HARISHWAR REDDY ABBAREDDY

YOUTUBE:
https://youtu.be/gBxSHXjnsAE

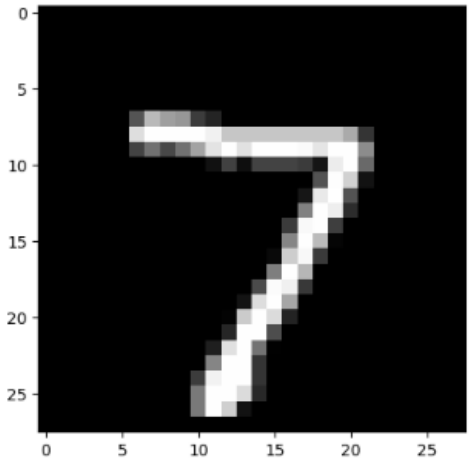
GITHUB:
https://github.com/Harishwar-reddi/ICP-6

```
[4]  import keras
     import pandas
     from keras.models import Sequential
     from keras.layers import Dense, Activation
     from sklearn.model_selection import train_test_split
     import pandas as pd
     import numpy as np

     # loading dataset
     url = 'https://drive.google.com/uc?id=1iX5lLJdKZjno5sEYs1xF1pEbxe9EMjJD'
     dataset = pd.read_csv(url, header=None).values

     #splitting the dataset
     X_train,X_test,Y_train,Y_test = train_test_split(dataset[:,0:8],dataset[:,8],test_size=0.1,random_state=30)
     np.random.seed(155)
     my_first_nn = Sequential() # create model
     my_first_nn.add(Dense(16, activation='relu', input_shape=(8,)))
     my_first_nn.add(Dense(8, activation='relu'))
     my_first_nn.add(Dense(64, activation='relu'))
     my_first_nn.add(Dense(1, activation='sigmoid'))

     #training the model
     my_first_nn.compile(loss='mean_squared_error', optimizer='adam', metrics=['acc'])
     my_first_nn.fit(X_train,Y_train,epochs=100,initial_epoch=0)

     print(my_first_nn.summary())
     print(my_first_nn.evaluate(X_test,Y_test))
```

```
Epoch 82/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1612 - acc: 0.7800
Epoch 83/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1614 - acc: 0.7728
Epoch 84/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1584 - acc: 0.7786
Epoch 85/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1590 - acc: 0.7656
Epoch 86/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1563 - acc: 0.7858
Epoch 87/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1601 - acc: 0.7713
Epoch 88/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1616 - acc: 0.7757
Epoch 89/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1556 - acc: 0.7829
Epoch 90/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1566 - acc: 0.7771
Epoch 91/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1657 - acc: 0.7496
Epoch 92/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1539 - acc: 0.7844
Epoch 93/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1570 - acc: 0.7742
Epoch 94/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1546 - acc: 0.7844
Epoch 95/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1556 - acc: 0.7728
Epoch 96/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1536 - acc: 0.7742
Epoch 97/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1532 - acc: 0.7829
Epoch 98/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1535 - acc: 0.7771
Epoch 99/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1534 - acc: 0.7858
Epoch 100/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1510 - acc: 0.8046
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 16)                144

 dense_5 (Dense)             (None, 8)                 136

 dense_6 (Dense)             (None, 64)                576

 dense_7 (Dense)             (None, 1)                 65

=================================================================
Total params: 921 (3.60 KB)
Trainable params: 921 (3.60 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
3/3 [==============================] - 0s 5ms/step - loss: 0.1589 - acc: 0.7532
[0.15886713564395905, 0.7532467246055603]
```

```
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical
import matplotlib.pyplot as plt

# Loading the data
(train_images,train_labels),(test_images, test_labels) = mnist.load_data()

# Processing the data
dimData = np.prod(train_images.shape[1:])
train_data = train_images.reshape(train_images.shape[0],dimData)
test_data = test_images.reshape(test_images.shape[0],dimData)

# Converting data to float
train_data = train_data.astype('float')
test_data = test_data.astype('float')

# Scaling data
train_data /=255.0
test_data /=255.0

# One-hot encoding the labels
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

# Creating the network with 3 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(dimData,)))
model.add(Dense(512, activation='tanh'))
model.add(Dense(512, activation='tanh'))
model.add(Dense(512, activation='sigmoid'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))

# Plotting an image from the test dataset
plt.imshow(test_images[0], cmap='gray')
plt.show()

# Predicting the class
test_image = test_data[0].reshape(1, dimData)
predicted_class = np.argmax(model.predict(test_image), axis=-1)
print(f"Predicted Class: {predicted_class[0]}, Actual Class: {test_labels[0]}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
Epoch 1/10
235/235 [==============================] - 20s 62ms/step - loss: 0.4049 - accuracy: 0.8730 - val_loss: 0.2844 - val_accuracy: 0.9111
Epoch 2/10
235/235 [==============================] - 10s 42ms/step - loss: 0.1853 - accuracy: 0.9430 - val_loss: 0.1747 - val_accuracy: 0.9448
Epoch 3/10
235/235 [==============================] - 12s 50ms/step - loss: 0.1214 - accuracy: 0.9623 - val_loss: 0.1475 - val_accuracy: 0.9531
Epoch 4/10
235/235 [==============================] - 12s 50ms/step - loss: 0.0881 - accuracy: 0.9729 - val_loss: 0.1379 - val_accuracy: 0.9571
Epoch 5/10
235/235 [==============================] - 12s 50ms/step - loss: 0.0666 - accuracy: 0.9791 - val_loss: 0.0894 - val_accuracy: 0.9725
Epoch 6/10
235/235 [==============================] - 13s 56ms/step - loss: 0.0486 - accuracy: 0.9844 - val_loss: 0.1390 - val_accuracy: 0.9552
Epoch 7/10
235/235 [==============================] - 10s 44ms/step - loss: 0.0372 - accuracy: 0.9881 - val_loss: 0.1293 - val_accuracy: 0.9620
Epoch 8/10
235/235 [==============================] - 12s 50ms/step - loss: 0.0263 - accuracy: 0.9918 - val_loss: 0.0735 - val_accuracy: 0.9774
Epoch 9/10
235/235 [==============================] - 12s 53ms/step - loss: 0.0212 - accuracy: 0.9931 - val_loss: 0.1023 - val_accuracy: 0.9717
Epoch 10/10
235/235 [==============================] - 12s 53ms/step - loss: 0.0159 - accuracy: 0.9948 - val_loss: 0.0662 - val_accuracy: 0.9806
```




```
1/1 [==============================] - 0s 167ms/step
Predicted Class: 7, Actual Class: 7
```