COMPUTER VISION

ASSIGNMENT 04

Group Members:

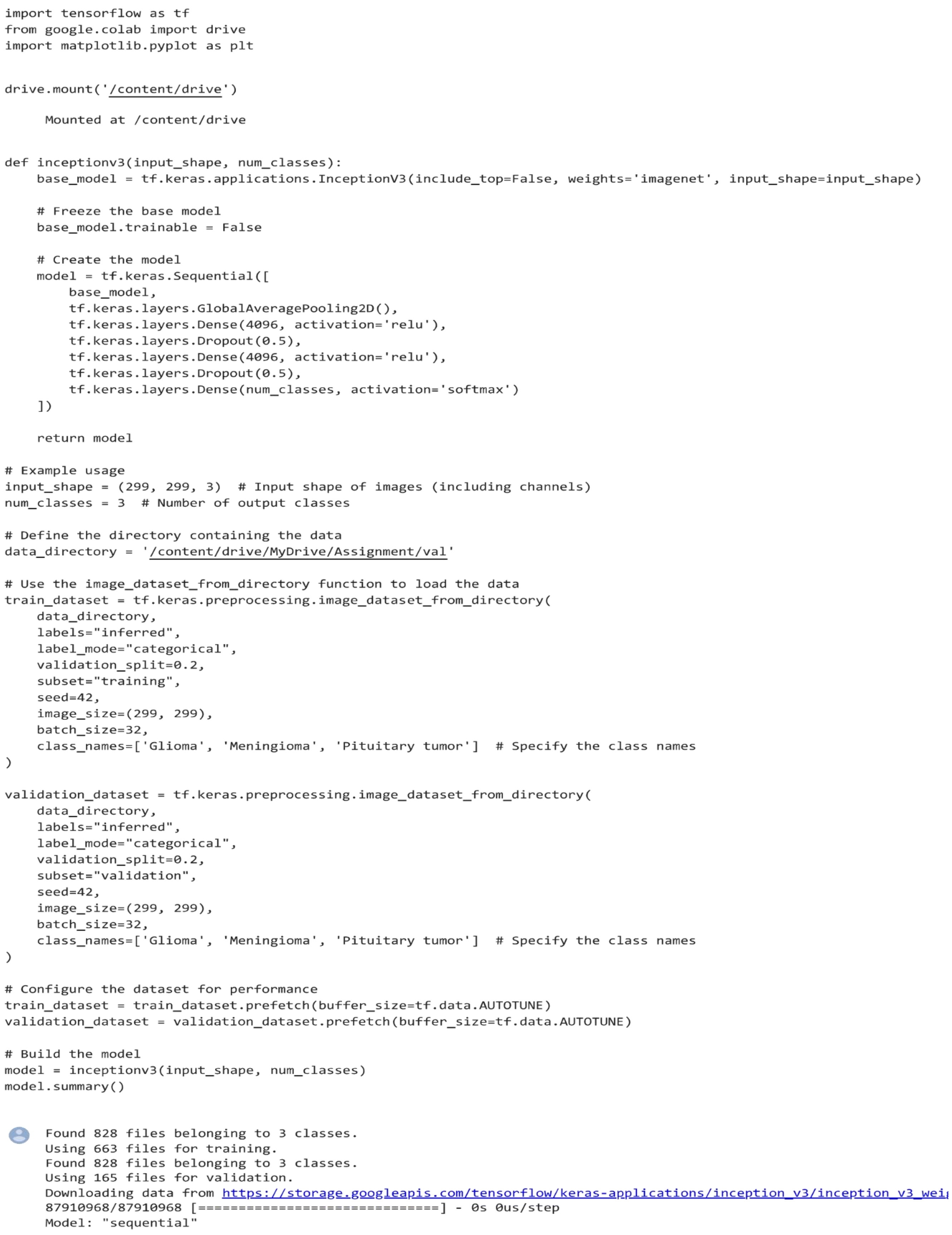
Abdullah Khalid Pasha (SP19-BCS-141)- G4

Ayesha Bibi (SP20-BCS-152)-G4

Abdullah Chaudhary (SP20-BCS-043)-G4

JUNE 13, 2023

INCEPTION

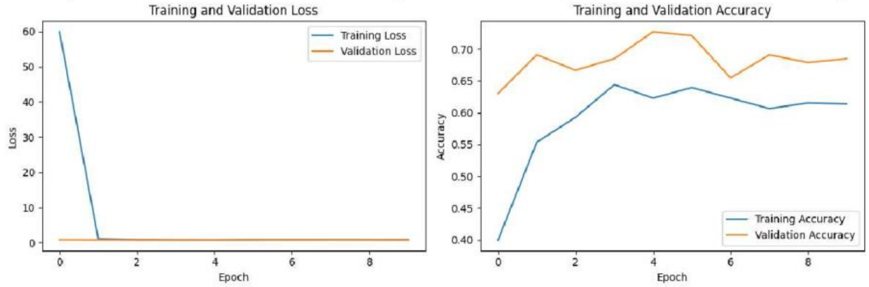


```
import tensorflow as tf
from google.colab import drive
import matplotlib.pyplot as plt


drive.mount('/content/drive')

     Mounted at /content/drive


def inceptionv3(input_shape, num_classes):
    base_model = tf.keras.applications.InceptionV3(include_top=False, weights='imagenet', input_shape=input_shape)

    # Freeze the base model
    base_model.trainable = False

    # Create the model
    model = tf.keras.Sequential([
        base_model,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(num_classes, activation='softmax')
    ])

    return model

# Example usage
input_shape = (299, 299, 3)  # Input shape of images (including channels)
num_classes = 3  # Number of output classes

# Define the directory containing the data
data_directory = '/content/drive/MyDrive/Assignment/val'

# Use the image_dataset_from_directory function to load the data
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    labels="inferred",
    label_mode="categorical",
    validation_split=0.2,
    subset="training",
    seed=42,
    image_size=(299, 299),
    batch_size=32,
    class_names=['Glioma', 'Meningioma', 'Pituitary tumor']  # Specify the class names
)

validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    labels="inferred",
    label_mode="categorical",
    validation_split=0.2,
    subset="validation",
    seed=42,
    image_size=(299, 299),
    batch_size=32,
    class_names=['Glioma', 'Meningioma', 'Pituitary tumor']  # Specify the class names
)

# Configure the dataset for performance
train_dataset = train_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)

# Build the model
model = inceptionv3(input_shape, num_classes)
model.summary()


     Found 828 files belonging to 3 classes.
     Using 663 files for training.
     Found 828 files belonging to 3 classes.
     Using 165 files for validation.
     Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_wei
     87910968/87910968 [==============================] - 0s 0us/step
     Model: "sequential"
```

```
 Layer (type)                Output Shape              Param #   
=================================================================
 inception_v3 (Functional)   (None, 8, 8, 2048)        21802784  
                                                                 
 global_average_pooling2d (G  (None, 2048)             0         
 lobalAveragePooling2D)                                          
                                                                 
 dense (Dense)               (None, 4096)              8392704   
                                                                 
 dropout (Dropout)           (None, 4096)              0         
                                                                 
 dense_1 (Dense)             (None, 4096)              16781312  
                                                                 
 dropout_1 (Dropout)         (None, 4096)              0         
                                                                 
 dense_2 (Dense)             (None, 3)                 12291     
                                                                 
=================================================================
Total params: 46,989,091
Trainable params: 25,186,307
Non-trainable params: 21,802,784
```

