Neural Networks & Deep Learning - ICP-6

Name : Naga Phaneendra Kumara Gupta Mogili

700757977

Git Hub Url:
https://github.com/nagaphaneendra2001/Deep_Learning_Neural_Networks

Program Dense:

C: > Users > User > Desktop > dense.py > {} pd

```python
import keras
import pandas
from keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
dataset = pd.read_csv('diabetes.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(123)
my_first_nn = Sequential()
my_first_nn.add(Dense(20, input_dim=8, activation='relu'))
my_first_nn.add(Dense(4, activation='relu'))
my_first_nn.add(Dense(1, activation='sigmoid'))
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=25,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
```

Output:

```
18/18 [==============================] - 1s 3ms/step - loss: 14.7099 - acc: 0.3385
Epoch 2/25
18/18 [==============================] - 0s 2ms/step - loss: 7.6133 - acc: 0.3333
Epoch 3/25
18/18 [==============================] - 0s 2ms/step - loss: 3.8574 - acc: 0.3333
Epoch 4/25
18/18 [==============================] - 0s 2ms/step - loss: 1.9134 - acc: 0.4132
Epoch 5/25
18/18 [==============================] - 0s 2ms/step - loss: 1.0924 - acc: 0.5573
Epoch 6/25
18/18 [==============================] - 0s 2ms/step - loss: 0.8369 - acc: 0.6163
Epoch 7/25
18/18 [==============================] - 0s 2ms/step - loss: 0.7381 - acc: 0.6372
Epoch 8/25
18/18 [==============================] - 0s 2ms/step - loss: 0.7035 - acc: 0.6580
Epoch 9/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6889 - acc: 0.6632
Epoch 10/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6824 - acc: 0.6667
Epoch 11/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6789 - acc: 0.6684
Epoch 12/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6758 - acc: 0.6701
Epoch 13/25
18/18 [==============================] - 0s 3ms/step - loss: 0.6731 - acc: 0.6719
Epoch 14/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6714 - acc: 0.6719
Epoch 15/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6699 - acc: 0.6719
Epoch 16/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6682 - acc: 0.6719
Epoch 17/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6669 - acc: 0.6719
Epoch 18/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6652 - acc: 0.6719
Epoch 19/25
18/18 [==============================] - 0s 3ms/step - loss: 0.6641 - acc: 0.6719
Epoch 20/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6624 - acc: 0.6719
Epoch 21/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6612 - acc: 0.6719
Epoch 22/25
18/18 [==============================] - 0s 3ms/step - loss: 0.6599 - acc: 0.6719
Epoch 23/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6585 - acc: 0.6736
Epoch 24/25
18/18 [==============================] - 0s 3ms/step - loss: 0.6571 - acc: 0.6719
Epoch 25/25
18/18 [==============================] - 0s 2ms/step - loss: 0.6566 - acc: 0.6701
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 0s/step - loss: 0.6647 - acc: 0.6354
[0.6647228598594666, 0.6354166865348816]
```

Program breast:

: > Users > User > Desktop > breast.py > {} pd

```
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from tensorflow.keras.layers import Dense
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()
my_nn.add(Dense(20, input_dim=30, activation='relu'))
my_nn.add(Dense(1, activation='sigmoid'))
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))
```

