Building a Brain Tumor Detection Model on Google Colab

A Step-by-Step Guide from Dataset Download to Model Training

NIELIT Chandigarh/Ropar

"Big Data is at the foundation of all the megatrends that are happening today."

Objective

- Download the dataset from Kaggle.
- Upload files to Google Drive and access them in Google Colab.
- Create and train a convolutional neural network (CNN) on the dataset.

Prerequisites

1. Kaggle (for downloading the dataset).
2. Google account (for Google Colab and Drive).
3. Basic understanding of Python and TensorFlow

Download Dataset from Kaggle

- **Steps 1**:

1. Go to the Kaggle dataset link: **Brain Tumor Dataset - Yes/No Class.**

https://www.kaggle.com/datasets/princelv84/brain-tumor-dataset-yesno-class

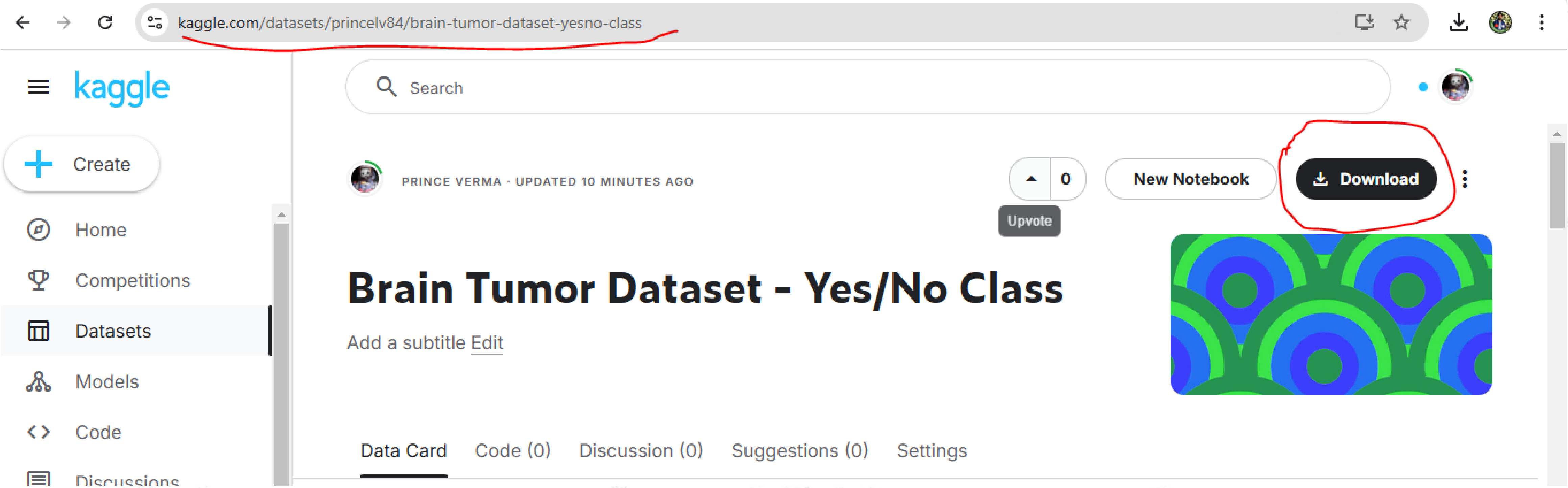
- **Step 2**:

1. Click on the **Download** button.
2. Save the .zip file to your local machine.

Download Dataset from Kaggle

Screenshot:

https://www.kaggle.com/datasets/princelv84/brain-tumor-dataset-yesno-class



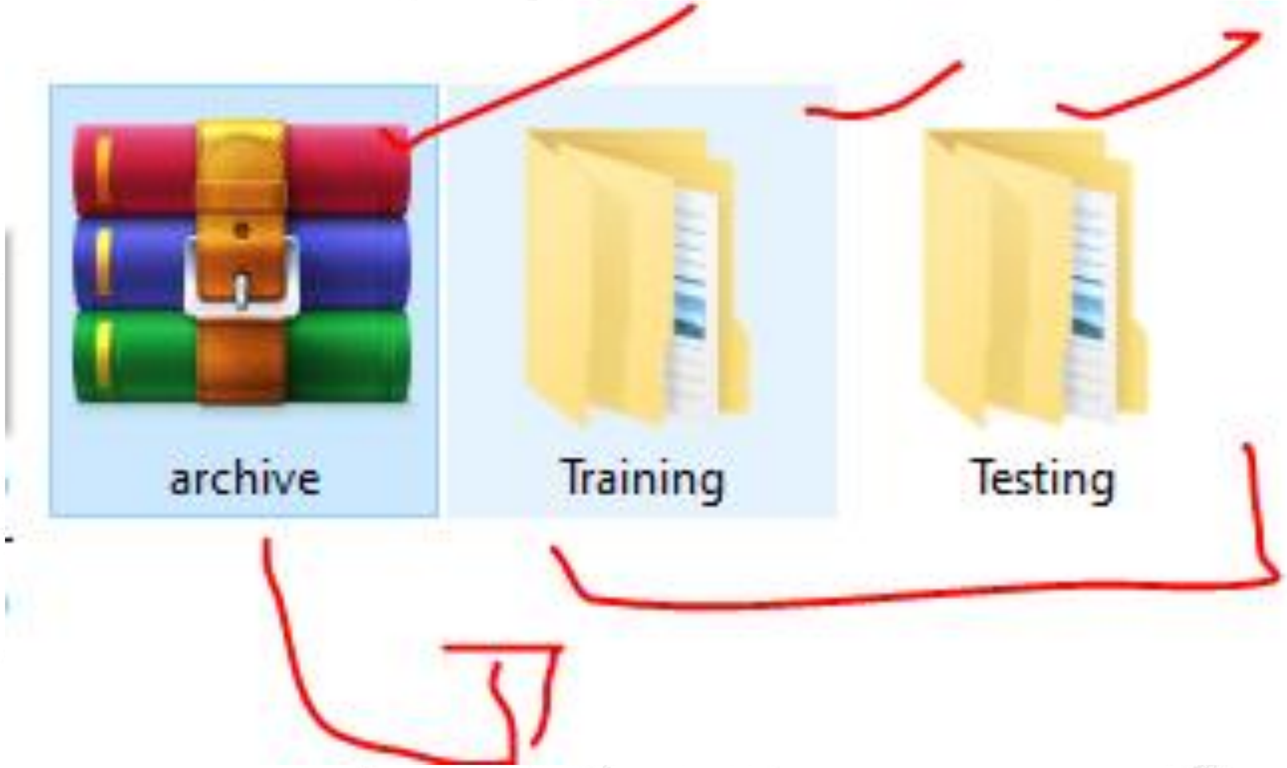
kaggle.com/datasets/princelv84/brain-tumor-dataset-yesno-class

Brain Tumor Dataset - Yes/No Class

Add a subtitle Edit

Extract the Dataset

- Use winrar or 7zip to extract that zip we just downloaded from Kaggle.

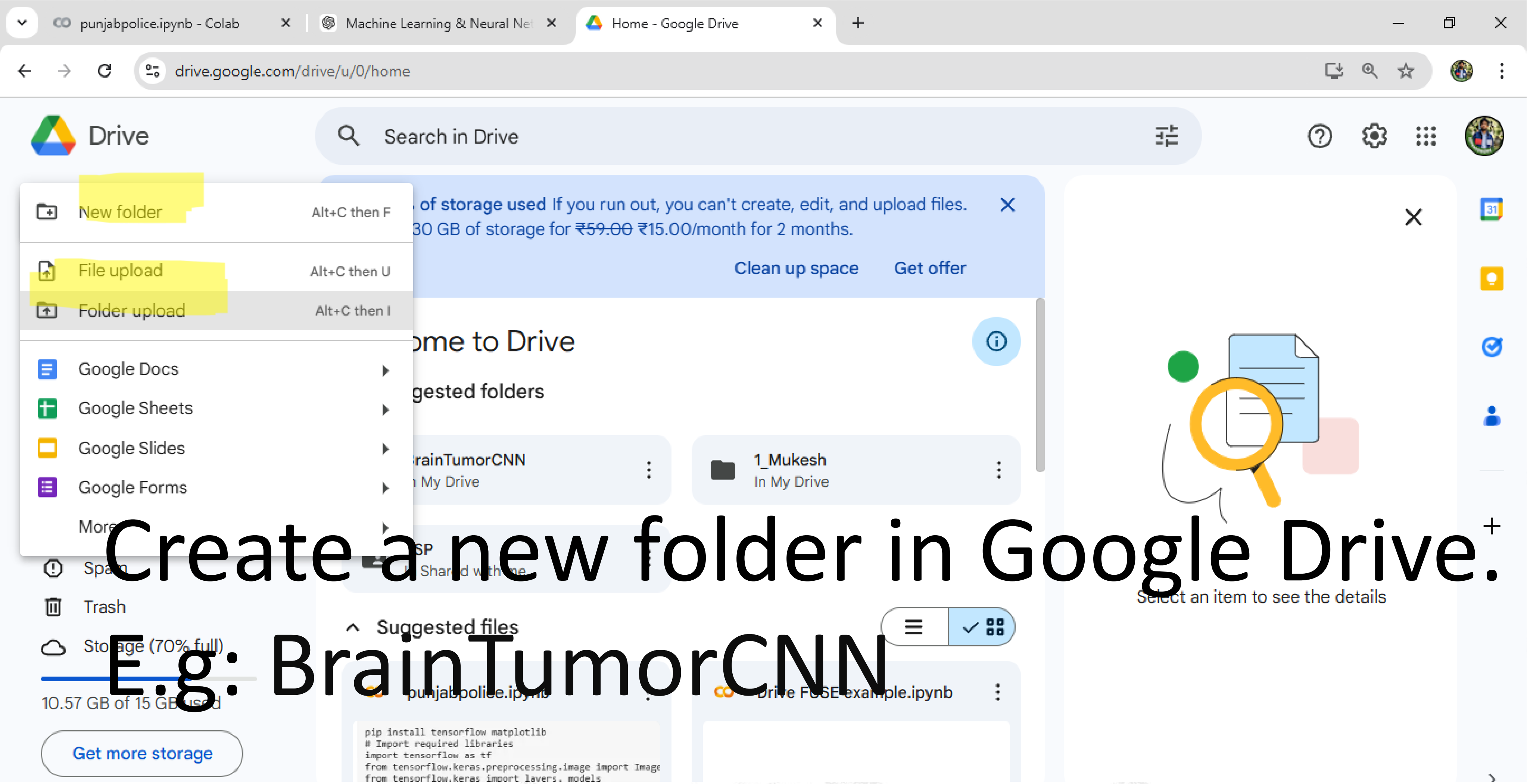


Upload Dataset to Google Drive

- **Steps**:

1. Open Google Drive.

Create a folder named BrainTumorCNN.



```
pip install tensorflow matplotlib
# Import required libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.image import Image
from tensorflow.keras import layers, models
```

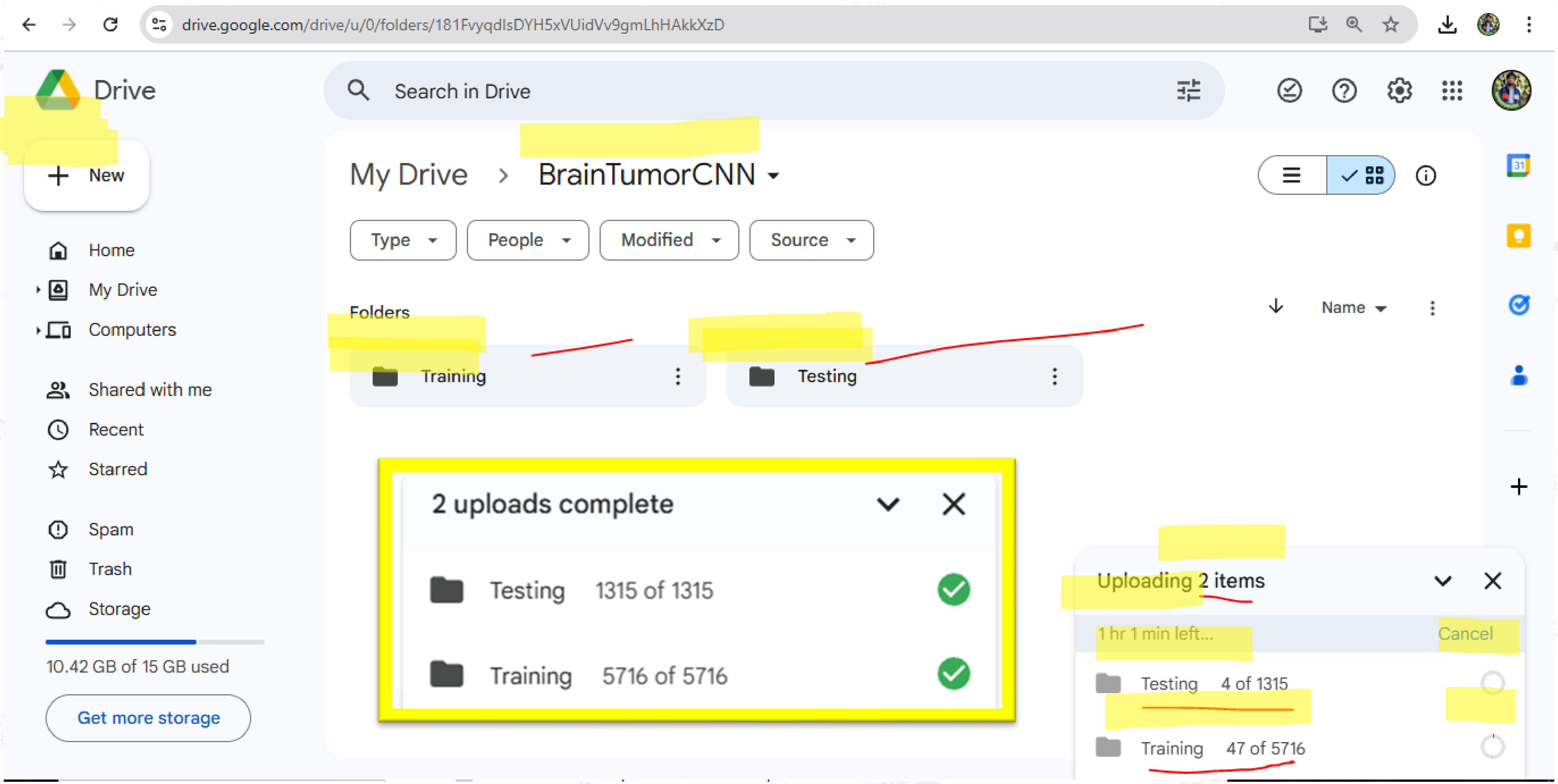
Create a new folder in Google Drive.

E.g: BrainTumorCNN

Upload Dataset to Drive

- Open that newly **created folder .**
- Then click on **New** again this this click on **folder upload.**
- Upload both **Testing** and **Training** folders we just extracted in our local pc to google drive inside that folder. To do So Drag and drop both Testing and Training folders we just extracted in our local pc to google drive inside that Drive folder. E.g: BrainTumorCNN is the name of that Drive folder **in my case**.
- You must Upload both Testing and Training folders **One by One**.
- Now **wait** for them to upload.

Upload Dataset to Drive



drive.google.com/drive/u/0/folders/181FvyqdlsDYH5xVUidVv9gmLhHAkkXzD

My Drive > BrainTumorCNN

Folders

Training

Testing

2 uploads complete

Testing 1315 of 1315 ✓

Training 5716 of 5716 ✓

Uploading 2 items

1 hr 1 min left... Cancel

Testing 4 of 1315

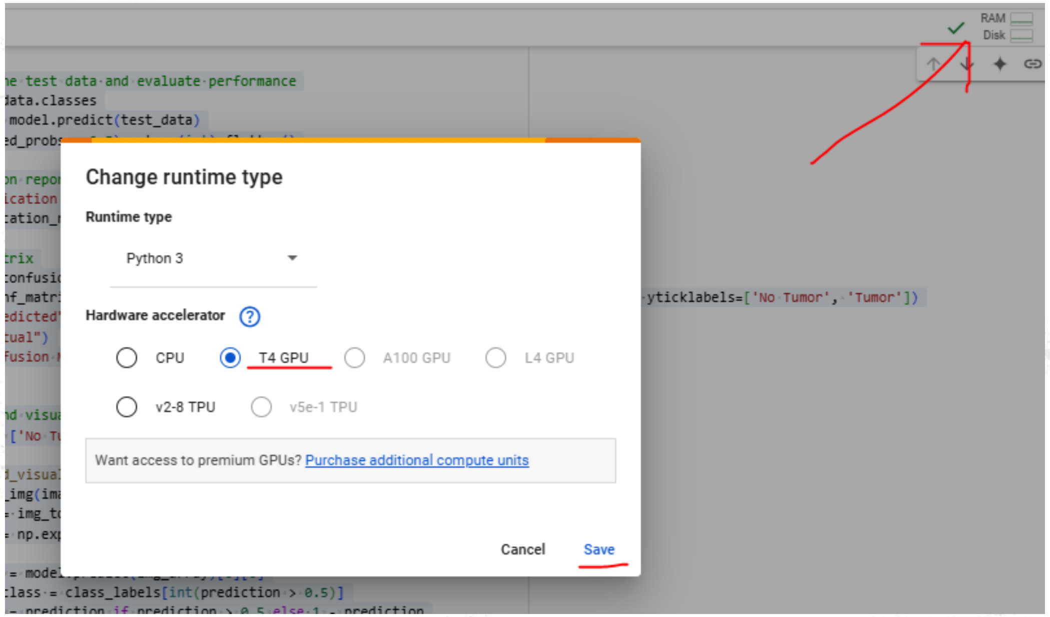
Training 47 of 5716

10.42 GB of 15 GB used

Setup in Google Colab

- **Steps**:

1. Open Google Colab.
2. Create a new notebook.
3. Set runtime type to T4 GPU
4. Mount Google Drive:



Change runtime type

Runtime type

Python 3

Hardware accelerator

○ CPU ● T4 GPU ○ A100 GPU ○ L4 GPU

○ v2-8 TPU ○ v5e-1 TPU

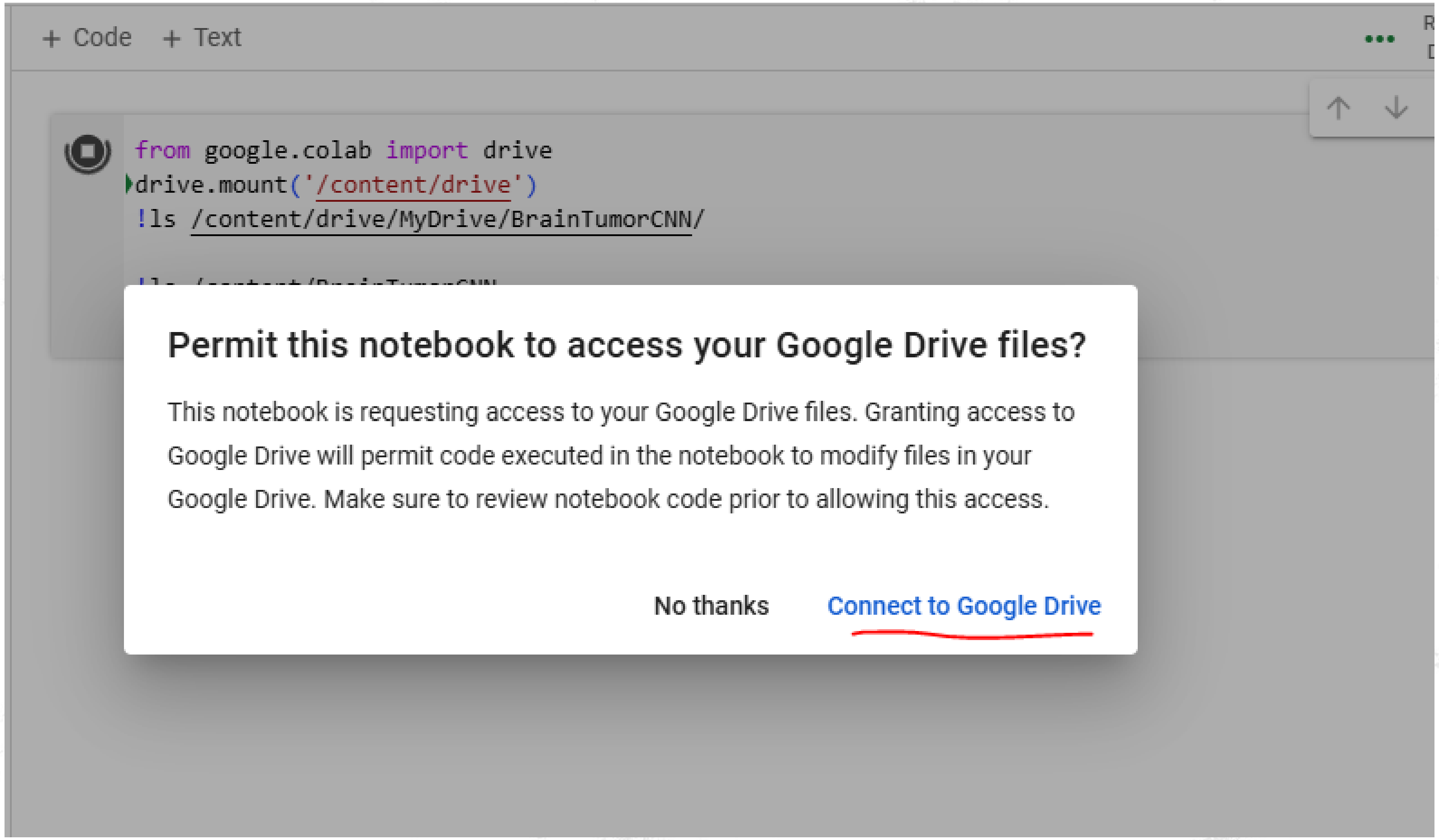
Want access to premium GPUs? Purchase additional compute units

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

- Navigate to the folder where the dataset is located:

```
!ls /content/drive/MyDrive/BrainTumorCNN/
```

Permit notebook access to Your Google Drive

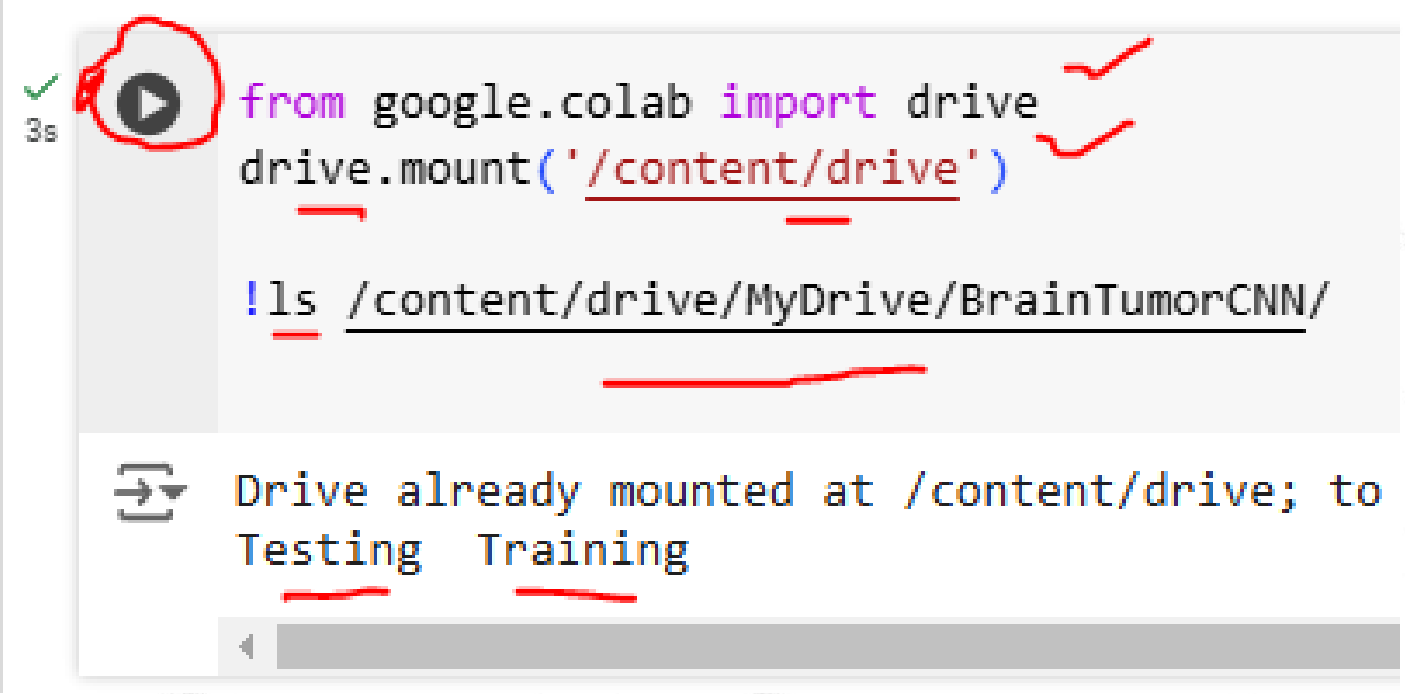


```
from google.colab import drive
drive.mount('/content/drive')
!ls /content/drive/MyDrive/BrainTumorCNN/

!ls /content/BrainTumorCNN
```

Permit this notebook to access your Google Drive files?

This notebook is requesting access to your Google Drive files. Granting access to Google Drive will permit code executed in the notebook to modify files in your Google Drive. Make sure to review notebook code prior to allowing this access.

Ensuring Drive Connection



```
from google.colab import drive
drive.mount('/content/drive')

!ls /content/drive/MyDrive/BrainTumorCNN/
```

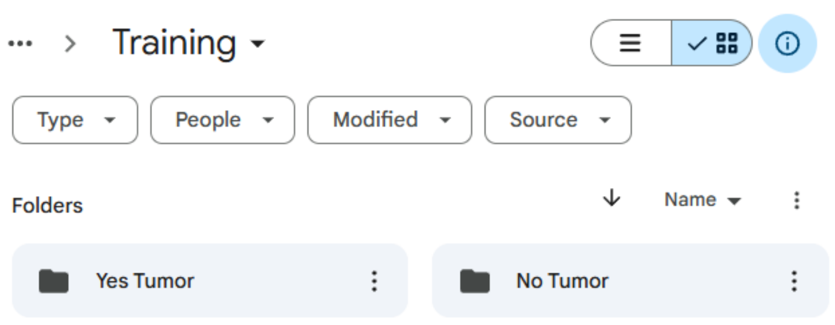
```
Drive already mounted at /content/drive; to
Testing  Training
```

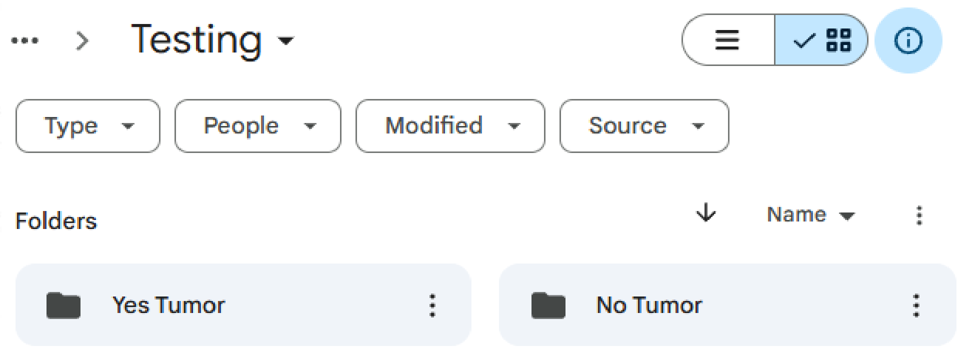
Dataset Preparation

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive’)

# Import necessary libraries
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import load_img, img_to_array
from tensorflow.keras import layers, models
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
```

Dataset paths

```
/content/drive/MyDrive/BrainTumorCNN/
├── Training/
│   ├── Tumor/
│   │   ├── image1.jpg
│   │   ├── image2.jpg
│   │   └── ...
│   ├── NoTumor/
│       ├── image1.jpg
│       ├── image2.jpg
│       └── ...
└── Testing/
    ├── Tumor/
    │   ├── image1.jpg
    │   ├── image2.jpg
    │   └── ...
    ├── NoTumor/
        ├── image1.jpg
        ├── image2.jpg
        └── ...
```





```
# Dataset paths

train_dir = '/content/drive/MyDrive/BrainTumorCNN/Training'

test_dir = '/content/drive/MyDrive/BrainTumorCNN/Testing’
# Verify the folder structure
print("Training folders:", os.listdir(train_dir))
print("Testing folders:", os.listdir(test_dir))
```

Data generators for training and testing

```
# Data generators for training and testing
train_datagen = ImageDataGenerator(rescale=1.0/255.0)
test_datagen = ImageDataGenerator(rescale=1.0/255.0)

# Load data from the organized folders
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary’
)

test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary',
    shuffle=False  # For evaluation consistency
)
```

Define the CNN model

```
# Define the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')  # Binary classification
])
```

Compile / Train/ Save the model

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

# Train the model
history = model.fit(train_data, validation_data=test_data, epochs=10)

# Evaluate the model
test_loss, test_acc = model.evaluate(test_data, verbose=2)
print(f"Test Accuracy: {test_acc:.2f}")

# Save the model
model_save_path = '/content/drive/MyDrive/braintumor_binary.h5'
model.save(model_save_path)
print(f"Model saved successfully at {model_save_path}!")
```

Load the model (if needed later)

1. **Save the model:**

```
model.save('/content/drive/MyDrive/braintumor.h5')
print("Model saved successfully!")
```

2. **Load the model later:**

```
from tensorflow.keras.models import load_model
model =
load_model('/content/drive/MyDrive/braintumor.h5')
```

Visualize Training Results

```
# Plot accuracy and loss graphs
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title("Accuracy over Epochs")
plt.show()

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title("Loss over Epochs")
plt.show()

# Predict on the test data and evaluate performance
y_true = test_data.classes
y_pred_probs = model.predict(test_data)
y_pred = (y_pred_probs > 0.5).astype(int).flatten()
```

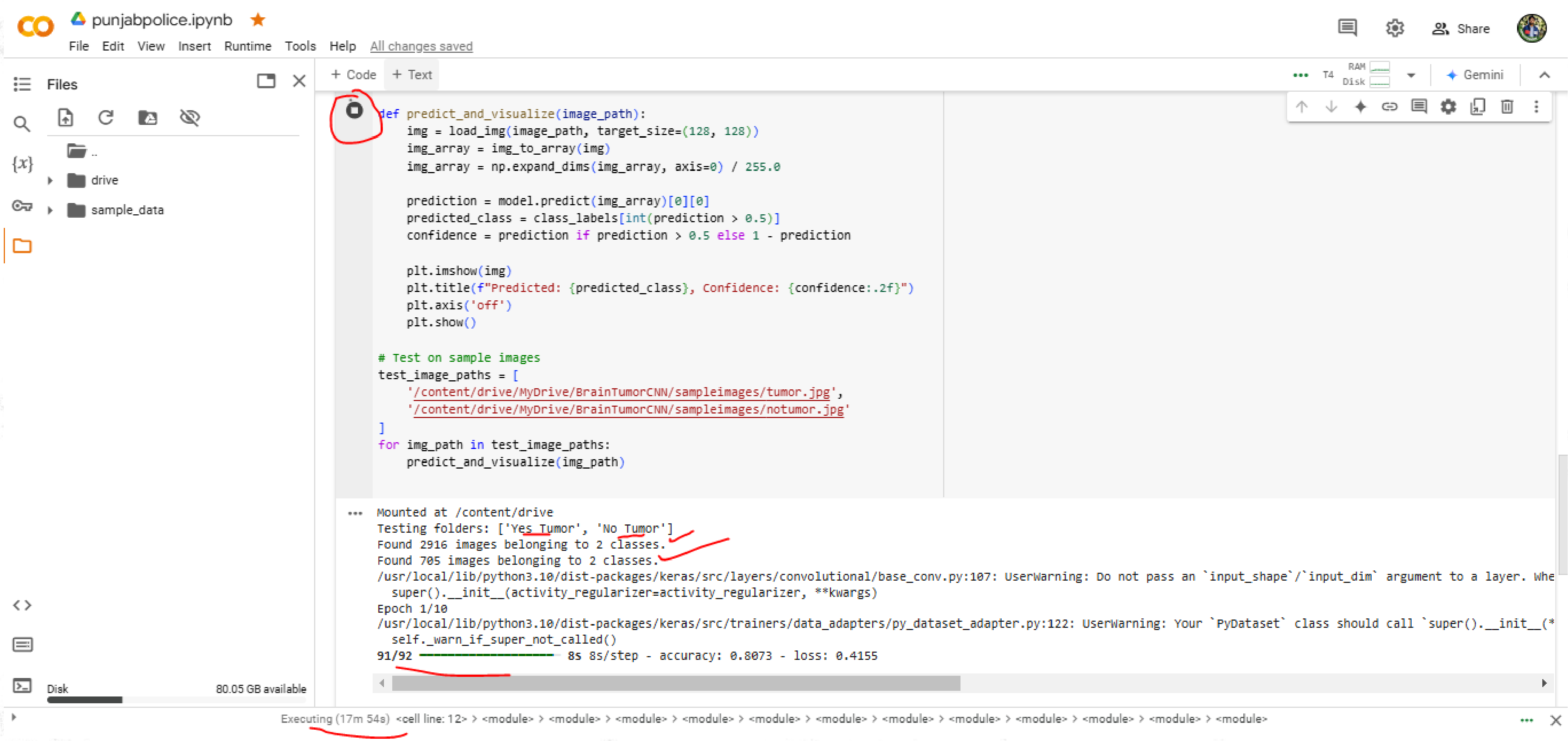
```
# Classification report
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=['No Tumor',
'Tumor’]))

# Confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=['No Tumor', 'Tumor'], yticklabels=['No Tumor', 'Tumor'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Visualize Training Results

```
# Prediction and visualization function

class_labels = ['No Tumor', 'Tumor']
def predict_and_visualize(image_path):

    img = load_img(image_path, target_size=(128, 128))

    img_array = img_to_array(img)

    img_array = np.expand_dims(img_array, axis=0) / 255.0


    prediction = model.predict(img_array)[0][0]

    predicted_class = class_labels[int(prediction > 0.5)]

    confidence = prediction if prediction > 0.5 else 1 - prediction


    plt.imshow(img)

    plt.title(f"Predicted: {predicted_class}, Confidence: {confidence:.2f}")

    plt.axis('off')

    plt.show()
```

```
# Test on sample images

test_image_paths =
['/content/drive/MyDrive/BrainTumorCNN/sampleimages/tumor.jpg’,

'/content/drive/MyDrive/BrainTumorCNN/sampleimages/notumor.jpg'

]

for img_path in test_image_paths:

    predict_and_visualize(img_path)
```

Run the Code and wait…….

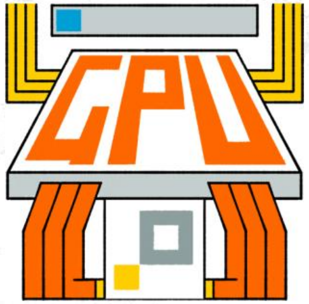


```python
def predict_and_visualize(image_path):
    img = load_img(image_path, target_size=(128, 128))
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0) / 255.0

    prediction = model.predict(img_array)[0][0]
    predicted_class = class_labels[int(prediction > 0.5)]
    confidence = prediction if prediction > 0.5 else 1 - prediction

    plt.imshow(img)
    plt.title(f"Predicted: {predicted_class}, Confidence: {confidence:.2f}")
    plt.axis('off')
    plt.show()

# Test on sample images
test_image_paths = [
    '/content/drive/MyDrive/BrainTumorCNN/sampleimages/tumor.jpg',
    '/content/drive/MyDrive/BrainTumorCNN/sampleimages/notumor.jpg'
]
for img_path in test_image_paths:
    predict_and_visualize(img_path)
```

```
Mounted at /content/drive
Testing folders: ['Yes Tumor', 'No Tumor']
Found 2916 images belonging to 2 classes.
Found 705 images belonging to 2 classes.
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Whe
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset` class should call `super().__init__(*
  self._warn_if_super_not_called()
91/92 ━━━━━━━━━━━━━━━━━━━━ 8s 8s/step - accuracy: 0.8073 - loss: 0.4155
```

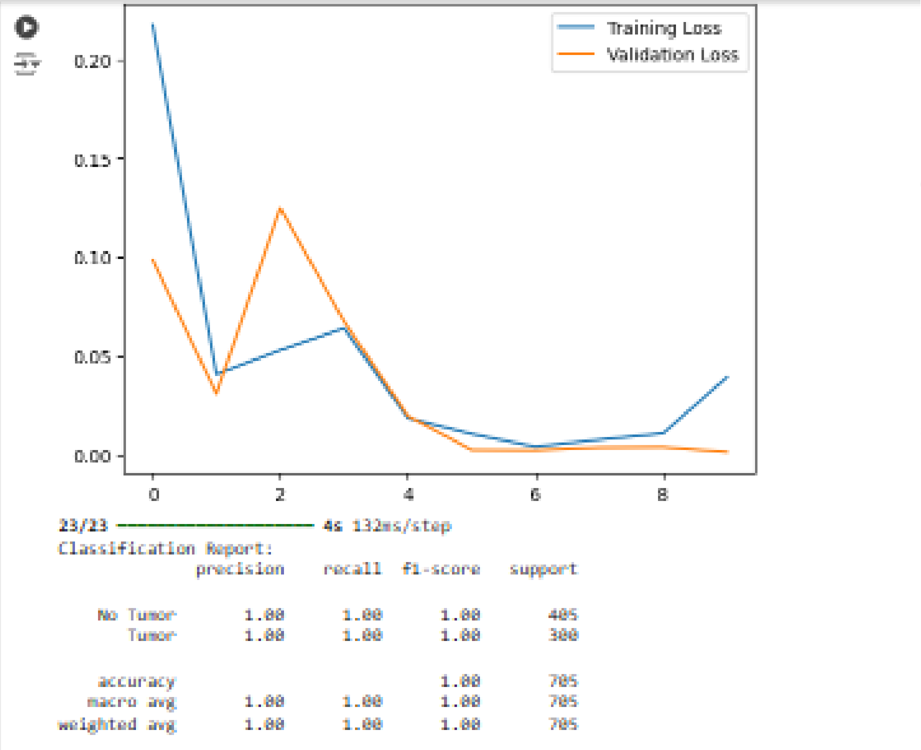
Executing (17m 54s) <cell line: 12> > <module> > <module> > <module> > <module> > <module> > <module> > <module> > <module> > <module> > <module> > <module> > <module>

Will take forever wait until model is created .....☺



Check results

https://colab.research.google.com/drive/1c7S07QIDgW4K73jo5AcxIaBMfcbvU2GL#scrollTo=Lq9o3b1gWPBE



```
23/23 ━━━━━━━━━━━━━━━━━━━━ 4s 132ms/step
Classification Report:
              precision    recall  f1-score   support

    No Tumor       1.00      1.00      1.00       405
       Tumor       1.00      1.00      1.00       300

    accuracy                           1.00       705
   macro avg       1.00      1.00      1.00       705
weighted avg       1.00      1.00      1.00       705
```

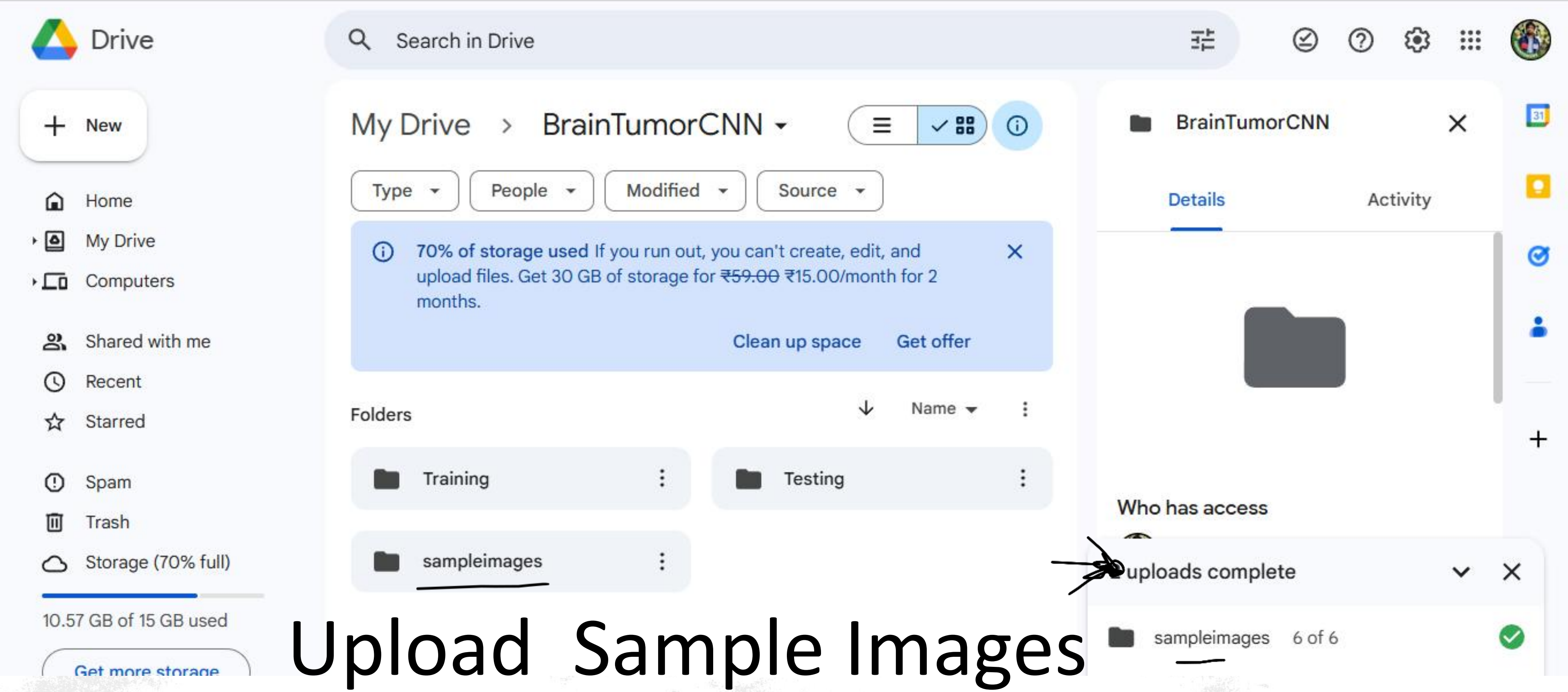
Results

- Achieved training and testing accuracy.
- Saved the trained model.

Next Steps:

- Visualized training progress with graphs.

Visualized training progress with graphs:



My Drive > BrainTumorCNN

70% of storage used If you run out, you can't create, edit, and upload files. Get 30 GB of storage for ~~₹59.00~~ ₹15.00/month for 2 months.

Clean up space Get offer

Folders

Training

Testing

sampleimages

BrainTumorCNN

Who has access

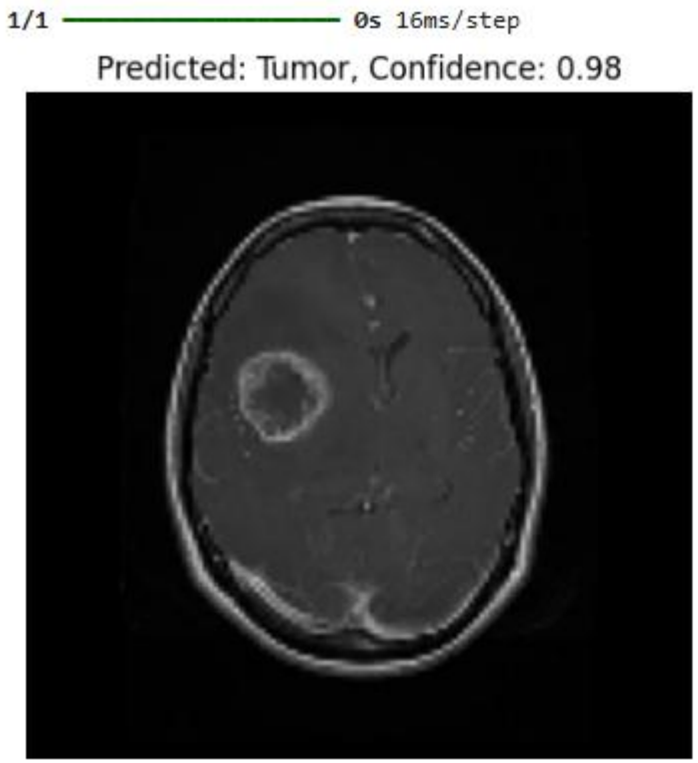
uploads complete

sampleimages 6 of 6

10.57 GB of 15 GB used

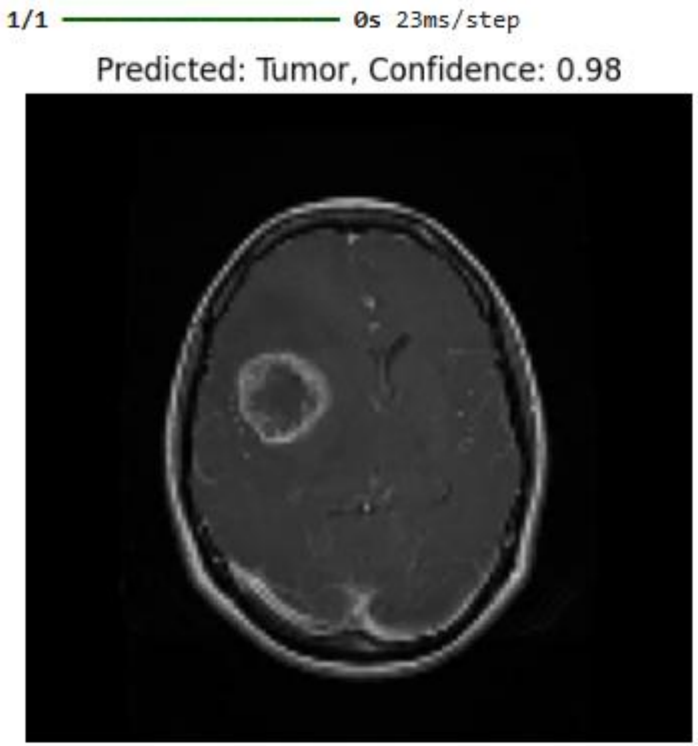
Upload Sample Images

See Results



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
```

Predicted: Tumor, Confidence: 0.98



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```

Predicted: Tumor, Confidence: 0.98

References

- Dataset: **Brain Tumor Dataset - Yes/No Class**

https://www.kaggle.com/datasets/princelv84/brain-tumor-dataset-yesno-class

- TensorFlow documentation https://www.tensorflow.org/api_docs/python/tf
- Google Colab https://colab.research.google.com/drive/1c7S07QIDgW4K73jo5AcxIaBMfcbvU2GL?usp=sharing