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


class PlotTrainingProgress(tf.keras.callbacks.Callback):
    def on_train_end(self, logs=None):
        fig, axes = plt.subplots(1, 2, figsize=(12, 4))

        axes[0].plot(self.model.history.history['loss'], label='Training Loss')
        axes[0].plot(self.model.history.history['val_loss'], label='Validation Loss')
        axes[0].set_xlabel('Epoch')
        axes[0].set_ylabel('Loss')
        axes[0].set_title('Training and Validation Loss')
        axes[0].legend()

        axes[1].plot(self.model.history.history['accuracy'], label='Training Accuracy')
        axes[1].plot(self.model.history.history['val_accuracy'], label='Validation Accuracy')
        axes[1].set_xlabel('Epoch')
        axes[1].set_ylabel('Accuracy')
        axes[1].set_title('Training and Validation Accuracy')
        axes[1].legend()

        plt.tight_layout()
        plt.show()  # Show the final figure

history = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=10,
    callbacks=[PlotTrainingProgress()]
)

model.save('/content/drive/MyDrive/Assignment/inception_model')
```

```
Epoch 1/10
21/21 [==============================] - 285s 12s/step - loss: 59.9507 - accuracy:
Epoch 2/10
21/21 [==============================] - 204s 10s/step - loss: 1.1048 - accuracy:
Epoch 3/10
21/21 [==============================] - 200s 10s/step - loss: 0.8327 - accuracy:
Epoch 4/10
21/21 [==============================] - 199s 10s/step - loss: 0.8062 - accuracy:
Epoch 5/10
21/21 [==============================] - 196s 9s/step - loss: 0.7730 - accuracy: 0
Epoch 6/10
21/21 [==============================] - 198s 9s/step - loss: 0.7929 - accuracy: 0
Epoch 7/10
21/21 [==============================] - 202s 10s/step - loss: 0.8538 - accuracy:
Epoch 8/10
21/21 [==============================] - 209s 10s/step - loss: 0.8552 - accuracy:
Epoch 9/10
21/21 [==============================] - 206s 10s/step - loss: 0.8252 - accuracy:
Epoch 10/10
21/21 [==============================] - 210s 10s/step - loss: 0.8422 - accuracy:
```



```
WARNING:absl:Found untraced functions such as _update_step_xla, _jit_compiled_conv
```

RESNET

```
import tensorflow as tf
from google.colab import drive
import matplotlib.pyplot as plt


drive.mount('/content/drive')
```

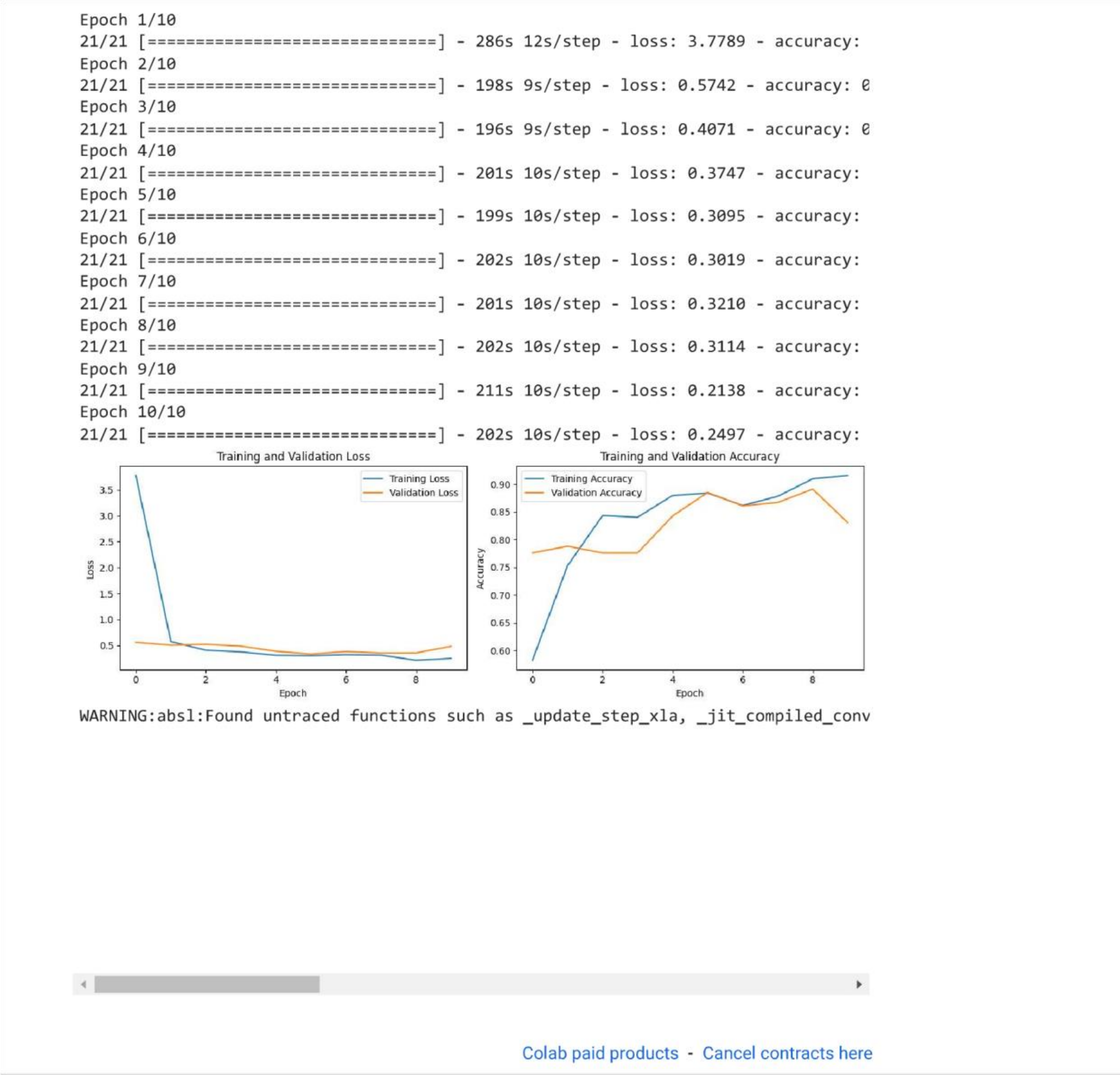
```
     Mounted at /content/drive
```

```
def resnet50(input_shape, num_classes):
    base_model = tf.keras.applications.ResNet50(include_top=False, weights='imagenet', input_shape=input_shape)

    # Freeze the base model
    base_model.trainable = False

    # Create the model
    model = tf.keras.Sequential([
        base_model,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(num_classes, activation='softmax')
    ])

    return model

# Example usage
input_shape = (224, 224, 3)  # Input shape of images (including channels)
num_classes = 3  # Number of output classes

# Define the directory containing the data
data_directory = '/content/drive/MyDrive/Assignment/val'

# Use the image_dataset_from_directory function to load the data
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    labels="inferred",
    label_mode="categorical",
    validation_split=0.2,
    subset="training",
    seed=42,
    image_size=(224, 224),
    batch_size=32,
    class_names=['Glioma', 'Meningioma', 'Pituitary tumor']  # Specify the class names
)

validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    labels="inferred",
    label_mode="categorical",
    validation_split=0.2,
    subset="validation",
    seed=42,
    image_size=(224, 224),
    batch_size=32,
    class_names=['Glioma', 'Meningioma', 'Pituitary tumor']  # Specify the class names
)

# Configure the dataset for performance
train_dataset = train_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)

# Build the model
model = resnet50(input_shape, num_classes)
model.summary()

# Compile and train the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
     Found 828 files belonging to 3 classes.
     Using 663 files for training.
     Found 828 files belonging to 3 classes.
```

```
Using 165 files for validation.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_orderi
94765736/94765736 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712  
                                                                 
 global_average_pooling2d (G  (None, 2048)             0         
 lobalAveragePooling2D)                                          
                                                                 
 dense (Dense)               (None, 4096)              8392704   
                                                                 
 dropout (Dropout)           (None, 4096)              0         
                                                                 
 dense_1 (Dense)             (None, 4096)              16781312  
                                                                 
 dropout_1 (Dropout)         (None, 4096)              0         
                                                                 
 dense_2 (Dense)             (None, 3)                 12291     
                                                                 
=================================================================
Total params: 48,774,019
Trainable params: 25,186,307
Non-trainable params: 23,587,712
_________________________________________________________________
```

```
class PlotTrainingProgress(tf.keras.callbacks.Callback):
    def on_train_end(self, logs=None):
        fig, axes = plt.subplots(1, 2, figsize=(12, 4))

        axes[0].plot(self.model.history.history['loss'], label='Training Loss')
        axes[0].plot(self.model.history.history['val_loss'], label='Validation Loss')
        axes[0].set_xlabel('Epoch')
        axes[0].set_ylabel('Loss')
        axes[0].set_title('Training and Validation Loss')
        axes[0].legend()

        axes[1].plot(self.model.history.history['accuracy'], label='Training Accuracy')
        axes[1].plot(self.model.history.history['val_accuracy'], label='Validation Accuracy')
        axes[1].set_xlabel('Epoch')
        axes[1].set_ylabel('Accuracy')
        axes[1].set_title('Training and Validation Accuracy')
        axes[1].legend()

        plt.tight_layout()
        plt.show()  # Show the final figure

history = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=10,
    callbacks=[PlotTrainingProgress()]
)

model.save('/content/drive/MyDrive/Assignment/resnet_model')
```



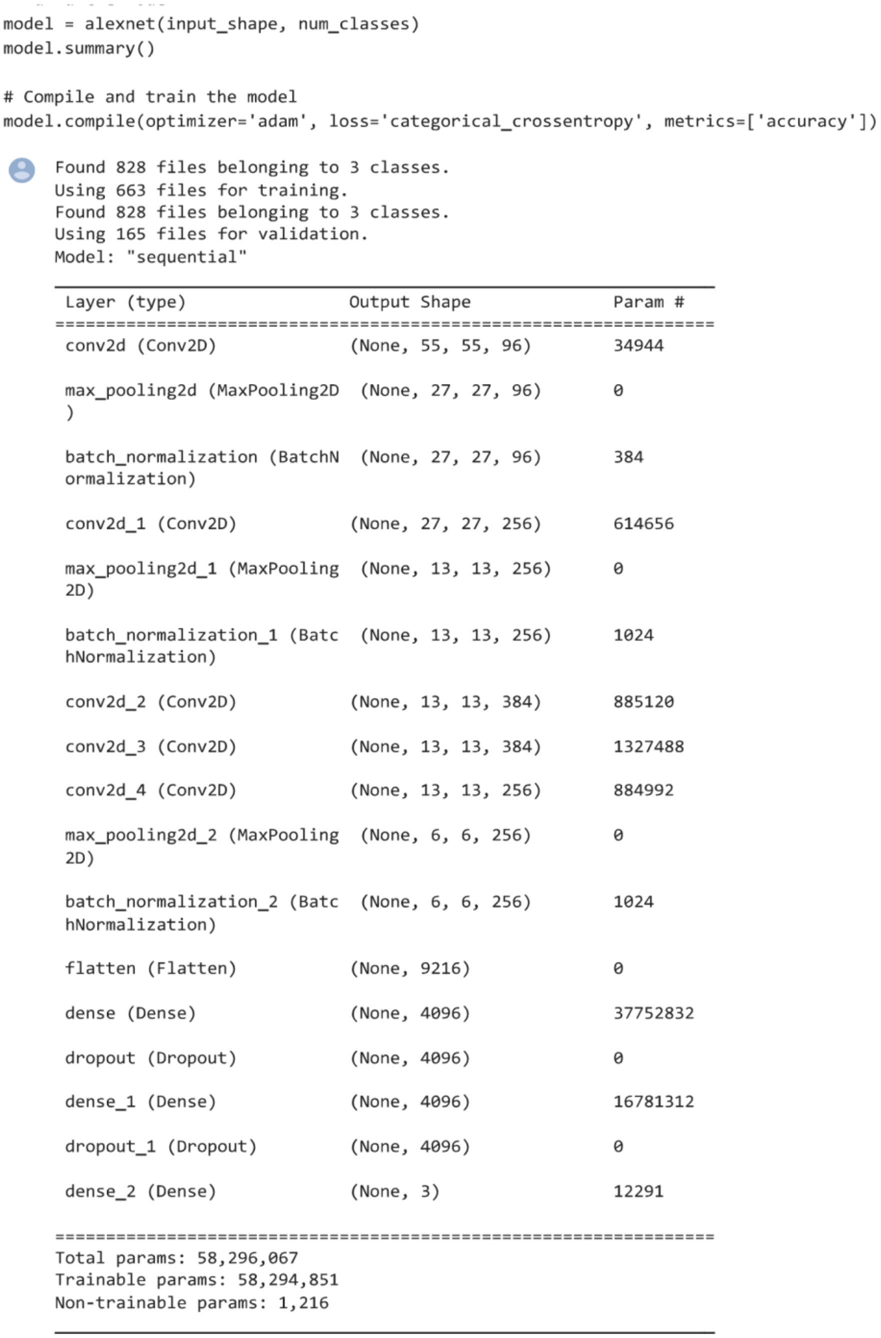
```
Epoch 1/10
21/21 [==============================] - 286s 12s/step - loss: 3.7789 - accuracy:
Epoch 2/10
21/21 [==============================] - 198s 9s/step - loss: 0.5742 - accuracy: 0
Epoch 3/10
21/21 [==============================] - 196s 9s/step - loss: 0.4071 - accuracy: 0
Epoch 4/10
21/21 [==============================] - 201s 10s/step - loss: 0.3747 - accuracy:
Epoch 5/10
21/21 [==============================] - 199s 10s/step - loss: 0.3095 - accuracy:
Epoch 6/10
21/21 [==============================] - 202s 10s/step - loss: 0.3019 - accuracy:
Epoch 7/10
21/21 [==============================] - 201s 10s/step - loss: 0.3210 - accuracy:
Epoch 8/10
21/21 [==============================] - 202s 10s/step - loss: 0.3114 - accuracy:
Epoch 9/10
21/21 [==============================] - 211s 10s/step - loss: 0.2138 - accuracy:
Epoch 10/10
21/21 [==============================] - 202s 10s/step - loss: 0.2497 - accuracy:
```

```
WARNING:absl:Found untraced functions such as _update_step_xla, _jit_compiled_conv
```

Colab paid products - Cancel contracts here

Alex Net

```
import tensorflow as tf
from google.colab import drive
import matplotlib.pyplot as plt


drive.mount('/content/drive')


     Mounted at /content/drive


import tensorflow as tf
import matplotlib.pyplot as plt

def alexnet(input_shape, num_classes):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input_shape=input_shape),
        tf.keras.layers.MaxPooling2D((3, 3), strides=(2, 2)),
        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Conv2D(256, (5, 5), padding='same', activation='relu'),
        tf.keras.layers.MaxPooling2D((3, 3), strides=(2, 2)),
        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Conv2D(384, (3, 3), padding='same', activation='relu'),
        tf.keras.layers.Conv2D(384, (3, 3), padding='same', activation='relu'),
        tf.keras.layers.Conv2D(256, (3, 3), padding='same', activation='relu'),
        tf.keras.layers.MaxPooling2D((3, 3), strides=(2, 2)),
        tf.keras.layers.BatchNormalization(),

        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(num_classes, activation='softmax')
    ])

    return model

# Example usage
input_shape = (227, 227, 3)  # Input shape of images (including channels)
num_classes = 3  # Number of output classes

# Define the directory containing the data
data_directory = '/content/drive/MyDrive/Assignment/val'

# Use the image_dataset_from_directory function to load the data
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    labels="inferred",
    label_mode="categorical",
    validation_split=0.2,
    subset="training",
    seed=42,
    image_size=(227, 227),
    batch_size=32,
    class_names=['Glioma', 'Meningioma', 'Pituitary tumor']  # Specify the class names
)

validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    labels="inferred",
    label_mode="categorical",
    validation_split=0.2,
    subset="validation",
    seed=42,
    image_size=(227, 227),
    batch_size=32,
    class_names=['Glioma', 'Meningioma', 'Pituitary tumor']  # Specify the class names
)

# Configure the dataset for performance
train_dataset = train_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)

# Build the model
```

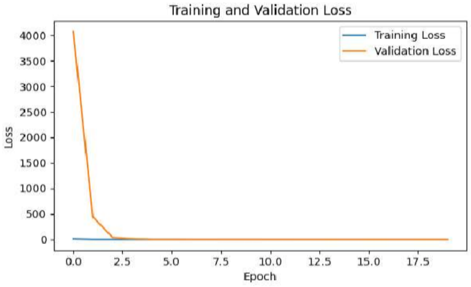


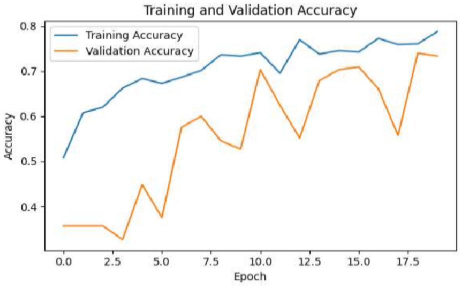
```
model = alexnet(input_shape, num_classes)
model.summary()

# Compile and train the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Found 828 files belonging to 3 classes.
Using 663 files for training.
Found 828 files belonging to 3 classes.
Using 165 files for validation.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 55, 55, 96)        34944     
                                                                 
 max_pooling2d (MaxPooling2D  (None, 27, 27, 96)       0         
 )                                                               
                                                                 
 batch_normalization (BatchN  (None, 27, 27, 96)       384       
 ormalization)                                                   
                                                                 
 conv2d_1 (Conv2D)           (None, 27, 27, 256)       614656    
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 13, 13, 256)      0         
 2D)                                                             
                                                                 
 batch_normalization_1 (Batc  (None, 13, 13, 256)      1024      
 hNormalization)                                                 
                                                                 
 conv2d_2 (Conv2D)           (None, 13, 13, 384)       885120    
                                                                 
 conv2d_3 (Conv2D)           (None, 13, 13, 384)       1327488   
                                                                 
 conv2d_4 (Conv2D)           (None, 13, 13, 256)       884992    
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 6, 6, 256)        0         
 2D)                                                             
                                                                 
 batch_normalization_2 (Batc  (None, 6, 6, 256)        1024      
 hNormalization)                                                 
                                                                 
 flatten (Flatten)           (None, 9216)              0         
                                                                 
 dense (Dense)               (None, 4096)              37752832  
                                                                 
 dropout (Dropout)           (None, 4096)              0         
                                                                 
 dense_1 (Dense)             (None, 4096)              16781312  
                                                                 
 dropout_1 (Dropout)         (None, 4096)              0         
                                                                 
 dense_2 (Dense)             (None, 3)                 12291     
                                                                 
=================================================================
Total params: 58,296,067
Trainable params: 58,294,851
Non-trainable params: 1,216
_________________________________________________________________
```

```
class PlotTrainingProgress(tf.keras.callbacks.Callback):
    def on_train_end(self, logs=None):
        fig, axes = plt.subplots(1, 2, figsize=(12, 4))

        axes[0].plot(self.model.history.history['loss'], label='Training Loss')
        axes[0].plot(self.model.history.history['val_loss'], label='Validation Loss')
        axes[0].set_xlabel('Epoch')
        axes[0].set_ylabel('Loss')
        axes[0].set_title('Training and Validation Loss')
        axes[0].legend()

        axes[1].plot(self.model.history.history['accuracy'], label='Training Accuracy')
        axes[1].plot(self.model.history.history['val_accuracy'], label='Validation Accuracy')
        axes[1].set_xlabel('Epoch')
        axes[1].set_ylabel('Accuracy')
        axes[1].set_title('Training and Validation Accuracy')
        axes[1].legend()

        plt.tight_layout()
        plt.show()  # Show the final figure

history = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=20,
    callbacks=[PlotTrainingProgress()]
)

model.save('/content/drive/MyDrive/Assignment/saved_model')
```

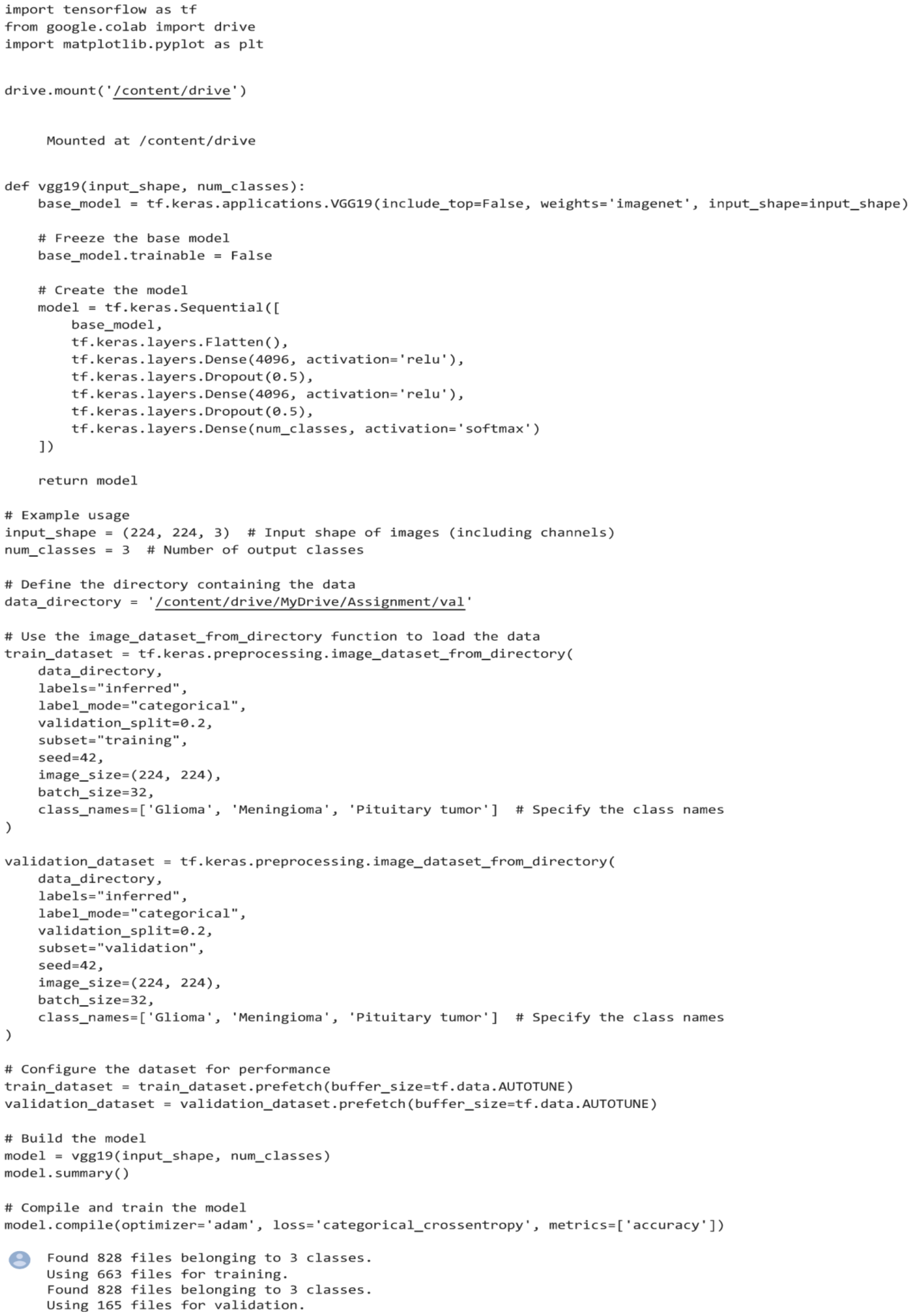
```
Epoch 1/20
21/21 [==============================] - 95s 2s/step - loss: 13.6835 - accuracy: 0
Epoch 2/20
21/21 [==============================] - 5s 136ms/step - loss: 2.2233 - accuracy:
Epoch 3/20
21/21 [==============================] - 4s 128ms/step - loss: 2.1160 - accuracy:
Epoch 4/20
21/21 [==============================] - 4s 130ms/step - loss: 1.2630 - accuracy:
Epoch 5/20
21/21 [==============================] - 6s 209ms/step - loss: 0.8997 - accuracy:
Epoch 6/20
21/21 [==============================] - 4s 131ms/step - loss: 0.9337 - accuracy:
Epoch 7/20
21/21 [==============================] - 4s 130ms/step - loss: 0.7255 - accuracy:
Epoch 8/20
21/21 [==============================] - 6s 228ms/step - loss: 0.7479 - accuracy:
Epoch 9/20
21/21 [==============================] - 5s 138ms/step - loss: 0.6522 - accuracy:
Epoch 10/20
21/21 [==============================] - 4s 129ms/step - loss: 0.7379 - accuracy:
Epoch 11/20
21/21 [==============================] - 4s 170ms/step - loss: 0.7503 - accuracy:
Epoch 12/20
21/21 [==============================] - 4s 131ms/step - loss: 0.7166 - accuracy:
Epoch 13/20
21/21 [==============================] - 4s 134ms/step - loss: 0.5411 - accuracy:
Epoch 14/20
21/21 [==============================] - 5s 205ms/step - loss: 0.5900 - accuracy:
Epoch 15/20
21/21 [==============================] - 4s 130ms/step - loss: 0.6419 - accuracy:
Epoch 16/20
21/21 [==============================] - 4s 132ms/step - loss: 0.5781 - accuracy:
Epoch 17/20
21/21 [==============================] - 4s 162ms/step - loss: 0.5139 - accuracy:
Epoch 18/20
21/21 [==============================] - 5s 137ms/step - loss: 0.6011 - accuracy:
Epoch 19/20
21/21 [==============================] - 4s 130ms/step - loss: 0.5990 - accuracy:
Epoch 20/20
21/21 [==============================] - 5s 176ms/step - loss: 0.5381 - accuracy:
```





```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_c
```

VGG19

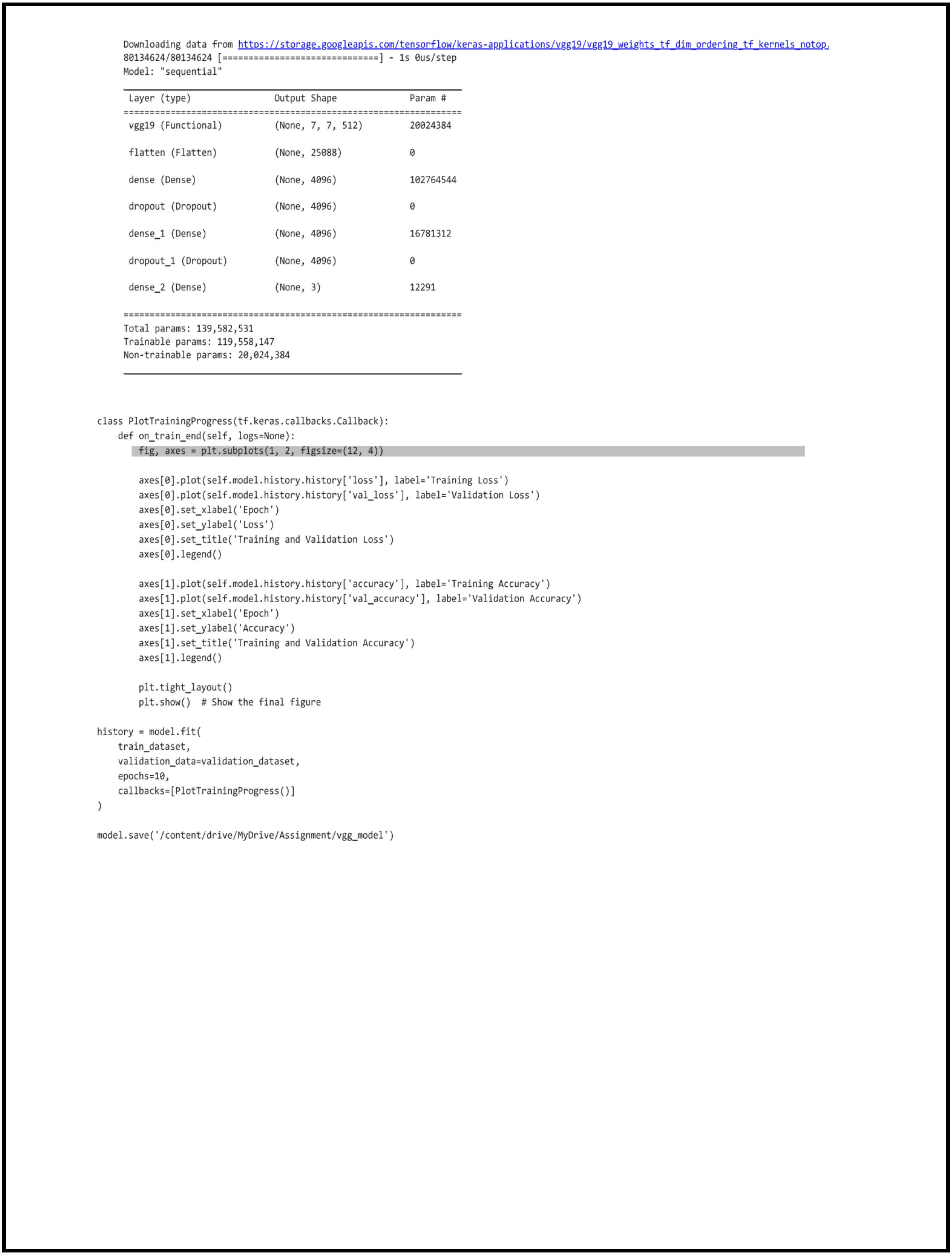


```
import tensorflow as tf
from google.colab import drive
import matplotlib.pyplot as plt


drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

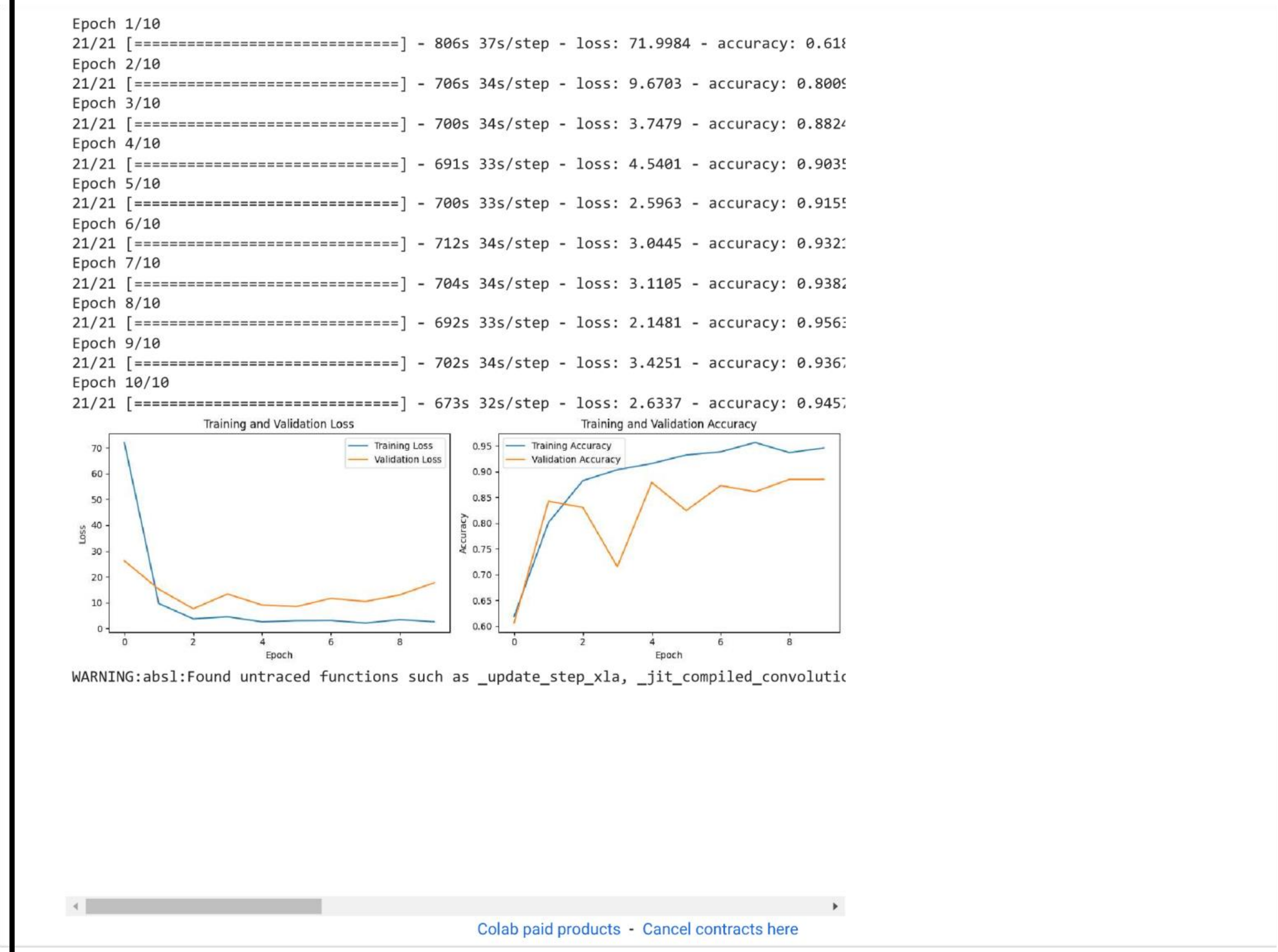
```
def vgg19(input_shape, num_classes):
    base_model = tf.keras.applications.VGG19(include_top=False, weights='imagenet', input_shape=input_shape)

    # Freeze the base model
    base_model.trainable = False

    # Create the model
    model = tf.keras.Sequential([
        base_model,
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(num_classes, activation='softmax')
    ])

    return model

# Example usage
input_shape = (224, 224, 3)  # Input shape of images (including channels)
num_classes = 3  # Number of output classes

# Define the directory containing the data
data_directory = '/content/drive/MyDrive/Assignment/val'

# Use the image_dataset_from_directory function to load the data
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    labels="inferred",
    label_mode="categorical",
    validation_split=0.2,
    subset="training",
    seed=42,
    image_size=(224, 224),
    batch_size=32,
    class_names=['Glioma', 'Meningioma', 'Pituitary tumor']  # Specify the class names
)

validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    data_directory,
    labels="inferred",
    label_mode="categorical",
    validation_split=0.2,
    subset="validation",
    seed=42,
    image_size=(224, 224),
    batch_size=32,
    class_names=['Glioma', 'Meningioma', 'Pituitary tumor']  # Specify the class names
)

# Configure the dataset for performance
train_dataset = train_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)

# Build the model
model = vgg19(input_shape, num_classes)
model.summary()

# Compile and train the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
     Found 828 files belonging to 3 classes.
     Using 663 files for training.
     Found 828 files belonging to 3 classes.
     Using 165 files for validation.
```



```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.
80134624/80134624 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 vgg19 (Functional)          (None, 7, 7, 512)         20024384  
                                                                 
 flatten (Flatten)           (None, 25088)             0         
                                                                 
 dense (Dense)               (None, 4096)              102764544 
                                                                 
 dropout (Dropout)           (None, 4096)              0         
                                                                 
 dense_1 (Dense)             (None, 4096)              16781312  
                                                                 
 dropout_1 (Dropout)         (None, 4096)              0         
                                                                 
 dense_2 (Dense)             (None, 3)                 12291     
                                                                 
=================================================================
Total params: 139,582,531
Trainable params: 119,558,147
Non-trainable params: 20,024,384
_________________________________________________________________
```

```python
class PlotTrainingProgress(tf.keras.callbacks.Callback):
    def on_train_end(self, logs=None):
        fig, axes = plt.subplots(1, 2, figsize=(12, 4))

        axes[0].plot(self.model.history.history['loss'], label='Training Loss')
        axes[0].plot(self.model.history.history['val_loss'], label='Validation Loss')
        axes[0].set_xlabel('Epoch')
        axes[0].set_ylabel('Loss')
        axes[0].set_title('Training and Validation Loss')
        axes[0].legend()

        axes[1].plot(self.model.history.history['accuracy'], label='Training Accuracy')
        axes[1].plot(self.model.history.history['val_accuracy'], label='Validation Accuracy')
        axes[1].set_xlabel('Epoch')
        axes[1].set_ylabel('Accuracy')
        axes[1].set_title('Training and Validation Accuracy')
        axes[1].legend()

        plt.tight_layout()
        plt.show()  # Show the final figure

history = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=10,
    callbacks=[PlotTrainingProgress()]
)

model.save('/content/drive/MyDrive/Assignment/vgg_model')
```



```
Epoch 1/10
21/21 [==============================] - 806s 37s/step - loss: 71.9984 - accuracy: 0.618
Epoch 2/10
21/21 [==============================] - 706s 34s/step - loss: 9.6703 - accuracy: 0.8009
Epoch 3/10
21/21 [==============================] - 700s 34s/step - loss: 3.7479 - accuracy: 0.8824
Epoch 4/10
21/21 [==============================] - 691s 33s/step - loss: 4.5401 - accuracy: 0.9035
Epoch 5/10
21/21 [==============================] - 700s 33s/step - loss: 2.5963 - accuracy: 0.9155
Epoch 6/10
21/21 [==============================] - 712s 34s/step - loss: 3.0445 - accuracy: 0.9321
Epoch 7/10
21/21 [==============================] - 704s 34s/step - loss: 3.1105 - accuracy: 0.9382
Epoch 8/10
21/21 [==============================] - 692s 33s/step - loss: 2.1481 - accuracy: 0.9563
Epoch 9/10
21/21 [==============================] - 702s 34s/step - loss: 3.4251 - accuracy: 0.9367
Epoch 10/10
21/21 [==============================] - 673s 32s/step - loss: 2.6337 - accuracy: 0.9457
```

```
WARNING:absl:Found untraced functions such as _update_step_xla, _jit_compiled_convolutic
```

Colab paid products - Cancel contracts here