DL_Lab2_201911027

September 25, 2020

[ ]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from tensorflow import keras
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
# from sklearn.preprocessing import MinMaxScaler, Normalizer
from sklearn.utils import shuffle
print(tf.__version__)
```

```
2.3.0

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

1 Binary Classification ANN

[ ]:
```
heart = pd.read_csv("/content/drive/My Drive/Deep Learning/Lab2/heart.csv")
print(heart.shape)
heart.head()
```

```
(303, 14)
```

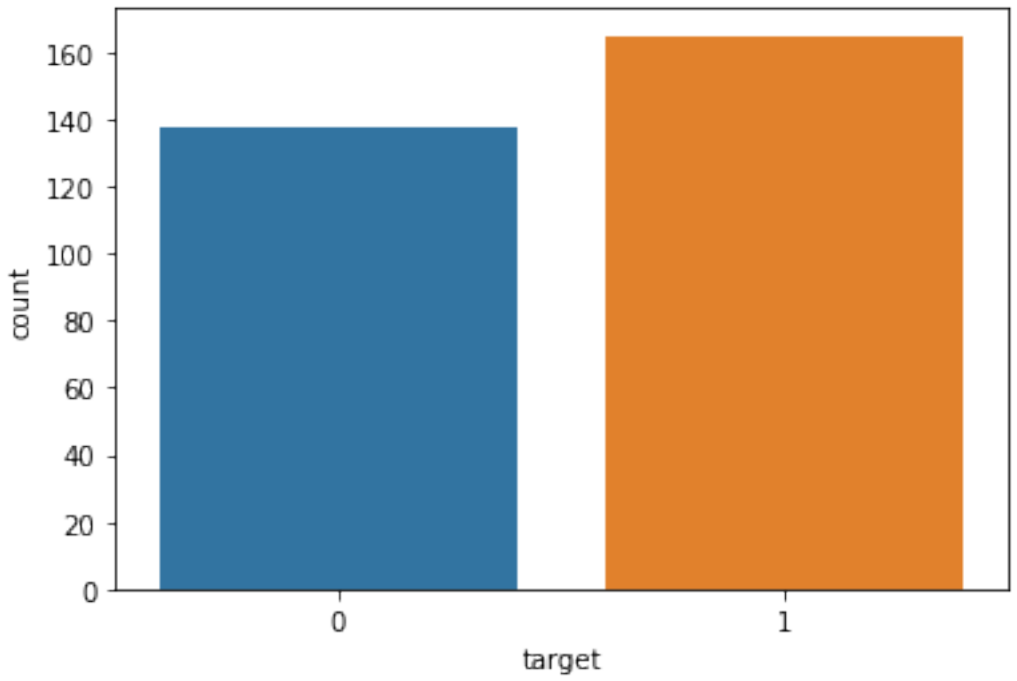
[ ]:
```
   age  sex  cp  trestbps  chol  fbs  ...  exang  oldpeak  slope  ca  thal
target
```

```
0   63    1   3       145   233   1   ...       0       2.3       0   0      1
1
1   37    1   2       130   250   0   ...       0       3.5       0   0      2
1
2   41    0   1       130   204   0   ...       0       1.4       2   0      2
1
3   56    1   1       120   236   0   ...       0       0.8       2   0      2
1
4   57    0   0       120   354   0   ...       1       0.6       2   0      2
1

[5 rows x 14 columns]
```

[ ]:
```
sns.countplot(heart['target'],label='Count')
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f275b746b00>
```



[ ]:
```
features = list(heart.columns.values)
features.remove('target')
X = heart[features]
y = heart['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=0)

print("Training Data :", X_train.shape)
```

```
print("Training Label :", y_train.shape)
print("Testing Data :", X_test.shape)
print("Testing Label :", y_test.shape)
X_train.head()

# train,test = train_test_split(heart, test_size=0.3, random_state=42)
# X_train,y_train = train.iloc[:,:-1],train.iloc[:,-1:]
# X_test, y_test = test.iloc[:,:-1], test.iloc[:,-1:]
```

```
Training Data : (212, 13)
Training Label : (212,)
Testing Data : (91, 13)
Testing Label : (91,)
```

[ ]:

	age	sex	cp	trestbps	chol	...	exang	oldpeak	slope	ca	thal
137	62	1	1	128	208	...	0	0.0	2	0	2
106	69	1	3	160	234	...	0	0.1	1	1	2
284	61	1	0	140	207	...	1	1.9	2	1	3
44	39	1	2	140	321	...	0	0.0	2	0	2
139	64	1	0	128	263	...	1	0.2	1	1	3

[5 rows x 13 columns]

[ ]:
```
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scalled = scaler.transform(X_train)
X_test_scalled = scaler.transform(X_test)
```

[ ]:
```
# model = keras.Sequential([
#     keras.layers.Flatten(input_shape=(13,)),
#     keras.layers.Dense(16, activation=tf.nn.relu),
#         keras.layers.Dense(16, activation=tf.nn.relu),
#     keras.layers.Dense(1, activation=tf.nn.sigmoid),
# ])
model = tf.keras.models.Sequential()
model.add(tf.keras.Input(shape=13))
model.add(tf.keras.layers.Dense(10, activation='relu'))
model.add(tf.keras.layers.Dense(6, activation='relu'))
model.add(tf.keras.layers.Dense(1,activation='sigmoid'))
model.output_shape
```

[ ]:
```
(None, 1)
```

[ ]:
```
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_17 (Dense)             (None, 10)                140
_________________________________________________________________
dense_18 (Dense)             (None, 6)                 66
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 7
=================================================================
Total params: 213
Trainable params: 213
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
model.fit(X_train_scalled, y_train, epochs=100, batch_size=1)
```

```
Epoch 1/50
212/212 [==============================] - 0s 786us/step - loss: 0.6690 -
accuracy: 0.5802
Epoch 2/50
212/212 [==============================] - 0s 772us/step - loss: 0.5890 -
accuracy: 0.7406
Epoch 3/50
212/212 [==============================] - 0s 783us/step - loss: 0.5172 -
accuracy: 0.8208
Epoch 4/50
212/212 [==============================] - 0s 772us/step - loss: 0.4674 -
accuracy: 0.8491
Epoch 5/50
212/212 [==============================] - 0s 809us/step - loss: 0.4332 -
accuracy: 0.8585
Epoch 6/50
212/212 [==============================] - 0s 884us/step - loss: 0.4089 -
accuracy: 0.8632
Epoch 7/50
212/212 [==============================] - 0s 860us/step - loss: 0.3890 -
accuracy: 0.8396
Epoch 8/50
212/212 [==============================] - 0s 834us/step - loss: 0.3750 -
accuracy: 0.8774
Epoch 9/50
212/212 [==============================] - 0s 857us/step - loss: 0.3626 -
accuracy: 0.8443
Epoch 10/50
212/212 [==============================] - 0s 859us/step - loss: 0.3507 -
accuracy: 0.8679
Epoch 11/50
212/212 [==============================] - 0s 1ms/step - loss: 0.3449 -
```

```
accuracy: 0.8726
Epoch 12/50
212/212 [==============================] - 0s 866us/step - loss: 0.3358 -
accuracy: 0.8774
Epoch 13/50
212/212 [==============================] - 0s 873us/step - loss: 0.3272 -
accuracy: 0.8868
Epoch 14/50
212/212 [==============================] - 0s 896us/step - loss: 0.3204 -
accuracy: 0.8821
Epoch 15/50
212/212 [==============================] - 0s 879us/step - loss: 0.3165 -
accuracy: 0.8821
Epoch 16/50
212/212 [==============================] - 0s 980us/step - loss: 0.3070 -
accuracy: 0.8774
Epoch 17/50
212/212 [==============================] - 0s 870us/step - loss: 0.3027 -
accuracy: 0.8915
Epoch 18/50
212/212 [==============================] - 0s 866us/step - loss: 0.2957 -
accuracy: 0.9009
Epoch 19/50
212/212 [==============================] - 0s 854us/step - loss: 0.2925 -
accuracy: 0.8868
Epoch 20/50
212/212 [==============================] - 0s 844us/step - loss: 0.2844 -
accuracy: 0.9057
Epoch 21/50
212/212 [==============================] - 0s 856us/step - loss: 0.2837 -
accuracy: 0.8962
Epoch 22/50
212/212 [==============================] - 0s 876us/step - loss: 0.2765 -
accuracy: 0.9104
Epoch 23/50
212/212 [==============================] - 0s 900us/step - loss: 0.2691 -
accuracy: 0.9009
Epoch 24/50
212/212 [==============================] - 0s 837us/step - loss: 0.2691 -
accuracy: 0.8962
Epoch 25/50
212/212 [==============================] - 0s 890us/step - loss: 0.2622 -
accuracy: 0.9104
Epoch 26/50
212/212 [==============================] - 0s 906us/step - loss: 0.2606 -
accuracy: 0.9104
Epoch 27/50
212/212 [==============================] - 0s 890us/step - loss: 0.2540 -
```

```
accuracy: 0.9057
Epoch 28/50
212/212 [==============================] - 0s 873us/step - loss: 0.2439 -
accuracy: 0.9198
Epoch 29/50
212/212 [==============================] - 0s 877us/step - loss: 0.2476 -
accuracy: 0.9292
Epoch 30/50
212/212 [==============================] - 0s 853us/step - loss: 0.2372 -
accuracy: 0.9151
Epoch 31/50
212/212 [==============================] - 0s 855us/step - loss: 0.2377 -
accuracy: 0.9151
Epoch 32/50
212/212 [==============================] - 0s 860us/step - loss: 0.2334 -
accuracy: 0.9198
Epoch 33/50
212/212 [==============================] - 0s 893us/step - loss: 0.2216 -
accuracy: 0.9245
Epoch 34/50
212/212 [==============================] - 0s 871us/step - loss: 0.2194 -
accuracy: 0.9198
Epoch 35/50
212/212 [==============================] - 0s 855us/step - loss: 0.2155 -
accuracy: 0.9340
Epoch 36/50
212/212 [==============================] - 0s 816us/step - loss: 0.2065 -
accuracy: 0.9434
Epoch 37/50
212/212 [==============================] - 0s 784us/step - loss: 0.2041 -
accuracy: 0.9387
Epoch 38/50
212/212 [==============================] - 0s 790us/step - loss: 0.1977 -
accuracy: 0.9387
Epoch 39/50
212/212 [==============================] - 0s 774us/step - loss: 0.1925 -
accuracy: 0.9434
Epoch 40/50
212/212 [==============================] - 0s 744us/step - loss: 0.1871 -
accuracy: 0.9434
Epoch 41/50
212/212 [==============================] - 0s 765us/step - loss: 0.1830 -
accuracy: 0.9434
Epoch 42/50
212/212 [==============================] - 0s 790us/step - loss: 0.1754 -
accuracy: 0.9481
Epoch 43/50
212/212 [==============================] - 0s 806us/step - loss: 0.1711 -
```

```
accuracy: 0.9528
Epoch 44/50
212/212 [==============================] - 0s 803us/step - loss: 0.1662 -
accuracy: 0.9575
Epoch 45/50
212/212 [==============================] - 0s 836us/step - loss: 0.1620 -
accuracy: 0.9575
Epoch 46/50
212/212 [==============================] - 0s 805us/step - loss: 0.1574 -
accuracy: 0.9528
Epoch 47/50
212/212 [==============================] - 0s 820us/step - loss: 0.1516 -
accuracy: 0.9575
Epoch 48/50
212/212 [==============================] - 0s 852us/step - loss: 0.1485 -
accuracy: 0.9623
Epoch 49/50
212/212 [==============================] - 0s 864us/step - loss: 0.1432 -
accuracy: 0.9670
Epoch 50/50
212/212 [==============================] - 0s 868us/step - loss: 0.1397 -
accuracy: 0.9623
```

[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f2751f2d128>
```

[ ]:
```
test_loss, test_acc = model.evaluate(X_test_scalled, y_test)
print('Test accuracy:', test_acc)
```

