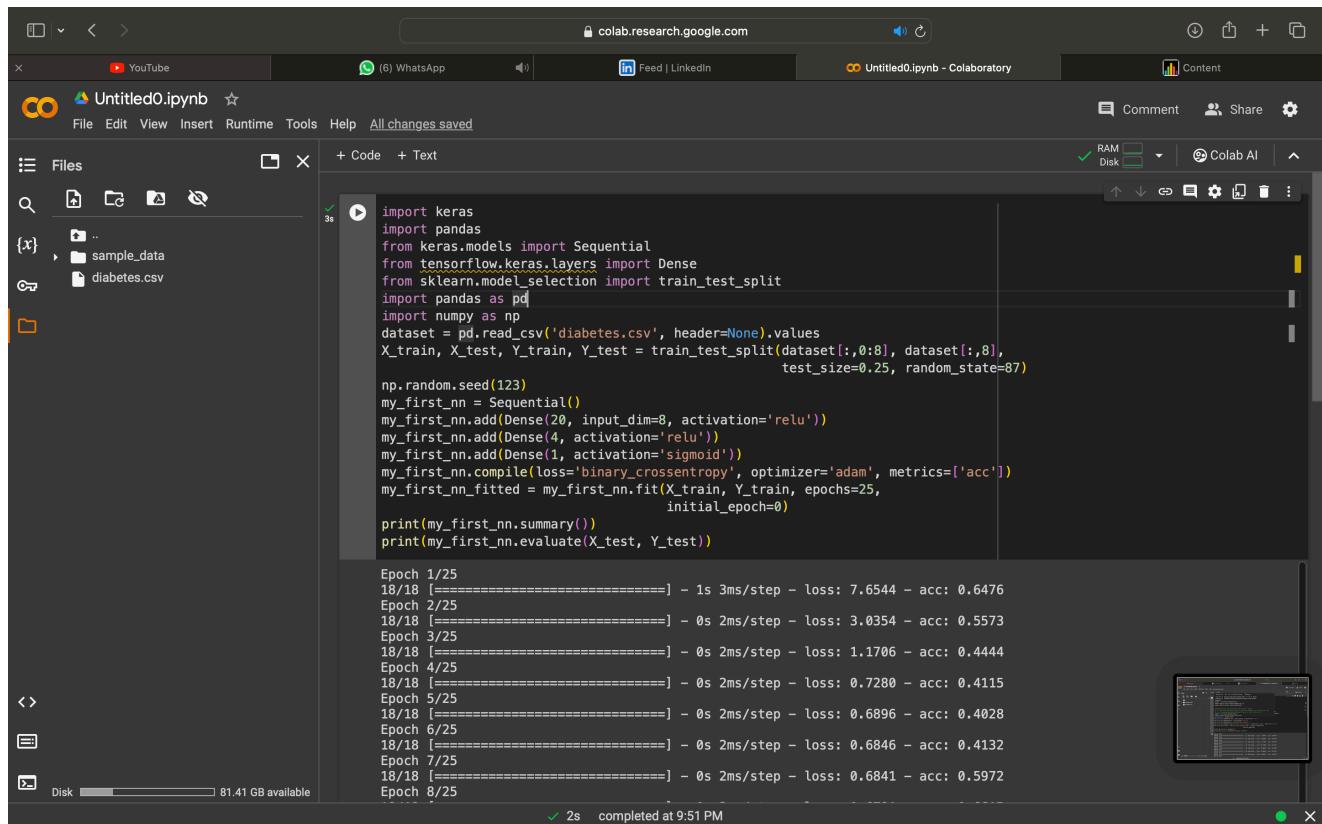


Neural Networks & Deep Learning - ICP-6

Git Hub Url:

<https://github.com/BillaBhavana7/neuralN>

Program Dense:

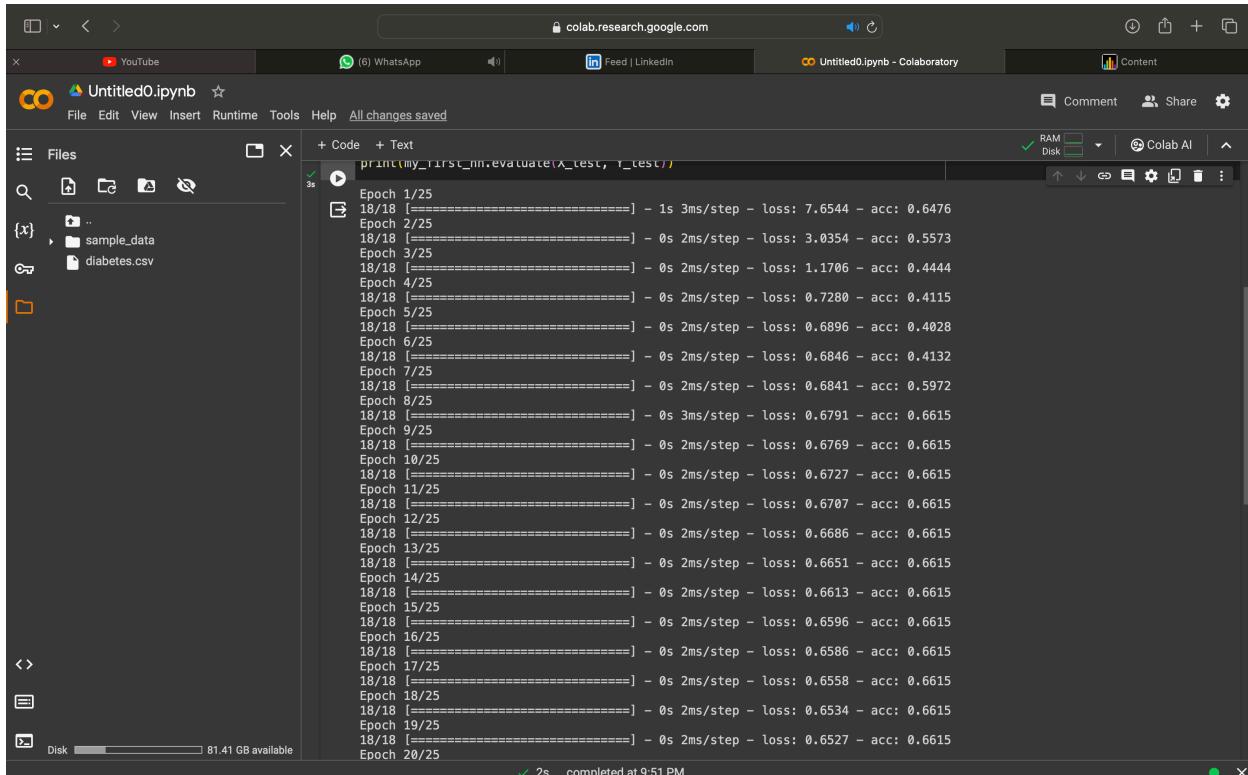


The screenshot shows a Google Colab interface with a Jupyter notebook titled "Untitled0.ipynb". The code cell contains Python code for building a simple neural network using Keras. The output shows the training progress with 8 epochs, indicating a loss of approximately 0.68 and an accuracy of about 0.5972.

```
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))

Epoch 1/25
18/18 [=====] - 1s 3ms/step - loss: 7.6544 - acc: 0.6476
Epoch 2/25
18/18 [=====] - 0s 2ms/step - loss: 3.0354 - acc: 0.5573
Epoch 3/25
18/18 [=====] - 0s 2ms/step - loss: 1.1706 - acc: 0.4444
Epoch 4/25
18/18 [=====] - 0s 2ms/step - loss: 0.7280 - acc: 0.4115
Epoch 5/25
18/18 [=====] - 0s 2ms/step - loss: 0.6896 - acc: 0.4028
Epoch 6/25
18/18 [=====] - 0s 2ms/step - loss: 0.6846 - acc: 0.4132
Epoch 7/25
18/18 [=====] - 0s 2ms/step - loss: 0.6841 - acc: 0.5972
Epoch 8/25
```

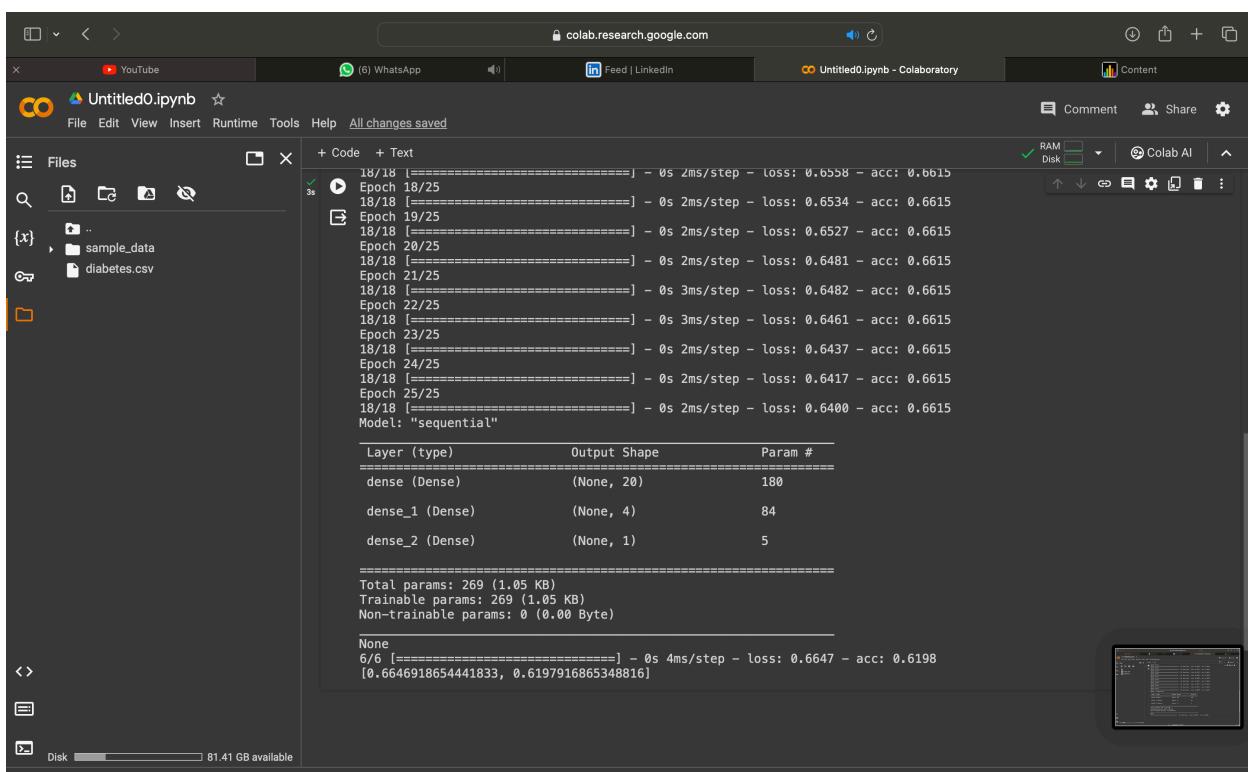
Output:



```
print('my_linear.evaluate(x_test, y_test)')

Epoch 1/25
18/18 [=====] - 1s 3ms/step - loss: 7.6544 - acc: 0.6476
Epoch 2/25
18/18 [=====] - 0s 2ms/step - loss: 3.0354 - acc: 0.5573
Epoch 3/25
18/18 [=====] - 0s 2ms/step - loss: 1.1706 - acc: 0.4444
Epoch 4/25
18/18 [=====] - 0s 2ms/step - loss: 0.7280 - acc: 0.4115
Epoch 5/25
18/18 [=====] - 0s 2ms/step - loss: 0.6896 - acc: 0.4028
Epoch 6/25
18/18 [=====] - 0s 2ms/step - loss: 0.6846 - acc: 0.4132
Epoch 7/25
18/18 [=====] - 0s 2ms/step - loss: 0.6841 - acc: 0.5972
Epoch 8/25
18/18 [=====] - 0s 3ms/step - loss: 0.6791 - acc: 0.6615
Epoch 9/25
18/18 [=====] - 0s 2ms/step - loss: 0.6769 - acc: 0.6615
Epoch 10/25
18/18 [=====] - 0s 2ms/step - loss: 0.6727 - acc: 0.6615
Epoch 11/25
18/18 [=====] - 0s 2ms/step - loss: 0.6707 - acc: 0.6615
Epoch 12/25
18/18 [=====] - 0s 2ms/step - loss: 0.6686 - acc: 0.6615
Epoch 13/25
18/18 [=====] - 0s 2ms/step - loss: 0.6651 - acc: 0.6615
Epoch 14/25
18/18 [=====] - 0s 2ms/step - loss: 0.6613 - acc: 0.6615
Epoch 15/25
18/18 [=====] - 0s 2ms/step - loss: 0.6596 - acc: 0.6615
Epoch 16/25
18/18 [=====] - 0s 2ms/step - loss: 0.6586 - acc: 0.6615
Epoch 17/25
18/18 [=====] - 0s 2ms/step - loss: 0.6558 - acc: 0.6615
Epoch 18/25
18/18 [=====] - 0s 2ms/step - loss: 0.6534 - acc: 0.6615
Epoch 19/25
18/18 [=====] - 0s 2ms/step - loss: 0.6527 - acc: 0.6615
Epoch 20/25
18/18 [=====] - 0s 2ms/step - loss: 0.6481 - acc: 0.6615
Epoch 21/25
18/18 [=====] - 0s 3ms/step - loss: 0.6482 - acc: 0.6615
Epoch 22/25
18/18 [=====] - 0s 3ms/step - loss: 0.6461 - acc: 0.6615
Epoch 23/25
18/18 [=====] - 0s 2ms/step - loss: 0.6437 - acc: 0.6615
Epoch 24/25
18/18 [=====] - 0s 2ms/step - loss: 0.6417 - acc: 0.6615
Epoch 25/25
18/18 [=====] - 0s 2ms/step - loss: 0.6400 - acc: 0.6615
Model: "sequential"
Layer (type)          Output Shape         Param #
dense (Dense)        (None, 20)           100
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)

6/6 [=====] - 0s 4ms/step - loss: 0.6647 - acc: 0.6198
[0.6646918654441833, 0.6197916865348816]
```

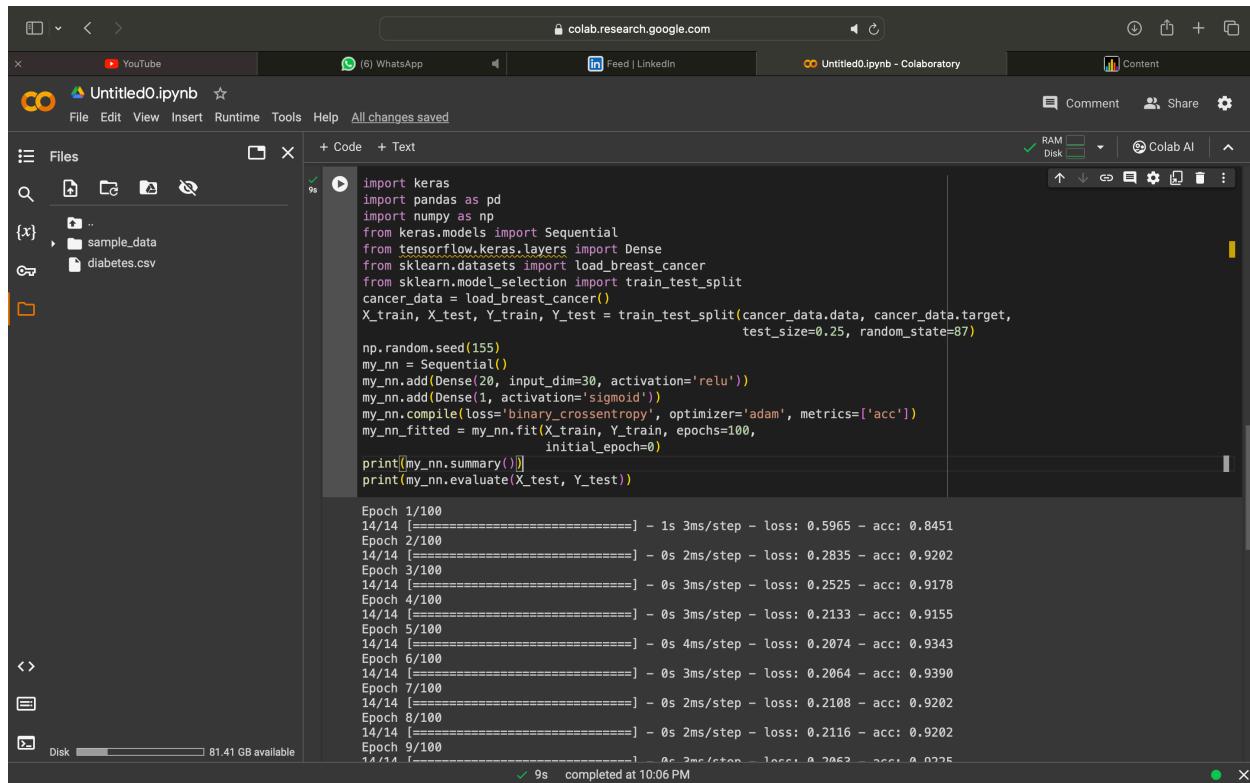


```
print('my_linear.evaluate(x_test, y_test)')

Epoch 18/25
18/18 [=====] - 0s 2ms/step - loss: 0.6558 - acc: 0.6615
Epoch 19/25
18/18 [=====] - 0s 2ms/step - loss: 0.6534 - acc: 0.6615
Epoch 20/25
18/18 [=====] - 0s 2ms/step - loss: 0.6527 - acc: 0.6615
Epoch 21/25
18/18 [=====] - 0s 2ms/step - loss: 0.6481 - acc: 0.6615
Epoch 22/25
18/18 [=====] - 0s 3ms/step - loss: 0.6482 - acc: 0.6615
Epoch 23/25
18/18 [=====] - 0s 3ms/step - loss: 0.6461 - acc: 0.6615
Epoch 24/25
18/18 [=====] - 0s 2ms/step - loss: 0.6437 - acc: 0.6615
Epoch 25/25
18/18 [=====] - 0s 2ms/step - loss: 0.6417 - acc: 0.6615
Model: "sequential"
Layer (type)          Output Shape         Param #
dense (Dense)        (None, 20)           100
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)

6/6 [=====] - 0s 4ms/step - loss: 0.6647 - acc: 0.6198
[0.6646918654441833, 0.6197916865348816]
```

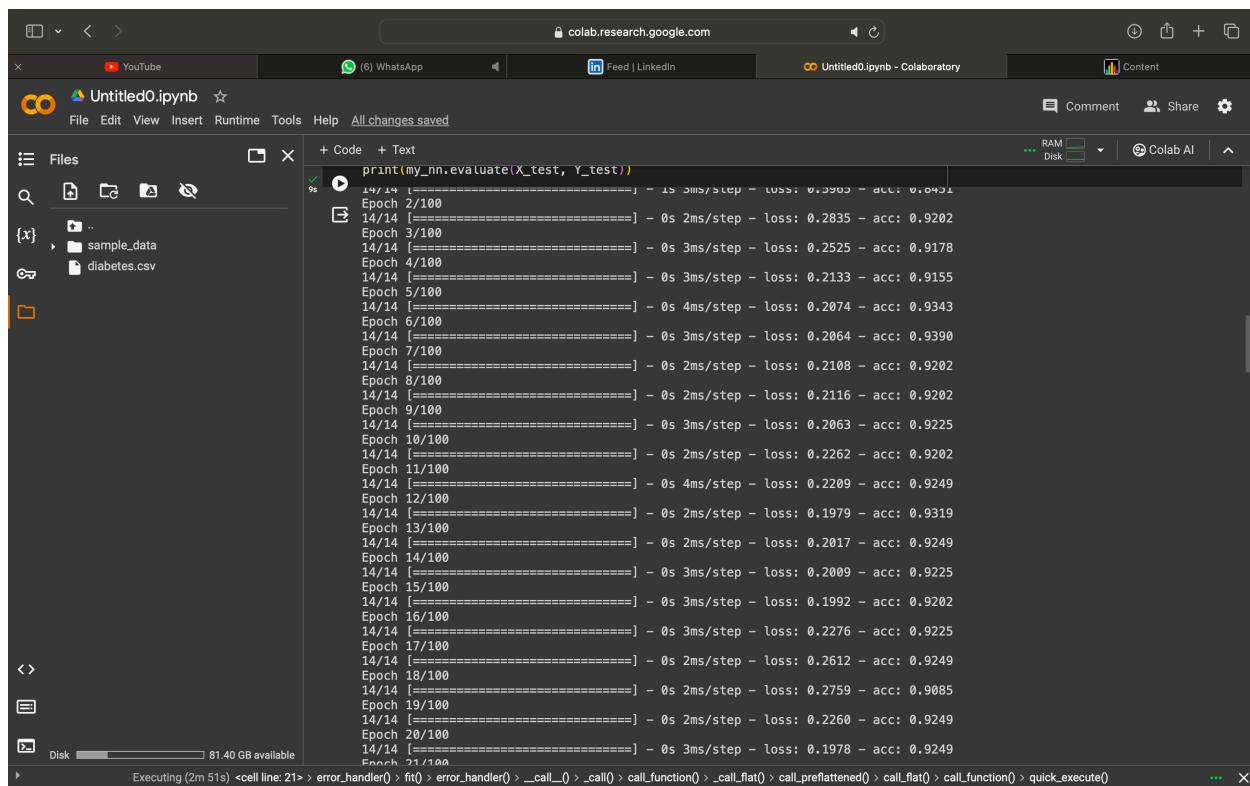
Program breast:



```
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 = 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))

Epoch 1/100
14/14 [=====] - 1s 3ms/step - loss: 0.5965 - acc: 0.8451
Epoch 2/100
14/14 [=====] - 0s 2ms/step - loss: 0.2835 - acc: 0.9202
Epoch 3/100
14/14 [=====] - 0s 3ms/step - loss: 0.2525 - acc: 0.9178
Epoch 4/100
14/14 [=====] - 0s 3ms/step - loss: 0.2133 - acc: 0.9155
Epoch 5/100
14/14 [=====] - 0s 4ms/step - loss: 0.2074 - acc: 0.9343
Epoch 6/100
14/14 [=====] - 0s 3ms/step - loss: 0.2064 - acc: 0.9390
Epoch 7/100
14/14 [=====] - 0s 2ms/step - loss: 0.2108 - acc: 0.9202
Epoch 8/100
14/14 [=====] - 0s 2ms/step - loss: 0.2116 - acc: 0.9202
Epoch 9/100
14/14 [=====] - 0s 3ms/step - loss: 0.2063 - acc: 0.9249
14/14 completed at 10:06 PM
```

Output:



```
print(my_nn.evaluate(X_test, Y_test))

Epoch 2/100
14/14 [=====] - 0s 3ms/step - loss: 0.5965 - acc: 0.8451
Epoch 3/100
14/14 [=====] - 0s 2ms/step - loss: 0.2835 - acc: 0.9202
Epoch 4/100
14/14 [=====] - 0s 3ms/step - loss: 0.2525 - acc: 0.9178
Epoch 5/100
14/14 [=====] - 0s 3ms/step - loss: 0.2133 - acc: 0.9155
Epoch 6/100
14/14 [=====] - 0s 4ms/step - loss: 0.2074 - acc: 0.9343
Epoch 7/100
14/14 [=====] - 0s 3ms/step - loss: 0.2064 - acc: 0.9390
Epoch 8/100
14/14 [=====] - 0s 2ms/step - loss: 0.2108 - acc: 0.9202
Epoch 9/100
14/14 [=====] - 0s 2ms/step - loss: 0.2116 - acc: 0.9202
Epoch 10/100
14/14 [=====] - 0s 3ms/step - loss: 0.2063 - acc: 0.9225
Epoch 11/100
14/14 [=====] - 0s 2ms/step - loss: 0.2262 - acc: 0.9202
Epoch 12/100
14/14 [=====] - 0s 4ms/step - loss: 0.2209 - acc: 0.9249
Epoch 13/100
14/14 [=====] - 0s 2ms/step - loss: 0.1979 - acc: 0.9319
Epoch 14/100
14/14 [=====] - 0s 2ms/step - loss: 0.2017 - acc: 0.9249
Epoch 15/100
14/14 [=====] - 0s 3ms/step - loss: 0.2009 - acc: 0.9225
Epoch 16/100
14/14 [=====] - 0s 3ms/step - loss: 0.1992 - acc: 0.9202
Epoch 17/100
14/14 [=====] - 0s 3ms/step - loss: 0.2276 - acc: 0.9225
Epoch 18/100
14/14 [=====] - 0s 2ms/step - loss: 0.2612 - acc: 0.9249
Epoch 19/100
14/14 [=====] - 0s 2ms/step - loss: 0.2759 - acc: 0.9085
Epoch 20/100
14/14 [=====] - 0s 2ms/step - loss: 0.2260 - acc: 0.9249
14/14 completed at 10:06 PM
```

Executing (2m 51s) <cell line: 21> > error_handler() > fit() > error_handler() > __call__() > call_function() > __call_flatten() > call_prestuffed() > call_flat() > call_function() > quick_execute()

The screenshot shows a Google Colab notebook titled "Untitled0.ipynb". The code cell contains a series of training log entries from epoch 49 to 68. Each entry shows a progress bar followed by the epoch number, a step count, and metrics: loss and accuracy. The logs indicate a learning rate of 0.001 and a batch size of 32. The accuracy generally increases over time, starting around 0.9343 and reaching approximately 0.9319 by epoch 68.

```
14/14 [=====] - 0s 14ms/step - loss: 0.2036 - acc: 0.9343
Epoch 49/100
14/14 [=====] - 0s 7ms/step - loss: 0.1576 - acc: 0.9413
Epoch 50/100
14/14 [=====] - 0s 7ms/step - loss: 0.1705 - acc: 0.9202
Epoch 51/100
14/14 [=====] - 0s 11ms/step - loss: 0.1593 - acc: 0.9296
Epoch 52/100
14/14 [=====] - 0s 8ms/step - loss: 0.1532 - acc: 0.9319
Epoch 53/100
14/14 [=====] - 0s 10ms/step - loss: 0.2163 - acc: 0.9131
Epoch 54/100
14/14 [=====] - 0s 7ms/step - loss: 0.1599 - acc: 0.9319
Epoch 55/100
14/14 [=====] - 0s 5ms/step - loss: 0.1716 - acc: 0.9272
Epoch 56/100
14/14 [=====] - 0s 4ms/step - loss: 0.1788 - acc: 0.9366
Epoch 57/100
14/14 [=====] - 0s 7ms/step - loss: 0.1961 - acc: 0.9202
Epoch 58/100
14/14 [=====] - 0s 6ms/step - loss: 0.1635 - acc: 0.9296
Epoch 59/100
14/14 [=====] - 0s 6ms/step - loss: 0.2259 - acc: 0.9319
Epoch 60/100
14/14 [=====] - 0s 5ms/step - loss: 0.2046 - acc: 0.9272
Epoch 61/100
14/14 [=====] - 0s 7ms/step - loss: 0.1758 - acc: 0.9390
Epoch 62/100
14/14 [=====] - 0s 9ms/step - loss: 0.1643 - acc: 0.9319
Epoch 63/100
14/14 [=====] - 0s 5ms/step - loss: 0.2147 - acc: 0.9155
Epoch 64/100
14/14 [=====] - 0s 8ms/step - loss: 0.1772 - acc: 0.9343
Epoch 65/100
14/14 [=====] - 0s 6ms/step - loss: 0.1483 - acc: 0.9366
Epoch 66/100
14/14 [=====] - 0s 7ms/step - loss: 0.1637 - acc: 0.9319
Epoch 67/100
14/14 [=====] - 0s 6ms/step - loss: 0.2033 - acc: 0.9155
Epoch 68/100
```

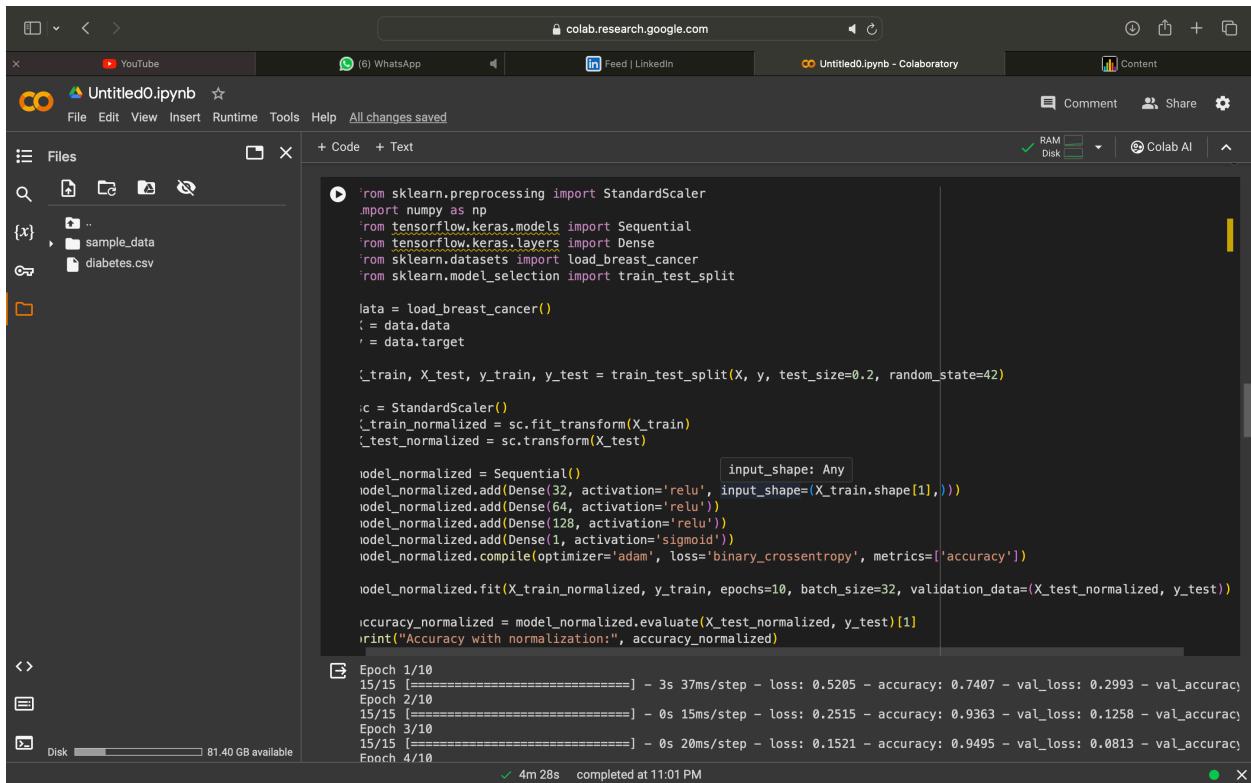
The screenshot shows a Google Colab notebook titled "Untitled0.ipynb". The code cell displays a model summary table and a final execution log. The model is named "sequential_1" and consists of two layers: "dense_3" (20 units) and "dense_4" (1 unit). The total parameters are 641 (2.50 KB), all of which are trainable. The final log entry shows the execution of a cell taking 3m 3s, resulting in a loss of 0.3002 and an accuracy of 0.8811.

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 20)	620
dense_4 (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: 0.3002 - acc: 0.8811
[0.3001939356327057, 0.881118893623352]
```

Program Normal:



```
from sklearn.preprocessing import StandardScaler
import numpy as np
from tensorflow.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

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

sc = StandardScaler()
X_train_normalized = sc.fit_transform(X_train)
X_test_normalized = sc.transform(X_test)

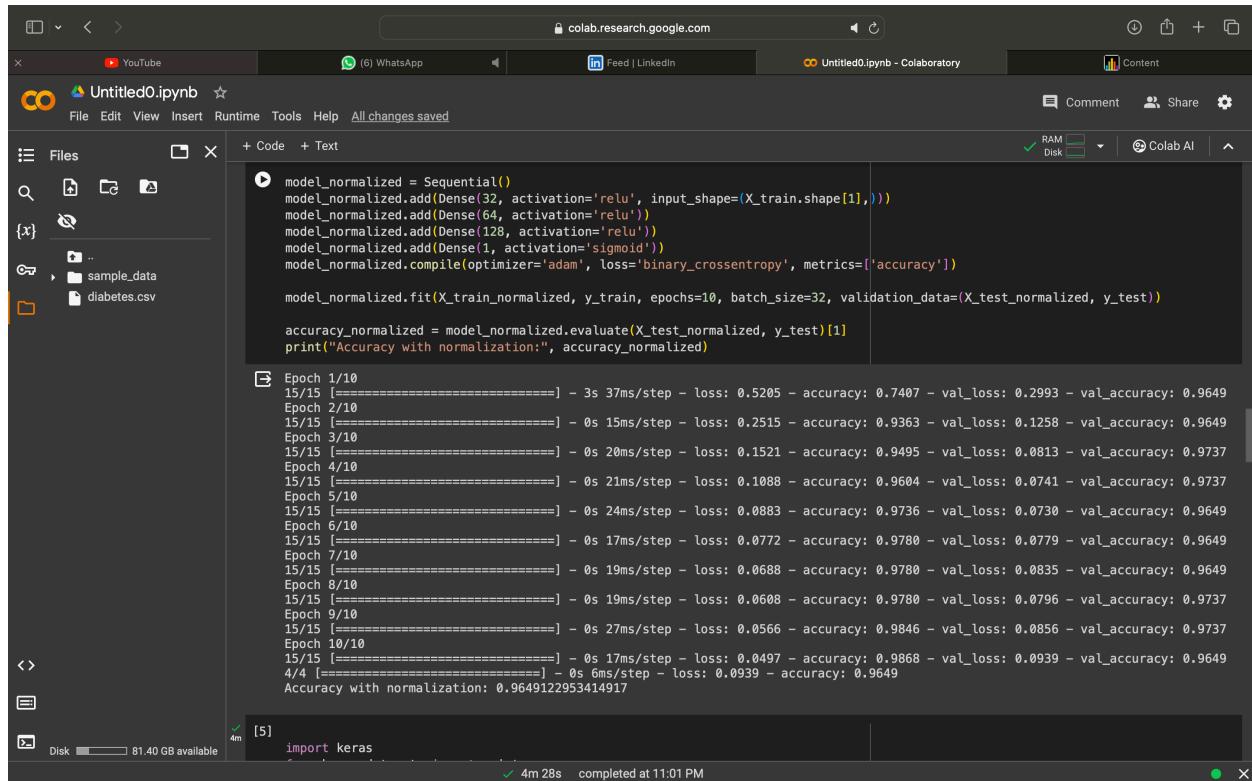
model_normalized = Sequential()
model_normalized.add(Dense(32, activation='relu', input_shape=(X_train.shape[1],)))
model_normalized.add(Dense(64, activation='relu'))
model_normalized.add(Dense(128, activation='relu'))
model_normalized.add(Dense(1, activation='sigmoid'))
model_normalized.compile(optimizer='adam', 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)
```

Epoch 1/10
15/15 [=====] - 3s 37ms/step - loss: 0.5205 - accuracy: 0.7407 - val_loss: 0.2993 - val_accuracy: 0.8140
Epoch 2/10
15/15 [=====] - 0s 15ms/step - loss: 0.2515 - accuracy: 0.9363 - val_loss: 0.1258 - val_accuracy: 0.9495
Epoch 3/10
15/15 [=====] - 0s 20ms/step - loss: 0.1521 - accuracy: 0.9495 - val_loss: 0.0813 - val_accuracy: 0.9545
Epoch 4/10
15/15 [=====] - 0s 15ms/step - loss: 0.1521 - accuracy: 0.9495 - val_loss: 0.0813 - val_accuracy: 0.9545

Output:



The screenshot shows a Google Colab interface with a Jupyter notebook titled "Untitled0.ipynb". The code cell contains Python code for building a neural network and training it on a dataset. The output cell displays the training logs for 10 epochs and the final accuracy.

```
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)
```

```
Epoch 1/10
15/15 [=====] - 3s 37ms/step - loss: 0.5205 - accuracy: 0.7407 - val_loss: 0.2993 - val_accuracy: 0.9649
Epoch 2/10
15/15 [=====] - 0s 15ms/step - loss: 0.2515 - accuracy: 0.9363 - val_loss: 0.1258 - val_accuracy: 0.9649
Epoch 3/10
15/15 [=====] - 0s 20ms/step - loss: 0.1521 - accuracy: 0.9495 - val_loss: 0.0813 - val_accuracy: 0.9737
Epoch 4/10
15/15 [=====] - 0s 21ms/step - loss: 0.1088 - accuracy: 0.9604 - val_loss: 0.0741 - val_accuracy: 0.9737
Epoch 5/10
15/15 [=====] - 0s 24ms/step - loss: 0.0883 - accuracy: 0.9736 - val_loss: 0.0730 - val_accuracy: 0.9649
Epoch 6/10
15/15 [=====] - 0s 17ms/step - loss: 0.0772 - accuracy: 0.9780 - val_loss: 0.0779 - val_accuracy: 0.9649
Epoch 7/10
15/15 [=====] - 0s 19ms/step - loss: 0.0688 - accuracy: 0.9780 - val_loss: 0.0835 - val_accuracy: 0.9649
Epoch 8/10
15/15 [=====] - 0s 19ms/step - loss: 0.0608 - accuracy: 0.9780 - val_loss: 0.0796 - val_accuracy: 0.9737
Epoch 9/10
15/15 [=====] - 0s 27ms/step - loss: 0.0566 - accuracy: 0.9846 - val_loss: 0.0856 - val_accuracy: 0.9737
Epoch 10/10
15/15 [=====] - 0s 17ms/step - loss: 0.0497 - accuracy: 0.9868 - val_loss: 0.0939 - val_accuracy: 0.9649
4/4 [=====] - 0s 6ms/step - loss: 0.0939 - accuracy: 0.9649
Accuracy with normalization: 0.96491229553414917
```

[5] import keras

4m 28s completed at 11:01 PM

Program Image 1:

The screenshot shows a Google Colab interface with a Jupyter notebook titled "Untitled0.ipynb". The code cell contains Python code for a neural network to classify digits from the MNIST dataset. The code includes data loading, model definition (Sequential), compilation (categorical_crossentropy, adam, accuracy), training (fit), and plotting (accuracy vs epoch). The output shows the plot and the training progress.

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

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

train_images = train_images.astype('float32') / 255
test_images = test_images.astype('float32') / 255

num_classes = 10
train_labels = keras.utils.to_categorical(train_labels, num_classes)
test_labels = keras.utils.to_categorical(test_labels, 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(train_images.reshape(-1, 784), train_labels, validation_data=(test_images.reshape(-1, 784), test_labels),
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')

4m 28s completed at 11:01 PM
```

This screenshot shows the same Google Colab interface as the previous one, but with additional code added to plot the loss history. The new code creates a second subplot showing 'Loss' and 'val_loss' over 4 epochs. The output shows both the accuracy and loss plots.

```
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(train_images.reshape(-1, 784), train_labels, validation_data=(test_images.reshape(-1, 784), test_labels),
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')

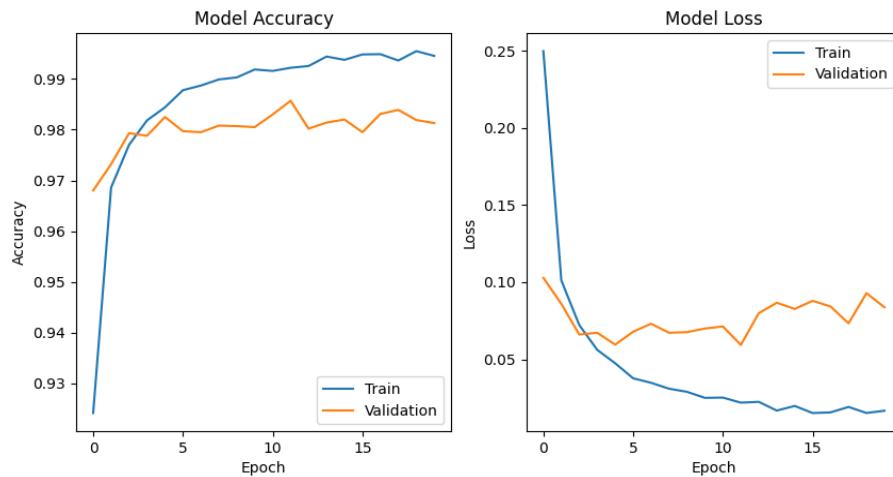
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()

11490434/11490434 [=====] - 0s 0us/step
Epoch 1/20
469/469 [=====] - 14s 25ms/step - loss: 0.2467 - accuracy: 0.9255 - val_loss: 0.1017 - va
Epoch 2/20
469/469 [=====] - 11s 23ms/step - loss: 0.1022 - accuracy: 0.9683 - val_loss: 0.0705 - va
Epoch 3/20
469/469 [=====] - 13s 28ms/step - loss: 0.0709 - accuracy: 0.9773 - val_loss: 0.0736 - val_ accuracy: 0.9772
Epoch 4/20
469/469 [=====] - 13s 28ms/step - loss: 0.0709 - accuracy: 0.9773 - val_loss: 0.0736 - val_ accuracy: 0.9772

4m 28s completed at 11:01 PM
```

Output:



Program Image 2:

```
[6] 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

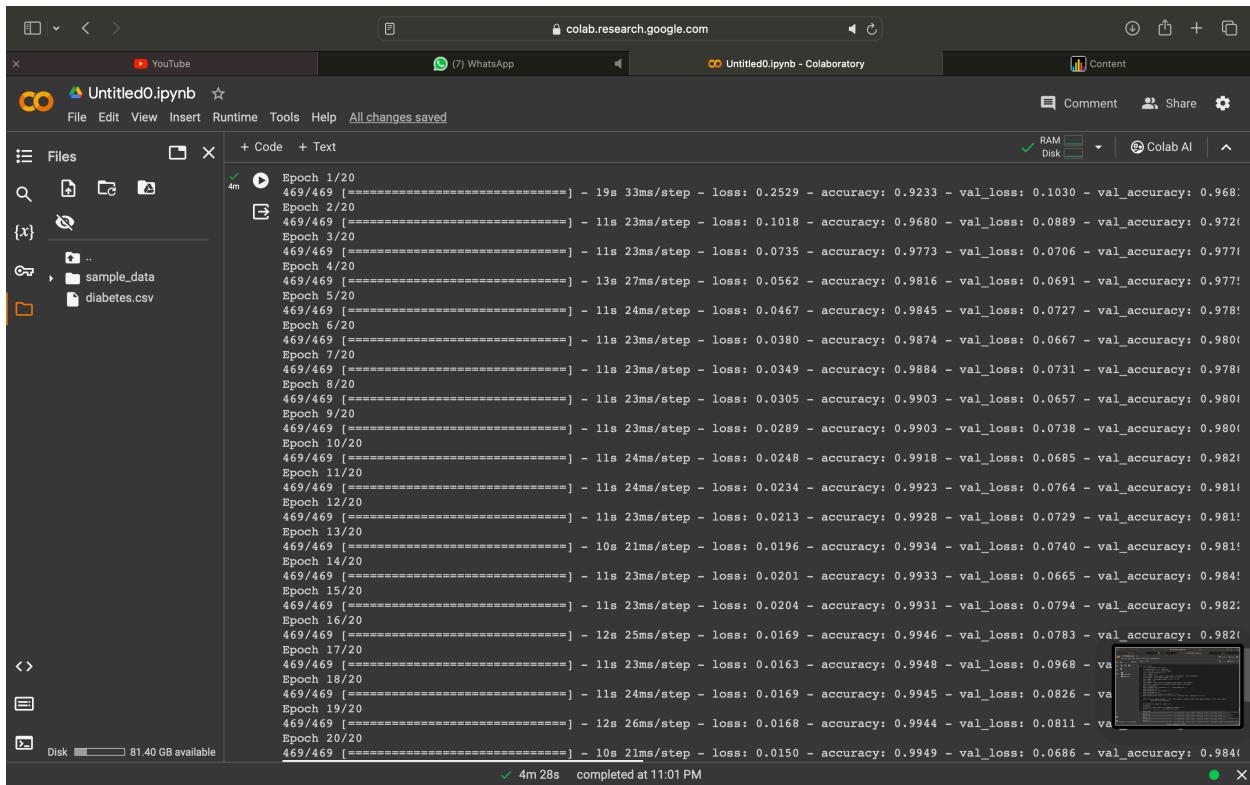
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.astype('float32') / 255
test_images = test_images.astype('float32') / 255
num_classes = 10
train_labels = keras.utils.to_categorical(train_labels, num_classes)
test_labels = keras.utils.to_categorical(test_labels, 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(train_images.reshape(-1, 784), train_labels, validation_data=(test_images.reshape(-1, 784), test_labels),
          epochs=20, batch_size=128)

plt.imshow(test_images[0], cmap='gray')
plt.show()
prediction = model.predict(test_images[0].reshape(1, -1))
print('Model prediction:', np.argmax(prediction))

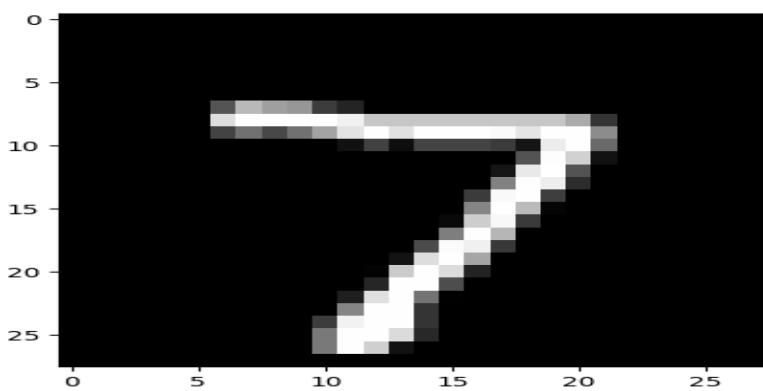
Epoch 5/20
469/469 [=====] - 1s 24ms/step - loss: 0.0467 - accuracy: 0.9845 - val_loss: 0.0727 - va
Epoch 6/20
469/469 [=====] - 1s 23ms/step - loss: 0.0380 - accuracy: 0.9874 - val_loss: 0.0667 - va
Epoch 7/20
469/469 [=====] - 1s 23ms/step - loss: 0.0349 - accuracy: 0.9884 - val_loss: 0.0731 - va
Epoch 8/20
469/469 [=====] - 1s 23ms/step - loss: 0.0305 - accuracy: 0.9903 - val_loss: 0.0657 - val_accuracy: 0.9801
```

Output:



The screenshot shows a Google Colab interface with the following details:

- Title:** Untitled0.ipynb
- Runtime:** colab.research.google.com
- File List:** Files, sample_data, diabetes.csv
- Logs:** Training logs for 469 epochs, showing metrics like loss, accuracy, and validation loss.
- Disk Usage:** Disk 81.40 GB available.
- Completion:** 4m 28s completed at 11:01 PM.



Program Image 3:

The screenshot shows a Google Colab notebook titled "Untitled0.ipynb". The code in the cell is as follows:

```
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

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

train_images = train_images.astype('float32') / 255
test_images = test_images.astype('float32') / 255

num_classes = 10
train_labels = keras.utils.to_categorical(train_labels, num_classes)
test_labels = keras.utils.to_categorical(test_labels, 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)
```

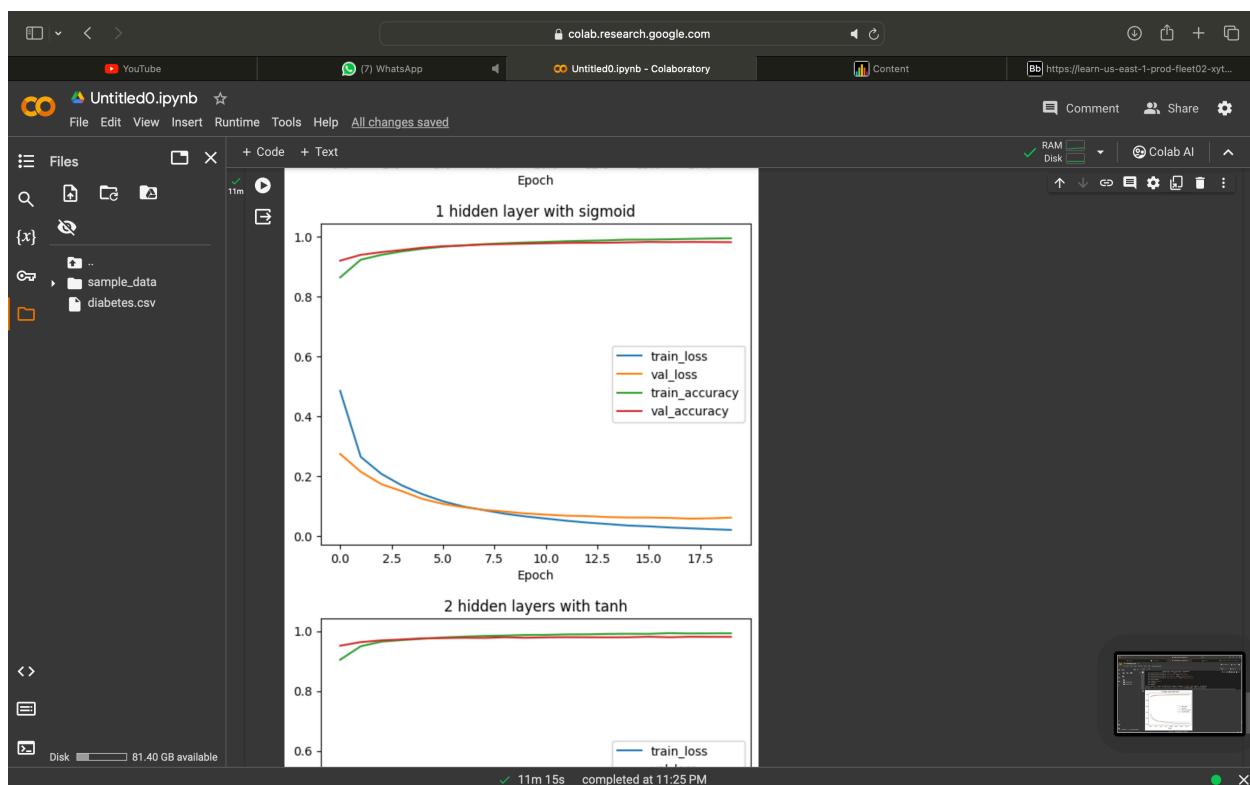
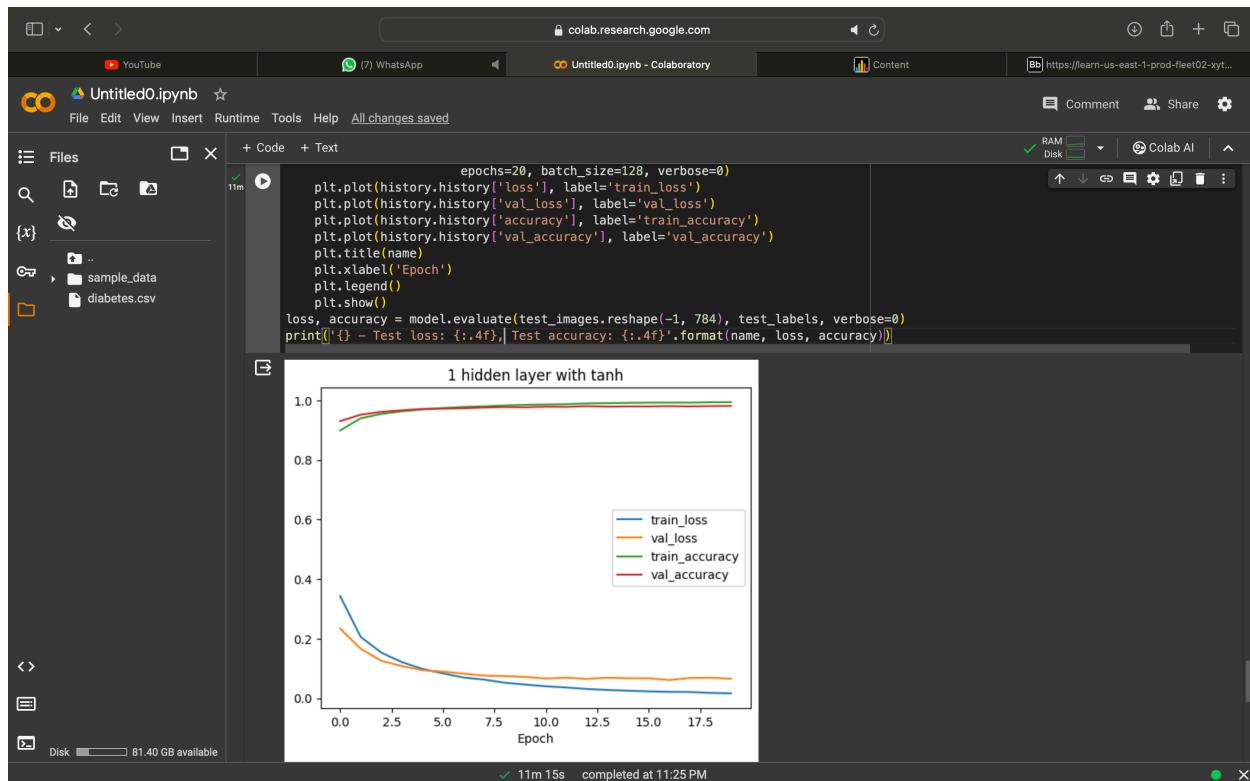
The cell has been run and completed at 11:25 PM.

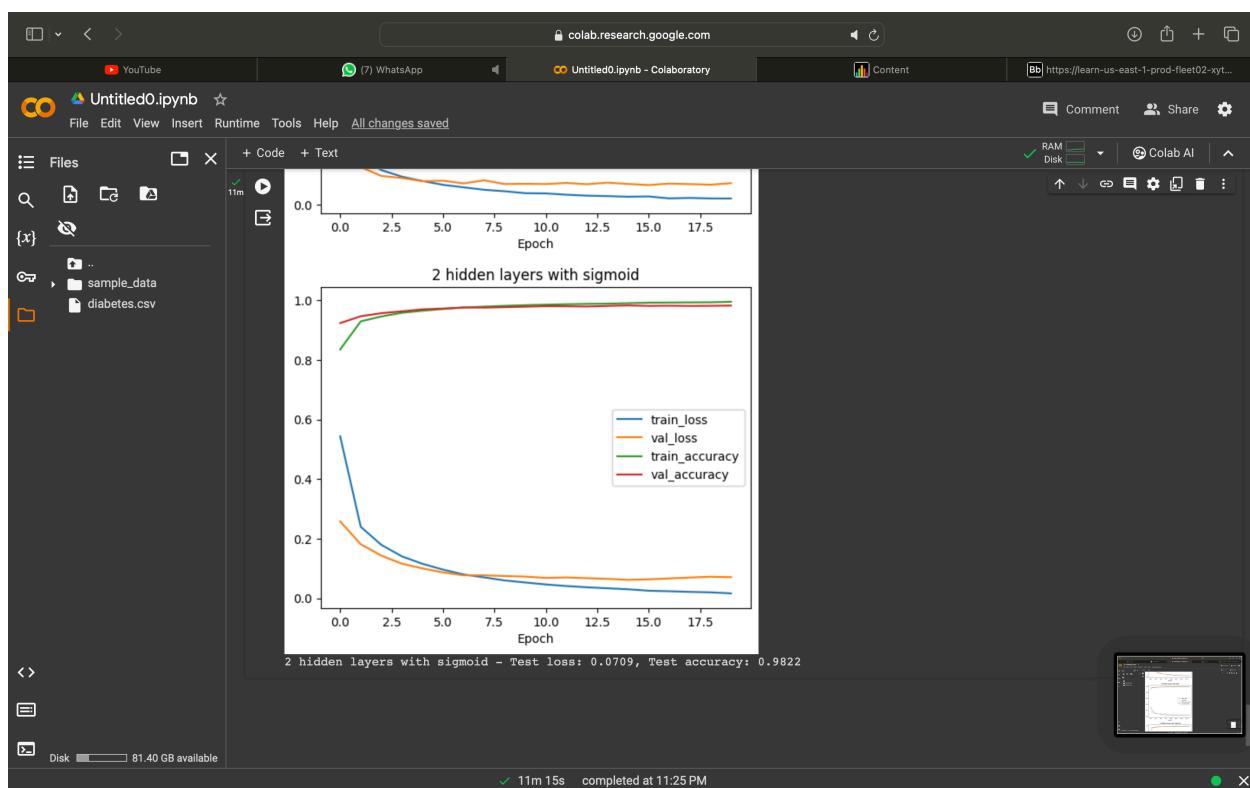
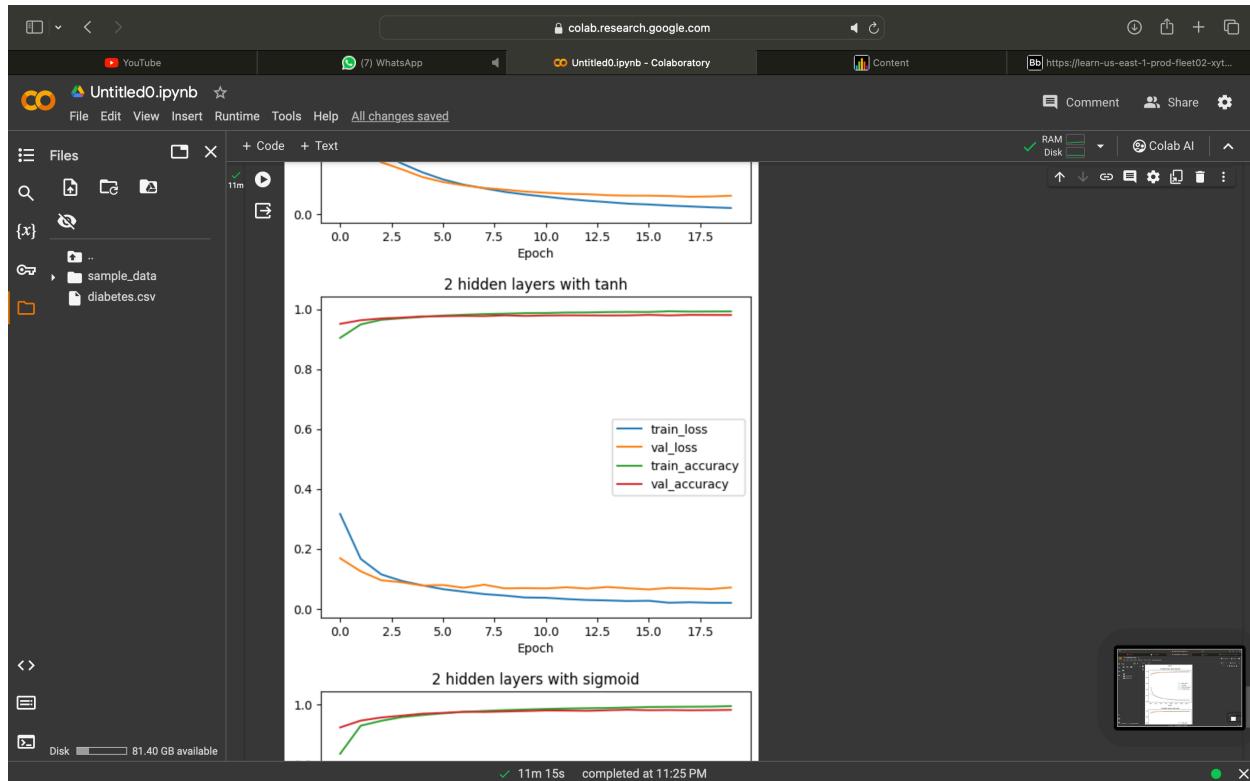
The screenshot shows the continuation of the Google Colab notebook "Untitled0.ipynb". The code in the cell is as follows:

```
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)
    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()

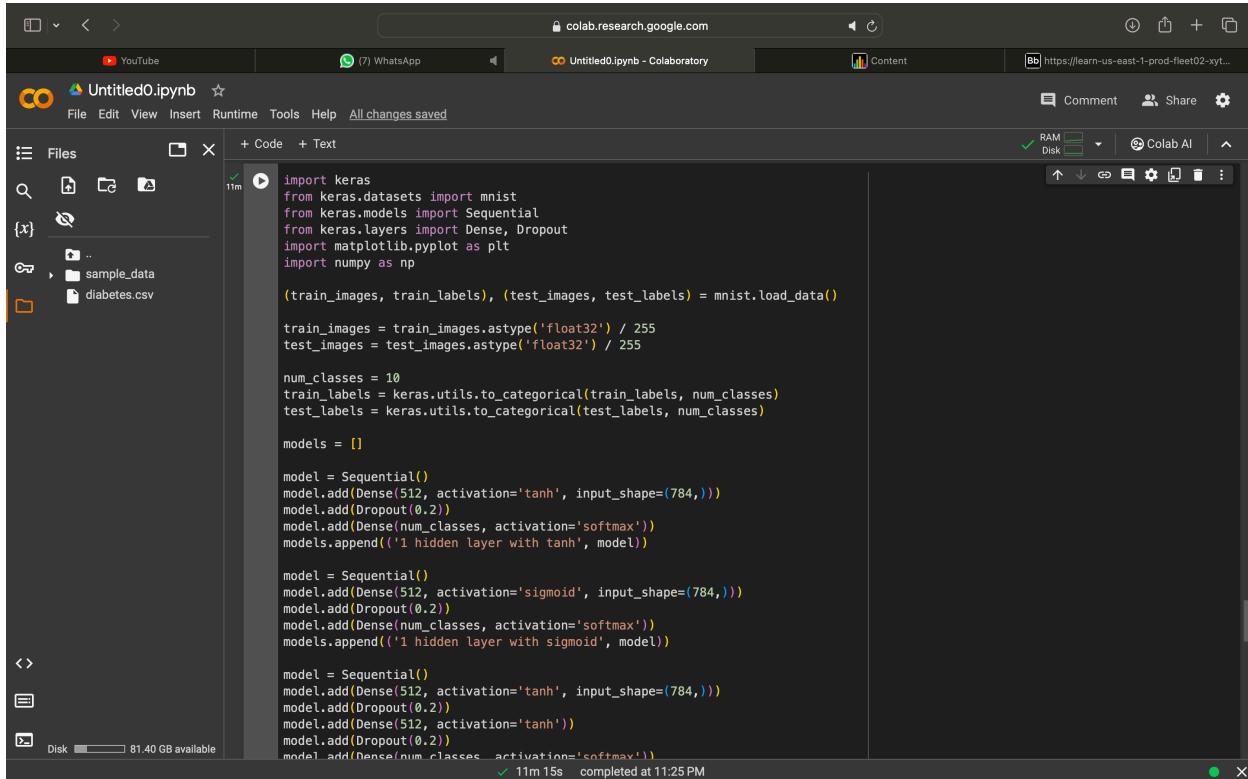
loss, accuracy = model.evaluate(test_images.reshape(-1, 784), test_labels, verbose=0)
print(f'{name} - Test loss: {loss:.4f}, Test accuracy: {accuracy:.4f}')
```

The cell has been run and completed at 11:25 PM.





Program Image 4:



A screenshot of the Google Colab interface. The left sidebar shows files: 'sample_data' and 'diabetes.csv'. The main area contains Python code for building a neural network. The code imports Keras, loads MNIST data, and defines three models. The first model has one hidden layer with tanh activation. The second has one hidden layer with sigmoid activation. The third has two hidden layers, each with tanh activation. All models include a softmax layer for 10 classes. The code is running, indicated by a green progress bar at the bottom.

```
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

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

train_images = train_images.astype('float32') / 255
test_images = test_images.astype('float32') / 255

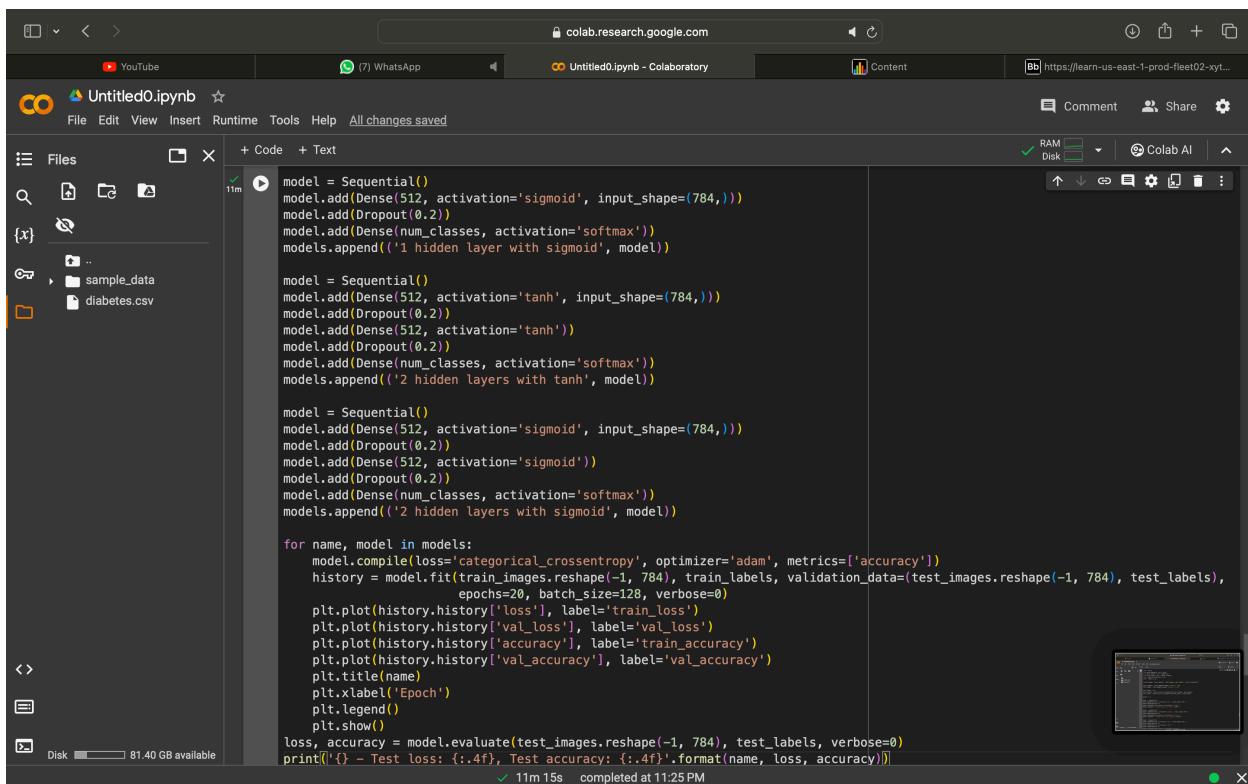
num_classes = 10
train_labels = keras.utils.to_categorical(train_labels, num_classes)
test_labels = keras.utils.to_categorical(test_labels, num_classes)

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])
```



A screenshot of the Google Colab interface showing the completed execution of the code from the previous image. The right sidebar shows a small thumbnail of the notebook. The status bar at the bottom indicates the execution took 11m 15s and was completed at 11:25 PM.

```
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])

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)
    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(f'{name} - Test loss: {loss:.4f}, Test accuracy: {accuracy:.4f}')
```

Output:

