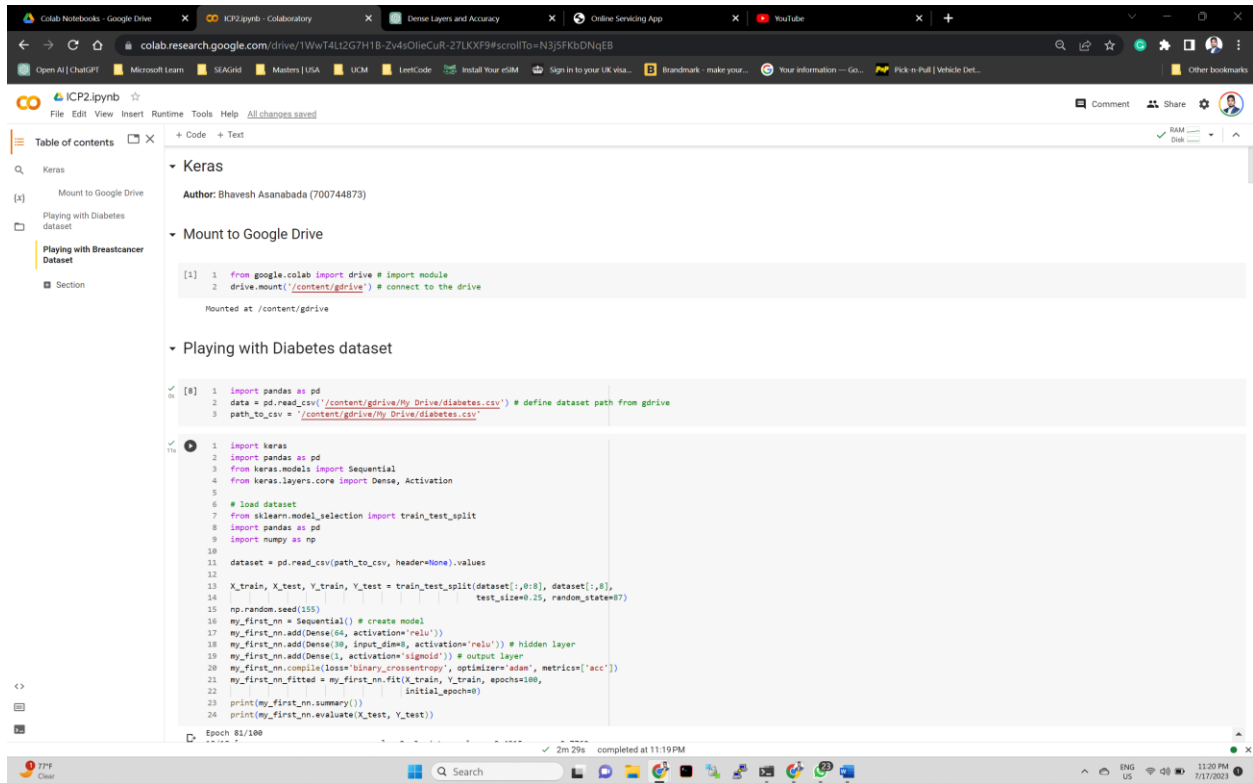


# ICP-2

Name: Bhavesh Asanabada

ID: 700744873

Code: <https://github.com/bhavesh-asana/Neural-Network/tree/main/Assignment-2>



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'Colab Notebooks', 'Google Drive', 'ICP2.ipynb - Colaboratory', 'Dense Layers and Accuracy', 'Online Servicing App', and 'YouTube'. The address bar shows the Colab URL. The notebook has a table of contents on the left with sections: 'Keras', 'Mount to Google Drive', 'Playing with Diabetes dataset', and 'Playing with Breastcancer Dataset'. The main code area is divided into three sections:

- Keras**: Author: Bhavesh Asanabada (700744873)
- Mount to Google Drive**:

```
[1] 1 from google.colab import drive # import module
    2 drive.mount('/content/gdrive') # connect to the drive
```

Output: Mounted at /content/gdrive
- Playing with Diabetes dataset**:

```
[8] 1 import pandas as pd
    2 data = pd.read_csv('/content/gdrive/My Drive/diabetes.csv') # define dataset path from gdrive
    3 path_to_csv = '/content/gdrive/My Drive/diabetes.csv'
```

```
11 1 import keras
    2 import pandas as pd
    3 from keras.models import Sequential
    4 from keras.layers.core import Dense, Activation
    5
    6 # Load dataset
    7 from sklearn.model_selection import train_test_split
    8 import pandas as pd
    9 import numpy as np
   10
   11 dataset = pd.read_csv(path_to_csv, header=None).values
   12
   13 X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
   14                                                    test_size=0.25, random_state=0)
   15 np.random.seed(155)
   16 my_first_nn = Sequential() # create model
   17 my_first_nn.add(Dense(40, activation='relu'))
   18 my_first_nn.add(Dense(30, input_dim=40, activation='relu')) # hidden layer
   19 my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
   20 my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
   21 my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
   22                                     initial_epoch=0)
   23 print(my_first_nn.summary())
   24 print(my_first_nn.evaluate(X_test, Y_test))
```

Output: Epoch 82/100, 2m 29s, completed at 11:10 PM

Colab Notebook interface showing a Jupyter Notebook with a table of contents and a code cell. The code cell contains a Keras model definition and training results.

Table of contents:

- Mount to Google Drive
- Playing with Diabetes dataset
- Playing with Breastcancer Dataset
- Section

Code cell output:

```
Epoch 83/100
18/18 [=====] - 0s 3ms/step - loss: 0.5820 - acc: 0.7535
Epoch 84/100
18/18 [=====] - 0s 3ms/step - loss: 0.4799 - acc: 0.7743
Epoch 85/100
18/18 [=====] - 0s 3ms/step - loss: 0.4713 - acc: 0.7743
Epoch 86/100
18/18 [=====] - 0s 3ms/step - loss: 0.4685 - acc: 0.7830
Epoch 87/100
18/18 [=====] - 0s 3ms/step - loss: 0.4840 - acc: 0.7795
Epoch 88/100
18/18 [=====] - 0s 3ms/step - loss: 0.5246 - acc: 0.7361
Epoch 89/100
18/18 [=====] - 0s 3ms/step - loss: 0.5885 - acc: 0.7274
Epoch 90/100
18/18 [=====] - 0s 3ms/step - loss: 0.5273 - acc: 0.7292
Epoch 91/100
18/18 [=====] - 0s 3ms/step - loss: 0.4864 - acc: 0.7788
Epoch 92/100
18/18 [=====] - 0s 4ms/step - loss: 0.4739 - acc: 0.7812
Epoch 93/100
18/18 [=====] - 0s 3ms/step - loss: 0.5719 - acc: 0.7326
Epoch 94/100
18/18 [=====] - 0s 3ms/step - loss: 0.5255 - acc: 0.7760
Epoch 95/100
18/18 [=====] - 0s 3ms/step - loss: 0.4910 - acc: 0.7517
Epoch 96/100
18/18 [=====] - 0s 3ms/step - loss: 0.4883 - acc: 0.7674
Epoch 97/100
18/18 [=====] - 0s 3ms/step - loss: 0.5187 - acc: 0.7517
Epoch 98/100
18/18 [=====] - 0s 3ms/step - loss: 0.5812 - acc: 0.7656
Epoch 99/100
18/18 [=====] - 0s 3ms/step - loss: 0.5250 - acc: 0.7378
Epoch 100/100
18/18 [=====] - 0s 3ms/step - loss: 0.5750 - acc: 0.7274
Model: "sequential_3"
Layer (type) Output Shape Param #
-----
dense_5 (Dense) (32, 64) 576
dense_6 (Dense) (32, 30) 1950
dense_7 (Dense) (32, 1) 31
-----
Total params: 2,557
Trainable params: 2,557
Non-trainable params: 0
None
6/6 [=====] - 0s 3ms/step - loss: 0.7418 - acc: 0.6486
[0.7418193817138672, 0.648625]
```

Colab Notebook interface showing a Jupyter Notebook with a table of contents and a code cell. The code cell contains a Keras model definition and training results.

Table of contents:

- Mount to Google Drive
- Playing with Diabetes dataset
- Playing with Breastcancer Dataset
- Section

Code cell output:

```
6/6 [=====] - 0s 3ms/step - loss: 0.7418 - acc: 0.6486
[0.7418193817138672, 0.648625]

1 data = pd.read_csv('/content/gdrive/My Drive/breastcancer.csv')
2 path_to_csv = '/content/gdrive/My Drive/diabetes.csv'

1 import keras
2 import pandas as pd
3 import numpy as np
4 from keras.models import Sequential
5 from keras.layers.core import Dense, Activation
6 from sklearn.datasets import load_breast_cancer
7 from sklearn.model_selection import train_test_split
8
9 # load dataset
10 cancer_data = load_breast_cancer()
11 X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
12                                                    test_size=0.25, random_state=0)
13 np.random.seed(155)
14 my_nn = Sequential() # create model
15 my_nn.add(Dense(20, input_dim=10, activation='relu')) # hidden layer 1
16 my_nn.add(Dense(1, activation='sigmoid')) # output layer
17 my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
18 my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
19                          initial_epoch=0)
20 print(my_nn.summary())
21 print(my_nn.evaluate(X_test, Y_test))

14/14 [=====] - 0s 6ms/step - loss: 0.2154 - acc: 0.9296
Epoch 51/100
14/14 [=====] - 0s 6ms/step - loss: 0.2567 - acc: 0.9085
Epoch 52/100
14/14 [=====] - 0s 7ms/step - loss: 0.2629 - acc: 0.9188
Epoch 53/100
14/14 [=====] - 0s 6ms/step - loss: 0.3216 - acc: 0.8756
Epoch 54/100
14/14 [=====] - 0s 8ms/step - loss: 0.3485 - acc: 0.9225
Epoch 55/100
14/14 [=====] - 0s 7ms/step - loss: 0.2262 - acc: 0.9085
Epoch 56/100
14/14 [=====] - 0s 7ms/step - loss: 0.2348 - acc: 0.9085
Epoch 57/100
14/14 [=====] - 0s 10ms/step - loss: 0.2204 - acc: 0.9249
Epoch 58/100
14/14 [=====] - 0s 12ms/step - loss: 0.2121 - acc: 0.9249
Epoch 59/100
14/14 [=====] - 0s 12ms/step - loss: 0.1994 - acc: 0.9249
Epoch 60/100
14/14 [=====] - 0s 10ms/step - loss: 0.2188 - acc: 0.9272
Epoch 61/100
14/14 [=====] - 0s 13ms/step - loss: 0.2057 - acc: 0.9272
Epoch 62/100
```

Colab Notebook interface showing a Keras model training process. The notebook is titled "Dense Layers and Accuracy". The code cell shows the training of a model on the Breast Cancer dataset. The output displays the training progress, including epochs, loss, and accuracy. The model architecture is summarized as follows:

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 20)	620
dense_4 (Dense)	(None, 1)	21

Total params: 641  
Trainable params: 641  
Non-trainable params: 0

The training progress shows the model converging over 100 epochs, with a final accuracy of approximately 0.8462.

Colab Notebook interface showing a Keras model training process. The notebook is titled "Dense Layers and Accuracy". The code cell shows the training of a model on the Breast Cancer dataset. The output displays the training progress, including epochs, loss, and accuracy. The model architecture is summarized as follows:

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 20)	620
dense_4 (Dense)	(None, 1)	21

Total params: 641  
Trainable params: 641  
Non-trainable params: 0

The training progress shows the model converging over 100 epochs, with a final accuracy of approximately 0.8462.

Colab Notebook interface showing a Jupyter Notebook with a table of contents and code cells. The notebook is titled "Dense Layers and Accuracy". The table of contents includes sections for "Keras", "Mount to Google Drive", "Playing with Diabetes dataset", and "Playing with Breastcancer Dataset". The code cells show the following output:

```
Epoch 88/100
14/14 [=====] - 0s 3ms/step - loss: 0.2921 - acc: 0.9319
Epoch 89/100
14/14 [=====] - 0s 4ms/step - loss: 0.1640 - acc: 0.9437
Epoch 90/100
14/14 [=====] - 0s 3ms/step - loss: 0.1683 - acc: 0.9587
Epoch 91/100
14/14 [=====] - 0s 3ms/step - loss: 0.1696 - acc: 0.9398
Epoch 92/100
14/14 [=====] - 0s 3ms/step - loss: 0.1589 - acc: 0.9413
Epoch 93/100
14/14 [=====] - 0s 3ms/step - loss: 0.1446 - acc: 0.9531
Epoch 94/100
14/14 [=====] - 0s 4ms/step - loss: 0.1684 - acc: 0.9468
Epoch 95/100
14/14 [=====] - 0s 3ms/step - loss: 0.1667 - acc: 0.9437
Epoch 96/100
14/14 [=====] - 0s 3ms/step - loss: 0.1354 - acc: 0.9437
Epoch 97/100
14/14 [=====] - 0s 4ms/step - loss: 0.1671 - acc: 0.9366
Epoch 98/100
14/14 [=====] - 0s 4ms/step - loss: 0.1341 - acc: 0.9587
Epoch 99/100
14/14 [=====] - 0s 3ms/step - loss: 0.1516 - acc: 0.9437
Epoch 100/100
14/14 [=====] - 0s 4ms/step - loss: 0.1542 - acc: 0.9413
Model: "sequential_3"
Layer (type) Output Shape Param #
-----
dense_8 (Dense) (None, 20) 620
dense_9 (Dense) (None, 1) 21
-----
Total params: 641
Trainable params: 641
Non-trainable params: 0
None
5/5 [=====] - 0s 5ms/step - loss: 0.3253 - acc: 0.9381
[0.3253431822167286, 0.9380699234088789]
```

The code cells show the following code:

```
[14]: 1 from sklearn.preprocessing import StandardScaler
      2 sc = StandardScaler()

[15]: 1 import keras
      2 import pandas as pd
```

Colab Notebook interface showing a Jupyter Notebook with a table of contents and code cells. The notebook is titled "Dense Layers and Accuracy". The table of contents includes sections for "Keras", "Mount to Google Drive", "Playing with Diabetes dataset", and "Playing with Breastcancer Dataset". The code cells show the following output:

```
[0.3253431822167286, 0.9380699234088789]

[ ] 1

[ ] 1

[14]: 1 from sklearn.preprocessing import StandardScaler
      2 sc = StandardScaler()

[15]: 1 import keras
      2 import pandas as pd
      3 import numpy as np
      4 from keras.models import Sequential
      5 from keras.layers.core import Dense, Activation
      6 from sklearn.datasets import load_breast_cancer
      7 from sklearn.model_selection import train_test_split
      8
      9 # load dataset
     10 cancer_data = load_breast_cancer()
     11 X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
     12                                                    test_size=0.25, random_state=0)
     13 np.random.seed(155)
     14 my_nn = Sequential() # create model
     15 my_nn.add(Dense(20, input_dim=10, activation='relu')) # hidden layer 1
     16 my_nn.add(Dense(1, activation='sigmoid')) # output layer
     17 my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
     18 my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
     19                          initial_epoch=0)
     20 print(my_nn.summary())
     21 print(my_nn.evaluate(X_test, Y_test))

Epoch 1/100
14/14 [=====] - 1s 3ms/step - loss: 5.8897 - acc: 0.4977
Epoch 2/100
14/14 [=====] - 0s 3ms/step - loss: 4.5850 - acc: 0.5618
Epoch 3/100
14/14 [=====] - 0s 4ms/step - loss: 3.5676 - acc: 0.6488
Epoch 4/100
14/14 [=====] - 0s 4ms/step - loss: 2.5789 - acc: 0.7019
Epoch 5/100
14/14 [=====] - 0s 3ms/step - loss: 2.1831 - acc: 0.7324
Epoch 6/100
14/14 [=====] - 0s 3ms/step - loss: 1.7571 - acc: 0.7582
Epoch 7/100
14/14 [=====] - 0s 3ms/step - loss: 1.6826 - acc: 0.7981
Epoch 8/100
14/14 [=====] - 0s 3ms/step - loss: 1.6264 - acc: 0.7864
Epoch 9/100
14/14 [=====] - 0s 3ms/step - loss: 1.4832 - acc: 0.8805
Epoch 10/100
```

The code cells show the following code:

```
[14]: 1 from sklearn.preprocessing import StandardScaler
      2 sc = StandardScaler()

[15]: 1 import keras
      2 import pandas as pd
      3 import numpy as np
      4 from keras.models import Sequential
      5 from keras.layers.core import Dense, Activation
      6 from sklearn.datasets import load_breast_cancer
      7 from sklearn.model_selection import train_test_split
      8
      9 # load dataset
     10 cancer_data = load_breast_cancer()
     11 X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
     12                                                    test_size=0.25, random_state=0)
     13 np.random.seed(155)
     14 my_nn = Sequential() # create model
     15 my_nn.add(Dense(20, input_dim=10, activation='relu')) # hidden layer 1
     16 my_nn.add(Dense(1, activation='sigmoid')) # output layer
     17 my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
     18 my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
     19                          initial_epoch=0)
     20 print(my_nn.summary())
     21 print(my_nn.evaluate(X_test, Y_test))
```

Colab Notebook interface showing a Keras model training log. The log displays training progress for epochs 84 to 100, including loss and accuracy metrics. The model architecture is summarized as follows:

Layer (type)	Output Shape	Param #
dense_10 (Dense)	(None, 20)	620
dense_11 (Dense)	(None, 1)	21

Summary statistics:  
Total params: 641  
Trainable params: 641  
Non-trainable params: 0

The code snippet shows the import of Keras and the loading of the MNIST dataset.

```
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
```

Colab Notebook interface showing the Keras model training code. The code defines a sequential model with two dense layers, trains it on the MNIST dataset, and plots the training and validation accuracy and loss curves.

```
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
7 # Load MNIST dataset
8 (x_train, y_train), (x_test, y_test) = mnist.load_data()
9
10 # normalize pixel values to range [0, 1]
11 x_train = x_train.astype('float32') / 255
12 x_test = x_test.astype('float32') / 255
13
14 # convert class labels to binary class matrices
15 num_classes = 10
16 y_train = keras.utils.to_categorical(y_train, num_classes)
17 y_test = keras.utils.to_categorical(y_test, num_classes)
18
19 # create a simple neural network model
20 model = Sequential()
21 model.add(Dense(20, activation='relu', input_shape=(784,)))
22 model.add(Dropout(0.2))
23 model.add(Dense(10, activation='relu'))
24 model.add(Dropout(0.2))
25 model.add(Dense(num_classes, activation='softmax'))
26
27 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
28
29 # train the model and record the training history
30 history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
31                    epochs=100, batch_size=128)
32
33 # plot the training and validation accuracy and loss curves
34 plt.figure(figsize=(10, 5))
35 plt.subplot(1, 2, 1)
36 plt.plot(history.history['accuracy'])
37 plt.plot(history.history['val_accuracy'])
38 plt.title('Model Accuracy')
39 plt.ylabel('Accuracy')
40 plt.xlabel('Epoch')
41 plt.legend(['Train', 'Validation'], loc='lower right')
42
43 plt.subplot(1, 2, 2)
44 plt.plot(history.history['loss'])
45 plt.plot(history.history['val_loss'])
46 plt.title('Model Loss')
47 plt.ylabel('Loss')
48 plt.xlabel('Epoch')
49 plt.legend(['Train', 'Validation'], loc='upper right')
```

Colab Notebook interface showing a Keras model training process. The code defines a Sequential model with a Dense layer, compiles it with categorical\_crossentropy loss and adam optimizer, and trains it on MNIST data. The training history is plotted, showing accuracy and loss over 20 epochs.

```
26 model.add(Dense(num_classes, activation='softmax'))
27
28 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
29
30 # train the model and record the training history
31 history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
32                   epochs=20, batch_size=128)
33
34 # plot the training and validation accuracy and loss curves
35 plt.figure(figsize=(10, 5))
36 plt.subplot(1, 2, 1)
37 plt.plot(history.history['accuracy'])
38 plt.plot(history.history['val_accuracy'])
39 plt.title('Model Accuracy')
40 plt.ylabel('Accuracy')
41 plt.xlabel('Epoch')
42 plt.legend(['Train', 'Validation'], loc='lower right')
43
44 plt.subplot(1, 2, 2)
45 plt.plot(history.history['loss'])
46 plt.plot(history.history['val_loss'])
47 plt.title('Model Loss')
48 plt.ylabel('Loss')
49 plt.xlabel('Epoch')
50 plt.legend(['Train', 'Validation'], loc='upper right')
51
52 plt.show()
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>

Epoch 1/20  
6s 5ms/step - loss: 0.2468 - accuracy: 0.9258 - val\_loss: 0.1891 - val\_accuracy: 0.9607

Epoch 2/20  
3s 5ms/step - loss: 0.1820 - accuracy: 0.9685 - val\_loss: 0.0926 - val\_accuracy: 0.9716

Epoch 3/20  
2s 4ms/step - loss: 0.0741 - accuracy: 0.9765 - val\_loss: 0.0725 - val\_accuracy: 0.9779

Epoch 4/20  
2s 4ms/step - loss: 0.0561 - accuracy: 0.9830 - val\_loss: 0.0713 - val\_accuracy: 0.9772

Epoch 5/20  
2s 4ms/step - loss: 0.0487 - accuracy: 0.9833 - val\_loss: 0.0711 - val\_accuracy: 0.9798

Epoch 6/20  
3s 7ms/step - loss: 0.0390 - accuracy: 0.9873 - val\_loss: 0.0638 - val\_accuracy: 0.9801

Epoch 7/20  
4s 9ms/step - loss: 0.0348 - accuracy: 0.9886 - val\_loss: 0.0890 - val\_accuracy: 0.9761

Epoch 8/20  
2s 5ms/step - loss: 0.0280 - accuracy: 0.9908 - val\_loss: 0.0768 - val\_accuracy: 0.9795

Epoch 9/20  
2s 4ms/step - loss: 0.0287 - accuracy: 0.9906 - val\_loss: 0.0746 - val\_accuracy: 0.9788

Epoch 10/20  
2s 4ms/step - loss: 0.0252 - accuracy: 0.9914 - val\_loss: 0.0852 - val\_accuracy: 0.9787

Epoch 11/20  
2s 4ms/step - loss: 0.0250 - accuracy: 0.9916 - val\_loss: 0.0782 - val\_accuracy: 0.9814

Epoch 12/20

Colab Notebook interface showing the same Keras model training process, but with the training history plotted as two line graphs: Model Accuracy and Model Loss. The Model Accuracy graph shows training accuracy (blue line) and validation accuracy (orange line) over 20 epochs. The Model Loss graph shows training loss (blue line) and validation loss (orange line) over 20 epochs.

Epoch 17/20  
2s 4ms/step - loss: 0.0180 - accuracy: 0.9941 - val\_loss: 0.0743 - val\_accuracy: 0.9833

Epoch 18/20  
2s 4ms/step - loss: 0.0162 - accuracy: 0.9943 - val\_loss: 0.0797 - val\_accuracy: 0.9807

Epoch 19/20  
2s 4ms/step - loss: 0.0152 - accuracy: 0.9950 - val\_loss: 0.0776 - val\_accuracy: 0.9827

Epoch 20/20  
2s 4ms/step - loss: 0.0183 - accuracy: 0.9943 - val\_loss: 0.0766 - val\_accuracy: 0.9825

Epoch 21/20  
2s 5ms/step - loss: 0.0148 - accuracy: 0.9951 - val\_loss: 0.0878 - val\_accuracy: 0.9820

Model Accuracy

Model Loss

```
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
8 # load MNIST dataset
9 (x_train, y_train), (x_test, y_test) = mnist.load_data()
10
11 # normalize pixel values to range [0, 1]
12 x_train = x_train.astype('float32') / 255
13 x_test = x_test.astype('float32') / 255
14
```

Colab Notebook interface showing a Keras model training process. The code defines a simple neural network with three layers: an input layer of 784 units, a hidden layer of 512 units, and an output layer of 10 units. The model is trained on the MNIST dataset for 20 epochs. The training progress is visualized by a progress bar at the top, which shows the current epoch (16) and the validation loss (0.0010).

```
[16] 0 5 10 15 Epoch 0 5 10 15 Epoch
```


```
[17] 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
8 # Load MNIST dataset
9 (x_train, y_train), (x_test, y_test) = mnist.load_data()
10
11 # normalize pixel values to range [0, 1]
12 x_train = x_train.astype('float32') / 255
13 x_test = x_test.astype('float32') / 255
14
15 # convert class labels to binary class matrices
16 num_classes = 10
17 y_train = keras.utils.to_categorical(y_train, num_classes)
18 y_test = keras.utils.to_categorical(y_test, num_classes)
19
20 # create a simple neural network model
21 model = Sequential()
22 model.add(Dense(512, activation='relu', input_shape=(784,)))
23 model.add(Dropout(0.2))
24 model.add(Dense(512, activation='relu'))
25 model.add(Dropout(0.2))
26 model.add(Dense(num_classes, activation='softmax'))
27
28 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
29
30 # train the model
31 model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
32         epochs=20, batch_size=128)
33
34 # plot one of the images in the test data
35 plt.imshow(x_test[0], cmap='gray')
36 plt.show()
37
38 # make a prediction on the image using the trained model
39 prediction = model.predict(x_test[0].reshape(1, -1))
40 print('Model prediction:', np.argmax(prediction))
41
```

Epoch 1/20

Colab Notebook interface showing the continuation of the Keras model training process. The code continues from the previous notebook, showing the training progress for epochs 16 to 20. The training progress is visualized by a progress bar at the top, which shows the current epoch (16) and the validation loss (0.0010).

```
Epoch 16/20: 2s 4ms/step - loss: 0.0219 - accuracy: 0.9927 - val_loss: 0.0615 - val_accuracy: 0.9841
Epoch 17/20: 2s 4ms/step - loss: 0.0212 - accuracy: 0.9927 - val_loss: 0.0740 - val_accuracy: 0.9824
Epoch 18/20: 2s 4ms/step - loss: 0.0233 - accuracy: 0.9920 - val_loss: 0.0717 - val_accuracy: 0.9808
Epoch 19/20: 2s 5ms/step - loss: 0.0182 - accuracy: 0.9938 - val_loss: 0.0655 - val_accuracy: 0.9844
Epoch 20/20: 2s 5ms/step - loss: 0.0196 - accuracy: 0.9934 - val_loss: 0.0811 - val_accuracy: 0.9820
Epoch 16/20: 2s 4ms/step - loss: 0.0233 - accuracy: 0.9948 - val_loss: 0.0985 - val_accuracy: 0.9806
Epoch 17/20: 2s 4ms/step - loss: 0.0165 - accuracy: 0.9946 - val_loss: 0.0752 - val_accuracy: 0.9835
Epoch 18/20: 2s 4ms/step - loss: 0.0189 - accuracy: 0.9940 - val_loss: 0.0795 - val_accuracy: 0.9815
Epoch 19/20: 2s 4ms/step - loss: 0.0166 - accuracy: 0.9948 - val_loss: 0.0880 - val_accuracy: 0.9827
Epoch 20/20: 2s 4ms/step - loss: 0.0137 - accuracy: 0.9951 - val_loss: 0.0819 - val_accuracy: 0.9837
```

Model prediction: 7



Colab Notebook - Google Drive | KP2.ipynb - Colaboratory | Dense Layers and Accuracy | Online Servicing App | YouTube

colab.research.google.com/drive/1WwT4L2G7H1B-Zv4S0lieCuR-27LXXF9#scrollTo=N3j5FKbDNqEB

Open AI | ChatGPT | Microsoft Learn | Kaggle | Medium | USA | UCM | LeetCode | Install Your eSIM | Sign in to your UK Visa... | Benchmark - make your... | Your information - Go... | Pick n Pull | Vehicle Det...

ICP2.ipynb | File | Edit | View | Insert | Runtime | Tools | Help | All changes saved

Table of contents | Keras | Mount to Google Drive | Playing with Diabetes dataset | Playing with Breastcancer Dataset | Section

```

1 1/1 [.....] - 0s 135ms/step
2 Model prediction: 7
3
4
5
6
7
8 # load MNIST dataset
9 (x_train, y_train), (x_test, y_test) = mnist.load_data()
10
11 # normalize pixel values to range [0, 1]
12 x_train = x_train.astype('float32') / 255
13 x_test = x_test.astype('float32') / 255
14
15 # convert class labels to binary class matrices
16 num_classes = 10
17 y_train = keras.utils.to_categorical(y_train, num_classes)
18 y_test = keras.utils.to_categorical(y_test, num_classes)
19
20 # create a list of models to train
21 models = []
22
23 # model with 1 hidden layer and tanh activation
24 model = Sequential()
25 model.add(Dense(512, activation='tanh', input_shape=(784,)))
26 model.add(Dropout(0.2))
27 model.add(Dense(num_classes, activation='softmax'))
28 models.append(('1 hidden layer with tanh', model))
29
30 # model with 1 hidden layer and sigmoid activation
31 model = Sequential()
32 model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
33 model.add(Dropout(0.2))
34 model.add(Dense(num_classes, activation='softmax'))
35 models.append(('1 hidden layer with sigmoid', model))
36
37 # model with 2 hidden layers and tanh activation
38 model = Sequential()
39 model.add(Dense(512, activation='tanh', input_shape=(784,)))
40 model.add(Dropout(0.2))
41 model.add(Dense(512, activation='tanh'))
42 model.add(Dropout(0.2))
43 model.add(Dense(num_classes, activation='softmax'))
44 models.append(('2 hidden layers with tanh', model))
45
46 # model with 2 hidden layers and sigmoid activation
47 model = Sequential()
48 model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
49 model.add(Dropout(0.2))
50 model.add(Dense(512, activation='sigmoid'))
51 model.add(Dropout(0.2))
52 model.add(Dense(num_classes, activation='softmax'))
53 models.append(('2 hidden layers with sigmoid', model))
54
55 # train each model and plot loss and accuracy curves
56 for name, model in models:
57     model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
58     history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
59                       epochs=20, batch_size=128, verbose=0)
60     # plot loss and accuracy curves
61     plt.plot(history.history['loss'], label='train_loss')
62     plt.plot(history.history['val_loss'], label='val_loss')
63     plt.plot(history.history['accuracy'], label='train_accuracy')
64     plt.plot(history.history['val_accuracy'], label='val_accuracy')
65     plt.title(name)
66     plt.xlabel('Epoch')
67     plt.legend()
68     plt.show()
69
70 # evaluate the model on test data
71 loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
72 print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
73

```

76°F Clear | Search | 2m 29s completed at 11:19PM

Colab Notebook - Google Drive | KP2.ipynb - Colaboratory | Dense Layers and Accuracy | Online Servicing App | YouTube

colab.research.google.com/drive/1WwT4L2G7H1B-Zv4S0lieCuR-27LXXF9#scrollTo=N3j5FKbDNqEB

Open AI | ChatGPT | Microsoft Learn | Kaggle | Medium | USA | UCM | LeetCode | Install Your eSIM | Sign in to your UK Visa... | Benchmark - make your... | Your information - Go... | Pick n Pull | Vehicle Det...

ICP2.ipynb | File | Edit | View | Insert | Runtime | Tools | Help | All changes saved

Table of contents | Keras | Mount to Google Drive | Playing with Diabetes dataset | Playing with Breastcancer Dataset | Section

```

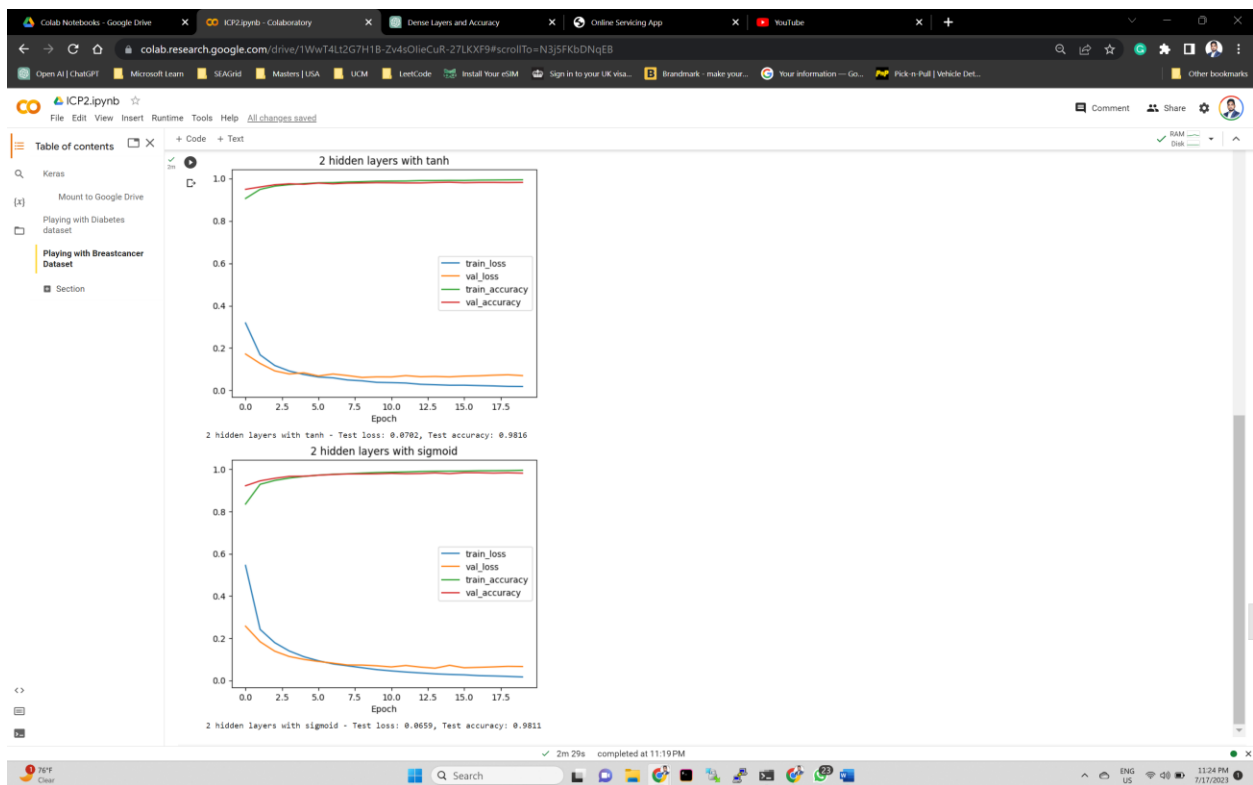
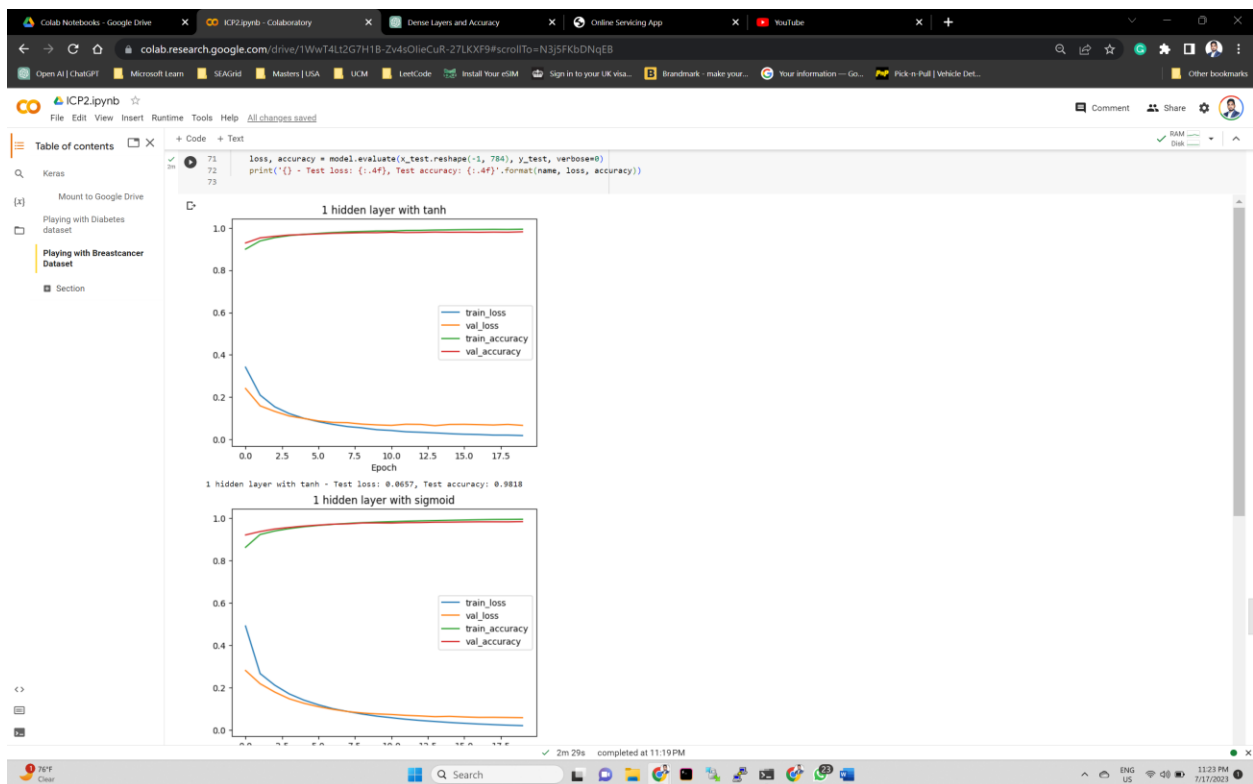
32 model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
33 model.add(Dropout(0.2))
34 model.add(Dense(num_classes, activation='softmax'))
35 models.append(('1 hidden layer with sigmoid', model))
36
37 # model with 2 hidden layers and tanh activation
38 model = Sequential()
39 model.add(Dense(512, activation='tanh', input_shape=(784,)))
40 model.add(Dropout(0.2))
41 model.add(Dense(512, activation='tanh'))
42 model.add(Dropout(0.2))
43 model.add(Dense(num_classes, activation='softmax'))
44 models.append(('2 hidden layers with tanh', model))
45
46 # model with 2 hidden layers and sigmoid activation
47 model = Sequential()
48 model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
49 model.add(Dropout(0.2))
50 model.add(Dense(512, activation='sigmoid'))
51 model.add(Dropout(0.2))
52 model.add(Dense(num_classes, activation='softmax'))
53 models.append(('2 hidden layers with sigmoid', model))
54
55 # train each model and plot loss and accuracy curves
56 for name, model in models:
57     model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
58     history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
59                       epochs=20, batch_size=128, verbose=0)
60     # plot loss and accuracy curves
61     plt.plot(history.history['loss'], label='train_loss')
62     plt.plot(history.history['val_loss'], label='val_loss')
63     plt.plot(history.history['accuracy'], label='train_accuracy')
64     plt.plot(history.history['val_accuracy'], label='val_accuracy')
65     plt.title(name)
66     plt.xlabel('Epoch')
67     plt.legend()
68     plt.show()
69
70 # evaluate the model on test data
71 loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
72 print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
73

```

1 hidden layer with tanh

76°F Clear | Search | 2m 29s completed at 11:19PM





Colab Notebook interface showing a Keras model training script. The script defines two models: one with 1 hidden layer and sigmoid activation, and another with 2 hidden layers and sigmoid activation. The training process is completed, showing a test loss of 0.0659 and test accuracy of 0.9831.

```
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
8 # load MNIST dataset
9 (x_train, y_train), (x_test, y_test) = mnist.load_data()
10
11 # convert class labels to binary class matrices
12 num_classes = 10
13 y_train = keras.utils.to_categorical(y_train, num_classes)
14 y_test = keras.utils.to_categorical(y_test, num_classes)
15
16 # create a list of models to train
17 models = []
18
19 # model with 1 hidden layer and tanh activation
20 model = Sequential()
21 model.add(Dense(512, activation='tanh', input_shape=(784,)))
22 model.add(Dropout(0.2))
23 model.add(Dense(num_classes, activation='softmax'))
24 models.append(('1 hidden layer with tanh', model))
25
26 # model with 1 hidden layer and sigmoid activation
27 model = Sequential()
28 model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
29 model.add(Dropout(0.2))
30 model.add(Dense(num_classes, activation='softmax'))
31 models.append(('1 hidden layer with sigmoid', model))
32
33 # model with 2 hidden layers and tanh activation
34 model = Sequential()
35 model.add(Dense(512, activation='tanh', input_shape=(784,)))
36 model.add(Dropout(0.2))
37 model.add(Dense(512, activation='tanh'))
38 model.add(Dropout(0.2))
39 model.add(Dense(num_classes, activation='softmax'))
40 models.append(('2 hidden layers with tanh', model))
41
42 # model with 2 hidden layers and sigmoid activation
43 model = Sequential()
44 model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
45 model.add(Dropout(0.2))
46 model.add(Dense(512, activation='sigmoid'))
47 model.add(Dropout(0.2))
48 model.add(Dense(num_classes, activation='softmax'))
```

2 hidden layers with sigmoid - Test loss: 0.0659, Test accuracy: 0.9831

RAM 11:24 PM 7/17/2021

Colab Notebook interface showing the same Keras model training script, but with a plot of training and validation loss and accuracy over 20 epochs. The plot shows that the 2 hidden layers model with sigmoid activation performs better than the 1 hidden layer model with sigmoid activation.

```
44 model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
45 model.add(Dropout(0.2))
46 model.add(Dense(512, activation='sigmoid'))
47 model.add(Dropout(0.2))
48 model.add(Dense(num_classes, activation='softmax'))
49 models.append(('2 hidden layers with sigmoid', model))
50
51 # train each model and plot loss and accuracy curves
52 for name, model in models:
53     model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
54     history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
55                       epochs=20, batch_size=128, verbose=0)
56     # plot loss and accuracy curves
57     plt.plot(history.history['loss'], label='train_loss')
58     plt.plot(history.history['val_loss'], label='val_loss')
59     plt.plot(history.history['accuracy'], label='train_accuracy')
60     plt.plot(history.history['val_accuracy'], label='val_accuracy')
61     plt.title(name)
62     plt.xlabel('Epoch')
63     plt.legend()
64     plt.show()
65
66 # evaluate the model on test data
67 loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
68 print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```

Epoch

1 hidden layer with sigmoid - Test loss: 0.1532, Test accuracy: 0.9539

2 hidden layers with tanh

RAM 11:24 PM 7/17/2021

