Deep Learning Practical Assignment 2A

April 27, 2023

Name - Takte Yash Santosh / Roll No. - 4264 / Batch - B7

Importing Dataset & Libraries

[125]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

[126]:
```
#url='https://archive.ics.uci.edu/ml/machine-learning-databases/
↪letter-recognition/letter-recognition.data'
```

[127]:
```
columns = ["lettr", "x-box", "y-box", "width", "height", "onpix", "x-bar",␣
↪"y-bar", "x2bar", "y2bar", "xybar", "x2ybr", "xy2br", "x-ege", "xegvy",␣
↪"y-ege", "yegvx"]
```

[128]:
```
#df = pd.read_csv(url, names=columns)
df = pd.read_csv('D:\DL Practical\letter-recognition.data', names=columns)
```

[129]:
```
df
```

[129]:

	lettr	x-box	y-box	width	height	onpix	x-bar	y-bar	x2bar	y2bar	\
0	T	2	8	3	5	1	8	13	0	6	
1	I	5	12	3	7	2	10	5	5	4	
2	D	4	11	6	8	6	10	6	2	6	
3	N	7	11	6	6	3	5	9	4	6	
4	G	2	1	3	1	1	8	6	6	6	
…	…	…	…	…	…	…	…	…	…	…	
19995	D	2	2	3	3	2	7	7	7	6	
19996	C	7	10	8	8	4	4	8	6	9	
19997	T	6	9	6	7	5	6	11	3	7	
19998	S	2	3	4	2	1	8	7	2	6	
19999	A	4	9	6	6	2	9	5	3	1	

	xybar	x2ybr	xy2br	x-ege	xegvy	y-ege	yegvx
0	6	10	8	0	8	0	8
1	13	3	9	2	8	4	10
2	10	3	7	3	7	3	9
3	4	4	10	6	10	2	8
4	6	5	9	1	7	5	10

```
…      …    …    …    …    …    …    …
19995    6    6    4    2    8    3    7
19996   12    9   13    2    9    3    7
19997   11    9    5    2   12    2    4
19998   10    6    8    1    9    5    8
19999    8    1    8    2    7    2    8

[20000 rows x 17 columns]
```

[130]:
```
x = df.drop("lettr", axis=1).values
y = df["lettr"].values
```

[131]:
```
x.shape
```

[131]:
```
(20000, 16)
```

[132]:
```
y.shape
```

[132]:
```
(20000,)
```

[133]:
```
np.unique(y)
```

[133]:
```
array(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
       'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'],
      dtype=object)
```

[134]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

[135]:
```
def shape():
      print("Train Shape :",x_train.shape)
      print("Test Shape :",x_test.shape)
      print("y_train shape :",y_train.shape)
      print("y_test shape :",y_test.shape)
shape()
```

```
Train Shape : (16000, 16)
Test Shape : (4000, 16)
y_train shape : (16000,)
y_test shape : (4000,)
```

[136]:
```
x_train[0]
```

[136]:
```
array([ 7,  8,  7,  6,  5,  7, 10,  3,  7, 10,  9,  5,  4, 11,  5,  5],
      dtype=int64)
```

[137]:
```
y_train[0]
```

[137]:
```
'T'
```

[138]:
```
class_names=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',␣
 ↪'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

[139]:
```
x_test[10]
```

[139]:
```
array([ 3,  7,  3,  5,  2,  5,  7,  7,  2,  6,  5, 11,  3,  8,  2, 11],
      dtype=int64)
```

[140]:
```
y_test[10]
```

[140]:
```
'K'
```

Preprocessing

[141]:
```
x_train = x_train/255
x_test = x_test/255
```

[142]:
```
from sklearn.preprocessing import LabelEncoder
```

[143]:
```
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.fit_transform(y_test)
```

Building our Model

[144]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

[145]:
```
model=Sequential()
model.add(Dense(512, activation='relu', input_shape=(16,)))
model.add(Dropout(0.2))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(26, activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 512)               8704

 dropout_8 (Dropout)         (None, 512)               0

 dense_13 (Dense)            (None, 256)               131328

 dropout_9 (Dropout)         (None, 256)               0
```

```
 dense_14 (Dense)            (None, 26)                6682

=================================================================
Total params: 146,714
Trainable params: 146,714
Non-trainable params: 0
_________________________________________________________________
```

Training our Model

[146]:
```
model.fit(x_train, y_train, epochs=50, batch_size=128, verbose=1,␣
↪validation_data=(x_test, y_test))
```

```
Epoch 1/50
125/125 [==============================] - 2s 9ms/step - loss: 3.1405 -
accuracy: 0.1376 - val_loss: 2.7789 - val_accuracy: 0.3200
Epoch 2/50
125/125 [==============================] - 1s 10ms/step - loss: 2.3794 -
accuracy: 0.3139 - val_loss: 2.0621 - val_accuracy: 0.3895
Epoch 3/50
125/125 [==============================] - 1s 10ms/step - loss: 1.9730 -
accuracy: 0.3999 - val_loss: 1.8093 - val_accuracy: 0.4757
Epoch 4/50
125/125 [==============================] - 1s 8ms/step - loss: 1.7710 -
accuracy: 0.4606 - val_loss: 1.6285 - val_accuracy: 0.5280
Epoch 5/50
125/125 [==============================] - 1s 8ms/step - loss: 1.6371 -
accuracy: 0.4986 - val_loss: 1.5359 - val_accuracy: 0.5512
Epoch 6/50
125/125 [==============================] - 1s 8ms/step - loss: 1.5522 -
accuracy: 0.5236 - val_loss: 1.4595 - val_accuracy: 0.5655
Epoch 7/50
125/125 [==============================] - 1s 9ms/step - loss: 1.4855 -
accuracy: 0.5507 - val_loss: 1.3863 - val_accuracy: 0.5950
Epoch 8/50
125/125 [==============================] - 1s 9ms/step - loss: 1.4229 -
accuracy: 0.5726 - val_loss: 1.3439 - val_accuracy: 0.6077
Epoch 9/50
125/125 [==============================] - 1s 9ms/step - loss: 1.3628 -
accuracy: 0.5936 - val_loss: 1.2928 - val_accuracy: 0.6215
Epoch 10/50
125/125 [==============================] - 1s 8ms/step - loss: 1.3110 -
accuracy: 0.6049 - val_loss: 1.2317 - val_accuracy: 0.6455
Epoch 11/50
125/125 [==============================] - 1s 11ms/step - loss: 1.2580 -
accuracy: 0.6259 - val_loss: 1.2051 - val_accuracy: 0.6445
Epoch 12/50
125/125 [==============================] - 1s 10ms/step - loss: 1.2156 -
```

```
accuracy: 0.6394 - val_loss: 1.1422 - val_accuracy: 0.6615
Epoch 13/50
125/125 [==============================] - 1s 12ms/step - loss: 1.1677 -
accuracy: 0.6544 - val_loss: 1.1078 - val_accuracy: 0.6812
Epoch 14/50
125/125 [==============================] - 1s 12ms/step - loss: 1.1246 -
accuracy: 0.6661 - val_loss: 1.0515 - val_accuracy: 0.7057
Epoch 15/50
125/125 [==============================] - 1s 10ms/step - loss: 1.0801 -
accuracy: 0.6757 - val_loss: 1.0259 - val_accuracy: 0.7103
Epoch 16/50
125/125 [==============================] - 2s 12ms/step - loss: 1.0401 -
accuracy: 0.6895 - val_loss: 0.9846 - val_accuracy: 0.7185
Epoch 17/50
125/125 [==============================] - 2s 18ms/step - loss: 1.0107 -
accuracy: 0.6977 - val_loss: 0.9620 - val_accuracy: 0.7310
Epoch 18/50
125/125 [==============================] - 1s 10ms/step - loss: 0.9809 -
accuracy: 0.7048 - val_loss: 0.9191 - val_accuracy: 0.7430
Epoch 19/50
125/125 [==============================] - 1s 9ms/step - loss: 0.9509 -
accuracy: 0.7166 - val_loss: 0.8989 - val_accuracy: 0.7362
Epoch 20/50
125/125 [==============================] - 1s 8ms/step - loss: 0.9227 -
accuracy: 0.7237 - val_loss: 0.8695 - val_accuracy: 0.7430
Epoch 21/50
125/125 [==============================] - 1s 9ms/step - loss: 0.9025 -
accuracy: 0.7265 - val_loss: 0.8429 - val_accuracy: 0.7595
Epoch 22/50
125/125 [==============================] - 1s 11ms/step - loss: 0.8832 -
accuracy: 0.7342 - val_loss: 0.8257 - val_accuracy: 0.7635
Epoch 23/50
125/125 [==============================] - 1s 8ms/step - loss: 0.8550 -
accuracy: 0.7431 - val_loss: 0.8138 - val_accuracy: 0.7588
Epoch 24/50
125/125 [==============================] - 1s 8ms/step - loss: 0.8424 -
accuracy: 0.7442 - val_loss: 0.7895 - val_accuracy: 0.7745
Epoch 25/50
125/125 [==============================] - 1s 8ms/step - loss: 0.8223 -
accuracy: 0.7498 - val_loss: 0.7715 - val_accuracy: 0.7690
Epoch 26/50
125/125 [==============================] - 1s 9ms/step - loss: 0.8051 -
accuracy: 0.7541 - val_loss: 0.7586 - val_accuracy: 0.7790
Epoch 27/50
125/125 [==============================] - 1s 12ms/step - loss: 0.7840 -
accuracy: 0.7611 - val_loss: 0.7334 - val_accuracy: 0.7865
Epoch 28/50
125/125 [==============================] - 2s 12ms/step - loss: 0.7714 -
```

```
accuracy: 0.7634 - val_loss: 0.7211 - val_accuracy: 0.7895
Epoch 29/50
125/125 [==============================] - 2s 13ms/step - loss: 0.7567 -
accuracy: 0.7707 - val_loss: 0.7056 - val_accuracy: 0.7862
Epoch 30/50
125/125 [==============================] - 1s 11ms/step - loss: 0.7449 -
accuracy: 0.7679 - val_loss: 0.6951 - val_accuracy: 0.7915
Epoch 31/50
125/125 [==============================] - 1s 11ms/step - loss: 0.7310 -
accuracy: 0.7771 - val_loss: 0.6808 - val_accuracy: 0.8030
Epoch 32/50
125/125 [==============================] - 1s 12ms/step - loss: 0.7154 -
accuracy: 0.7815 - val_loss: 0.6613 - val_accuracy: 0.8077
Epoch 33/50
125/125 [==============================] - 1s 11ms/step - loss: 0.7131 -
accuracy: 0.7806 - val_loss: 0.6534 - val_accuracy: 0.8062
Epoch 34/50
125/125 [==============================] - 1s 11ms/step - loss: 0.6975 -
accuracy: 0.7847 - val_loss: 0.6485 - val_accuracy: 0.8050
Epoch 35/50
125/125 [==============================] - 1s 10ms/step - loss: 0.6858 -
accuracy: 0.7878 - val_loss: 0.6474 - val_accuracy: 0.8117
Epoch 36/50
125/125 [==============================] - 1s 10ms/step - loss: 0.6695 -
accuracy: 0.7921 - val_loss: 0.6366 - val_accuracy: 0.8050
Epoch 37/50
125/125 [==============================] - 2s 18ms/step - loss: 0.6597 -
accuracy: 0.7969 - val_loss: 0.6163 - val_accuracy: 0.8158
Epoch 38/50
125/125 [==============================] - 2s 14ms/step - loss: 0.6471 -
accuracy: 0.8005 - val_loss: 0.6028 - val_accuracy: 0.8213
Epoch 39/50
125/125 [==============================] - 2s 15ms/step - loss: 0.6358 -
accuracy: 0.8024 - val_loss: 0.6036 - val_accuracy: 0.8160
Epoch 40/50
125/125 [==============================] - 2s 15ms/step - loss: 0.6317 -
accuracy: 0.8009 - val_loss: 0.5704 - val_accuracy: 0.8292
Epoch 41/50
125/125 [==============================] - 1s 12ms/step - loss: 0.6176 -
accuracy: 0.8100 - val_loss: 0.5628 - val_accuracy: 0.8345
Epoch 42/50
125/125 [==============================] - 1s 8ms/step - loss: 0.6128 -
accuracy: 0.8126 - val_loss: 0.5568 - val_accuracy: 0.8315
Epoch 43/50
125/125 [==============================] - 1s 11ms/step - loss: 0.5959 -
accuracy: 0.8153 - val_loss: 0.5499 - val_accuracy: 0.8340
Epoch 44/50
125/125 [==============================] - 1s 10ms/step - loss: 0.5873 -
```

```
accuracy: 0.8156 - val_loss: 0.5428 - val_accuracy: 0.8338
Epoch 45/50
125/125 [==============================] - 1s 9ms/step - loss: 0.5856 -
accuracy: 0.8190 - val_loss: 0.5228 - val_accuracy: 0.8425
Epoch 46/50
125/125 [==============================] - 1s 11ms/step - loss: 0.5731 -
accuracy: 0.8192 - val_loss: 0.5127 - val_accuracy: 0.8443
Epoch 47/50
125/125 [==============================] - 1s 10ms/step - loss: 0.5553 -
accuracy: 0.8252 - val_loss: 0.5084 - val_accuracy: 0.8505
Epoch 48/50
125/125 [==============================] - 1s 10ms/step - loss: 0.5561 -
accuracy: 0.8240 - val_loss: 0.4960 - val_accuracy: 0.8528
Epoch 49/50
125/125 [==============================] - 1s 7ms/step - loss: 0.5460 -
accuracy: 0.8283 - val_loss: 0.4918 - val_accuracy: 0.8520
Epoch 50/50
125/125 [==============================] - 1s 8ms/step - loss: 0.5385 -
accuracy: 0.8300 - val_loss: 0.4860 - val_accuracy: 0.8558
```

[146]:
```
<keras.callbacks.History at 0x154db5b69a0>
```

Testing our Model

[147]:
```
predictions = model.predict(x_test)
```

```
125/125 [==============================] - 0s 2ms/step
```

[148]:
```
index=10
print(predictions[index])
final_value=np.argmax(predictions[index])
print("Actual label :",y_test[index])
print("Predicted label :",final_value)
print("Class (A-Z) :",class_names[final_value])
```

```
[2.82419956e-06 3.09114297e-11 8.58481682e-04 1.52923052e-09
 1.75701853e-08 6.90554991e-09 2.59319018e-03 2.73245550e-03
 1.89313641e-06 6.92704276e-08 9.91127431e-01 2.80531793e-04
 1.50894982e-06 1.11913309e-04 1.95911690e-03 4.95337504e-09
 1.19699944e-04 1.13634174e-04 4.14209552e-07 1.65963798e-09
 1.98035650e-06 1.63241438e-07 1.56543487e-10 9.46799773e-05
 1.75718503e-13 1.22214403e-14]
Actual label : 10
Predicted label : 10
Class (A-Z) : K
```

Evaluating our Model

[149]:
```
loss, accuracy = model.evaluate(x_test, y_test)
print("Loss :",loss)
print("Accuracy (Test Data) :",accuracy*100)
```

```
125/125 [==============================] - 0s 3ms/step - loss: 0.4860 -
accuracy: 0.8558
Loss : 0.48596155643463135
Accuracy (Test Data) : 85.5750024318695
```