Assignment 6

Sai Pavan Reddy Pogula

700741739

Code Samples:

Sample diabetes.

```
import pandas as pd
data = pd.read_csv('diabetes.csv')
```

[41] ✓ 0.0s

```
path_to_csv = 'diabetes.csv'
```

[42] ✓ 0.0s

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

dataset = pd.read_csv(path_to_csv, header=None).values

X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(4, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = 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))
```

[43] ✓ 2.2s

```
... Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/100
18/18 [==============================] - 0s 919us/step - loss: 0.6920 - acc: 0.6597
Epoch 2/100
18/18 [==============================] - 0s 882us/step - loss: 0.6887 - acc: 0.6615
Epoch 3/100
18/18 [==============================] - 0s 1ms/step - loss: 0.6859 - acc: 0.6615
Epoch 4/100
```

Output and accuracy for diabetes dataset

Dataset of breast cancer

```
[44]  data = pd.read_csv('breastcancer.csv')
      ✓ 0.0s

[45]  path_to_csv = 'sample_data/breastcancer.csv'
      ✓ 0.0s

[46]  import keras
      import pandas as pd
      import numpy as np
      from keras.models import Sequential
      from keras.layers.core import Dense, Activation
      from sklearn.datasets import load_breast_cancer
      from sklearn.model_selection import train_test_split

      cancer_data = load_breast_cancer()
      X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                          test_size=0.25, random_state=87)
      np.random.seed(155)
      my_nn = Sequential() # create model
      my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
      my_nn.add(Dense(1, activation='sigmoid')) # output layer
      my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
      my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                               initial_epoch=0)
      print(my_nn.summary())
      print(my_nn.evaluate(X_test, Y_test))

      ✓ 1.7s

··    Output exceeds the size limit. Open the full output data in a text editor
      Epoch 1/100
      14/14 [==============================] - 0s 901us/step - loss: 18.6157 - acc: 0.2629
      Epoch 2/100
      14/14 [==============================] - 0s 714us/step - loss: 15.5858 - acc: 0.2066
      Epoch 3/100
      14/14 [==============================] - 0s 739us/step - loss: 13.5255 - acc: 0.1549
      Epoch 4/100
      14/14 [==============================] - 0s 824us/step - loss: 11.7932 - acc: 0.1432
      Epoch 5/100
```

```
14/14 [==============================] - 0s 845us/step - loss: 6.8262 - acc: 0.2418
Epoch 8/100
14/14 [==============================] - 0s 771us/step - loss: 5.4824 - acc: 0.3169
Epoch 9/100
14/14 [==============================] - 0s 695us/step - loss: 4.5334 - acc: 0.4671
Epoch 10/100
14/14 [==============================] - 0s 768us/step - loss: 3.4156 - acc: 0.4601
Epoch 11/100
14/14 [==============================] - 0s 716us/step - loss: 2.7149 - acc: 0.5164
Epoch 12/100
14/14 [==============================] - 0s 772us/step - loss: 2.0432 - acc: 0.6221
Epoch 13/100
14/14 [==============================] - 0s 693us/step - loss: 1.6146 - acc: 0.6455
Epoch 14/100
14/14 [==============================] - 0s 723us/step - loss: 1.2739 - acc: 0.7042
Epoch 15/100
14/14 [==============================] - 0s 775us/step - loss: 1.0605 - acc: 0.7300
Epoch 16/100
14/14 [==============================] - 0s 855us/step - loss: 0.8901 - acc: 0.7770
Epoch 17/100
14/14 [==============================] - 0s 788us/step - loss: 0.7916 - acc: 0.7817
Epoch 18/100
14/14 [==============================] - 0s 782us/step - loss: 0.6475 - acc: 0.8239
Epoch 19/100
14/14 [==============================] - 0s 787us/step - loss: 0.5848 - acc: 0.8169
Epoch 20/100
14/14 [==============================] - 0s 789us/step - loss: 0.5237 - acc: 0.8216
Epoch 21/100
14/14 [==============================] - 0s 707us/step - loss: 0.5227 - acc: 0.8427
Epoch 22/100
14/14 [==============================] - 0s 795us/step - loss: 0.4631 - acc: 0.8592
Epoch 23/100
...
_________________________________________________________________
None
5/5 [==============================] - 0s 1ms/step - loss: 0.1808 - acc: 0.9580
[0.1807614266872406, 0.9580419659614563]
```

