final accuracy we find that the dropout(0.2) give the best model

so the code for the best model is:::---

mport tensorflow as tf

import numpy as np

mnist=tf.keras.datasets.mnist

(x_train,y_train),(x_test,y_test)=mnist.load_data()

```
image_size = x_train.shape[1]

# resize and normalize

x_train = np.reshape(x_train,[-1, image_size, image_size, 1])

x_test = np.reshape(x_test,[-1, image_size, image_size, 1])

x_train = x_train.astype('float32') / 255

x_test = x_test.astype('float32') / 255

x_train.shape


model=tf.keras.models.Sequential([

                                        tf.keras.layers.MaxPooling2D((2, 2),strides=(2, 2)),

                                        tf.keras.layers.Flatten(),

                                        tf.keras.layers.Dropout(0.2),

                                        tf.keras.layers.Dense(10,activation=tf.nn.softmax)


])


opt = tf.keras.optimizers.Nadam(lr=.0050 )

model.compile(optimizer=opt,loss=tf.keras.losses.SparseCategoricalCrossentropy(),metrics=["accuracy"]
)


model.fit(x_train,y_train,epochs=10,batch_size=32)


model.evaluate(x_test,y_test,batch_size=32)


model.summary()
```

23) Add new layer

1- tf.keras.layers.Conv2D(32, kernel_size=(2, 2), activation=tf.nn.relu), as first layer in CNN

```
Epoch 1/10
1875/1875 [==============================] - 28s 15ms/step - loss: 0.3923 -
accuracy: 0.8866
Epoch 2/10
1875/1875 [==============================] - 27s 15ms/step - loss: 0.2941 -
accuracy: 0.9144
Epoch 3/10
1875/1875 [==============================] - 27s 14ms/step - loss: 0.3167 -
accuracy: 0.9036
Epoch 4/10
1875/1875 [==============================] - 27s 15ms/step - loss: 0.3240 -
accuracy: 0.9034
Epoch 5/10
1875/1875 [==============================] - 27s 14ms/step - loss: 0.3045 -
accuracy: 0.9073
Test  accuracy: 0.94
```

24) Basic Model best accuracy and run time

```
model2=tf.keras.models.Sequential([
                                    tf.keras.layers.Flatten(),
                                    tf.keras.layers.Dense(512,activation=tf.
nn.relu),
                                    tf.keras.layers.Dense(128,activation=tf.
nn.relu),
                                    #tf.keras.layers.Dropout(0.2),
                                    tf.keras.layers.Dense(10,activation=tf.n
n.softmax)
])


model2.compile(optimizer=opt,loss=tf.keras.losses.SparseCategoricalCrossen
tropy(),metrics=["accuracy"])
model2.fit(x_train,y_train,epochs=10,batch_size=32)
model2.evaluate(x_test,y_test,batch_size=32)


Epoch 1/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3209 -
accuracy: 0.9045
Epoch 2/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0811 -
accuracy: 0.9752
Epoch 3/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0536 -
accuracy: 0.9837
Epoch 4/10
```

```
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0393 -
accuracy: 0.9881
Epoch 5/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0285 -
accuracy: 0.9900
Test [0.09931602329015732, 0.9785000085830688]
```

If we use dropout 0.2 before last layer

```
Epoch 1/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3606 -
accuracy: 0.8896
Epoch 2/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0956 -
accuracy: 0.9692
Epoch 3/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0639 -
accuracy: 0.9799
Epoch 4/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0452 -
accuracy: 0.9854
Epoch 5/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0379 -
accuracy: 0.9884
Test [0.09314420074224472, 0.9789999723434448]
```

If we use tow dropouts 0.2

```
Epoch 1/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3883 -
accuracy: 0.8832
Epoch 2/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.1093 -
accuracy: 0.9666
Epoch 3/10
1875/1875 [==============================] - 10s 6ms/step - loss: 0.0804 -
accuracy: 0.9751
Epoch 4/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0614 -
accuracy: 0.9806
Epoch 5/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0544 -
accuracy: 0.9828
Test [0.06850088387727737, 0.9819999933242798]
```

If we use 3 dropouts 0.2

```
Epoch 1/10
1875/1875 [==============================] - 12s 6ms/step - loss: 0.4473 -
accuracy: 0.8598
Epoch 2/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1451 -
accuracy: 0.9544
Epoch 3/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1128 -
accuracy: 0.9648
Epoch 4/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0935 -
accuracy: 0.9709
Epoch 5/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0782 -
accuracy: 0.9748
[0.06345418840646744, 0.9819999933242798]
2 dropouts is the best
model2=tf.keras.models.Sequential([
                                    tf.keras.layers.Flatten(),
                                    #tf.keras.layers.Dropout(0.2),
                                    tf.keras.layers.Dense(512,activation=tf.
nn.relu),
                                    tf.keras.layers.Dropout(0.2),
                                    tf.keras.layers.Dense(128,activation=tf.
nn.relu),
                                    tf.keras.layers.Dropout(0.2),
                                    tf.keras.layers.Dense(10,activation=tf.n
n.softmax)
])
#model2.compile(optimizer="adam",loss="sparse_categorical_crossentropy",me
trics=["accuracy"])
opt = tf.keras.optimizers.Adam()
model2.compile(optimizer=opt,loss=tf.keras.losses.SparseCategoricalCrossen
tropy(),metrics=["accuracy"])
model2.fit(x_train,y_train,epochs=10,batch_size=32)
model2.evaluate(x_test,y_test,batch_size=32)
```

```
Conclusion
Model number 1 is slowly with accuracy in range 91%
Model number 2 is very Easy and fast with accuracy in range 98%
```