American International University- Bangladesh (AIUB)

Course Name:	Machine learning	**Course Code:**	CSC4232
Semester:	Spring 2024-25	**Submission date:**	26/04/25
Section	D		

Group Information

No	Name	ID
1	MD MEHEDI HASAN POLASH	22-46566-1
2	TRIDIB SARKAR	22-46444-1
3	SAJIN MAHMUD ARPON	22-46629-1
4	TALHA HOSSAIN SIFAT	22-46344-1

Import necessary libraries

```
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras import layers, models
```

Upload the path of google drive and set parameters

```
# Set the path to your dataset on Google Drive
path = '/content/drive/MyDrive/dataset'

# Image and training parameters
img_size = (224, 224)
batch_size = 32
epochs = 10
```

Data augmentation and preprocessing part

```
# Data preprocessing using ImageDataGenerator
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.3,
    horizontal_flip=True,
    vertical_flip=True
)

# Grayscale loader function
def to_grayscale(image):
    return tf.image.rgb_to_grayscale(image)

# Apply grayscale transformation
def preprocess(image):
    image = tf.image.resize(image, img_size)
    image = tf.image.rgb_to_grayscale(image)
    image = tf.image.grayscale_to_rgb(image)  # Convert back to 3-channel
    return image

# Train generator
train_generator = train_datagen.flow_from_directory(
    path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='training',
    shuffle=True
)

# Validation/Test generator
test_generator = train_datagen.flow_from_directory(
    path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='validation',
    shuffle=False
)
```

```
Found 178 images belonging to 2 classes.
Found 75 images belonging to 2 classes.
```

Choose DenseNet121 model and create model

```
# Load base model
base_model = DenseNet121(include_top=False, weights='imagenet', input_shape=(224, 224, 3))
base_model.trainable = False  # Freeze base model

# Add custom layers
model = models.Sequential([
    layers.Input(shape=(224, 224, 3)),
    layers.Lambda(preprocess),  # Apply grayscale conversion
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_
29084464/29084464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lambda (Lambda)                  │ (None, 224, 224, 3)    │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ densenet121 (Functional)         │ (None, 7, 7, 1024)     │     7,037,504 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ global_average_pooling2d         │ (None, 1024)           │             0 │
│ (GlobalAveragePooling2D)         │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                    │ (None, 256)            │       262,400 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)                │ (None, 256)            │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ (None, 1)              │           257 │
└──────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 7,300,161 (27.85 MB)
```

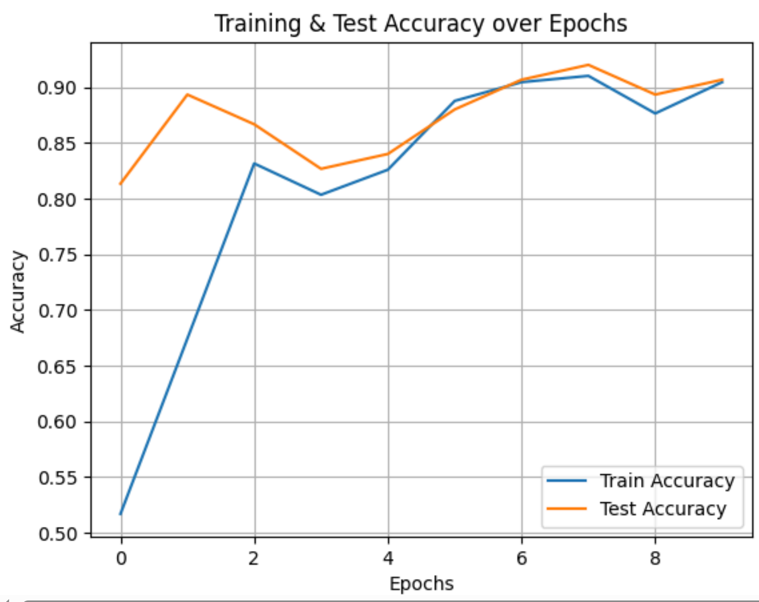
The process of train the Model

```
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=test_generator
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
Epoch 1/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 79s 11s/step - accuracy: 0.4654 - loss: 1.1437 - val_accuracy: 0.8133 - val_loss: 0.4719
Epoch 2/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 47s 9s/step - accuracy: 0.6564 - loss: 0.6768 - val_accuracy: 0.8933 - val_loss: 0.3934
Epoch 3/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 46s 8s/step - accuracy: 0.8283 - loss: 0.4384 - val_accuracy: 0.8667 - val_loss: 0.3532
Epoch 4/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 46s 8s/step - accuracy: 0.8292 - loss: 0.3920 - val_accuracy: 0.8267 - val_loss: 0.3423
Epoch 5/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 46s 8s/step - accuracy: 0.8328 - loss: 0.4100 - val_accuracy: 0.8400 - val_loss: 0.3750
Epoch 6/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 46s 8s/step - accuracy: 0.8747 - loss: 0.3454 - val_accuracy: 0.8800 - val_loss: 0.2902
Epoch 7/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 47s 8s/step - accuracy: 0.9074 - loss: 0.2355 - val_accuracy: 0.9067 - val_loss: 0.3101
Epoch 8/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 46s 8s/step - accuracy: 0.9252 - loss: 0.2314 - val_accuracy: 0.9200 - val_loss: 0.3145
Epoch 9/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 47s 8s/step - accuracy: 0.8913 - loss: 0.2841 - val_accuracy: 0.8933 - val_loss: 0.2753
Epoch 10/10
6/6 ━━━━━━━━━━━━━━━━━━━━ 54s 9s/step - accuracy: 0.9130 - loss: 0.2321 - val_accuracy: 0.9067 - val_loss: 0.2990
```

Plot Training & Validation Accuracy

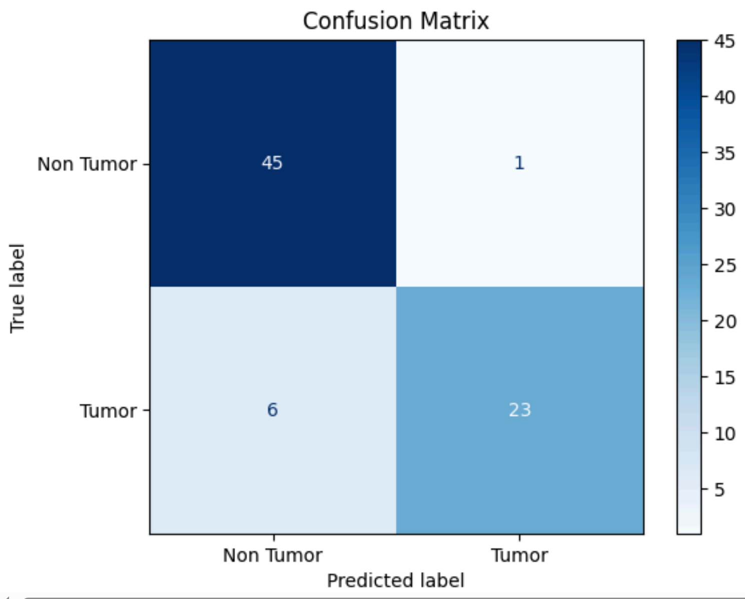
```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Test Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training & Test Accuracy over Epochs')
plt.legend()
plt.grid(True)
plt.show()
```



Finally Evaluate the Model & it's Metrics

```
# Predict on test set
y_pred_probs = model.predict(test_generator)
y_pred = (y_pred_probs > 0.5).astype(int).reshape(-1)
y_true = test_generator.classes

# Classification report
print("Classification Report:\n")
print(classification_report(y_true, y_pred, target_names=['Non Tumor', 'Tumor']))

# Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Non Tumor', 'Tumor'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 28s 7s/step
Classification Report:

              precision    recall  f1-score   support

   Non Tumor       0.88      0.98      0.93        46
       Tumor       0.96      0.79      0.87        29

    accuracy                           0.91        75
   macro avg       0.92      0.89      0.90        75
weighted avg       0.91      0.91      0.90        75
```


Confusion Matrix
True label
Predicted label
Non Tumor
Tumor
Non Tumor
Tumor