DL_Assignment_2

September 27, 2022

[1]:
```
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import classification_report
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential

from keras import layers
from keras.layers import Dense
from keras.optimizers import SGD
from keras.datasets import mnist
from keras import backend as K
import matplotlib.pyplot as plt
import numpy as np
import argparse
```

[4]:
```
print("[INFO] accessing MNIST...")
((trainX, trainY), (testX, testY)) = mnist.load_data()
```

```
[INFO] accessing MNIST…
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434 [==============================] - 40s 3us/step
```

[5]:
```
trainX = trainX.reshape((trainX.shape[0], 28 * 28 * 1))
testX = testX.reshape((testX.shape[0], 28 * 28 * 1))
```

[6]:
```
trainX = trainX.astype("float32") / 255.0
testX = testX.astype("float32") / 255.0
```

[7]:
```
lb = LabelBinarizer()
trainY = lb.fit_transform(trainY)
testY = lb.transform(testY)
```

[8]:
```
model = Sequential()
model.add(Dense(256, input_shape=(784,), activation="sigmoid"))
model.add(Dense(128, activation="sigmoid"))
model.add(Dense(10, activation="softmax"))
```

[9]:
```
print("[INFO] training network...")
sgd = SGD(0.01)
model.compile(loss="categorical_crossentropy",␣
 ↪optimizer=sgd,metrics=["accuracy"])
#H = model.fit(trainX, trainY, validation_data=(testX, testY),epochs=100,␣
 ↪batch_size=128)
m1=model.fit(trainX, trainY, validation_split=0.33,epochs=100, batch_size=128,␣
 ↪verbose=0)
print("[INFO] evaluating network...")
loss, acc = model.evaluate(testX, testY, verbose=0)
print('Test Accuracy: %.3f' % acc)
```

```
[INFO] training network…
[INFO] evaluating network…
Test Accuracy: 0.913
```

[10]:
```
predictions = model.predict(testX, batch_size=128)
print(classification_report(testY.argmax(axis=1),predictions.
 ↪argmax(axis=1),target_names=[str(x) for x in lb.classes_]))
```