Output:

```
14/14 [==============================] - 1s 4ms/step - loss: 27.5164 - acc: 0.3803
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 6.1277 - acc: 0.4789
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: 2.1547 - acc: 0.6221
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: 1.1755 - acc: 0.7230
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: 0.7657 - acc: 0.7793
Epoch 6/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5271 - acc: 0.8498
Epoch 7/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4687 - acc: 0.8850
Epoch 8/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4580 - acc: 0.8873
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4492 - acc: 0.8897
Epoch 10/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4214 - acc: 0.8920
Epoch 11/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4080 - acc: 0.8944
Epoch 12/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4088 - acc: 0.8920
Epoch 13/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4136 - acc: 0.8991
Epoch 14/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4060 - acc: 0.8991
Epoch 15/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3576 - acc: 0.9014
Epoch 16/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3419 - acc: 0.8920
Epoch 17/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3415 - acc: 0.8991
Epoch 18/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3225 - acc: 0.9061
Epoch 19/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3170 - acc: 0.9085
Epoch 20/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3105 - acc: 0.9131
Epoch 21/100
```

```
Epoch 23/100
14/14 [==============================] - 0s 13ms/step - loss: 0.3051 - acc: 0.9131
Epoch 24/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2823 - acc: 0.9131
Epoch 25/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2797 - acc: 0.9225
Epoch 26/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2841 - acc: 0.9014
Epoch 27/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2952 - acc: 0.9202
Epoch 28/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2868 - acc: 0.9131
Epoch 29/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2547 - acc: 0.9225
Epoch 30/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2643 - acc: 0.9155
Epoch 31/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2335 - acc: 0.9131
Epoch 32/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2407 - acc: 0.9272
Epoch 33/100
14/14 [==============================] - 0s 1ms/step - loss: 0.2342 - acc: 0.9178
Epoch 34/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2394 - acc: 0.9178
Epoch 35/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2570 - acc: 0.9249
Epoch 36/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2411 - acc: 0.9272
Epoch 37/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2046 - acc: 0.9319
Epoch 38/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2421 - acc: 0.9202
Epoch 39/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2074 - acc: 0.9296
Epoch 40/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2101 - acc: 0.9272
Epoch 41/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1984 - acc: 0.9249
Epoch 42/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2318 - acc: 0.9319
```

```
Epoch 90/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1505 - acc: 0.9366
Epoch 91/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1505 - acc: 0.9343
Epoch 92/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1409 - acc: 0.9366
Epoch 93/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2067 - acc: 0.9413
Epoch 94/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1733 - acc: 0.9460
Epoch 95/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1433 - acc: 0.9413
Epoch 96/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1845 - acc: 0.9460
Epoch 97/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1494 - acc: 0.9366
Epoch 98/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1885 - acc: 0.9272
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1725 - acc: 0.9437
Epoch 100/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1790 - acc: 0.9366
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 20)                620

 dense_1 (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 4ms/step - loss: 0.3496 - acc: 0.9091
[0.3496224880218506, 0.9090909361839294]

C:\Users\User\Desktop>
```

Program Normal:

C: > Users > User > Desktop > normal.py > {} np

```python
from sklearn.preprocessing import StandardScaler
import numpy as np
sc = StandardScaler()
X_train_normalized = sc.fit_transform()
X_test_normalized = sc.transform()
model_normalized = Sequential()
model_normalized.add(Dense(32, activation='relu', input_shape=(X_train.shape[1],)))
model_normalized.add(Dense(64, activation='relu'))
model_normalized.add(Dense(128, activation='relu'))
model_normalized.add(Dense(1, activation='sigmoid'))
model_normalized.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model_normalized.fit(X_train_normalized, y_train, epochs=10, batch_size=32, validation_data=(X_test_normalized, y_test))
accuracy_normalized = model_normalized.evaluate(X_test_normalized, y_test)[1]
print("Accuracy with normalization:", accuracy_normalized)

```

Output:

```
Epoch 90/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1505 - acc: 0.9366
Epoch 91/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1505 - acc: 0.9343
Epoch 92/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1409 - acc: 0.9366
Epoch 93/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2067 - acc: 0.9413
Epoch 94/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1733 - acc: 0.9460
Epoch 95/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1433 - acc: 0.9413
Epoch 96/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1845 - acc: 0.9460
Epoch 97/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1494 - acc: 0.9366
Epoch 98/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1885 - acc: 0.9272
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1725 - acc: 0.9437
Epoch 100/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1790 - acc: 0.9366
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 20)                620

 dense_1 (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 4ms/step - loss: 0.3496 - acc: 0.9091
[0.3496224880218506, 0.9090909361839294]

C:\Users\User\Desktop>
```