```
data = pd.read_csv('breastcancer.csv')
```

[47] ✓ 0.0s

```
path_to_csv = 'sample_data/breastcancer.csv'
```

✓ 0.0s

```
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

cancer_data = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_nn = Sequential() # create model
my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
my_nn.add(Dense(1, activation='sigmoid')) # output layer
my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                         initial_epoch=0)
print(my_nn.summary())
print(my_nn.evaluate(X_test, Y_test))
```

✓ 1.7s

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/100
14/14 [==============================] - 0s 901us/step - loss: 18.6157 - acc: 0.2629
Epoch 2/100
14/14 [==============================] - 0s 714us/step - loss: 15.5858 - acc: 0.2066
Epoch 3/100
14/14 [==============================] - 0s 739us/step - loss: 13.5255 - acc: 0.1549
Epoch 4/100
14/14 [==============================] - 0s 824us/step - loss: 11.7932 - acc: 0.1432
```

Output of Breast cancer dataset

```
14/14 [==============================] - 0s 768us/step - loss: 1.0570 - acc: 0.8146
Epoch 19/100
14/14 [==============================] - 0s 793us/step - loss: 1.0469 - acc: 0.8451
Epoch 20/100
14/14 [==============================] - 0s 884us/step - loss: 1.1133 - acc: 0.8263
Epoch 21/100
14/14 [==============================] - 0s 764us/step - loss: 0.9183 - acc: 0.8404
Epoch 22/100
14/14 [==============================] - 0s 734us/step - loss: 0.8815 - acc: 0.8333
Epoch 23/100
...
_________________________________________________________________
None
5/5 [==============================] - 0s 1ms/step - loss: 0.2498 - acc: 0.8951
[0.24975551664829254, 0.8951048851013184]
```

