Lab Report 1

Goutham Krishnan – 21BAI1007

Aim

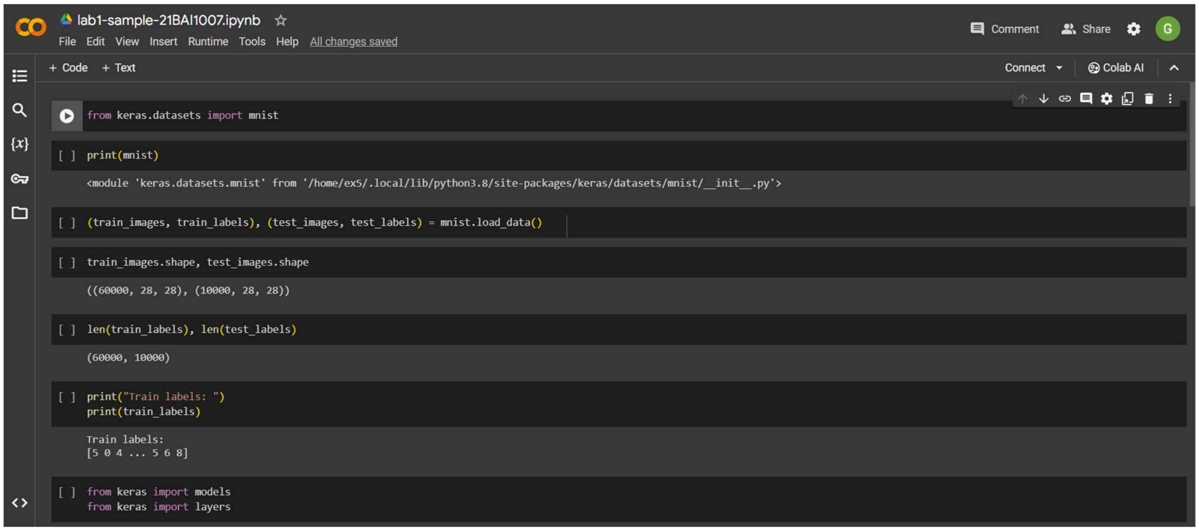
1. Classify the mnist handwritten digit dataset.
2. Import Fashion MNIST Dataset and perform classification for the following labels
3. Import PatchCamelyon (PCam) and perform classification
4. Import Cat and Dog Dataset for classification

1.Classify the mnist handwritten digit dataset.

Code:

```
from keras.datasets import mnist


print(mnist)


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


train_images.shape, test_images.shape


len(train_labels), len(test_labels)


print("Train labels: ")
print(train_labels)


from keras import models
from keras import layers


from keras.models import Sequential
from keras.layers import Dense


model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(28*28,)))
model.add(Dense(10, activation='softmax'))


model.compile(optimizer='adam', loss='categorical_crossentropy', metrics='mean_absolute_error')


train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255


from keras.utils import to_categorical


train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)




model.fit(train_images, train_labels, epochs=200, batch_size=128)


test_loss, mae = model.evaluate(test_images, test_labels)
```

Output


lab1-sample-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
G
+ Code + Text
Connect
Colab AI

```
from keras.datasets import mnist
```

```
[ ] print(mnist)

    <module 'keras.datasets.mnist' from '/home/ex5/.local/lib/python3.8/site-packages/keras/datasets/mnist/__init__.py'>
```

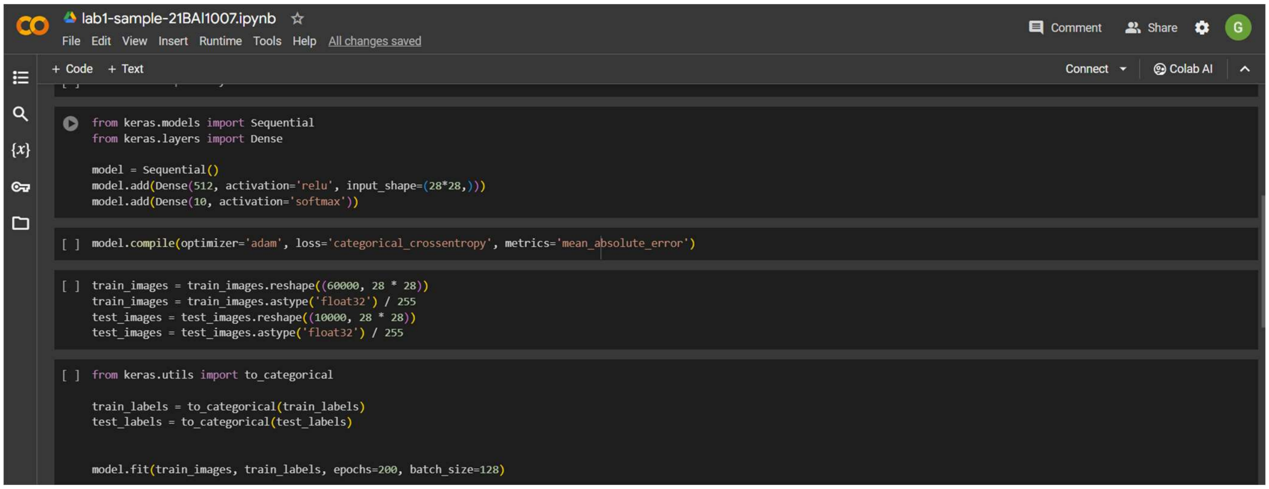
```
[ ] (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
[ ] train_images.shape, test_images.shape

    ((60000, 28, 28), (10000, 28, 28))
```

```
[ ] len(train_labels), len(test_labels)

    (60000, 10000)
```

```
[ ] print("Train labels: ")
    print(train_labels)

    Train labels:
    [5 0 4 ... 5 6 8]
```

```
[ ] from keras import models
    from keras import layers
```

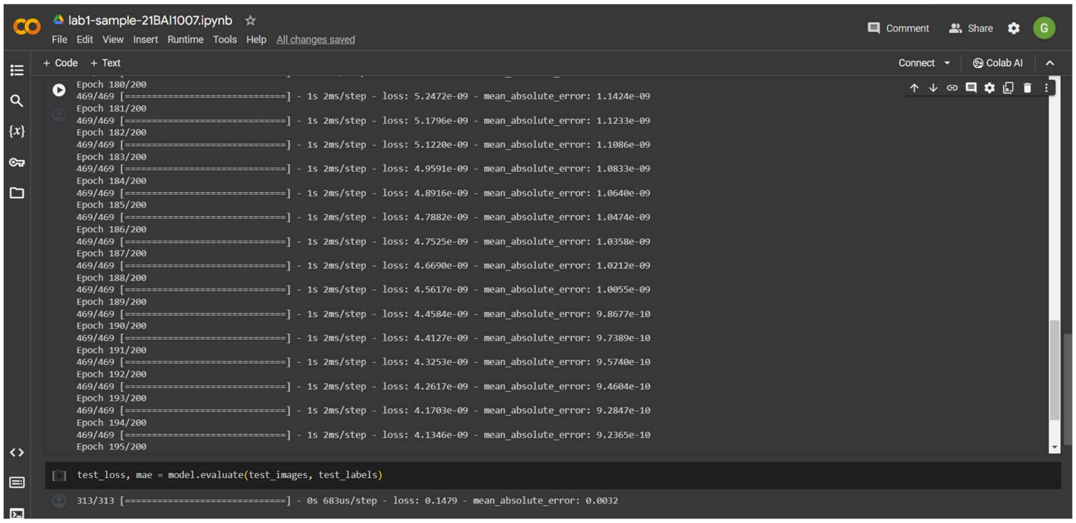

lab1-sample-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
G
+ Code + Text
Connect
Colab AI

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(28*28,)))
model.add(Dense(10, activation='softmax'))
```

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

```
[ ] train_images = train_images.reshape((60000, 28 * 28))
    train_images = train_images.astype('float32') / 255
    test_images = test_images.reshape((10000, 28 * 28))
    test_images = test_images.astype('float32') / 255
```

```
[ ] from keras.utils import to_categorical

    train_labels = to_categorical(train_labels)
    test_labels = to_categorical(test_labels)


    model.fit(train_images, train_labels, epochs=200, batch_size=128)
```


lab1-sample-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
G
+ Code + Text
Connect
Colab AI

```
Epoch 180/200
469/469 [==============================] - 1s 2ms/step - loss: 5.2472e-09 - mean_absolute_error: 1.1424e-09
Epoch 181/200
469/469 [==============================] - 1s 2ms/step - loss: 5.1796e-09 - mean_absolute_error: 1.1233e-09
Epoch 182/200
469/469 [==============================] - 1s 2ms/step - loss: 5.1220e-09 - mean_absolute_error: 1.1086e-09
Epoch 183/200
469/469 [==============================] - 1s 2ms/step - loss: 4.9591e-09 - mean_absolute_error: 1.0833e-09
Epoch 184/200
469/469 [==============================] - 1s 2ms/step - loss: 4.8916e-09 - mean_absolute_error: 1.0640e-09
Epoch 185/200
469/469 [==============================] - 1s 2ms/step - loss: 4.7882e-09 - mean_absolute_error: 1.0474e-09
Epoch 186/200
469/469 [==============================] - 1s 2ms/step - loss: 4.7525e-09 - mean_absolute_error: 1.0358e-09
Epoch 187/200
469/469 [==============================] - 1s 2ms/step - loss: 4.6690e-09 - mean_absolute_error: 1.0212e-09
Epoch 188/200
469/469 [==============================] - 1s 2ms/step - loss: 4.5617e-09 - mean_absolute_error: 1.0055e-09
Epoch 189/200
469/469 [==============================] - 1s 2ms/step - loss: 4.4584e-09 - mean_absolute_error: 9.8677e-10
Epoch 190/200
469/469 [==============================] - 1s 2ms/step - loss: 4.4127e-09 - mean_absolute_error: 9.7389e-10
Epoch 191/200
469/469 [==============================] - 1s 2ms/step - loss: 4.3253e-09 - mean_absolute_error: 9.5740e-10
Epoch 192/200
469/469 [==============================] - 1s 2ms/step - loss: 4.2617e-09 - mean_absolute_error: 9.4604e-10
Epoch 193/200
469/469 [==============================] - 1s 2ms/step - loss: 4.1703e-09 - mean_absolute_error: 9.2847e-10
Epoch 194/200
469/469 [==============================] - 1s 2ms/step - loss: 4.1346e-09 - mean_absolute_error: 9.2365e-10
Epoch 195/200
```

```
[ ] test_loss, mae = model.evaluate(test_images, test_labels)

313/313 [==============================] - 0s 683us/step - loss: 0.1479 - mean_absolute_error: 0.0032
```

2.Import Fashion MNIST Dataset and perform classification for the following labels

Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal

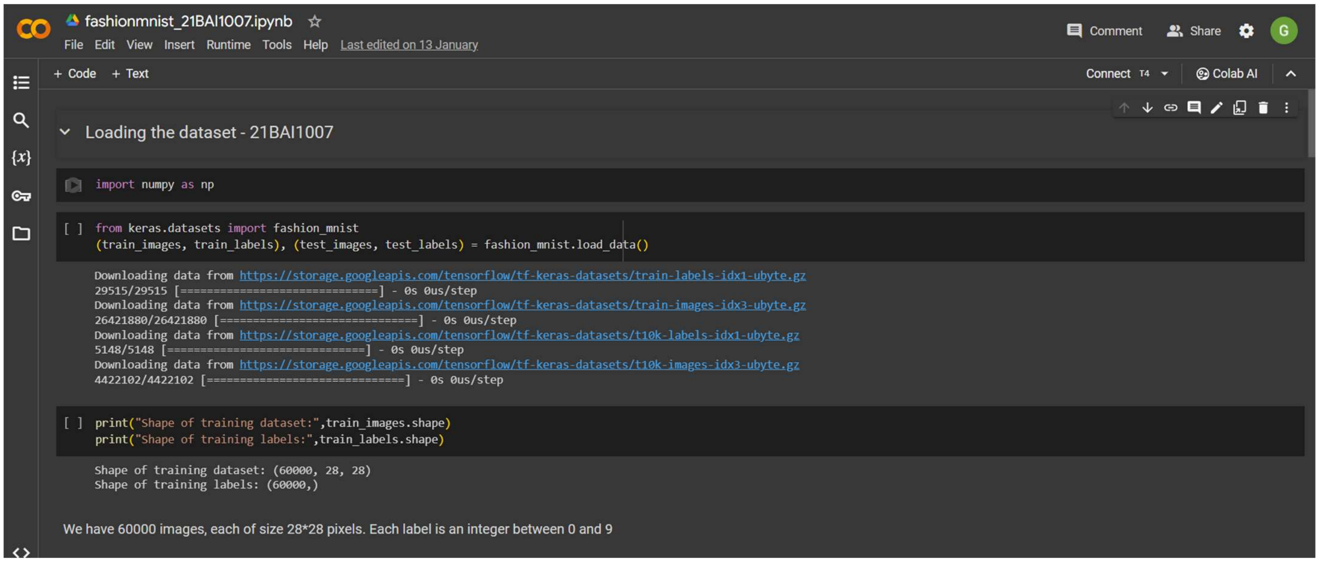
Code

import numpy as np

from keras.datasets import fashion_mnist

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

print("Shape of training dataset:",train_images.shape)
print("Shape of training labels:",train_labels.shape)

"""We have 60000 images, each of size 28*28 pixels. Each label is an integer between 0 and 9"""

print("Length of training dataset: ",len(train_images))
print("Length of testing dataset: ",len(test_images))

"""### Normalizing the dataset - 21BAI1007"""

import matplotlib.pyplot as plt

## PLotting the image of the first object in the dataset
plt.figure()
plt.imshow(train_images[0])

## To normalize the dataset, we divide each value by 255 so that the data is minimized to a value between 0 and 1
train_images = train_images/255.0
test_images = test_images/255.0

"""### Build the model - 21BAI1007"""

from keras.models import Sequential
from keras.layers import Dense, Flatten

model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
model.summary()


"""### Training the model - 21BAI1007"""


model.fit(train_images, train_labels, validation_data=(test_images, test_labels), epochs=100)


test_loss, test_acc = model.evaluate(test_images, test_labels)


"""### Predicting the classes of the test dataset - 21BAI1007"""


pred = model.predict(test_images)


# The predictions are a probability distribution of each class. To convert it into the required class
# Predicting the class of the first element in the testing dataset
np.argmax(pred[0])


# Checking if the prediction is correct
test_labels[0]


class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']


"""### Predciting a random image from the test dataset - 21BAI1007"""


image = test_images[1] ## An image of a pullover
print(image.shape)


plt.imshow(test_images[1])


# To process the image in the model, we have to reshape it no include an extra dimension
image = (np.expand_dims(image, 0))
print(image.shape)


# Function to map the class number to the class name
def get_item_name(item_number, class_names):
    try:
        item_name = class_names[item_number]
        return item_name
    except IndexError:
        return "Item not found"

final_prediction = model.predict(image)
ans = np.argmax(final_prediction)
item_name = get_item_name(ans, class_names)
item_name

"""Hence we correctly predicted the class of the image"""
```

Output



fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share G
+ Code + Text
Connect T4
Colab AI

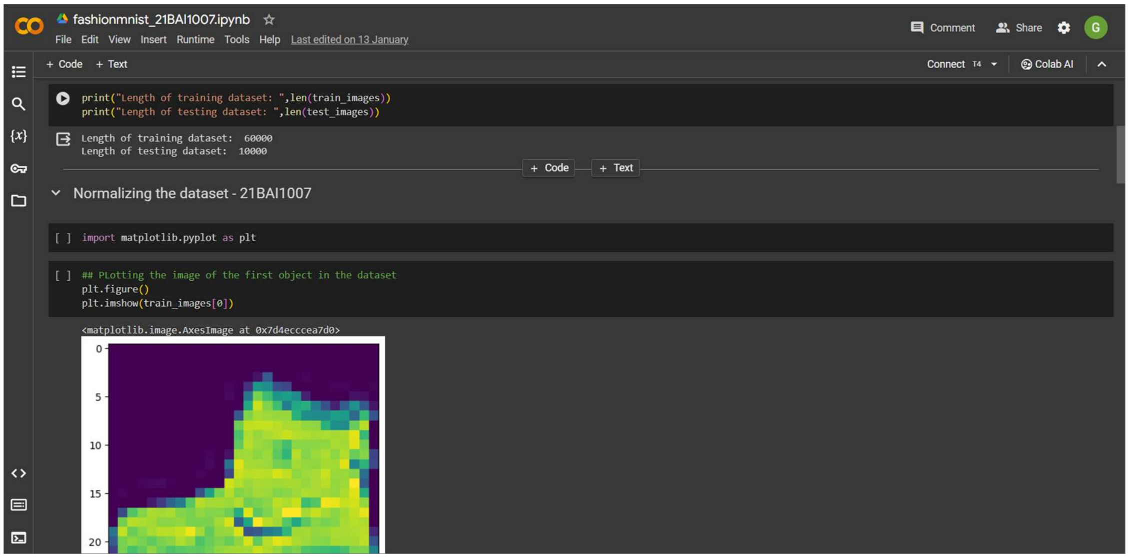
Loading the dataset - 21BAI1007

```
import numpy as np
```

```
[ ] from keras.datasets import fashion_mnist
    (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
    29515/29515 [==============================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
    26421880/26421880 [==============================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
    5148/5148 [==============================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
    4422102/4422102 [==============================] - 0s 0us/step
```

```
[ ] print("Shape of training dataset:",train_images.shape)
    print("Shape of training labels:",train_labels.shape)

    Shape of training dataset: (60000, 28, 28)
    Shape of training labels: (60000,)
```

We have 60000 images, each of size 28*28 pixels. Each label is an integer between 0 and 9



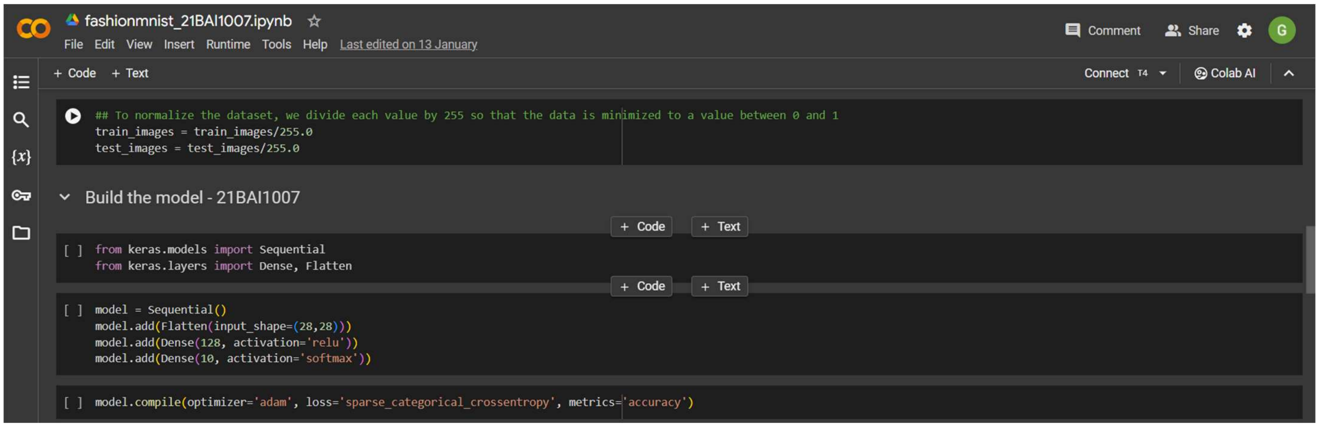
fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share G
+ Code + Text
Connect T4
Colab AI
+ Code + Text
0
5
10
15
20

```
print("Length of training dataset: ",len(train_images))
print("Length of testing dataset: ",len(test_images))

Length of training dataset:  60000
Length of testing dataset:  10000
```

Normalizing the dataset - 21BAI1007

```
[ ] import matplotlib.pyplot as plt
```

```
[ ] ## PLotting the image of the first object in the dataset
    plt.figure()
    plt.imshow(train_images[0])

    <matplotlib.image.AxesImage at 0x7d4ecccea7d0>
```


fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share
+ Code + Text
Connect T4 Colab AI

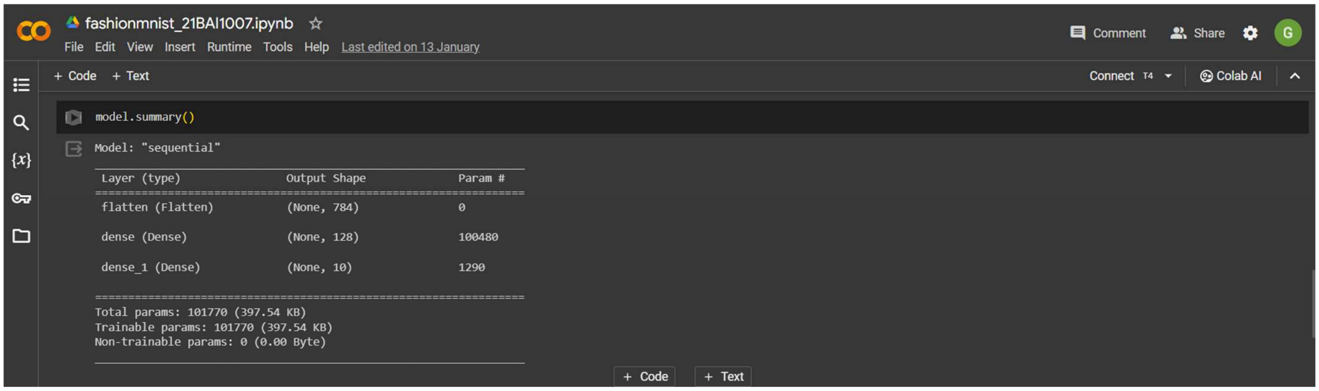
```
## To normalize the dataset, we divide each value by 255 so that the data is minimized to a value between 0 and 1
train_images = train_images/255.0
test_images = test_images/255.0
```

Build the model - 21BAI1007

```
from keras.models import Sequential
from keras.layers import Dense, Flatten
```

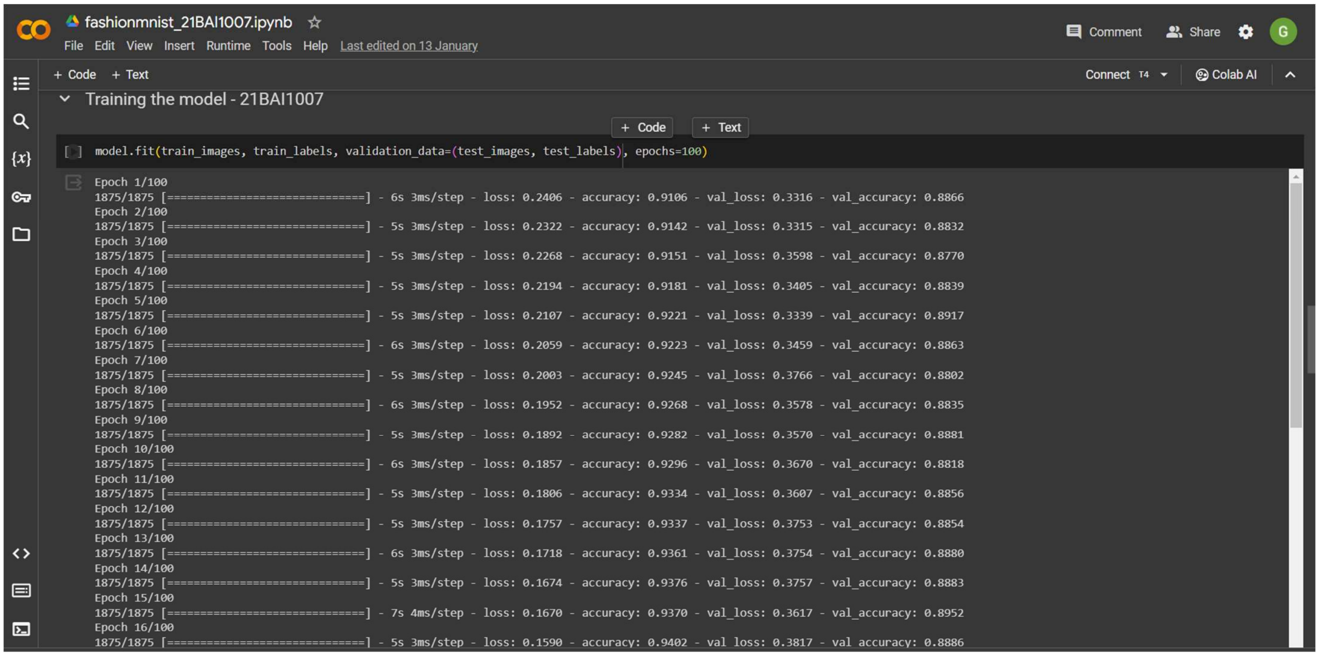
```
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
```


fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share
+ Code + Text
Connect T4 Colab AI

```
model.summary()
```

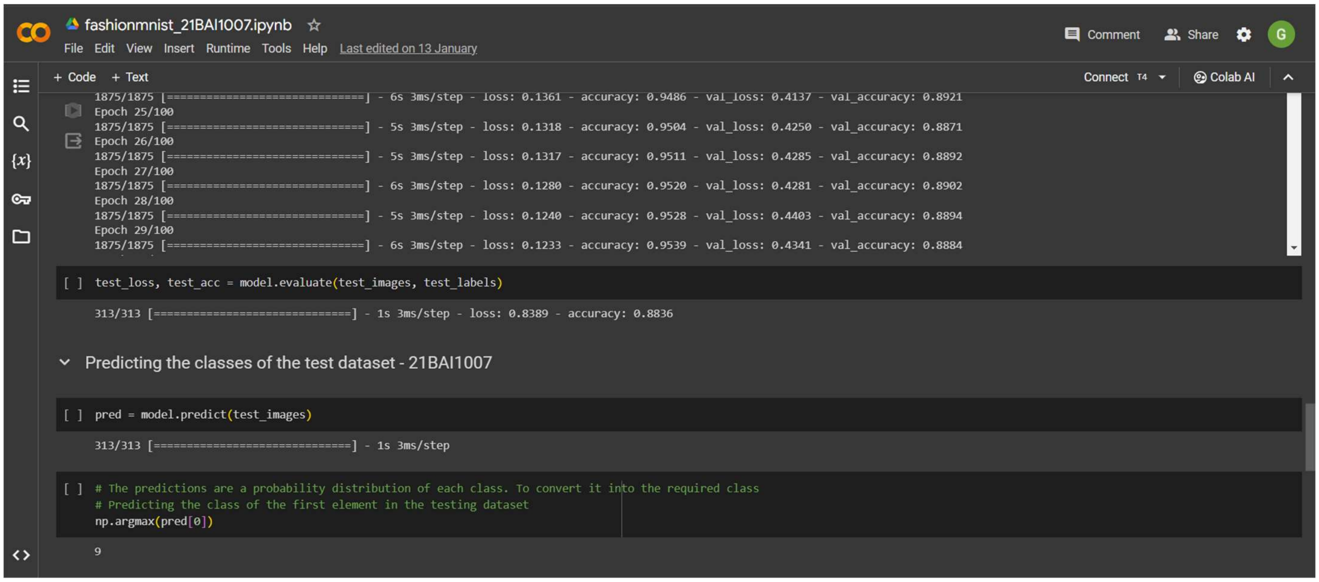
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```


fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share
+ Code + Text
Connect T4 Colab AI

Training the model - 21BAI1007

```
model.fit(train_images, train_labels, validation_data=(test_images, test_labels), epochs=100)
```

```
Epoch 1/100
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2406 - accuracy: 0.9106 - val_loss: 0.3316 - val_accuracy: 0.8866
Epoch 2/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2322 - accuracy: 0.9142 - val_loss: 0.3315 - val_accuracy: 0.8832
Epoch 3/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2268 - accuracy: 0.9151 - val_loss: 0.3598 - val_accuracy: 0.8770
Epoch 4/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2194 - accuracy: 0.9181 - val_loss: 0.3405 - val_accuracy: 0.8839
Epoch 5/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2107 - accuracy: 0.9221 - val_loss: 0.3339 - val_accuracy: 0.8917
Epoch 6/100
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2059 - accuracy: 0.9223 - val_loss: 0.3459 - val_accuracy: 0.8863
Epoch 7/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2003 - accuracy: 0.9245 - val_loss: 0.3766 - val_accuracy: 0.8802
Epoch 8/100
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1952 - accuracy: 0.9268 - val_loss: 0.3578 - val_accuracy: 0.8835
Epoch 9/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1892 - accuracy: 0.9282 - val_loss: 0.3570 - val_accuracy: 0.8881
Epoch 10/100
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1857 - accuracy: 0.9296 - val_loss: 0.3670 - val_accuracy: 0.8818
Epoch 11/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1806 - accuracy: 0.9334 - val_loss: 0.3607 - val_accuracy: 0.8856
Epoch 12/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1757 - accuracy: 0.9337 - val_loss: 0.3753 - val_accuracy: 0.8854
Epoch 13/100
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1718 - accuracy: 0.9361 - val_loss: 0.3754 - val_accuracy: 0.8880
Epoch 14/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1674 - accuracy: 0.9376 - val_loss: 0.3757 - val_accuracy: 0.8883
Epoch 15/100
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1670 - accuracy: 0.9370 - val_loss: 0.3617 - val_accuracy: 0.8952
Epoch 16/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1590 - accuracy: 0.9402 - val_loss: 0.3817 - val_accuracy: 0.8886
```


fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share
+ Code + Text
Connect T4
Colab AI

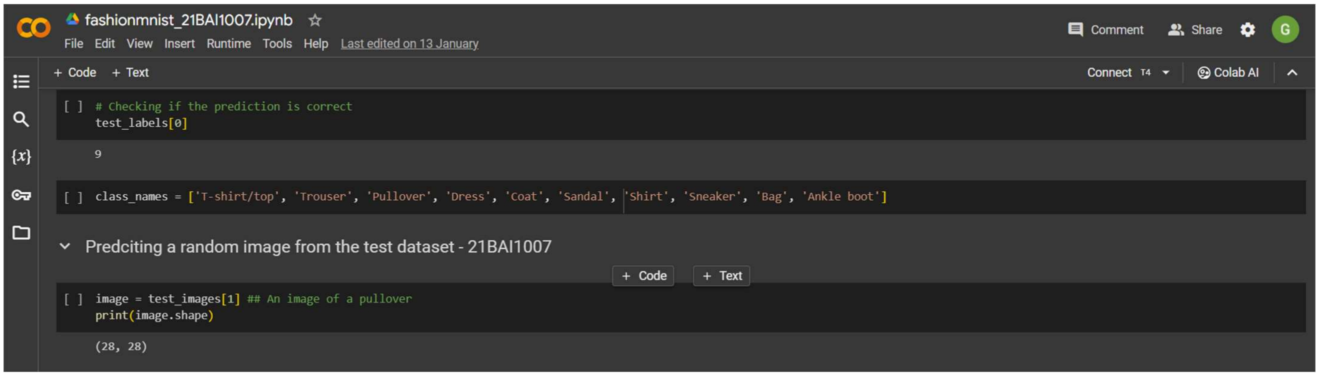
```
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1361 - accuracy: 0.9486 - val_loss: 0.4137 - val_accuracy: 0.8921
Epoch 25/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1318 - accuracy: 0.9504 - val_loss: 0.4250 - val_accuracy: 0.8871
Epoch 26/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1317 - accuracy: 0.9511 - val_loss: 0.4285 - val_accuracy: 0.8892
Epoch 27/100
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1280 - accuracy: 0.9520 - val_loss: 0.4281 - val_accuracy: 0.8902
Epoch 28/100
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1240 - accuracy: 0.9528 - val_loss: 0.4403 - val_accuracy: 0.8894
Epoch 29/100
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1233 - accuracy: 0.9539 - val_loss: 0.4341 - val_accuracy: 0.8884
```

```
[ ] test_loss, test_acc = model.evaluate(test_images, test_labels)

    313/313 [==============================] - 1s 3ms/step - loss: 0.8389 - accuracy: 0.8836
```

Predicting the classes of the test dataset - 21BAI1007

```
[ ] pred = model.predict(test_images)

    313/313 [==============================] - 1s 3ms/step
```

```
[ ] # The predictions are a probability distribution of each class. To convert it into the required class
    # Predicting the class of the first element in the testing dataset
    np.argmax(pred[0])

    9
```

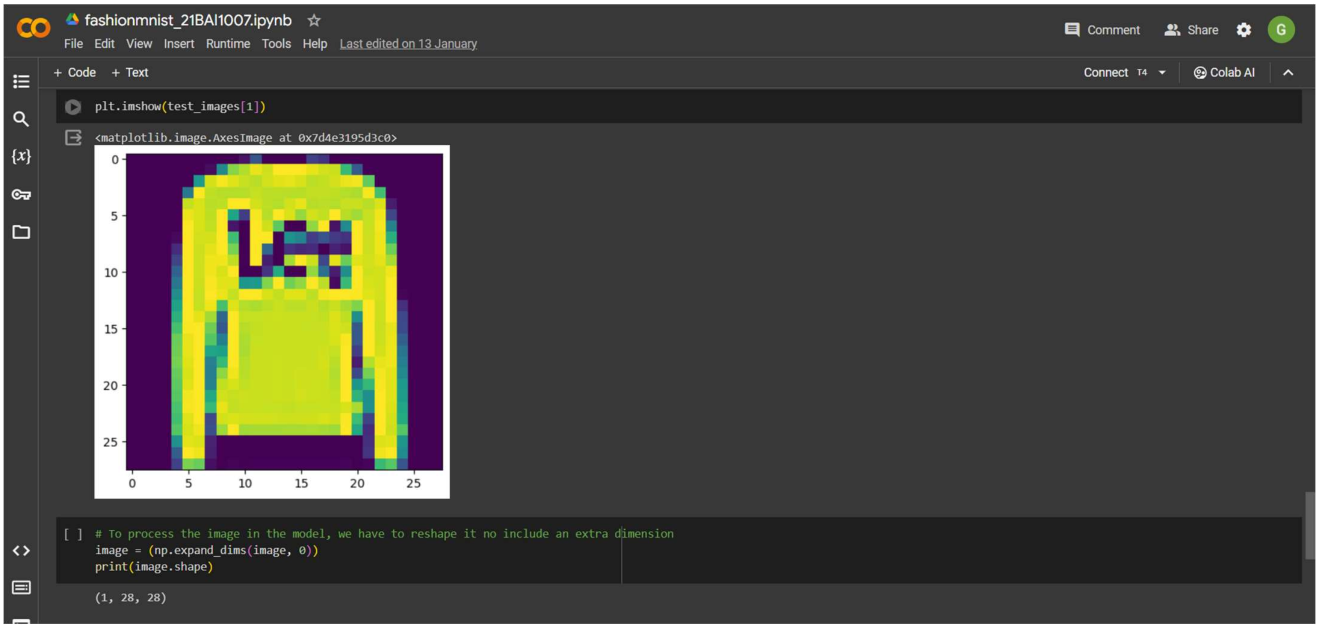

fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share
+ Code + Text
Connect T4
Colab AI

```
[ ] # Checking if the prediction is correct
    test_labels[0]

    9
```

```
[ ] class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

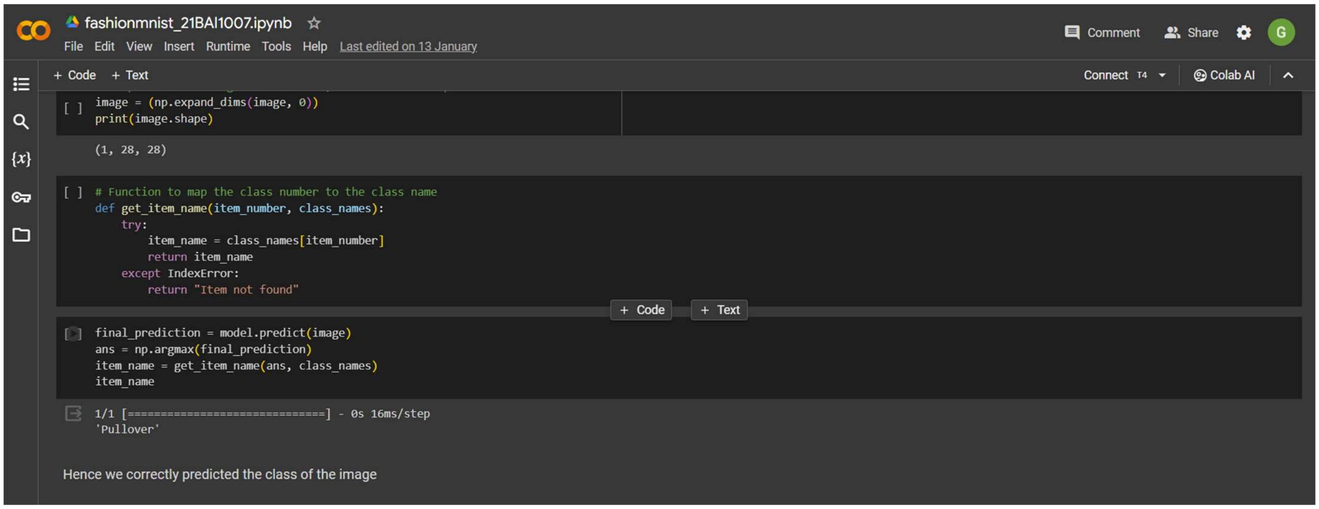
Predciting a random image from the test dataset - 21BAI1007

```
[ ] image = test_images[1] ## An image of a pullover
    print(image.shape)

    (28, 28)
```


fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share
+ Code + Text
Connect T4
Colab AI
0
5
10
15
20
25
0
5
10
15
20
25

```
plt.imshow(test_images[1])

<matplotlib.image.AxesImage at 0x7d4e3195d3c0>
```

```
[ ] # To process the image in the model, we have to reshape it no include an extra dimension
    image = (np.expand_dims(image, 0))
    print(image.shape)

    (1, 28, 28)
```


fashionmnist_21BAI1007.ipynb
File Edit View Insert Runtime Tools Help Last edited on 13 January
Comment Share
+ Code + Text
Connect T4
Colab AI
+ Code + Text

```
image = (np.expand_dims(image, 0))
print(image.shape)
```

```
(1, 28, 28)
```

```
# Function to map the class number to the class name
def get_item_name(item_number, class_names):
    try:
        item_name = class_names[item_number]
        return item_name
    except IndexError:
        return "Item not found"
```

```
final_prediction = model.predict(image)
ans = np.argmax(final_prediction)
item_name = get_item_name(ans, class_names)
item_name
```

```
1/1 [==============================] - 0s 16ms/step
'Pullover'
```

Hence we correctly predicted the class of the image

3.Import PatchCamelyon (PCam) and perform classification

Code

!mkdir -p ~/.kaggle

!cp kaggle.json ~/.kaggle/

!kaggle competitions download -c histopathologic-cancer-detection

"""### Loading Dataset - 21BAI1007"""

import zipfile

zip_ref = zipfile.ZipFile('/content/histopathologic-cancer-detection.zip', 'r')

zip_ref.extractall('/content')

zip_ref.close()

# Commented out IPython magic to ensure Python compatibility.

import pandas as pd

import numpy as np

import tensorflow as tf

import keras

from keras.preprocessing.image import ImageDataGenerator

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten, Activation

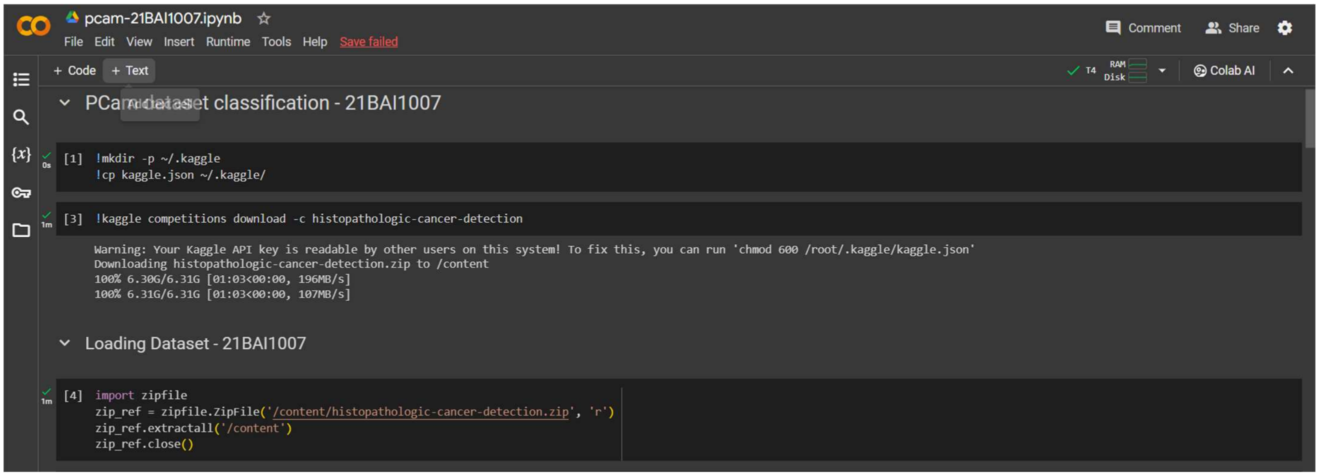
```
import cv2
import matplotlib.pyplot as plt
# %matplotlib inline
import os

test_path = '../content/test/'
train_path = '../content/train/'
train_data = pd.read_csv('../content/train_labels.csv')

"""Labels <br>
0 = no tumor
1 = tumor
"""

# No of images in each folder
print(len(os.listdir('../content/train')))
print(len(os.listdir('../content/test')))

train_data.info()
print("")
print(train_data.head())
print("")
print(train_data.describe())
print("")
print(len(os.listdir(test_path)))

"""### Preprocessing and imageGeneration - 21BAI1007"""

train_data["id"] = train_data["id"].apply(lambda x: x + ".tif")
train_data["label"] = train_data["label"].astype(str)

datagen = ImageDataGenerator(rescale=1./255., validation_split=0.2)
```

```
train_generator = datagen.flow_from_dataframe(
    dataframe=train_data,
    directory=train_path,
    x_col="id",
    y_col="label",
    subset="training",
    batch_size=256,
    seed=13,
    class_mode="binary",
    target_size=(64,64),
    shuffle=True)

valid_generator = datagen.flow_from_dataframe(
    dataframe=train_data,
    directory=train_path,
    x_col="id",
    y_col="label",
    subset="validation",
    batch_size=256,
    seed=13,
    class_mode="binary",
    target_size=(64,64),
    shuffle=True)

"""### Creating the model - 21BAI1007"""

model = Sequential()

model.add(Conv2D(filters=16, kernel_size=(3,3)))
model.add(Conv2D(filters=16, kernel_size=(3,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(3,3)))
model.add(Conv2D(filters=32, kernel_size=(3,3)))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))

model.build(input_shape=(32, 64, 64, 3))

model.compile(loss='binary_crossentropy', metrics=['accuracy'])

model.summary()

model.fit(train_generator,steps_per_epoch=687,epochs = 5,validation_data =
valid_generator,validation_steps=171,verbose=1)

test_data = pd.DataFrame({'id':os.listdir(test_path)})
test_data.head()

"""### Prediciting the test dataset - 21BAI1007"""

datagen_test = ImageDataGenerator(rescale=1./255.)

test_generator = datagen_test.flow_from_dataframe(
    dataframe=test_data,
    directory=test_path,
    x_col='id',
    y_col=None,
    target_size=(64,64),
    batch_size=1,
    shuffle=False,
    class_mode=None)

results = model.predict(test_generator, verbose=1)

results

results = np.transpose(results)[0]
```

answer = list(map(lambda x: 0 if x < 0.5 else 1, results))

answer

Output



PCam dataset classification - 21BAI1007

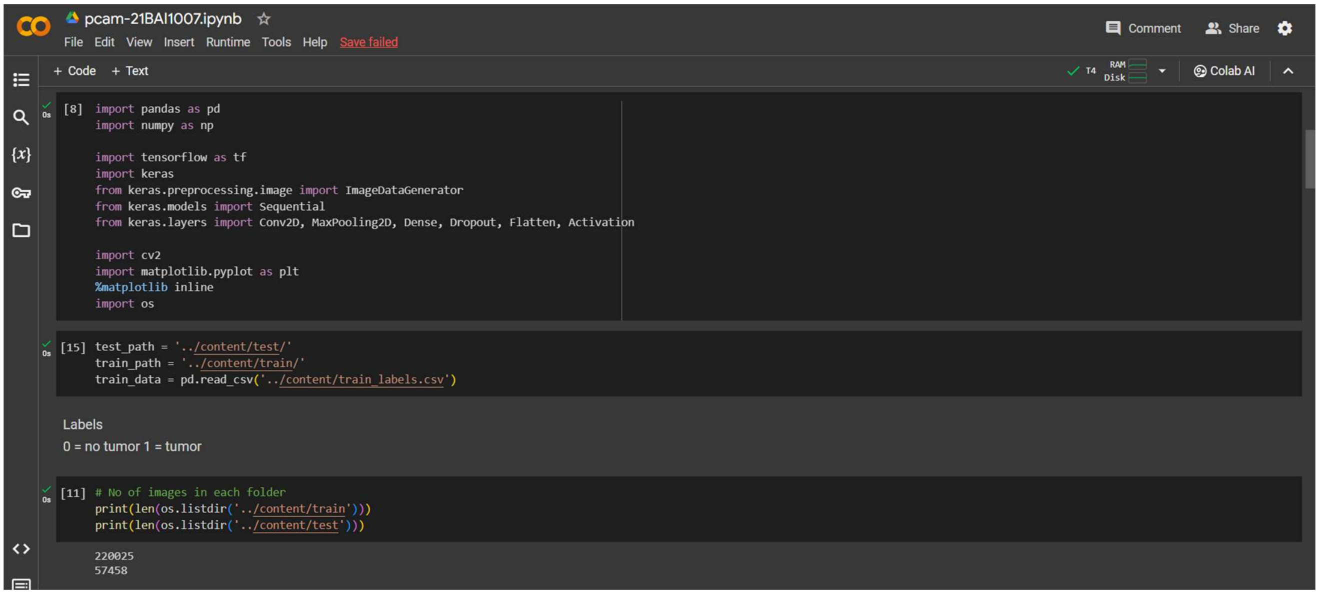
```
[1]  !mkdir -p ~/.kaggle
     !cp kaggle.json ~/.kaggle/
```

```
[3]  !kaggle competitions download -c histopathologic-cancer-detection
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading histopathologic-cancer-detection.zip to /content
100% 6.30G/6.31G [01:03<00:00, 196MB/s]
100% 6.31G/6.31G [01:03<00:00, 107MB/s]
```

Loading Dataset - 21BAI1007

```
[4]  import zipfile
     zip_ref = zipfile.ZipFile('/content/histopathologic-cancer-detection.zip', 'r')
     zip_ref.extractall('/content')
     zip_ref.close()
```



```
[8]  import pandas as pd
     import numpy as np

     import tensorflow as tf
     import keras
     from keras.preprocessing.image import ImageDataGenerator
     from keras.models import Sequential
     from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten, Activation

     import cv2
     import matplotlib.pyplot as plt
     %matplotlib inline
     import os
```

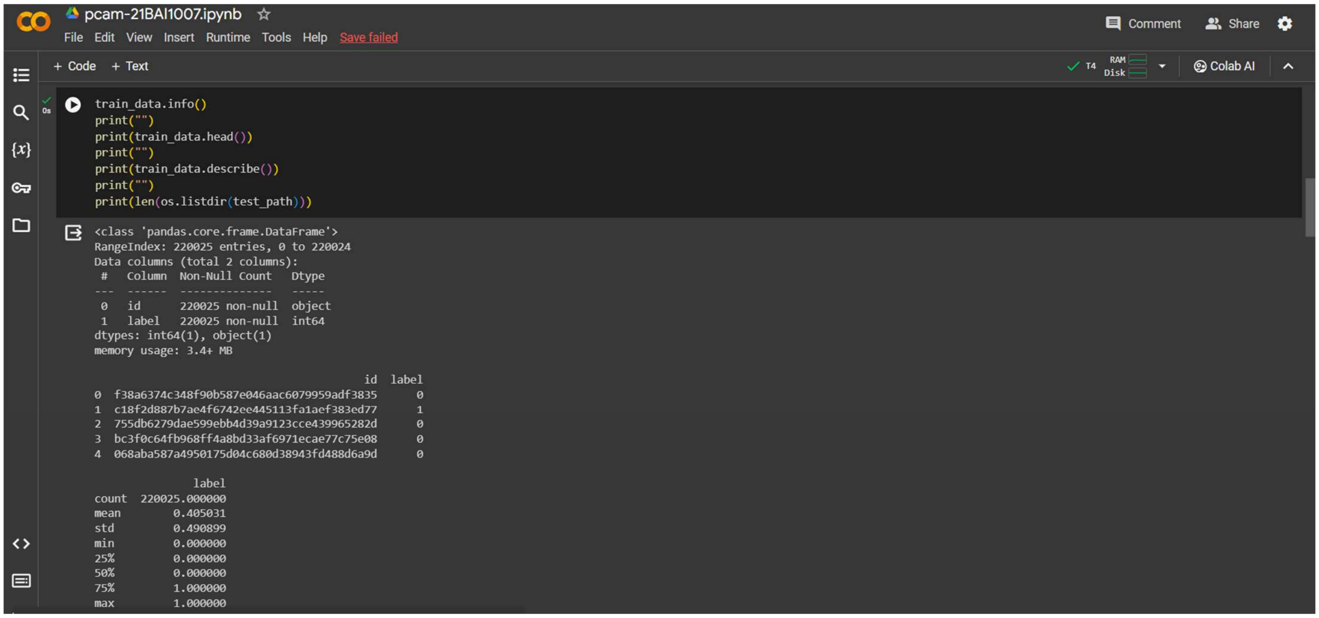
```
[15] test_path = '../content/test/'
     train_path = '../content/train/'
     train_data = pd.read_csv('../content/train_labels.csv')
```

Labels

0 = no tumor 1 = tumor

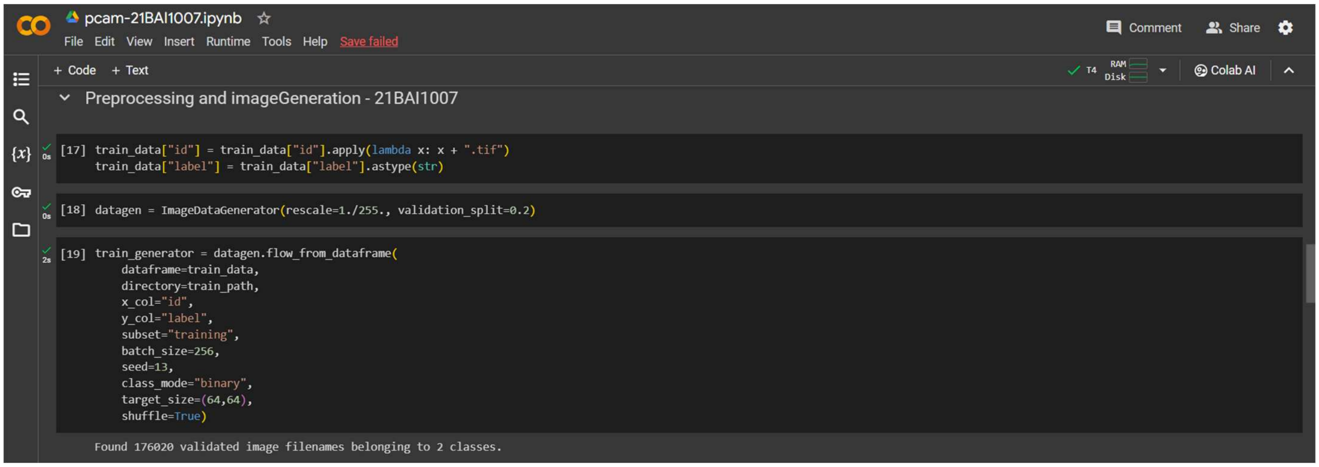
```
[11] # No of images in each folder
     print(len(os.listdir('../content/train')))
     print(len(os.listdir('../content/test')))
```

```
220025
57458
```



```
train_data.info()
print("")
print(train_data.head())
print("")
print(train_data.describe())
print("")
print(len(os.listdir(test_path)))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 220025 entries, 0 to 220024
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   id      220025 non-null  object
 1   label   220025 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 3.4+ MB

                                         id  label
0  f38a6374c348f90b587e046aac6079959adf3835      0
1  c18f2d887b7ae4f6742ee445113fa1aef383ed77      1
2  755db6279dae599ebb4d39a9123cce439965282d      0
3  bc3f0c64fb968ff4a8bd33af6971ecae77c75e08      0
4  068aba587a4950175d04c680d38943fd488d6a9d      0

               label
count  220025.000000
mean        0.405031
std         0.490899
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         1.000000
```



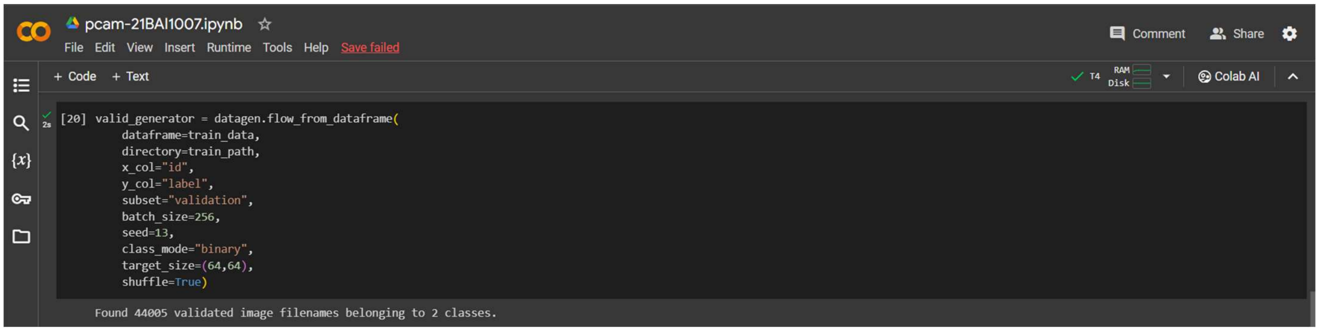
Preprocessing and imageGeneration - 21BAI1007

```
[17] train_data["id"] = train_data["id"].apply(lambda x: x + ".tif")
     train_data["label"] = train_data["label"].astype(str)
```

```
[18] datagen = ImageDataGenerator(rescale=1./255., validation_split=0.2)
```

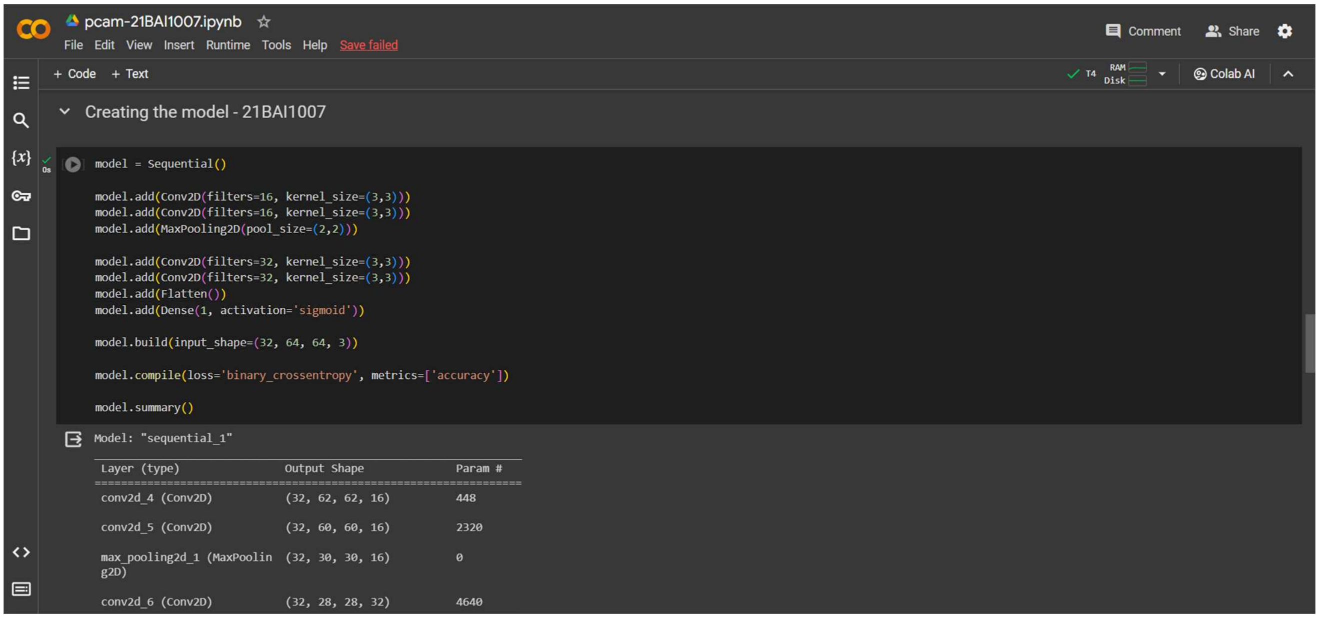
```
[19] train_generator = datagen.flow_from_dataframe(
         dataframe=train_data,
         directory=train_path,
         x_col="id",
         y_col="label",
         subset="training",
         batch_size=256,
         seed=13,
         class_mode="binary",
         target_size=(64,64),
         shuffle=True)
```

```
Found 176020 validated image filenames belonging to 2 classes.
```



```
[20] valid_generator = datagen.flow_from_dataframe(
         dataframe=train_data,
         directory=train_path,
         x_col="id",
         y_col="label",
         subset="validation",
         batch_size=256,
         seed=13,
         class_mode="binary",
         target_size=(64,64),
         shuffle=True)
```

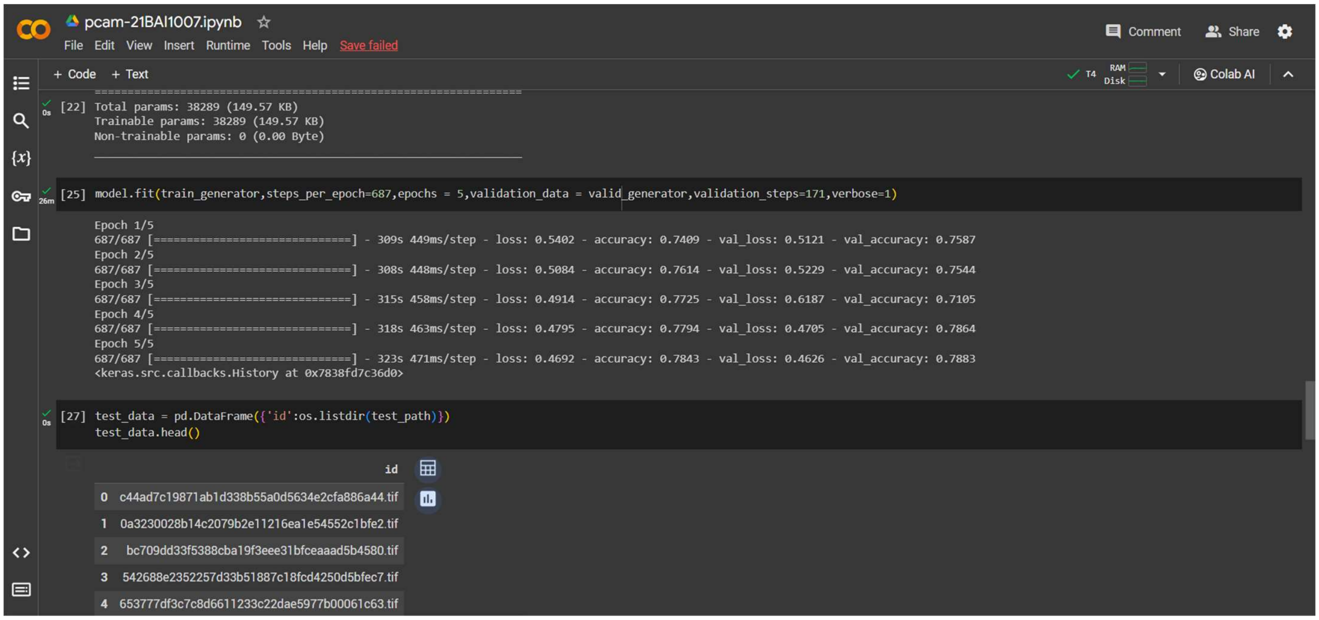
```
Found 44005 validated image filenames belonging to 2 classes.
```



Creating the model - 21BAI1007

```
model = Sequential()

model.add(Conv2D(filters=16, kernel_size=(3,3)))
model.add(Conv2D(filters=16, kernel_size=(3,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(3,3)))
model.add(Conv2D(filters=32, kernel_size=(3,3)))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))

model.build(input_shape=(32, 64, 64, 3))

model.compile(loss='binary_crossentropy', metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (32, 62, 62, 16)          448

 conv2d_5 (Conv2D)           (32, 60, 60, 16)          2320

 max_pooling2d_1 (MaxPoolin  (32, 30, 30, 16)          0
 g2D)

 conv2d_6 (Conv2D)           (32, 28, 28, 32)          4640
```



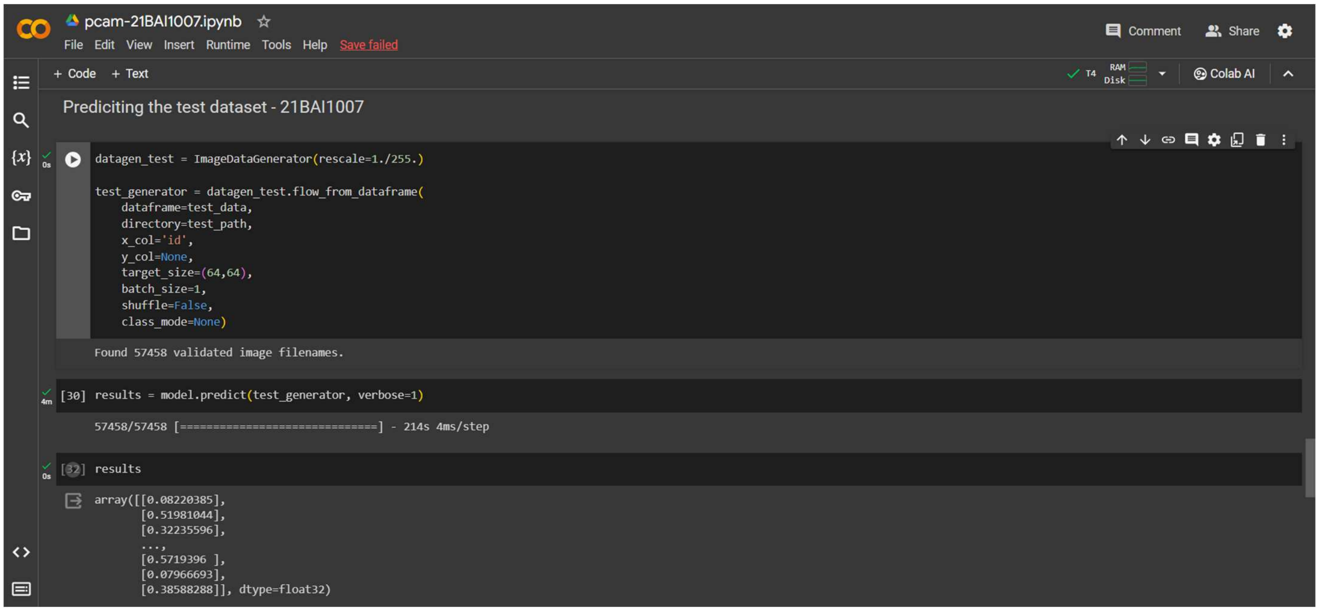
```
=================================================================
[22] Total params: 38289 (149.57 KB)
     Trainable params: 38289 (149.57 KB)
     Non-trainable params: 0 (0.00 Byte)
     _________________________________________________________________
```

```
[25] model.fit(train_generator,steps_per_epoch=687,epochs = 5,validation_data = valid_generator,validation_steps=171,verbose=1)
```

```
Epoch 1/5
687/687 [==============================] - 309s 449ms/step - loss: 0.5402 - accuracy: 0.7409 - val_loss: 0.5121 - val_accuracy: 0.7587
Epoch 2/5
687/687 [==============================] - 308s 448ms/step - loss: 0.5084 - accuracy: 0.7614 - val_loss: 0.5229 - val_accuracy: 0.7544
Epoch 3/5
687/687 [==============================] - 315s 458ms/step - loss: 0.4914 - accuracy: 0.7725 - val_loss: 0.6187 - val_accuracy: 0.7105
Epoch 4/5
687/687 [==============================] - 318s 463ms/step - loss: 0.4795 - accuracy: 0.7794 - val_loss: 0.4705 - val_accuracy: 0.7864
Epoch 5/5
687/687 [==============================] - 323s 471ms/step - loss: 0.4692 - accuracy: 0.7843 - val_loss: 0.4626 - val_accuracy: 0.7883
<keras.src.callbacks.History at 0x7838fd7c36d0>
```

```
[27] test_data = pd.DataFrame({'id':os.listdir(test_path)})
     test_data.head()
```

	id
0	c44ad7c19871ab1d338b55a0d5634e2cfa886a44.tif
1	0a3230028b14c2079b2e11216ea1e54552c1bfe2.tif
2	bc709dd33f5388cba19f3eee31bfceaaad5b4580.tif
3	542688e2352257d33b51887c18fcd4250d5bfec7.tif
4	653777df3c7c8d6611233c22dae5977b00061c63.tif



Prediciting the test dataset - 21BAI1007

```
datagen_test = ImageDataGenerator(rescale=1./255.)

test_generator = datagen_test.flow_from_dataframe(
    dataframe=test_data,
    directory=test_path,
    x_col='id',
    y_col=None,
    target_size=(64,64),
    batch_size=1,
    shuffle=False,
    class_mode=None)
```

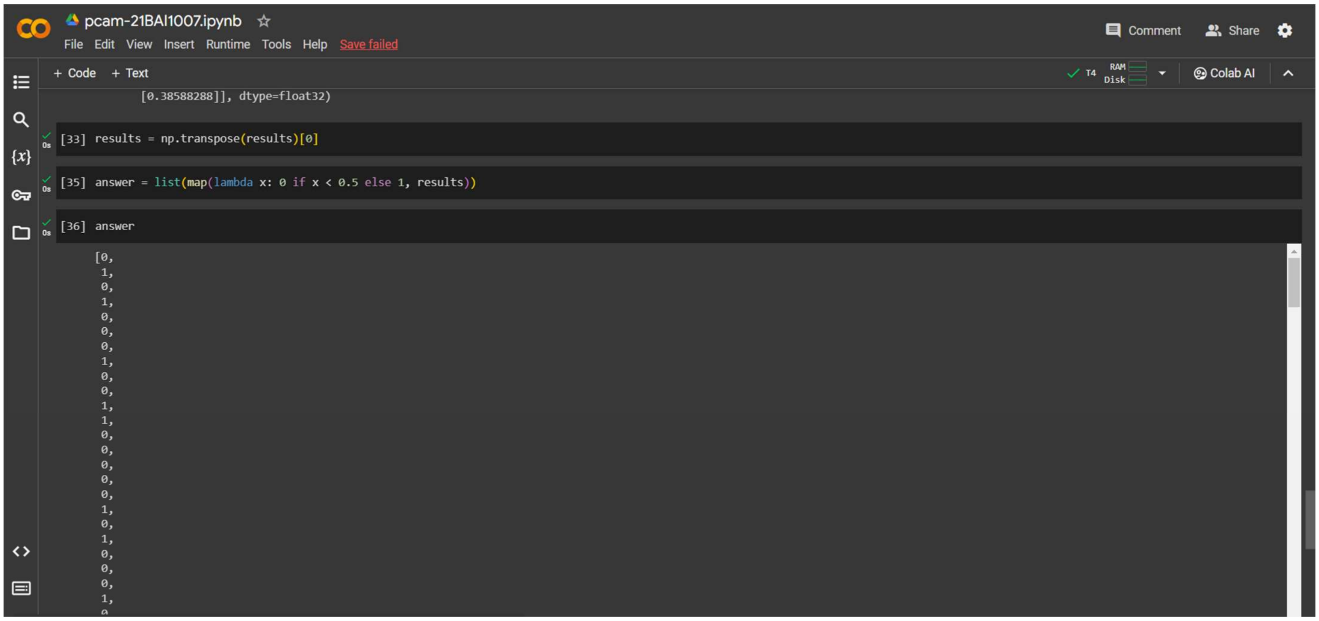
```
Found 57458 validated image filenames.
```

```
[30] results = model.predict(test_generator, verbose=1)
```

```
57458/57458 [==============================] - 214s 4ms/step
```

```
[32] results
```

```
array([[0.08220385],
       [0.51981044],
       [0.32235596],
       ...,
       [0.5719396 ],
       [0.07966693],
       [0.38588288]], dtype=float32)
```



```
              [0.38588288]], dtype=float32)

[33] results = np.transpose(results)[0]

[35] answer = list(map(lambda x: 0 if x < 0.5 else 1, results))

[36] answer

    [0,
     1,
     0,
     1,
     0,
     0,
     0,
     1,
     0,
     0,
     1,
     1,
     0,
     0,
     0,
     0,
     0,
     1,
     0,
     1,
     0,
     0,
     0,
     1,
     0
```

4.Import Cat and Dog Dataset for classification

Code

```
!mkdir -p ~/.kaggle

!cp kaggle.json ~/.kaggle/


!kaggle datasets download -d salader/dogs-vs-cats


# Data Loaded from kaggle is in zip format. Need to unzip it

import zipfile

zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip', 'r')

zip_ref.extractall('/content')

zip_ref.close()


"""### Loading the data - 21BAI1007"""


import tensorflow as tf

import keras

from keras.models import Sequential

from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, BatchNormalization, Dropout


train = keras.utils.image_dataset_from_directory(
    directory = '/content/train',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)

validation = keras.utils.image_dataset_from_directory(
    directory = '/content/test',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)

## Normalizing dataset values to a value between 0 and 1
def process(image,label):
    image = tf.cast(image/255. ,tf.float32)
    return image,label

train = train.map(process)
validation = validation.map(process)

"""### Creating the model - 21BAI1007"""

model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())

model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))

model.summary()

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

model.fit(train,epochs=10,validation_data=validation)

loss, acc = model.evaluate(validation)

"""### Predicting a random image - 21BAI1007"""

import cv2
import matplotlib.pyplot as plt
test_image = cv2.imread('/content/cat.jpg')

plt.imshow(test_image)

test_image = cv2.resize(test_image,(256,256))

test_input = test_image.reshape((1,256,256,3))

model.predict(test_input)

"""Array[0] -> Cat <br>
Hence, the model has correctly predicted the given image of a cat
"""

test_image2 = cv2.imread('/content/download.jpeg')

plt.imshow(test_image2)

test_image2 = cv2.resize(test_image2, (256, 256))
```

test_input2 = test_image2.reshape((1, 256, 256, 3))

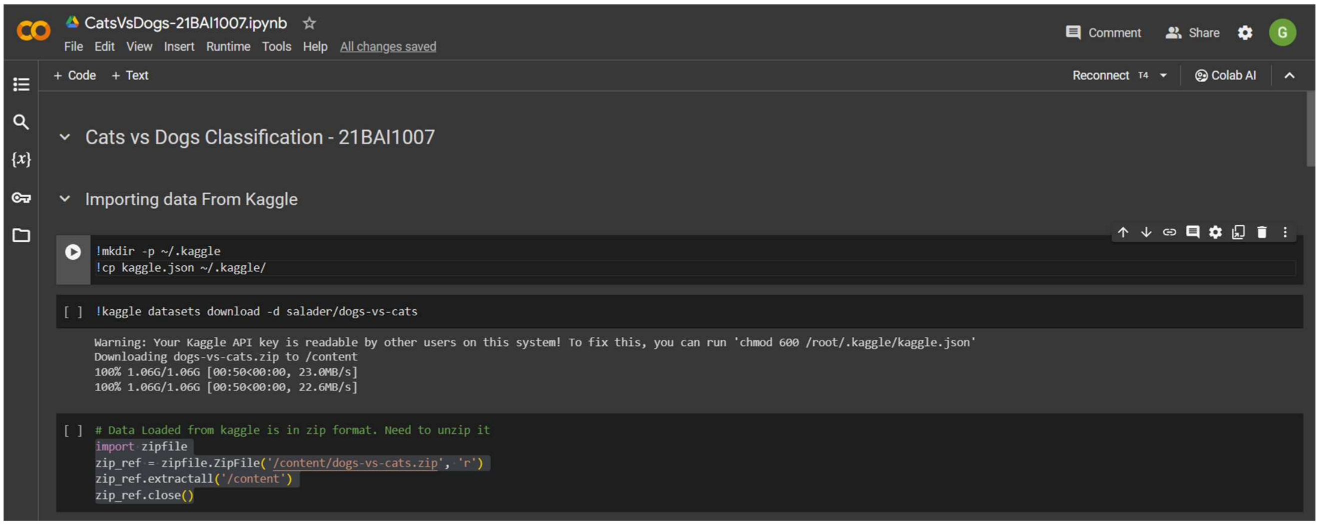
model.predict(test_input2)

"""Array[1] -> Dog <br>

Hence the model has successfully predicted both the images of a cat and a dog

"""

Output


CatsVsDogs-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect T4
Colab AI

Cats vs Dogs Classification - 21BAI1007

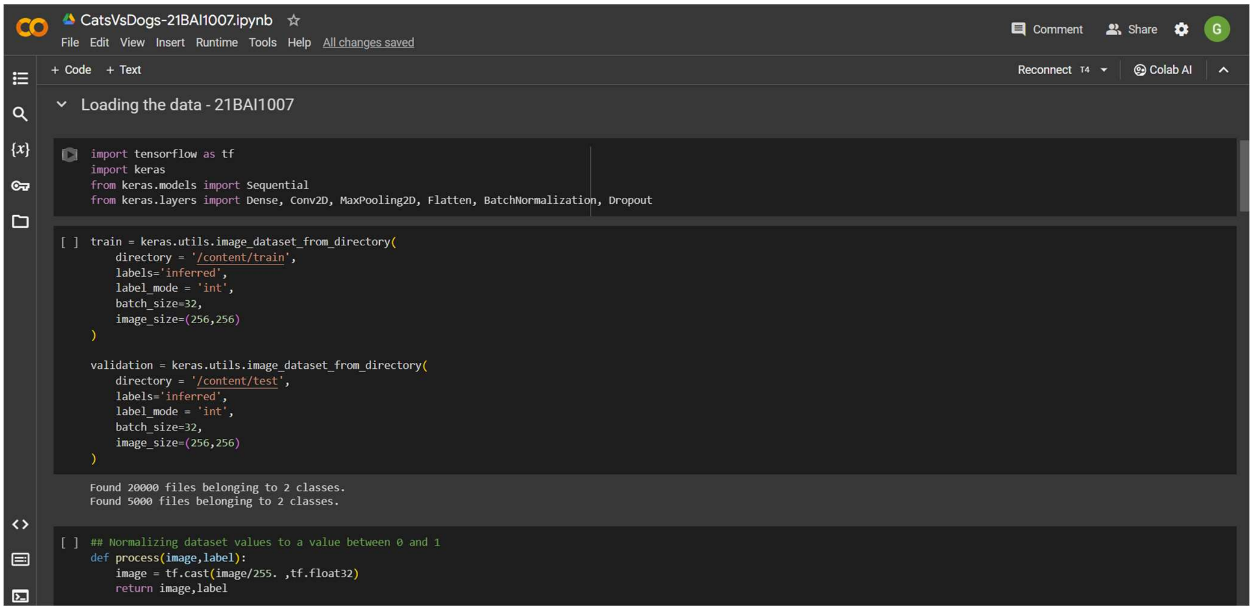
Importing data From Kaggle

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
[ ] !kaggle datasets download -d salader/dogs-vs-cats
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Downloading dogs-vs-cats.zip to /content
100% 1.06G/1.06G [00:50<00:00, 23.0MB/s]
100% 1.06G/1.06G [00:50<00:00, 22.6MB/s]
```

```
[ ] # Data Loaded from kaggle is in zip format. Need to unzip it
    import zipfile
    zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip', 'r')
    zip_ref.extractall('/content')
    zip_ref.close()
```


CatsVsDogs-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect T4
Colab AI

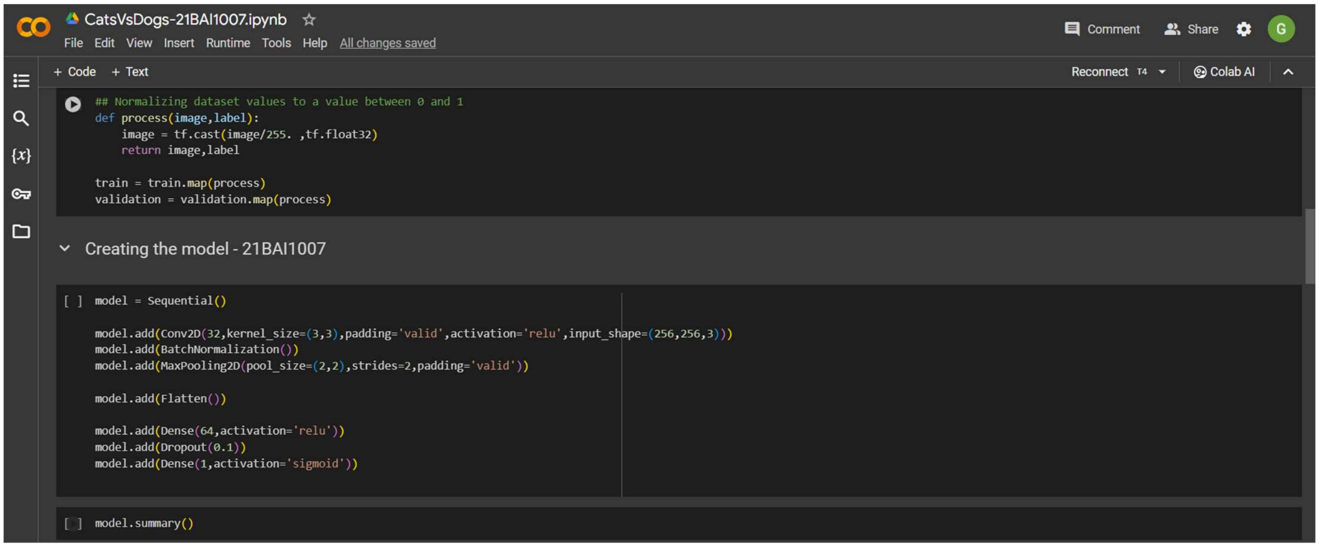
Loading the data - 21BAI1007

```
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, BatchNormalization, Dropout
```

```
[ ] train = keras.utils.image_dataset_from_directory(
        directory = '/content/train',
        labels='inferred',
        label_mode = 'int',
        batch_size=32,
        image_size=(256,256)
    )

    validation = keras.utils.image_dataset_from_directory(
        directory = '/content/test',
        labels='inferred',
        label_mode = 'int',
        batch_size=32,
        image_size=(256,256)
    )
```

```
Found 20000 files belonging to 2 classes.
Found 5000 files belonging to 2 classes.
```

```
[ ] ## Normalizing dataset values to a value between 0 and 1
    def process(image,label):
        image = tf.cast(image/255. ,tf.float32)
        return image,label
```

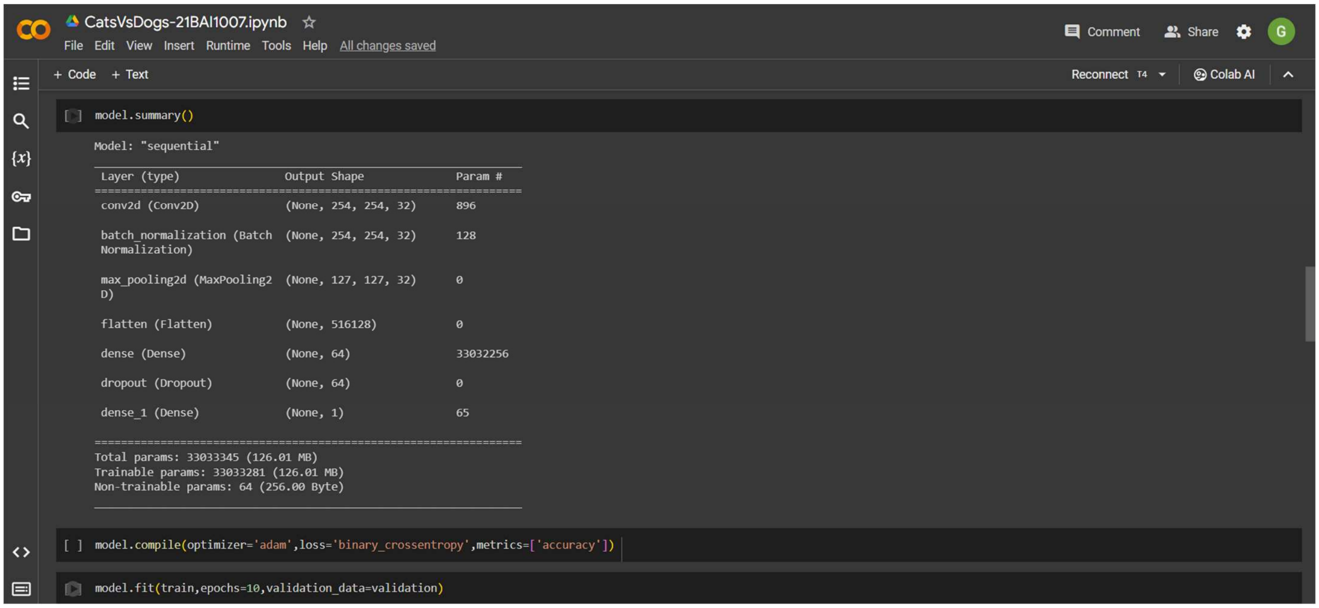

CatsVsDogs-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect T4
Colab AI

```
## Normalizing dataset values to a value between 0 and 1
def process(image,label):
    image = tf.cast(image/255. ,tf.float32)
    return image,label

train = train.map(process)
validation = validation.map(process)
```

Creating the model - 21BAI1007

```
model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(256,256,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())

model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))
```

```
model.summary()
```

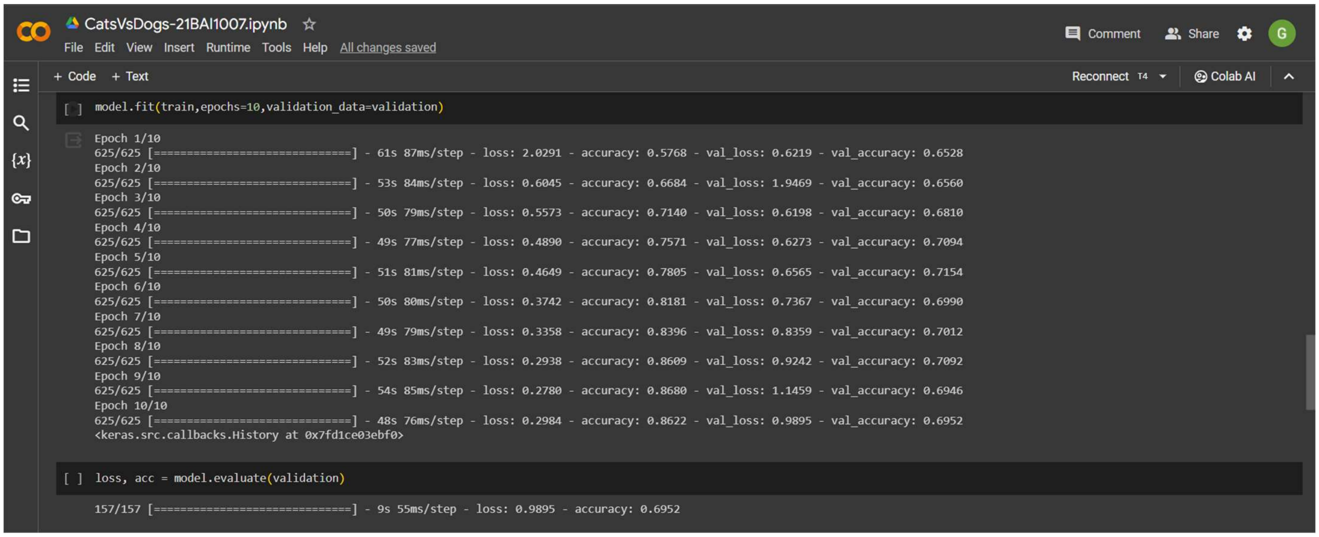

CatsVsDogs-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect T4
Colab AI

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 batch_normalization (Batch  (None, 254, 254, 32)      128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 127, 127, 32)      0
 D)

 flatten (Flatten)           (None, 516128)            0

 dense (Dense)               (None, 64)                33032256

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 33033345 (126.01 MB)
Trainable params: 33033281 (126.01 MB)
Non-trainable params: 64 (256.00 Byte)
_________________________________________________________________
```

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

```
model.fit(train,epochs=10,validation_data=validation)
```

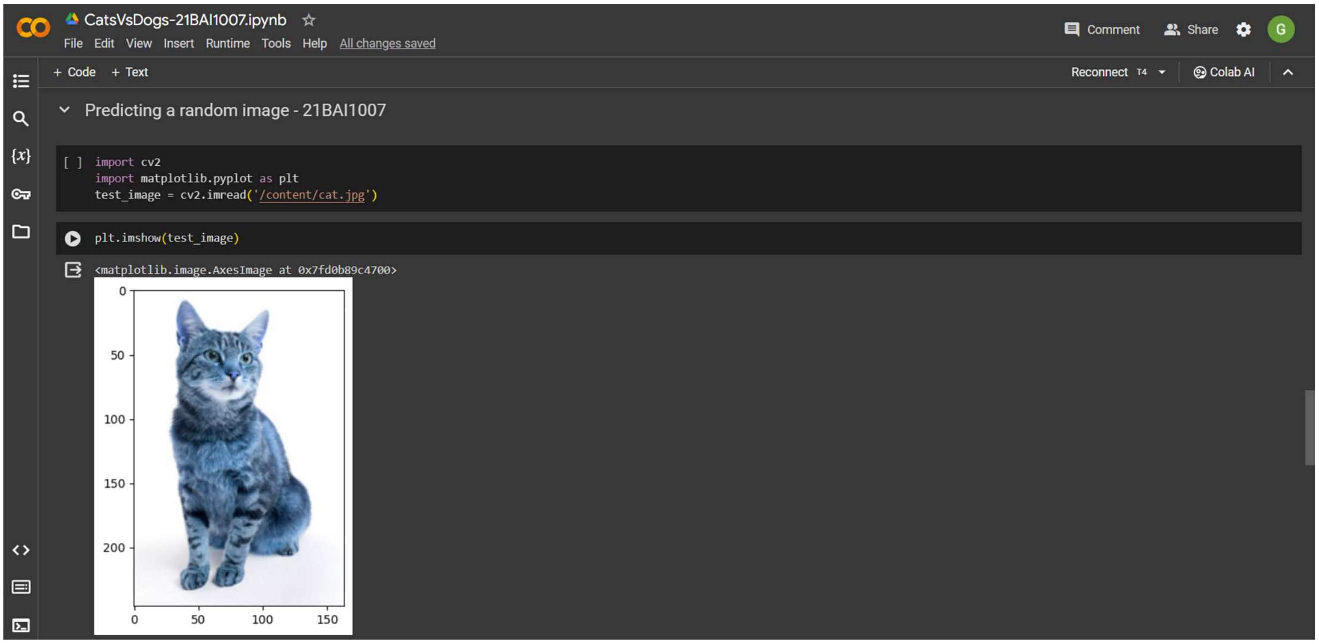

CatsVsDogs-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect T4
Colab AI

```
model.fit(train,epochs=10,validation_data=validation)
```

```
Epoch 1/10
625/625 [==============================] - 61s 87ms/step - loss: 2.0291 - accuracy: 0.5768 - val_loss: 0.6219 - val_accuracy: 0.6528
Epoch 2/10
625/625 [==============================] - 53s 84ms/step - loss: 0.6045 - accuracy: 0.6684 - val_loss: 1.9469 - val_accuracy: 0.6560
Epoch 3/10
625/625 [==============================] - 50s 79ms/step - loss: 0.5573 - accuracy: 0.7140 - val_loss: 0.6198 - val_accuracy: 0.6810
Epoch 4/10
625/625 [==============================] - 49s 77ms/step - loss: 0.4890 - accuracy: 0.7571 - val_loss: 0.6273 - val_accuracy: 0.7094
Epoch 5/10
625/625 [==============================] - 51s 81ms/step - loss: 0.4649 - accuracy: 0.7805 - val_loss: 0.6565 - val_accuracy: 0.7154
Epoch 6/10
625/625 [==============================] - 50s 80ms/step - loss: 0.3742 - accuracy: 0.8181 - val_loss: 0.7367 - val_accuracy: 0.6990
Epoch 7/10
625/625 [==============================] - 49s 79ms/step - loss: 0.3358 - accuracy: 0.8396 - val_loss: 0.8359 - val_accuracy: 0.7012
Epoch 8/10
625/625 [==============================] - 52s 83ms/step - loss: 0.2938 - accuracy: 0.8609 - val_loss: 0.9242 - val_accuracy: 0.7092
Epoch 9/10
625/625 [==============================] - 54s 85ms/step - loss: 0.2780 - accuracy: 0.8680 - val_loss: 1.1459 - val_accuracy: 0.6946
Epoch 10/10
625/625 [==============================] - 48s 76ms/step - loss: 0.2984 - accuracy: 0.8622 - val_loss: 0.9895 - val_accuracy: 0.6952
<keras.src.callbacks.History at 0x7fd1ce03ebf0>
```

```
loss, acc = model.evaluate(validation)
```

```
157/157 [==============================] - 9s 55ms/step - loss: 0.9895 - accuracy: 0.6952
```

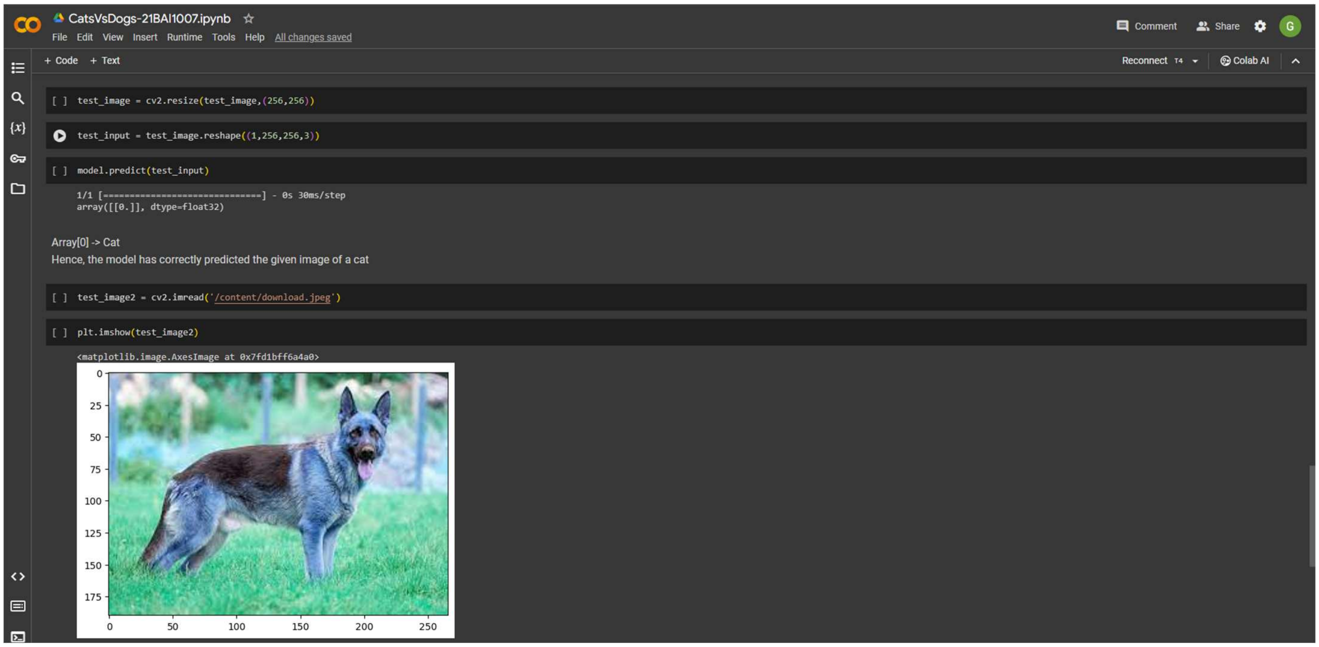

CatsVsDogs-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
G
+ Code + Text
Reconnect T4
Colab AI
0
50
100
150
200
0
50
100
150

Predicting a random image - 21BAI1007

```
import cv2
import matplotlib.pyplot as plt
test_image = cv2.imread('/content/cat.jpg')
```

```
plt.imshow(test_image)
```

```
<matplotlib.image.AxesImage at 0x7fd0b89c4700>
```


CatsVsDogs-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
G
+ Code + Text
Reconnect T4
Colab AI
0
25
50
75
100
125
150
175
0
50
100
150
200
250

```
test_image = cv2.resize(test_image,(256,256))
```

```
test_input = test_image.reshape((1,256,256,3))
```

```
model.predict(test_input)
```

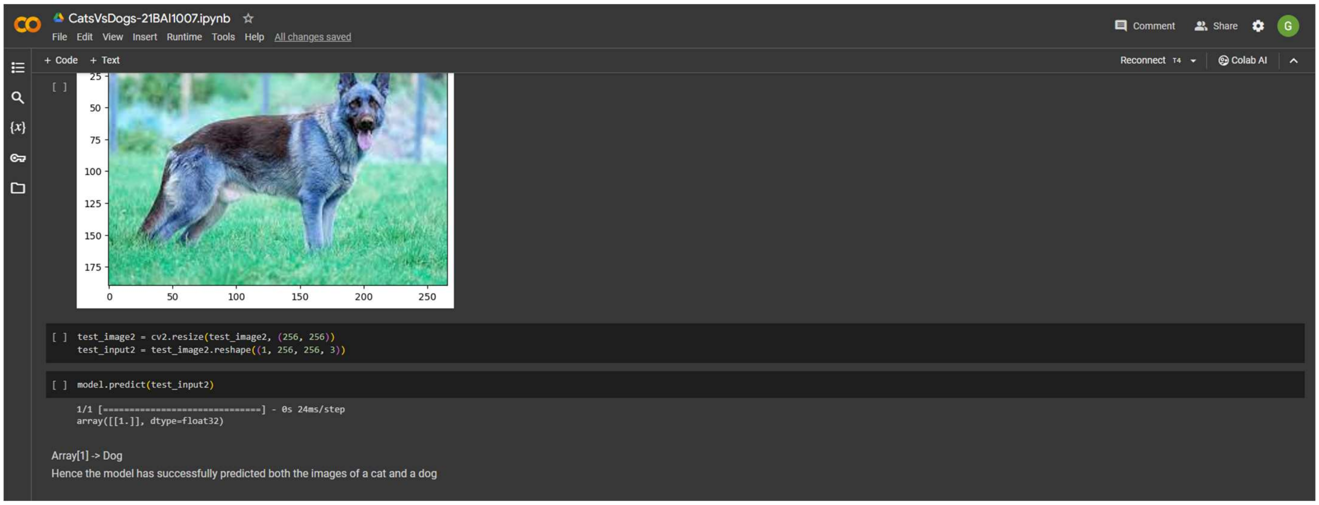
```
1/1 [==============================] - 0s 30ms/step
array([[0.]], dtype=float32)
```

Array[0] -> Cat
Hence, the model has correctly predicted the given image of a cat

```
test_image2 = cv2.imread('/content/download.jpeg')
```

```
plt.imshow(test_image2)
```

```
<matplotlib.image.AxesImage at 0x7fd1bff6a4a0>
```


CatsVsDogs-21BAI1007.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
G
+ Code + Text
Reconnect T4
Colab AI
25
50
75
100
125
150
175
0
50
100
150
200
250

```
test_image2 = cv2.resize(test_image2, (256, 256))
test_input2 = test_image2.reshape((1, 256, 256, 3))
```

```
model.predict(test_input2)
```

```
1/1 [==============================] - 0s 24ms/step
array([[1.]], dtype=float32)
```

Array[1] -> Dog
Hence the model has successfully predicted both the images of a cat and a dog