```
import os
import numpy as np
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import (
    accuracy_score,
    f1_score,
    confusion_matrix,
    precision_score,
    recall_score,
    classification_report,
)
from tensorflow import keras
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras.preprocessing.image import ImageDataGenerator
from google.colab import files
```

```
#Preprocess images using ImageDataGenerator
def preprocessing_images(path, augmentation=False):
    if augmentation:
        image_data = ImageDataGenerator(
            zoom_range=0.2,
            shear_range=0.2,
            rescale=1/255,
            horizontal_flip=True,
            rotation_range=20,
            width_shift_range=0.2,
            height_shift_range=0.2,
            fill_mode='nearest'
        )
    else:
        image_data = ImageDataGenerator(rescale=1/255)

    image_generator = image_data.flow_from_directory(
        directory=path,
        target_size=(227, 227),
        batch_size=32,
        class_mode='binary'
    )

    return image_generator


# Specify paths
train_path = r"/content/drive/MyDrive/DDSM/train"
test_path = r"/content/drive/MyDrive/DDSM/test"
val_path = r"/content/drive/MyDrive/DDSM/val"

# Data augmentation for training but not for validation set
train_data = preprocessing_images(train_path, augmentation=True)
val_data = preprocessing_images(val_path, augmentation=False)
```

```
Found 8433 images belonging to 2 classes.
Found 1808 images belonging to 2 classes.
```

```
# Model definition
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(227, 227, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['ac


# Model checkpoint
mc = ModelCheckpoint(
    monitor="val_accuracy",
    filepath="./21jan-ddsm-bestmodel.h5",
    verbose=1,
    save_best_only=True,
    mode='auto',
    save_freq='epoch'
)

# Learning rate scheduler function
def lr_schedule(epoch):
    learning_rate = 1e-4
    if epoch > 30:
        learning_rate *= 1e-1
    return learning_rate
# Learning rate scheduler callback
lr_scheduler = LearningRateScheduler(lr_schedule)

# Training
history = model.fit(
    train_data,
    steps_per_epoch=264,
    epochs=50,
    verbose=1,
    validation_data=val_data,
    validation_steps=57,
    callbacks=[mc, lr_scheduler]
)
```

```
Epoch 1/50
264/264 [==============================] - ETA: 0s - loss: 0.9692 - accuracy: 0.6218
Epoch 1: val_accuracy improved from -inf to 0.50055, saving model to ./21jan-ddsm-bes
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarnin
  saving_api.save_model(
264/264 [==============================] - 2166s 8s/step - loss: 0.9692 - accuracy: 0
Epoch 2/50
264/264 [==============================] - ETA: 0s - loss: 0.4616 - accuracy: 0.7742
Epoch 2: val_accuracy improved from 0.50055 to 0.72732, saving model to ./21jan-ddsm-
264/264 [==============================] - 121s 460ms/step - loss: 0.4616 - accuracy:
Epoch 3/50
264/264 [==============================] - ETA: 0s - loss: 0.3752 - accuracy: 0.8298
Epoch 3: val_accuracy improved from 0.72732 to 0.95852, saving model to ./21jan-ddsm-
264/264 [==============================] - 123s 467ms/step - loss: 0.3752 - accuracy:
Epoch 4/50
264/264 [==============================] - ETA: 0s - loss: 0.3545 - accuracy: 0.8477
Epoch 4: val_accuracy did not improve from 0.95852
264/264 [==============================] - 118s 448ms/step - loss: 0.3545 - accuracy:
Epoch 5/50
264/264 [==============================] - ETA: 0s - loss: 0.3376 - accuracy: 0.8583
Epoch 5: val_accuracy did not improve from 0.95852
264/264 [==============================] - 117s 444ms/step - loss: 0.3376 - accuracy:
Epoch 6/50
264/264 [==============================] - ETA: 0s - loss: 0.2997 - accuracy: 0.8654
Epoch 6: val_accuracy did not improve from 0.95852
264/264 [==============================] - 117s 443ms/step - loss: 0.2997 - accuracy:
Epoch 7/50
264/264 [==============================] - ETA: 0s - loss: 0.2781 - accuracy: 0.8847
Epoch 7: val_accuracy improved from 0.95852 to 0.97179, saving model to ./21jan-ddsm-
264/264 [==============================] - 120s 454ms/step - loss: 0.2781 - accuracy:
Epoch 8/50
264/264 [==============================] - ETA: 0s - loss: 0.2517 - accuracy: 0.8968
Epoch 8: val_accuracy did not improve from 0.97179
264/264 [==============================] - 121s 457ms/step - loss: 0.2517 - accuracy:
Epoch 9/50
264/264 [==============================] - ETA: 0s - loss: 0.2340 - accuracy: 0.9051
Epoch 9: val_accuracy did not improve from 0.97179
264/264 [==============================] - 117s 442ms/step - loss: 0.2340 - accuracy:
Epoch 10/50
264/264 [==============================] - ETA: 0s - loss: 0.2294 - accuracy: 0.9037
Epoch 10: val_accuracy improved from 0.97179 to 0.98341, saving model to ./21jan-ddsm
264/264 [==============================] - 119s 450ms/step - loss: 0.2294 - accuracy:
Epoch 11/50
264/264 [==============================] - ETA: 0s - loss: 0.2129 - accuracy: 0.9152
Epoch 11: val_accuracy did not improve from 0.98341
264/264 [==============================] - 120s 443ms/step - loss: 0.2129 - accuracy:
Epoch 12/50
264/264 [==============================] - ETA: 0s - loss: 0.1971 - accuracy: 0.9215
Epoch 12: val_accuracy improved from 0.98341 to 0.98838, saving model to ./21jan-ddsm
264/264 [==============================] - 119s 451ms/step - loss: 0.1971 - accuracy:
Epoch 13/50
264/264 [==============================] - ETA: 0s - loss: 0.1843 - accuracy: 0.9300
Epoch 13: val_accuracy did not improve from 0.98838
264/264 [==============================] - 121s 457ms/step - loss: 0.1843 - accuracy:
Epoch 14/50
264/264 [==============================] - ETA: 0s - loss: 0.1755 - accuracy: 0.9344
```