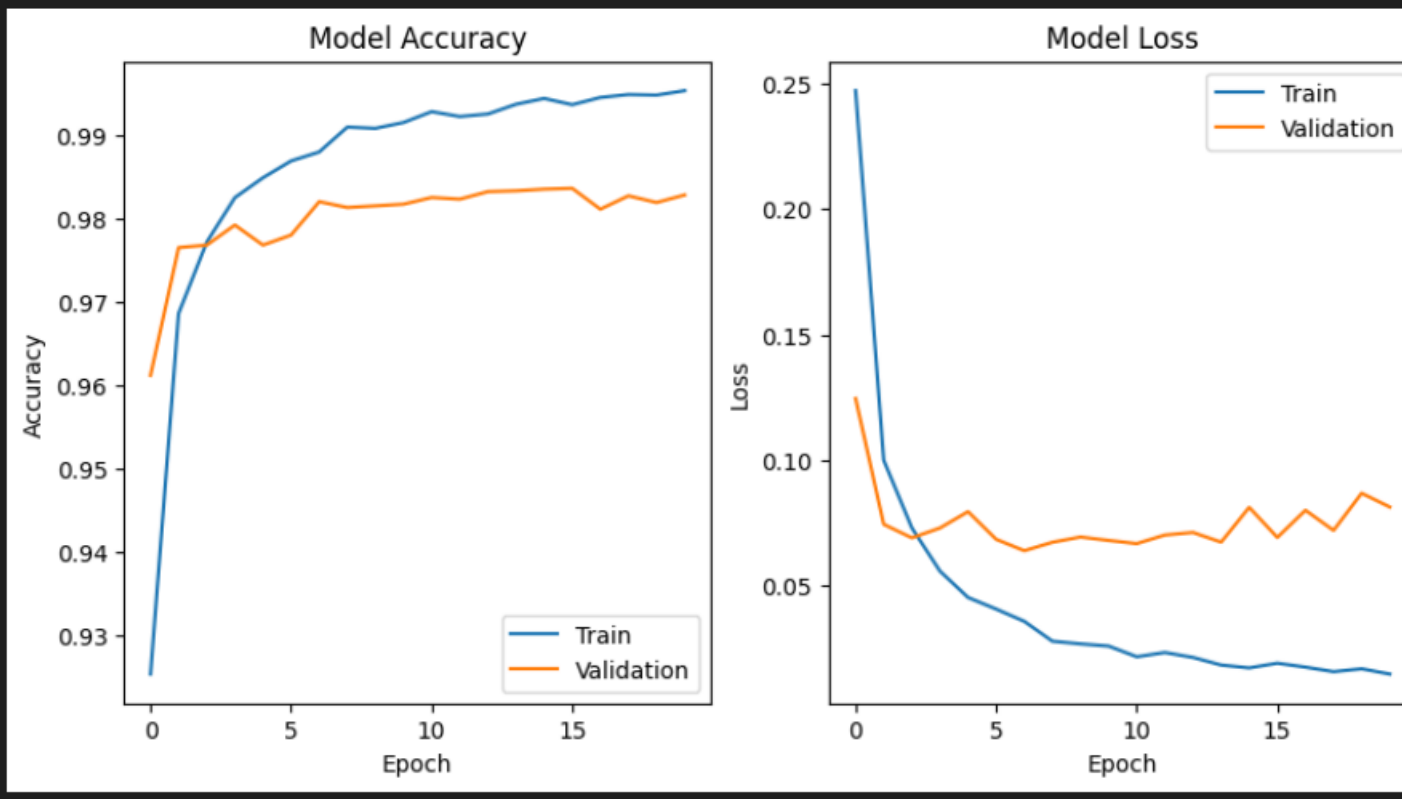
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt

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

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

num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                    epochs=20, batch_size=128)
```

```
                    epochs=20, batch_size=128)

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower right')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()
```

✓ 1m 31.3s

```
Epoch 1/20
469/469 [==============================] - 4s 8ms/step - loss: 0.2473 - accuracy: 0.9254 - val_loss: 0.1246 - v
Epoch 2/20
469/469 [==============================] - 4s 8ms/step - loss: 0.1000 - accuracy: 0.9686 - val_loss: 0.0744 - v
Epoch 3/20
469/469 [==============================] - 4s 8ms/step - loss: 0.0730 - accuracy: 0.9772 - val_loss: 0.0690 - v
Epoch 4/20
469/469 [==============================] - 4s 8ms/step - loss: 0.0557 - accuracy: 0.9825 - val_loss: 0.0729 - v
Epoch 5/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0452 - accuracy: 0.9849 - val_loss: 0.0794 - v
Epoch 6/20
469/469 [==============================] - 4s 8ms/step - loss: 0.0406 - accuracy: 0.9869 - val_loss: 0.0684 - v
Epoch 7/20
469/469 [==============================] - 4s 8ms/step - loss: 0.0357 - accuracy: 0.9880 - val_loss: 0.0639 - v
Epoch 8/20
469/469 [==============================] - 4s 8ms/step - loss: 0.0278 - accuracy: 0.9909 - val_loss: 0.0672 - v
Epoch 9/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0267 - accuracy: 0.9908 - val_loss: 0.0693 - v
Epoch 10/20
469/469 [==============================] - 5s 10ms/step - loss: 0.0258 - accuracy: 0.9915 - val_loss: 0.0680 -
Epoch 11/20
```

```
469/469 [==============================] - 4s 9ms/step - loss: 0.0216 - accuracy: 0.9928 - val_loss: 0.0667 -
Epoch 12/20
469/469 [==============================] - 5s 11ms/step - loss: 0.0233 - accuracy: 0.9922 - val_loss: 0.0701
Epoch 13/20
469/469 [==============================] - 5s 10ms/step - loss: 0.0213 - accuracy: 0.9925 - val_loss: 0.0712
Epoch 14/20
469/469 [==============================] - 5s 11ms/step - loss: 0.0183 - accuracy: 0.9937 - val_loss: 0.0673
Epoch 15/20
469/469 [==============================] - 5s 10ms/step - loss: 0.0172 - accuracy: 0.9944 - val_loss: 0.0812
Epoch 16/20
469/469 [==============================] - 5s 10ms/step - loss: 0.0190 - accuracy: 0.9936 - val_loss: 0.0692
Epoch 17/20
426/469 [==========================>...] - ETA: 0s - loss: 0.0165 - accuracy: 0.9948
```



Model accuracy using mnist with breast cancer dataset

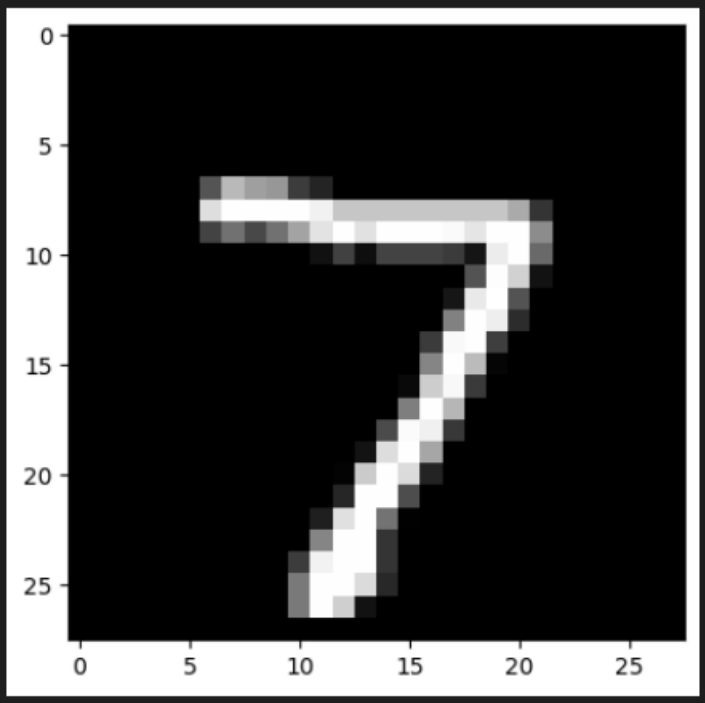
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np


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

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

num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
          epochs=20, batch_size=128)

plt.imshow(x_test[0], cmap='gray')
plt.show()

prediction = model.predict(x_test[0].reshape(1, -1))
print('Model prediction:', np.argmax(prediction))
```

✓ 1m 55.4s

```
Epoch 1/20
469/469 [==============================] - 6s 13ms/step - loss: 0.2495 - accuracy: 0.9251 - val_loss: 0.1098 -
Epoch 2/20
```

```
Epoch 1/20
469/469 [==============================] - 6s 13ms/step - loss: 0.2495 - accuracy: 0.9251 - val_loss: 0.1098
Epoch 2/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0994 - accuracy: 0.9690 - val_loss: 0.0777
Epoch 3/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0721 - accuracy: 0.9773 - val_loss: 0.0776
Epoch 4/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0555 - accuracy: 0.9822 - val_loss: 0.0701
Epoch 5/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0454 - accuracy: 0.9851 - val_loss: 0.0658
Epoch 6/20
469/469 [==============================] - 5s 11ms/step - loss: 0.0401 - accuracy: 0.9861 - val_loss: 0.0678
Epoch 7/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0349 - accuracy: 0.9887 - val_loss: 0.0694
Epoch 8/20
469/469 [==============================] - 5s 12ms/step - loss: 0.0300 - accuracy: 0.9901 - val_loss: 0.0640
Epoch 9/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0275 - accuracy: 0.9908 - val_loss: 0.0666
Epoch 10/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0253 - accuracy: 0.9917 - val_loss: 0.0736
Epoch 11/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0225 - accuracy: 0.9926 - val_loss: 0.0822
Epoch 12/20
469/469 [==============================] - 5s 12ms/step - loss: 0.0218 - accuracy: 0.9928 - val_loss: 0.0702
Epoch 13/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0225 - accuracy: 0.9924 - val_loss: 0.0781
Epoch 14/20
469/469 [==============================] - 6s 13ms/step - loss: 0.0164 - accuracy: 0.9947 - val_loss: 0.0696
Epoch 15/20
469/469 [==============================] - 6s 13ms/step - loss: 0.0209 - accuracy: 0.9928 - val_loss: 0.0820
Epoch 16/20
469/469 [==============================] - 6s 13ms/step - loss: 0.0198 - accuracy: 0.9934 - val_loss: 0.0747
Epoch 17/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0174 - accuracy: 0.9941 - val_loss: 0.0788
Epoch 18/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0165 - accuracy: 0.9947 - val_loss: 0.0797
Epoch 19/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0151 - accuracy: 0.9952 - val_loss: 0.0699
Epoch 20/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0157 - accuracy: 0.9949 - val_loss: 0.0811
```

```
Epoch 20/20
469/469 [==============================] - 6s 12ms/step - loss: 0.0157 - accuracy: 0.9949 - val_loss: 0.081
```




```
1/1 [==============================] - 0s 82ms/step
Model prediction: 7
```

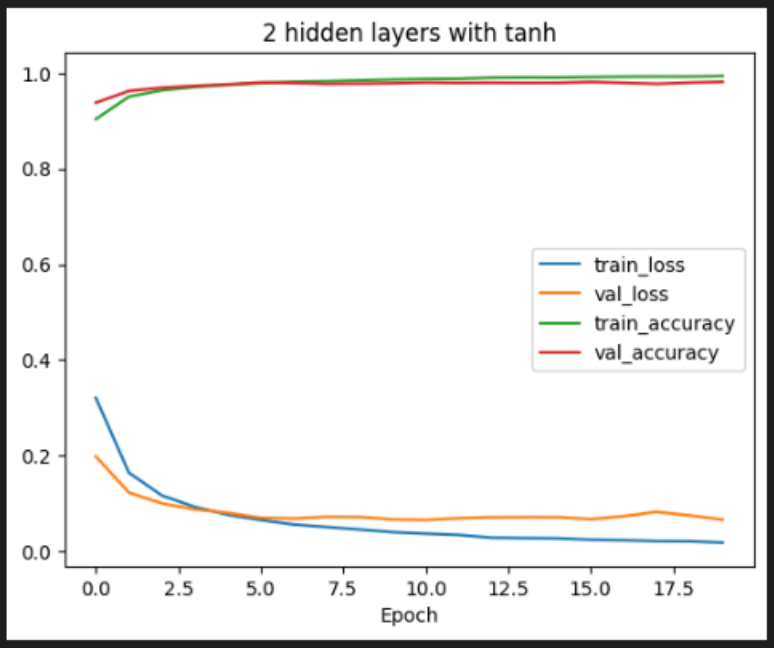
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

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

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

num_classes = 10
```

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

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

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

num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

models = []

model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with tanh', model))

model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
```

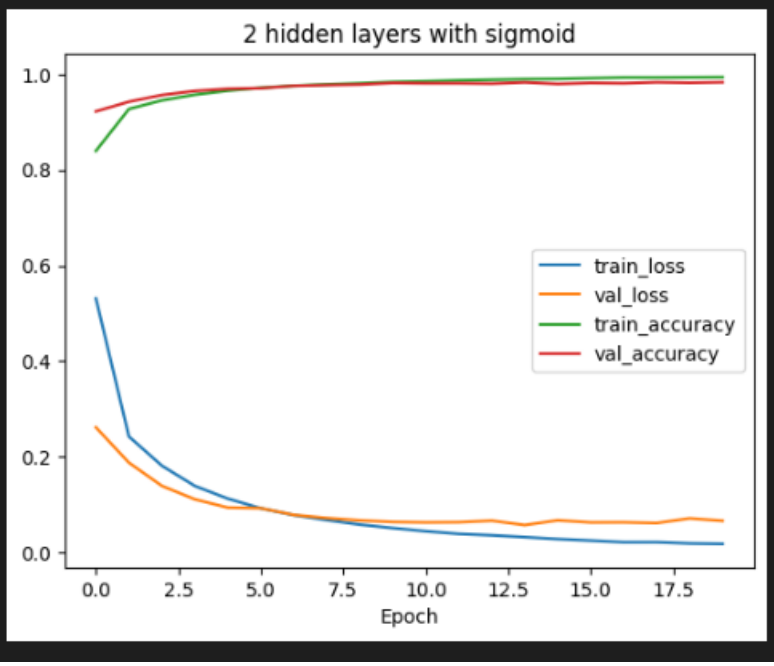
```
models.append(('2 hidden layers with tanh', model))

model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))

for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test
                        epochs=20, batch_size=128, verbose=0)
    plt.plot(history.history['loss'], label='train_loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.plot(history.history['accuracy'], label='train_accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.title(name)
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

    loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```

✓ 6m 11.4s

```
1 hidden layer with sigmoid - Test loss: 0.0615, Test accuracy: 0.9824
```



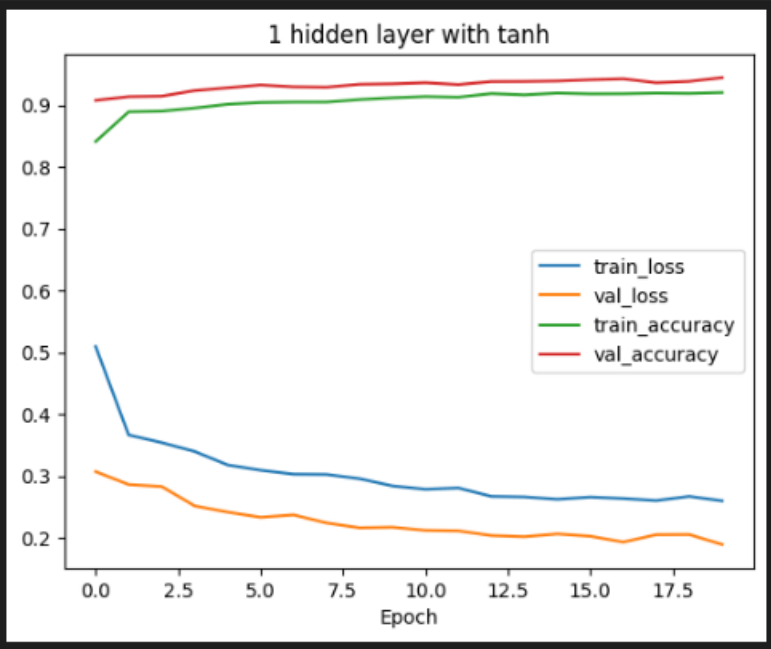

```
2 hidden layers with tanh - Test loss: 0.0658, Test accuracy: 0.9817
```



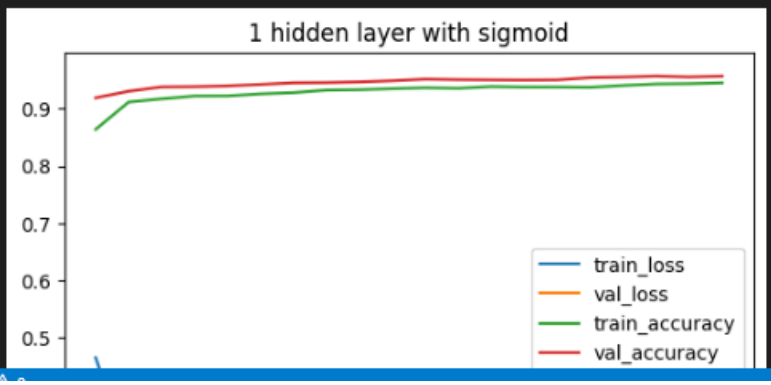
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

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

num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

models = []

model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with tanh', model))

model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))

for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                        epochs=20, batch_size=128, verbose=0)
    plt.plot(history.history['loss'], label='train_loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.plot(history.history['accuracy'], label='train_accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
```

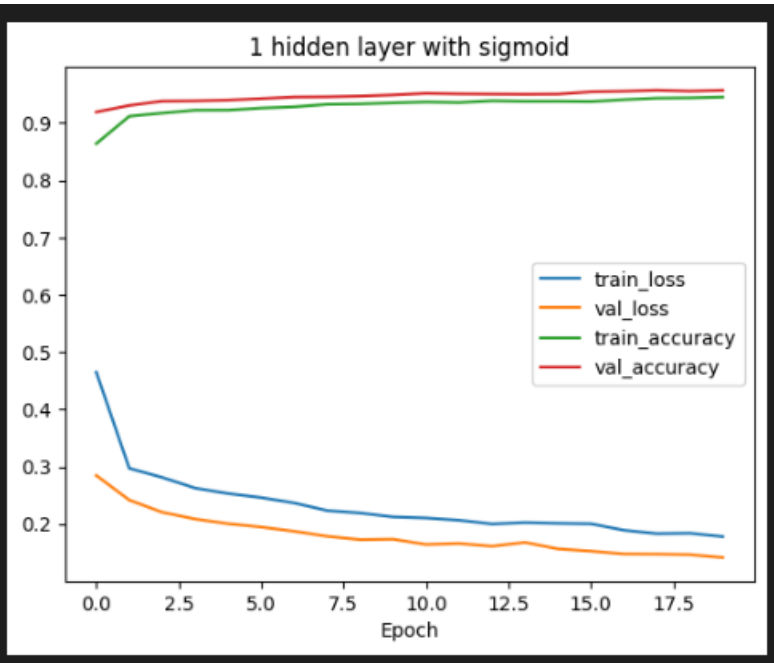
```
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.plot(history.history['accuracy'], label='train_accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.title(name)
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

    loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```

[54] 5m 38.2s

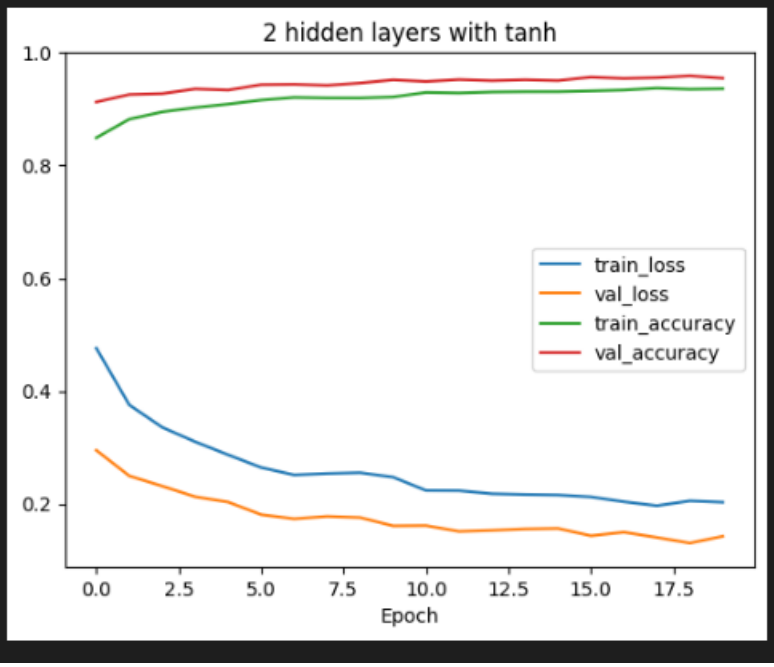


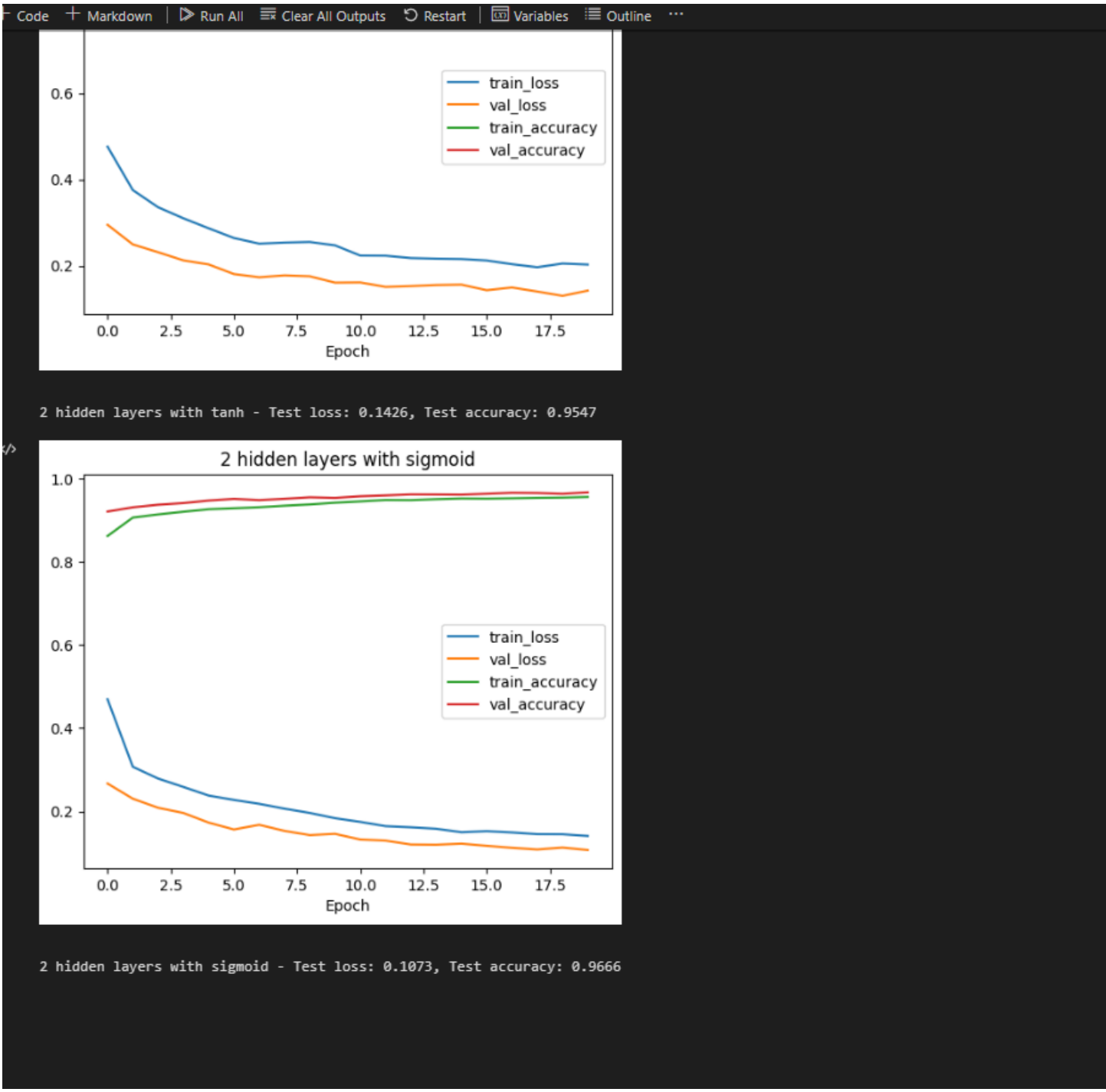

```
1 hidden layer with tanh - Test loss: 0.1896, Test accuracy: 0.9442
```






```
1 hidden layer with sigmoid - Test loss: 0.1414, Test accuracy: 0.9570
```





GitHub : https://github.com/pavan-reddy-28/ICP6.git