```
3/3 [==============================] - 0s 2ms/step - loss: 0.7794 - accuracy:
0.7912
Test accuracy: 0.791208803653717
```

2 Multi Classification ANN

[ ]:
```
mobileDS = pd.read_csv("/content/drive/My Drive/Deep Learning/Lab2/
 ↪datasets_11167_15520_train_Mobile.csv")
print(mobileDS.shape)
mobileDS.head()
```

```
(2000, 21)
```

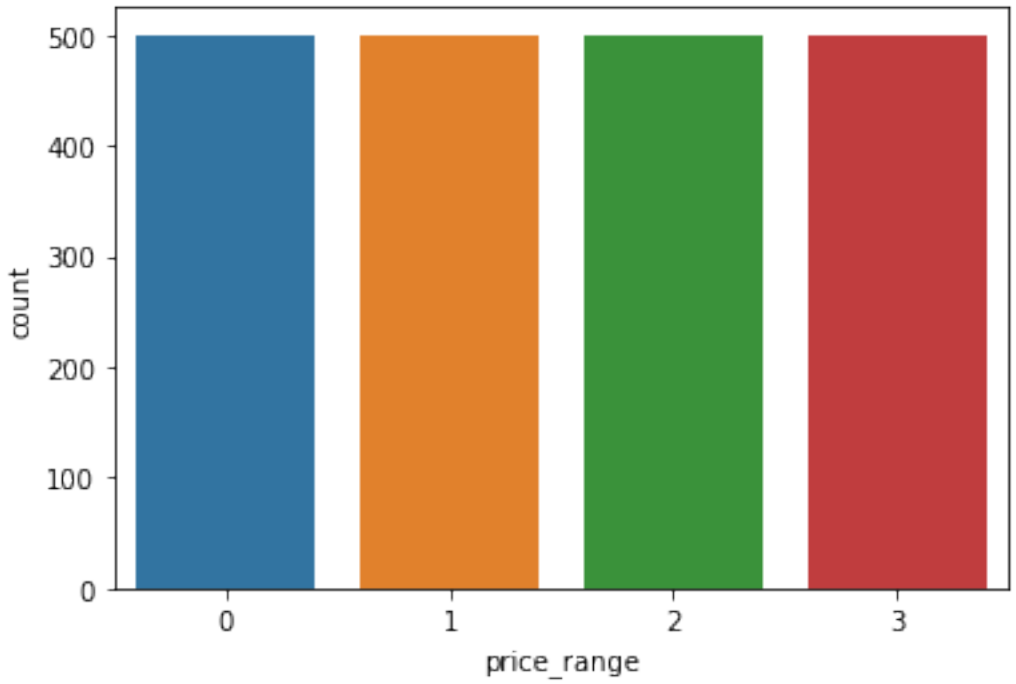
[ ]:

	battery_power	blue	clock_speed	...	touch_screen	wifi	price_range
0	842	0	2.2	...	0	1	1
1	1021	1	0.5	...	1	0	2
2	563	1	0.5	...	1	0	2
3	615	1	2.5	...	0	0	2
4	1821	1	1.2	...	1	0	1

```
[5 rows x 21 columns]
```

[ ]:
```
sns.countplot(mobileDS['price_range'],label='Count')
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa404e482b0>
```



[ ]:
```
features = list(mobileDS.columns.values)
features.remove('price_range')
X = mobileDS[features]
y = mobileDS['price_range']
y = pd.DataFrame(y.values.reshape(-1,1))
print(X.shape, y.shape)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=0)

print("Training Data :", X_train.shape)
print("Training Label :", y_train.shape)
print("Testing Data :", X_test.shape)
print("Testing Label :", y_test.shape)
X_train.head()
```

```
(2000, 20) (2000, 1)
Training Data : (1400, 20)
Training Label : (1400, 1)
```

```
Testing Data : (600, 20)
Testing Label : (600, 1)
```

[ ]:
```
      battery_power  blue  clock_speed  ...  three_g  touch_screen  wifi
1719            833     1          0.6  ...        1             0     1
1702           1424     1          2.9  ...        1             0     0
1287            860     1          1.3  ...        1             1     1
482            1330     1          1.3  ...        0             0     1
768            1149     1          2.2  ...        1             0     0