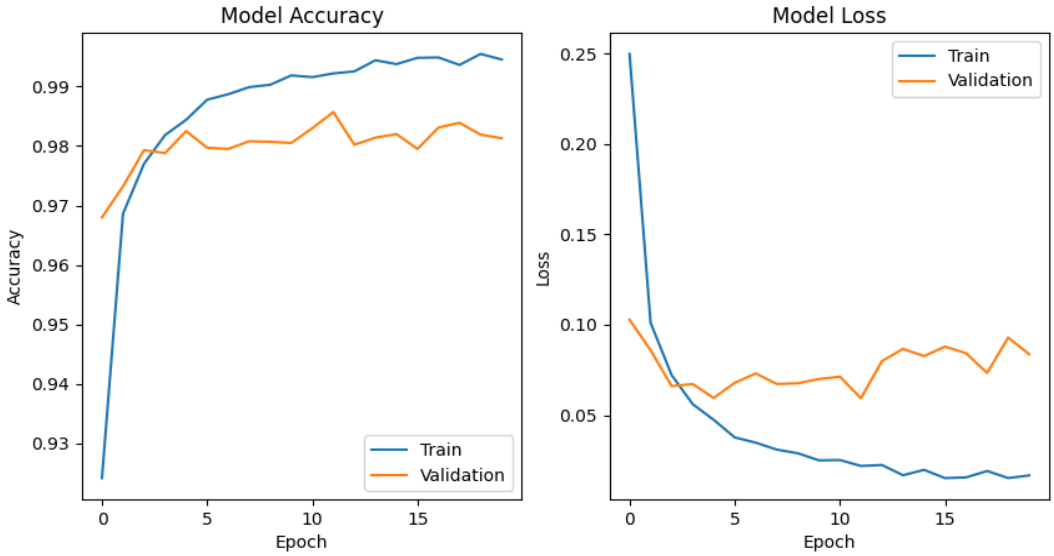
Program Image 1:

C: > Users > User > Desktop > image1.py > {} plt

```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
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

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

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

# 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(train_images.reshape(-1, 784), train_labels, validation_data=(test_images.reshape(-1, 784), test_labels),
                    epochs=20, batch_size=128)

# plot the training and validation accuracy and loss curves
plt.figure(figsize=(10, 5))
```

```python
history = model.fit(train_images.reshape(-1, 784), train_labels, validation_data=(test_images.reshape(-1, 784), test_labels),
                    epochs=20, batch_size=128)

# 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()
```

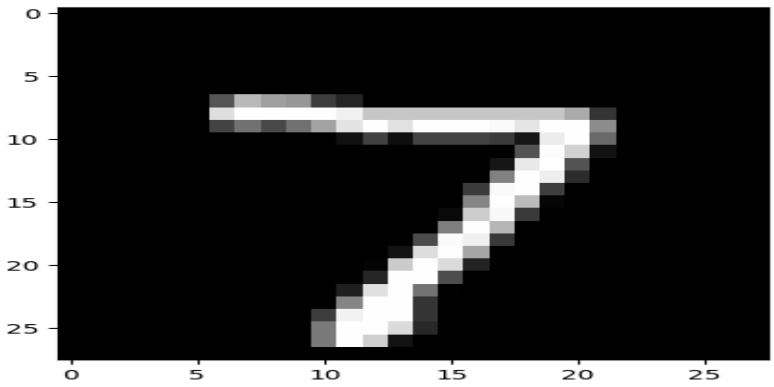
Output:



Program Image 2:

```
C: > Users > User > Desktop > image2.py > {} plt
1   import keras
2   from keras.datasets import mnist
3   from keras.models import Sequential
4   from keras.layers import Dense, Dropout
5   import matplotlib.pyplot as plt
6   import numpy as np
7   (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
8   train_images = train_images.astype('float32') / 255
9   test_images = test_images.astype('float32') / 255
10  num_classes = 10
11  train_labels = keras.utils.to_categorical(train_labels, num_classes)
12  test_labels = keras.utils.to_categorical(test_labels, num_classes)
13  model = Sequential()
14  model.add(Dense(512, activation='relu', input_shape=(784,)))
15  model.add(Dropout(0.2))
16  model.add(Dense(512, activation='relu'))
17  model.add(Dropout(0.2))
18  model.add(Dense(num_classes, activation='softmax'))
19  model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
20
21  model.fit(train_images.reshape(-1, 784), train_labels, validation_data=(test_images.reshape(-1, 784), test_labels),
22            epochs=20, batch_size=128)
23
24  plt.imshow(test_images[0], cmap='gray')
25  plt.show()
26  prediction = model.predict(test_images[0].reshape(1, -1))
27  print('Model prediction:', np.argmax(prediction))
```

Output:



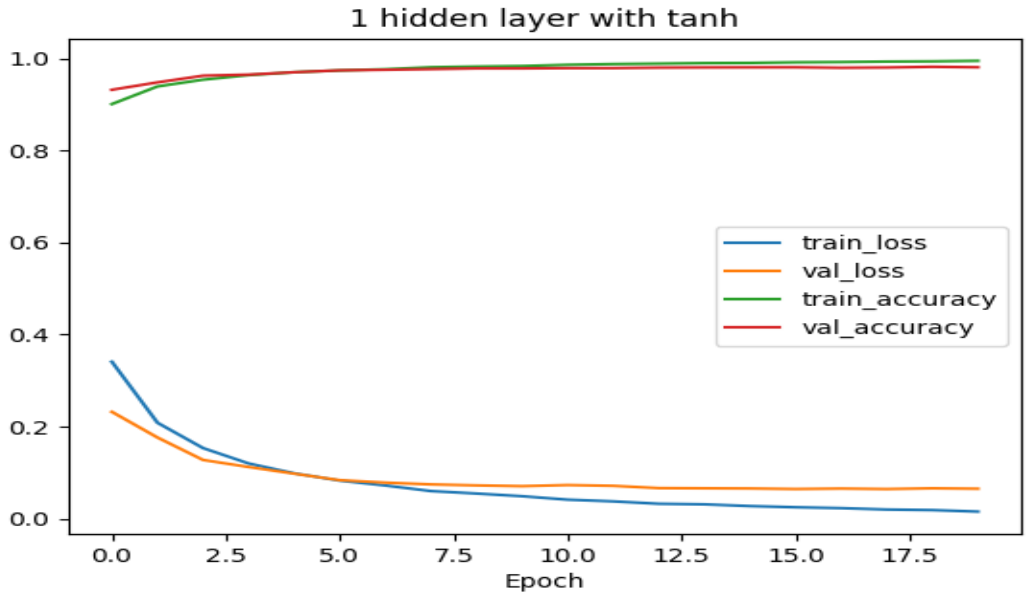
Program Image 3:

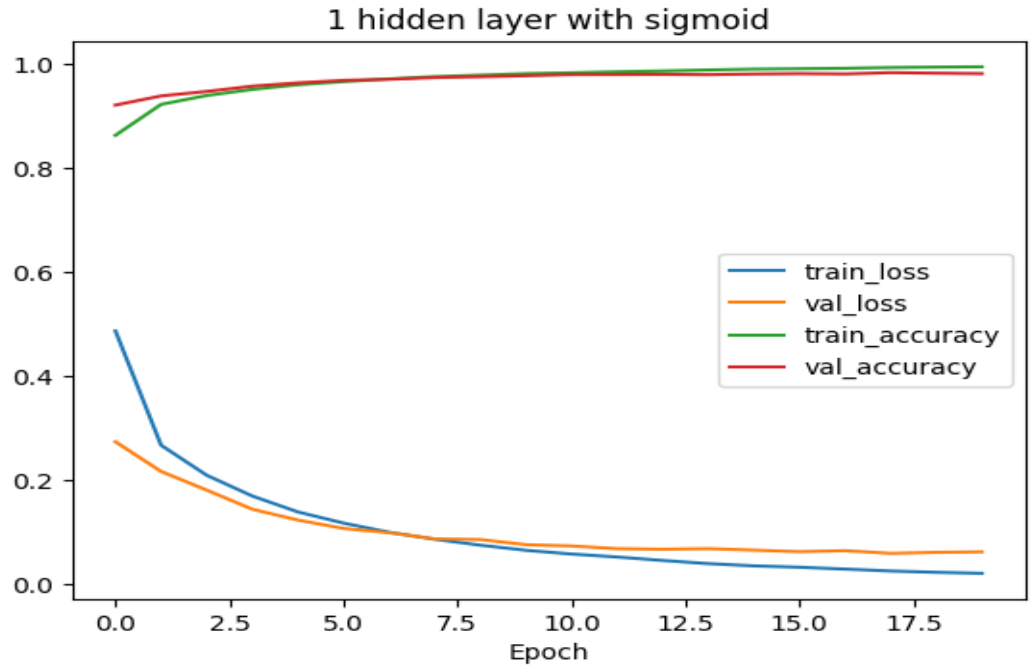
C: > Users > User > Desktop > image3.py > {} plt

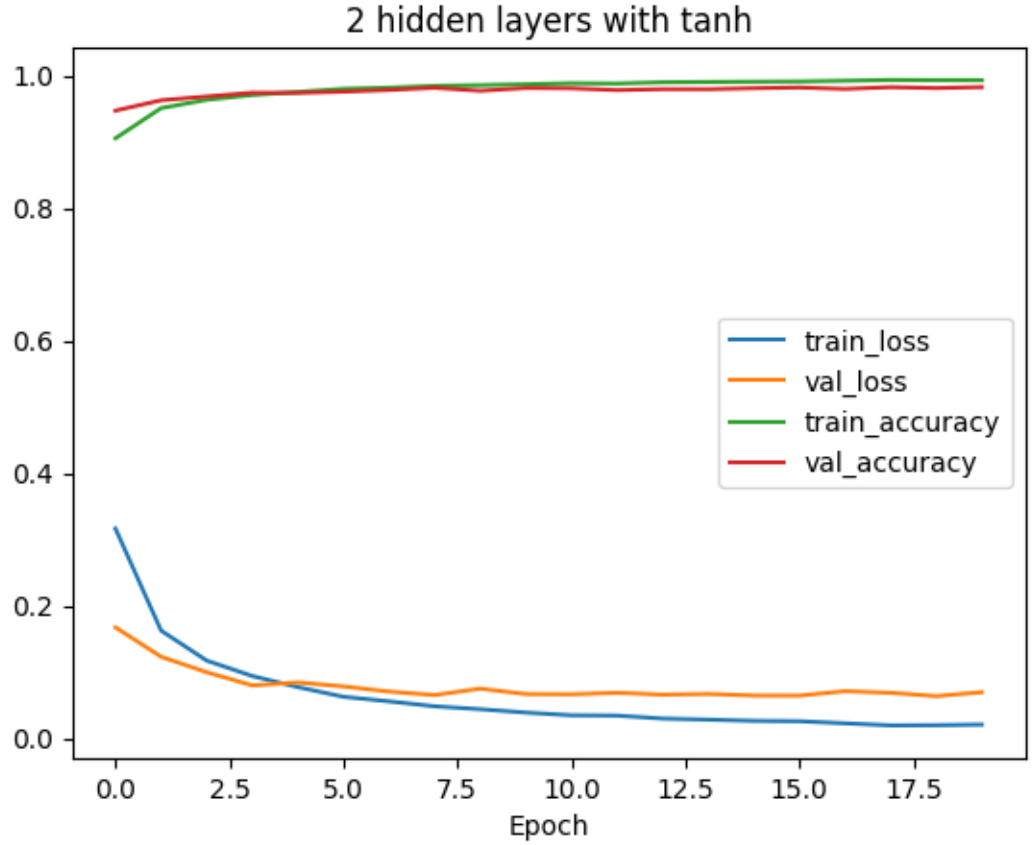
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

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

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

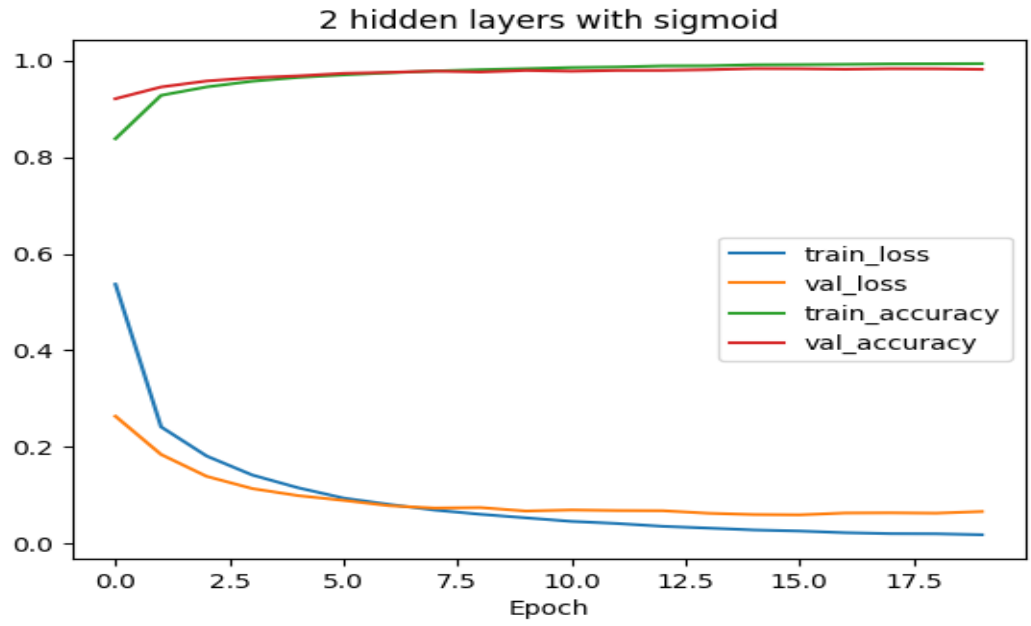
```
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

# 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))

# 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(train_images.reshape(-1, 784), train_labels, validation_data=(test_images.reshape(-1, 784), test_labels),
                        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()
loss, accuracy = model.evaluate(test_images.reshape(-1, 784), test_labels, verbose=0)
print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```

Output:




1 hidden layer with sigmoid
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
train_loss
val_loss
train_accuracy
val_accuracy


2 hidden layers with tanh
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
train_loss
val_loss
train_accuracy
val_accuracy



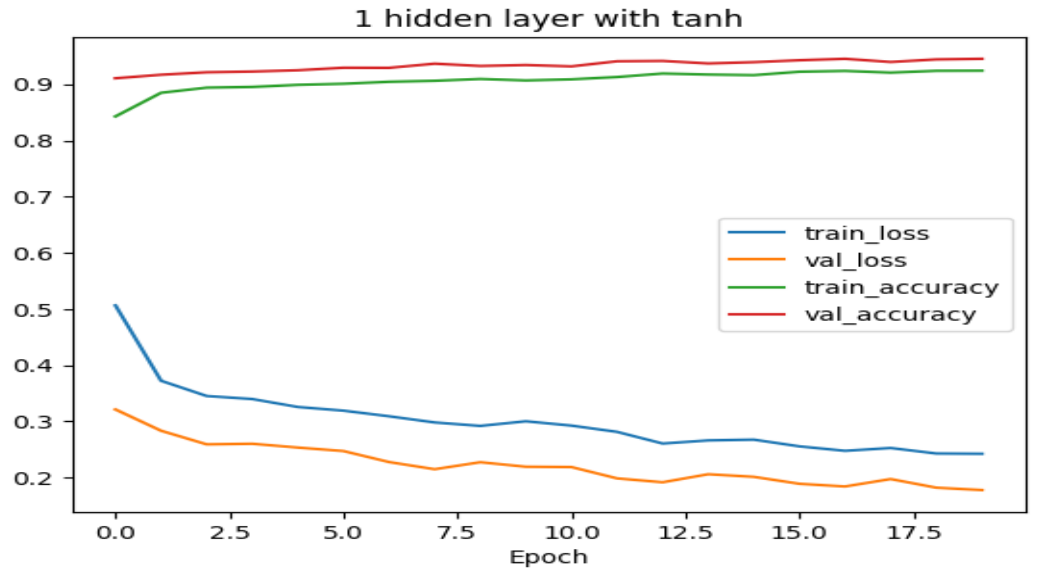
Program Image 4:

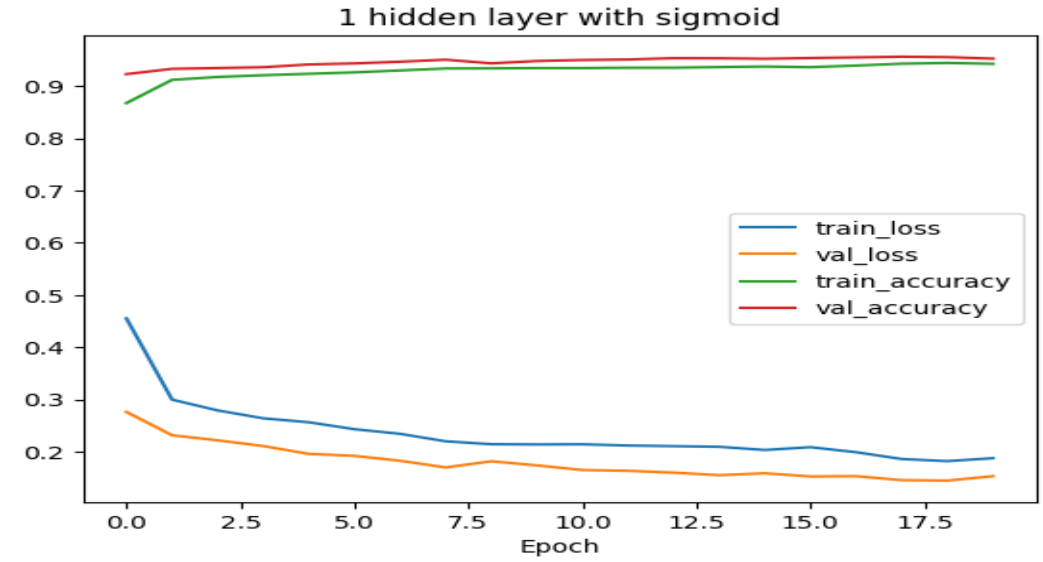
C: > Users > User > Desktop > image3.py > {} plt

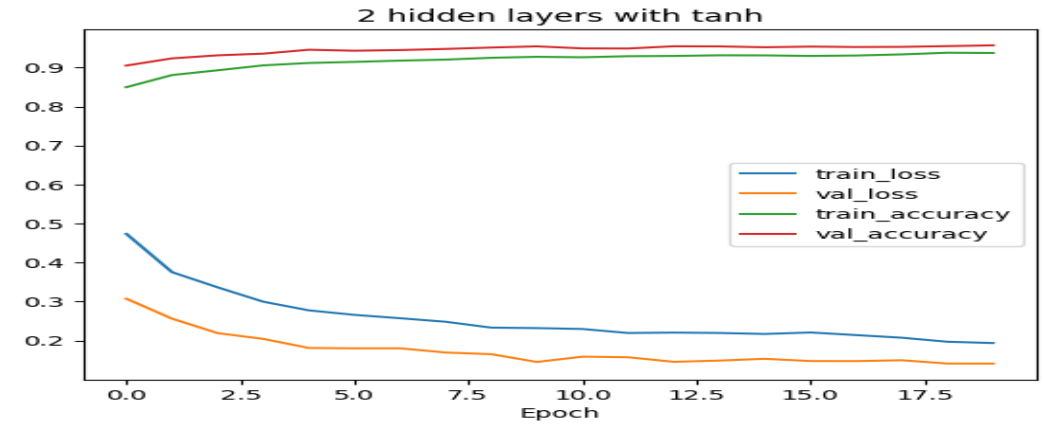
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

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

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

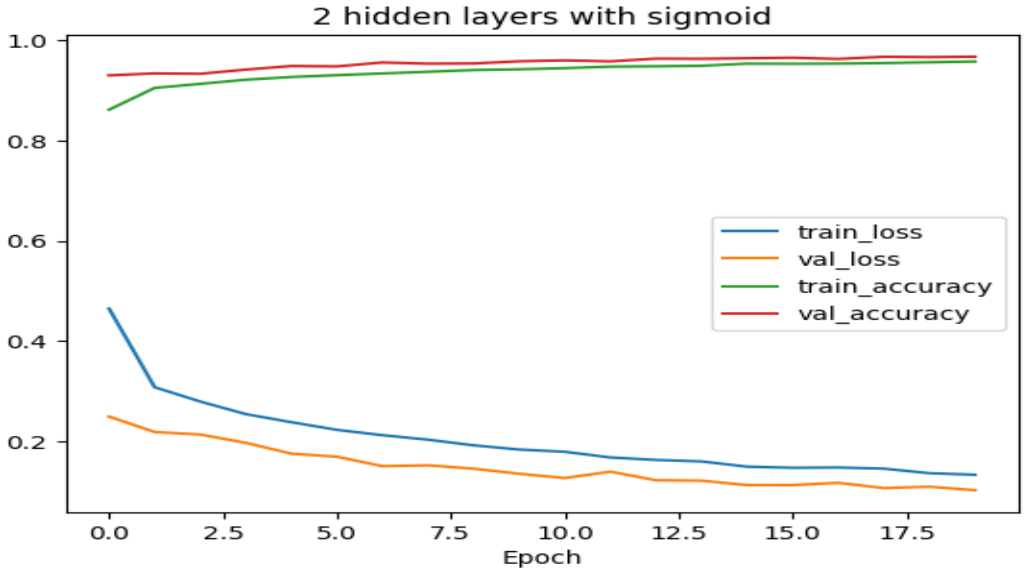
```
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

# 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))

# 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(train_images.reshape(-1, 784), train_labels, validation_data=(test_images.reshape(-1, 784), test_labels),
                        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()
loss, accuracy = model.evaluate(test_images.reshape(-1, 784), test_labels, verbose=0)
print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```

Output:




1 hidden layer with sigmoid
train_loss
val_loss
train_accuracy
val_accuracy
Epoch


2 hidden layers with tanh
train_loss
val_loss
train_accuracy
val_accuracy
Epoch


2 hidden layers with sigmoid
train_loss
val_loss
train_accuracy
val_accuracy
Epoch