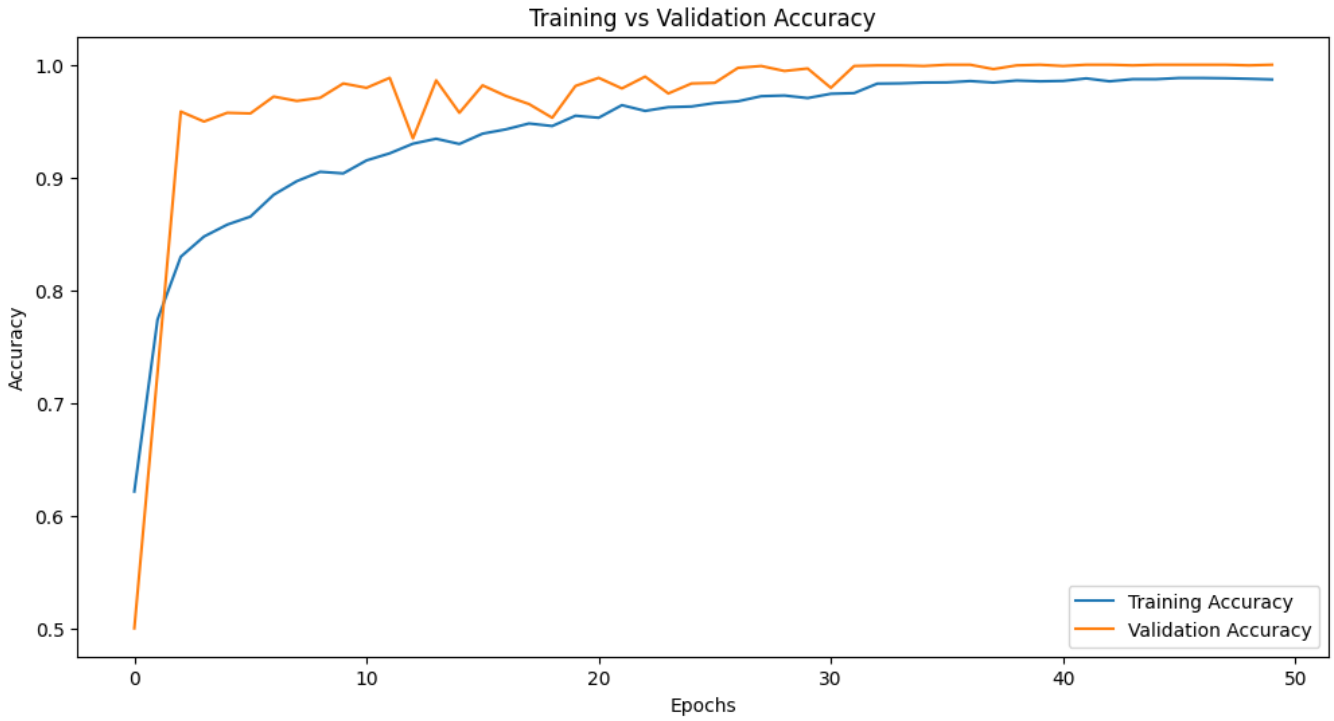
```
Epoch 14: val_accuracy did not improve from 0.98838
```

```
# Save the best model
model.save("/content/21jan-ddsm-bestmodel.h5")

# Download the best model
files.download("/content/21jan-ddsm-bestmodel.h5")
```

Downloading "21jan-ddsm-bestmodel.h5":

```
# Visualizations
# Training vs Validation Loss Graph
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training vs Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.savefig("/content/drive/MyDrive/DDSM/training_vs_validation_loss_21j.png")
plt.show()

# Learning Curve
plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training vs Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig("/content/drive/MyDrive/DDSM/training_vs_validation_accuracy_21j.png")
plt.show()
```


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epochs
0
1
2
3
4
0
10
20
30
40
50


Training vs Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epochs
0.5
0.6
0.7
0.8
0.9
1.0
0
10
20
30
40
50

```
# Load the best model
model = load_model("/content/21jan-ddsm-bestmodel.h5")
```

```
# Evaluation on the Test Set
# Create an ImageDataGenerator for rescaling
test_data = ImageDataGenerator(rescale=1 / 255)

test_generator = test_data.flow_from_directory(
    directory=test_path,
    target_size=(227, 227),
    batch_size=32,
    class_mode=None,  # Set to None for the test set
    shuffle=False
)

# Calculate the number of steps for evaluation
test_steps = int(np.ceil(test_generator.samples / test_generator.batch_size))

# Evaluate the model
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_steps)

# Predictions
y_pred = model.predict(test_generator, steps=test_steps)
y_true = test_generator.classes.astype(float)

# Metrics
accuracy = accuracy_score(y_true, np.round(y_pred))
precision = precision_score(y_true, np.round(y_pred))
recall = recall_score(y_true, np.round(y_pred))
f1 = f1_score(y_true, np.round(y_pred))
```