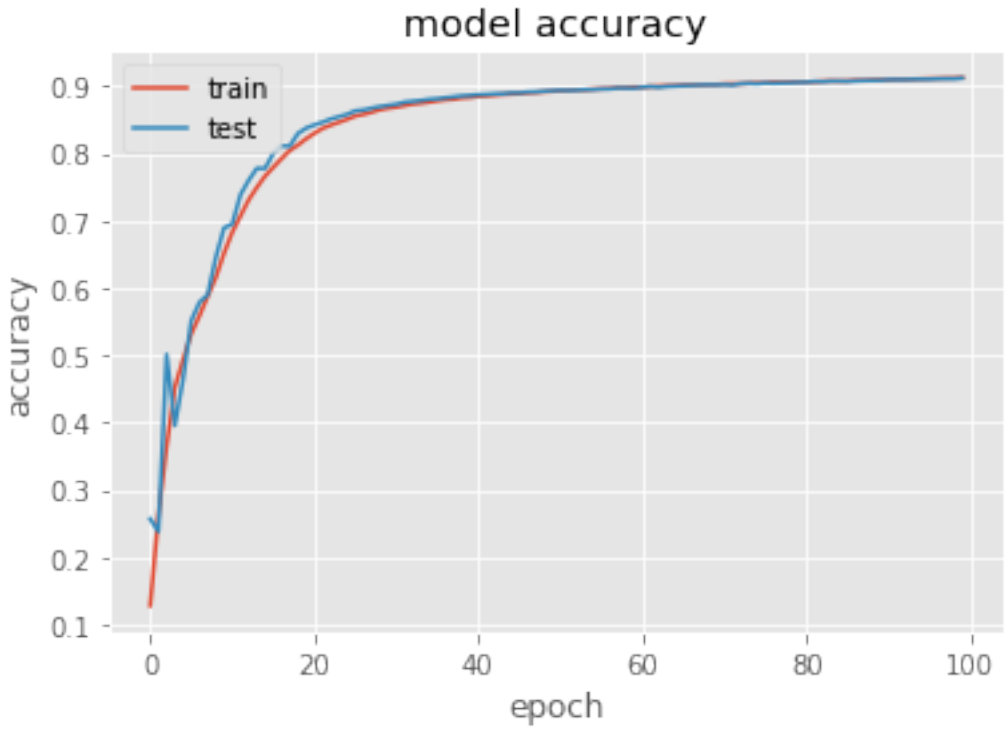
```
79/79 [==============================] - 0s 3ms/step
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       980
           1       0.96      0.97      0.97      1135
           2       0.91      0.90      0.90      1032
           3       0.90      0.89      0.90      1010
           4       0.91      0.93      0.92       982
           5       0.88      0.85      0.86       892
           6       0.92      0.94      0.93       958
           7       0.93      0.91      0.92      1028
           8       0.88      0.87      0.88       974
           9       0.91      0.88      0.89      1009

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

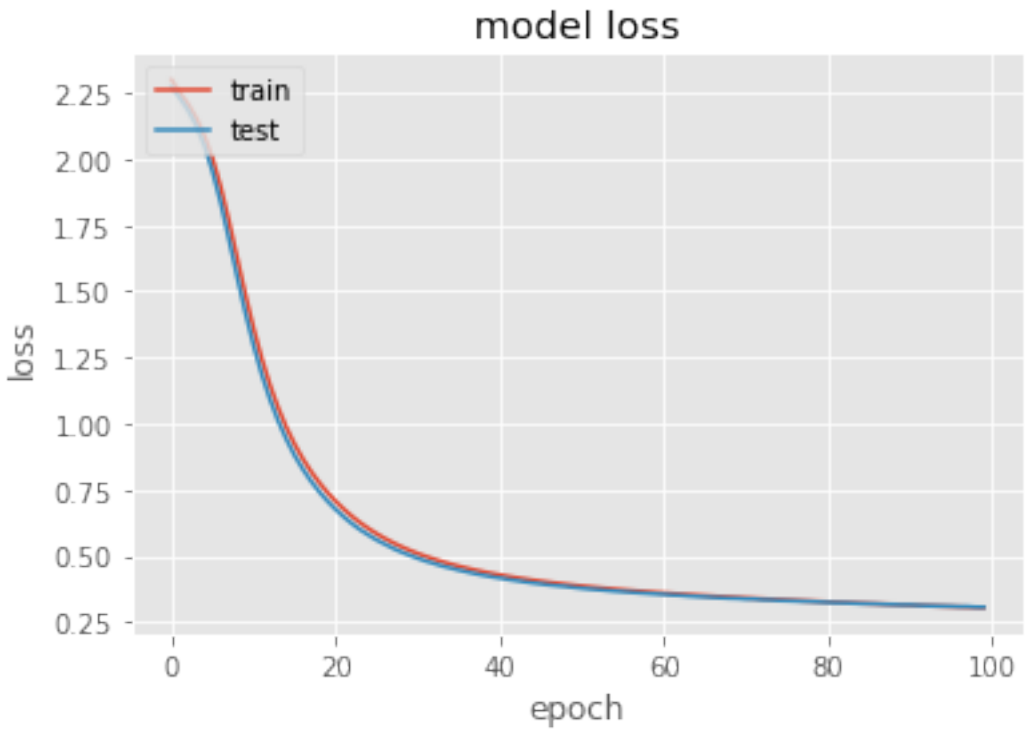
[11]:
```
# plot the training loss and accuracy
plt.style.use("ggplot")
plt.figure()
plt.plot(m1.history['accuracy'])
plt.plot(m1.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
```

```
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.savefig('mnist_accuracy.png')
```




```
<Figure size 432x288 with 0 Axes>
```

[12]:
```
plt.figure()
plt.plot(m1.history['loss'])
plt.plot(m1.history['val_loss'])
# plt.plot(np.arange(0, 100), m1.history["loss"], label="train_loss")
# plt.plot(np.arange(0, 100), m1.history["val_loss"], label="val_loss")
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.savefig('mnist_loss.png')
```




```
<Figure size 432x288 with 0 Axes>
```

[ ]: