Spring 2024: CS5720

Neural Networks & Deep Learning - ICP-6

ICP_Basics in Keras

NAME: Sai Ram Uppalapati   ID: 700740307

GITHUB LINK: https://github.com/sxu03070/ICP_6

CODE & SCREENSHOTS FOR RESULTS:

Use Case Description:
Predicting the diabetes disease

Programming elements:
Keras Basics

In class programming:

1. Use the use case in the class:
   a. Add more Dense layers to the existing code and check how the accuracy changes.

2. Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model.

3. Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below).

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
```

Breast Cancer dataset is designated to predict if a patient has Malignant (M) or Benign = B cancer

```
[1] #read the data
    import pandas as pd
    path_to_csv = '/content/diabetes.csv'
```

```
import keras
import pandas
from keras.models import Sequential
from keras.layers import Dense, Activation

# load dataset
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)
```

```python
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))
```

```
Epoch 1/100
18/18 [==============================] - 1s 8ms/step - loss: 1.8322 - acc: 0.6337
Epoch 2/100
18/18 [==============================] - 0s 5ms/step - loss: 1.2988 - acc: 0.5955
Epoch 3/100
18/18 [==============================] - 0s 4ms/step - loss: 1.1608 - acc: 0.6059
Epoch 4/100
18/18 [==============================] - 0s 7ms/step - loss: 1.0353 - acc: 0.6024
Epoch 5/100
18/18 [==============================] - 0s 4ms/step - loss: 0.9891 - acc: 0.5729
Epoch 6/100
18/18 [==============================] - 0s 4ms/step - loss: 0.9128 - acc: 0.6111
Epoch 7/100
18/18 [==============================] - 0s 4ms/step - loss: 0.8358 - acc: 0.6615
Epoch 8/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7745 - acc: 0.6649
Epoch 9/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7802 - acc: 0.6597
Epoch 10/100
18/18 [==============================] - 0s 2ms/step - loss: 0.7386 - acc: 0.6719
```

```
Epoch 95/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5619 - acc: 0.6979
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5337 - acc: 0.7153
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5379 - acc: 0.7257
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5452 - acc: 0.7118
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5332 - acc: 0.7170
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5493 - acc: 0.7014
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 20)                180

 dense_1 (Dense)             (None, 4)                 84

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 269 (1.05 KB)
Trainable params: 269 (1.05 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6974 - acc: 0.6146
[0.6974080204963684, 0.6145833134651184]
```

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

# load dataset
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))
```

```
Epoch 95/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3150 - acc: 0.9155
Epoch 96/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2942 - acc: 0.9249
Epoch 97/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2966 - acc: 0.9272
Epoch 98/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3716 - acc: 0.9014
Epoch 99/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2673 - acc: 0.9296
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2768 - acc: 0.9272
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_13 (Dense)            (None, 20)                620

 dense_14 (Dense)            (None, 1)                 21

=================================================================
Total params: 641 (2.50 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
5/5 [==============================] - 0s 3ms/step - loss: 0.7015 - acc: 0.8671
[0.7014830708503723, 0.867132842540741]
```

```
[6] from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
```

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

# load dataset
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))
```

```
Epoch 95/100
14/14 [==============================] - 0s 4ms/step - loss: 0.2502 - acc: 0.9155
Epoch 96/100
14/14 [==============================] - 0s 5ms/step - loss: 0.2224 - acc: 0.9225
Epoch 97/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2115 - acc: 0.9296
Epoch 98/100
14/14 [==============================] - 0s 4ms/step - loss: 0.2071 - acc: 0.9202
Epoch 99/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2020 - acc: 0.9272
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2072 - acc: 0.9390
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 20)                620

 dense_16 (Dense)            (None, 1)                 21

=================================================================
Total params: 641 (2.50 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
5/5 [==============================] - 0s 5ms/step - loss: 0.4221 - acc: 0.8951
[0.42209988832473755, 0.8951048851013184]
```

In class programming:

Use Image Classification on the hand written digits data set (mnist)

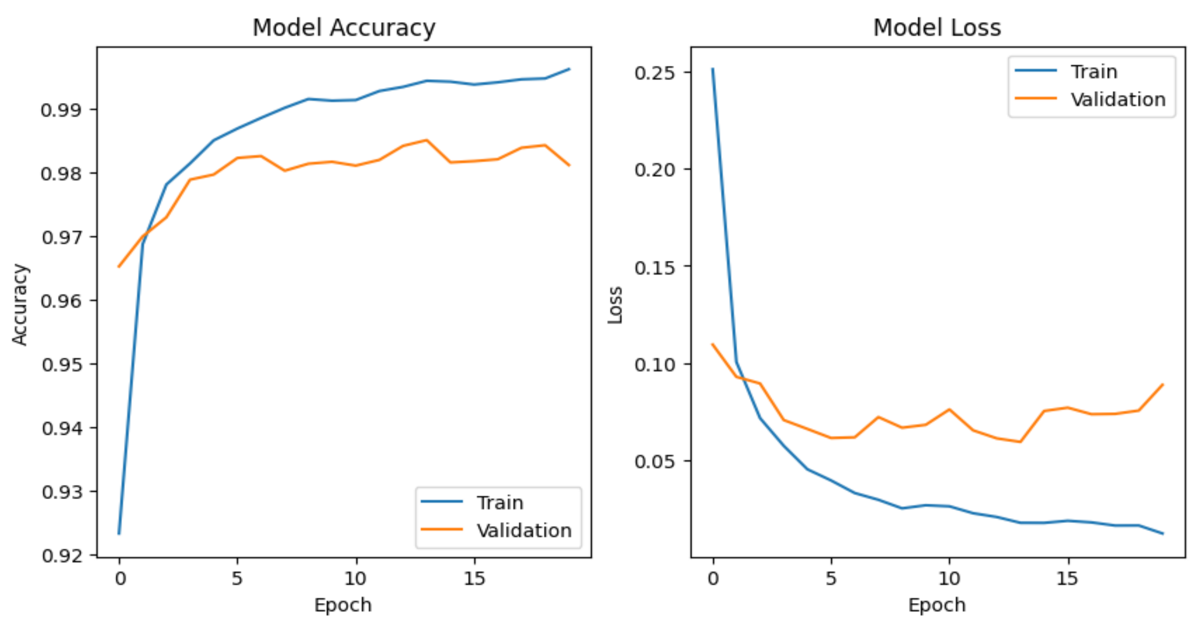
1. Plot the loss and accuracy for both training data and validation data using the history object in the source code.
2. Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.
3. We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.
4. Run the same code without scaling the images and check the performance?

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

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```python
# create a simple neural network model
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'])

# train the model and record the training history
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                    epochs=20, batch_size=128)
```

```
Epoch 10/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0265 - accuracy: 0.9912 - val_loss: 0.0680 - val_accuracy: 0.9816
Epoch 11/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0260 - accuracy: 0.9913 - val_loss: 0.0759 - val_accuracy: 0.9810
Epoch 12/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0224 - accuracy: 0.9927 - val_loss: 0.0652 - val_accuracy: 0.9819
Epoch 13/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0205 - accuracy: 0.9934 - val_loss: 0.0610 - val_accuracy: 0.9841
Epoch 14/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0175 - accuracy: 0.9943 - val_loss: 0.0592 - val_accuracy: 0.9850
Epoch 15/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0174 - accuracy: 0.9942 - val_loss: 0.0751 - val_accuracy: 0.9815
Epoch 16/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0185 - accuracy: 0.9937 - val_loss: 0.0768 - val_accuracy: 0.9817
Epoch 17/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0177 - accuracy: 0.9941 - val_loss: 0.0735 - val_accuracy: 0.9820
Epoch 18/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0161 - accuracy: 0.9945 - val_loss: 0.0736 - val_accuracy: 0.9838
Epoch 19/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0161 - accuracy: 0.9947 - val_loss: 0.0754 - val_accuracy: 0.9842
Epoch 20/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0120 - accuracy: 0.9961 - val_loss: 0.0886 - val_accuracy: 0.9811
```

```python
# plot the training and validation accuracy and loss curves
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()
```

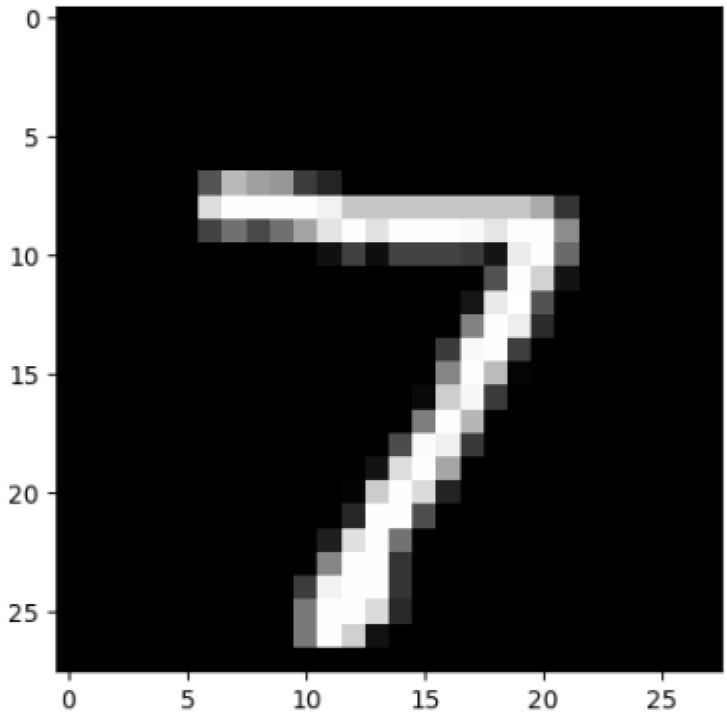


```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```python
# create a simple neural network model
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'])

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

```
Epoch 15/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0167 - accuracy: 0.9945 - val_loss: 0.0701 - val_accuracy: 0.9819
Epoch 16/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0186 - accuracy: 0.9942 - val_loss: 0.0730 - val_accuracy: 0.9825
Epoch 17/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0182 - accuracy: 0.9936 - val_loss: 0.0872 - val_accuracy: 0.9814
Epoch 18/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0170 - accuracy: 0.9947 - val_loss: 0.0745 - val_accuracy: 0.9836
Epoch 19/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0148 - accuracy: 0.9948 - val_loss: 0.0772 - val_accuracy: 0.9832
Epoch 20/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0146 - accuracy: 0.9956 - val_loss: 0.0738 - val_accuracy: 0.9836
<keras.src.callbacks.History at 0x790d1b141900>
```

```python
# plot one of the images in the test data
plt.imshow(x_test[0], cmap='gray')
plt.show()

# make a prediction on the image using the trained model
prediction = model.predict(x_test[0].reshape(1, -1))
print('Model prediction:', np.argmax(prediction))
```

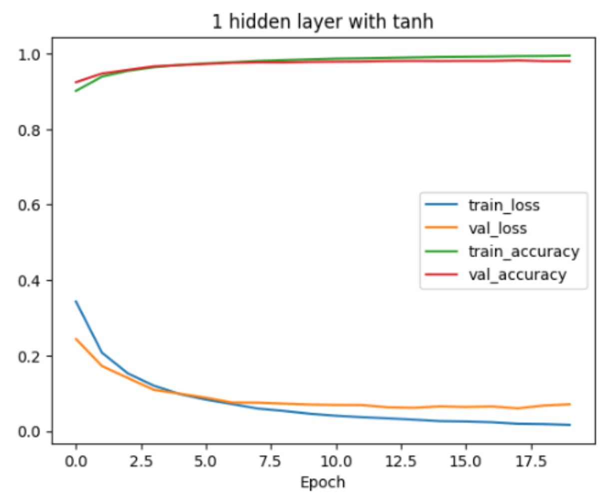



```
1/1 [==============================] - 0s 98ms/step
Model prediction: 7
```

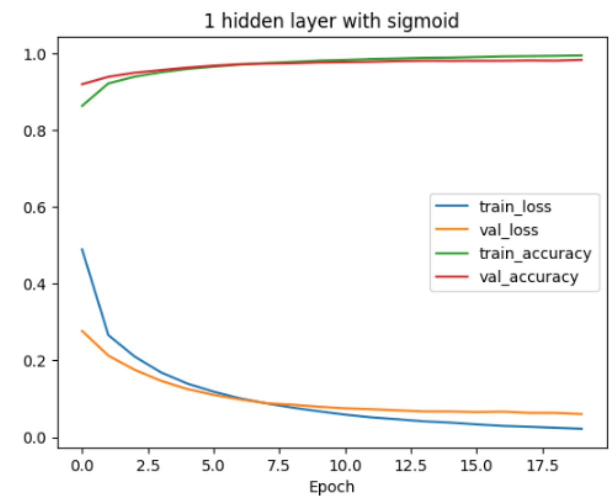
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```python
# create a list of models to train
models = []

# model with 1 hidden layer and tanh activation
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 with 1 hidden layer and sigmoid activation
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 with 2 hidden layers and tanh activation
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 with 2 hidden layers and sigmoid activation
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))
```

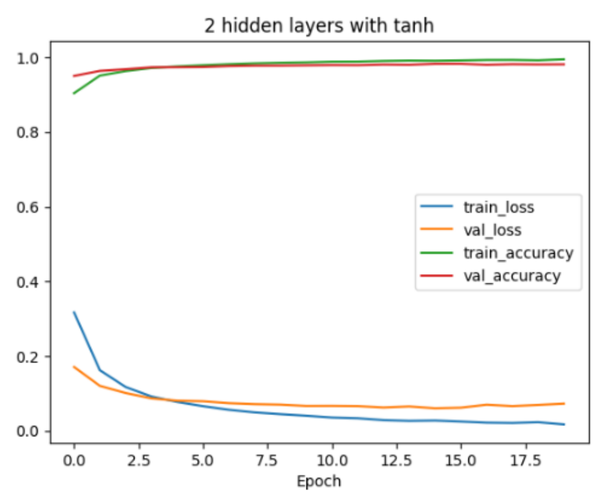
```python
# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                        epochs=20, batch_size=128, verbose=0)
    # plot loss and accuracy curves
    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()

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



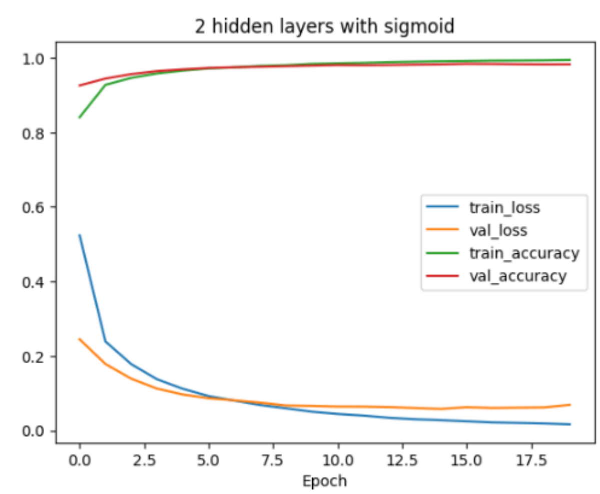

```
1 hidden layer with tanh - Test loss: 0.0709, Test accuracy: 0.9803
```




```
1 hidden layer with sigmoid - Test loss: 0.0604, Test accuracy: 0.9827
```




```
2 hidden layers with tanh - Test loss: 0.0726, Test accuracy: 0.9810
```



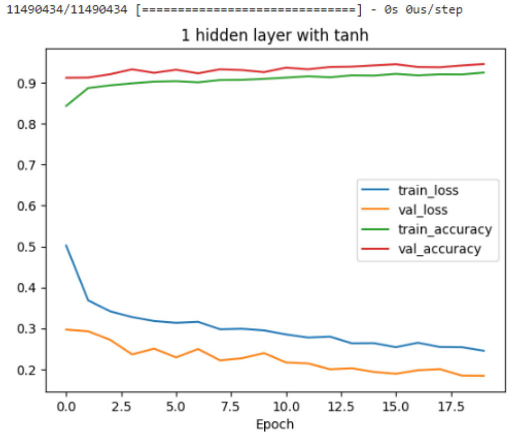

```
2 hidden layers with sigmoid - Test loss: 0.0689, Test accuracy: 0.9829
```

```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

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a list of models to train
models = []

# model with 1 hidden layer and tanh activation
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 with 1 hidden layer and sigmoid activation
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))
```

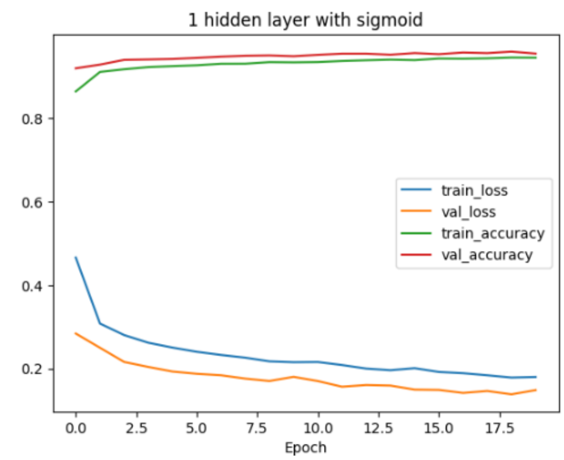
```python
# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden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

# model with 2 hidden layers and sigmoid activation
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))
```

```python
# train each model and plot loss and accuracy curves
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)
    # plot loss and accuracy curves
    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()

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

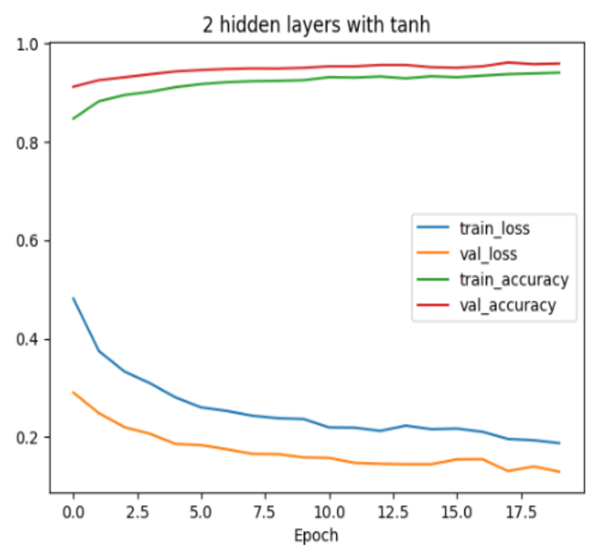
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```



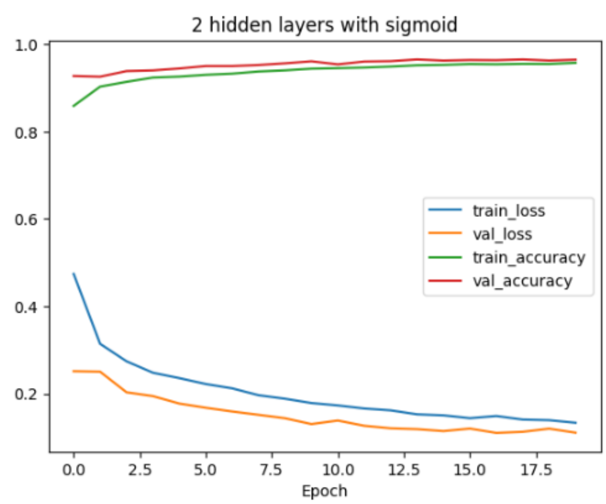

```
1 hidden layer with tanh - Test loss: 0.1839, Test accuracy: 0.9458
```




```
1 hidden layer with sigmoid - Test loss: 0.1486, Test accuracy: 0.9541
```




```
2 hidden layers with tanh - Test loss: 0.1294, Test accuracy: 0.9584
```




```
2 hidden layers with sigmoid - Test loss: 0.1111, Test accuracy: 0.9646
```