[5 rows x 20 columns]
```

[ ]:
```
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scalled = scaler.transform(X_train)
X_test_scalled = scaler.transform(X_test)

encoder = preprocessing.OneHotEncoder(sparse=False).fit(y_train)
print("categories",encoder.categories_)

y_train_encoded = encoder.transform(y_train)
y_test_encoded = encoder.transform(y_test)
print(y_train_encoded.shape)
```

```
categories [array([0, 1, 2, 3])]
(1400, 4)
```

[ ]:
```
model2 = tf.keras.models.Sequential()
model2.add(tf.keras.Input(shape=20))
# model2.add(tf.keras.layers.Dense(15, activation='relu'))
model2.add(tf.keras.layers.Dense(12, activation='relu'))
model2.add(tf.keras.layers.Dense(8, activation='relu'))
model2.add(tf.keras.layers.Dense(4,activation='softmax'))
model2.output_shape
```

[ ]:
```
(None, 4)
```

[ ]:
```
model2.compile(optimizer='adam',
               loss='categorical_crossentropy',
               metrics=['accuracy'])
model2.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 12)                252
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 104
```

```
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 36
=================================================================
Total params: 392
Trainable params: 392
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
model2.fit(X_train_scalled, y_train_encoded, epochs=100, batch_size=1)
```

```
Epoch 1/100
1400/1400 [==============================] - 2s 2ms/step - loss: 1.3521 -
accuracy: 0.3043
Epoch 2/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.9046 -
accuracy: 0.6107
Epoch 3/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.4458 -
accuracy: 0.8393
Epoch 4/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.2991 -
accuracy: 0.8979
Epoch 5/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.2280 -
accuracy: 0.9236
Epoch 6/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.1918 -
accuracy: 0.9307
Epoch 7/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.1619 -
accuracy: 0.9450
Epoch 8/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.1444 -
accuracy: 0.9543
Epoch 9/100
1400/1400 [==============================] - 3s 2ms/step - loss: 0.1271 -
accuracy: 0.9543
Epoch 10/100
1400/1400 [==============================] - 3s 2ms/step - loss: 0.1153 -
accuracy: 0.9650
Epoch 11/100
1400/1400 [==============================] - 3s 2ms/step - loss: 0.1071 -
accuracy: 0.9593
Epoch 12/100
1400/1400 [==============================] - 3s 2ms/step - loss: 0.0976 -
accuracy: 0.9664
Epoch 13/100
```

```
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0951 -
accuracy: 0.9657
Epoch 14/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0850 -
accuracy: 0.9743
Epoch 15/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0815 -
accuracy: 0.9721
Epoch 16/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0756 -
accuracy: 0.9750
Epoch 17/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0694 -
accuracy: 0.9779
Epoch 18/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0683 -
accuracy: 0.9793
Epoch 19/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0620 -
accuracy: 0.9779
Epoch 20/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0573 -
accuracy: 0.9821
Epoch 21/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0549 -
accuracy: 0.9807
Epoch 22/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0582 -
accuracy: 0.9800
Epoch 23/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0477 -
accuracy: 0.9871
Epoch 24/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0512 -
accuracy: 0.9843
Epoch 25/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0469 -
accuracy: 0.9829
Epoch 26/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0412 -
accuracy: 0.9893
Epoch 27/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0409 -
accuracy: 0.9857
Epoch 28/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0384 -
accuracy: 0.9886
Epoch 29/100
```

```
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0417 -
accuracy: 0.9836
Epoch 30/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0338 -
accuracy: 0.9900
Epoch 31/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0316 -
accuracy: 0.9907
Epoch 32/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0336 -
accuracy: 0.9893
Epoch 33/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0300 -
accuracy: 0.9914
Epoch 34/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0311 -
accuracy: 0.9893
Epoch 35/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0250 -
accuracy: 0.9936
Epoch 36/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0293 -
accuracy: 0.9907
Epoch 37/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0294 -
accuracy: 0.9921
Epoch 38/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0267 -
accuracy: 0.9914
Epoch 39/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0221 -
accuracy: 0.9943
Epoch 40/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0174 -
accuracy: 0.9943
Epoch 41/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0286 -
accuracy: 0.9900
Epoch 42/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0166 -
accuracy: 0.9964
Epoch 43/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0238 -
accuracy: 0.9886
Epoch 44/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0196 -
accuracy: 0.9950
Epoch 45/100
```

```
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0176 -
accuracy: 0.9943
Epoch 46/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0167 -
accuracy: 0.9950
Epoch 47/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0196 -
accuracy: 0.9936
Epoch 48/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0258 -
accuracy: 0.9907
Epoch 49/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0116 -
accuracy: 0.9964
Epoch 50/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0152 -
accuracy: 0.9964
Epoch 51/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0164 -
accuracy: 0.9964
Epoch 52/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0154 -
accuracy: 0.9950
Epoch 53/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0093 -
accuracy: 0.9979
Epoch 54/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0227 -
accuracy: 0.9914
Epoch 55/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0096 -
accuracy: 0.9979
Epoch 56/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0155 -
accuracy: 0.9936
Epoch 57/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0120 -
accuracy: 0.9971
Epoch 58/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0069 -
accuracy: 0.9986
Epoch 59/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0147 -
accuracy: 0.9964
Epoch 60/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0144 -
accuracy: 0.9943
Epoch 61/100
```

```
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0118 -
accuracy: 0.9950
Epoch 62/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0102 -
accuracy: 0.9986
Epoch 63/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0144 -
accuracy: 0.9943
Epoch 64/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0097 -
accuracy: 0.9964
Epoch 65/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0168 -
accuracy: 0.9957
Epoch 66/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0085 -
accuracy: 0.9993
Epoch 67/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0102 -
accuracy: 0.9964
Epoch 68/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0195 -
accuracy: 0.9914
Epoch 69/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0046 -
accuracy: 0.9993
Epoch 70/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0033 -
accuracy: 0.9993
Epoch 71/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0183 -
accuracy: 0.9950
Epoch 72/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0041 -
accuracy: 0.9993
Epoch 73/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0035 -
accuracy: 1.0000
Epoch 74/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0202 -
accuracy: 0.9907
Epoch 75/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0058 -
accuracy: 0.9986
Epoch 76/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0083 -
accuracy: 0.9964
Epoch 77/100
```

```
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0092 -
accuracy: 0.9979
Epoch 78/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0088 -
accuracy: 0.9971
Epoch 79/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0027 -
accuracy: 1.0000
Epoch 80/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0094 -
accuracy: 0.9957
Epoch 81/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0258 -
accuracy: 0.9900
Epoch 82/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0023 -
accuracy: 1.0000
Epoch 83/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0013 -
accuracy: 1.0000
Epoch 84/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0177 -
accuracy: 0.9929
Epoch 85/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0055 -
accuracy: 0.9986
Epoch 86/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0104 -
accuracy: 0.9964
Epoch 87/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0029 -
accuracy: 0.9993
Epoch 88/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0132 -
accuracy: 0.9943
Epoch 89/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0123 -
accuracy: 0.9950
Epoch 90/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0044 -
accuracy: 0.9993
Epoch 91/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0013 -
accuracy: 1.0000
Epoch 92/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0011 -
accuracy: 1.0000
Epoch 93/100
```

```
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0162 -
accuracy: 0.9957
Epoch 94/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0063 -
accuracy: 0.9979
Epoch 95/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0127 -
accuracy: 0.9943
Epoch 96/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0063 -
accuracy: 0.9986
Epoch 97/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0032 -
accuracy: 0.9993
Epoch 98/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0206 -
accuracy: 0.9929
Epoch 99/100
1400/1400 [==============================] - 2s 2ms/step - loss: 0.0047 -
accuracy: 0.9986
Epoch 100/100
1400/1400 [==============================] - 2s 2ms/step - loss: 9.4502e-04 -
accuracy: 1.0000
```

[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7fa3a85e27b8>
```

[ ]:
```
test_loss, test_acc = model2.evaluate(X_test_scalled, y_test_encoded)
print('Test accuracy:', test_acc)
```

```
19/19 [==============================] - 0s 2ms/step - loss: 0.4376 - accuracy:
0.9250
Test accuracy: 0.925000011920929
```

[ ]: