```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, BatchNormalization, Dropout
from matplotlib import pyplot as plt
import os
os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report
from tensorflow.keras.applications import DenseNet169 #ResNet50, ResNet101, MobileNetV2
from tensorflow.keras.regularizers import l2

# Data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    directory='picture/NewR22/train',
```

```
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary',
    subset='training'
)
val_generator = train_datagen.flow_from_directory(
    directory = 'picture/NewR22/train',
    target_size=(256,256),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)
#for evaluating results
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    directory = 'picture/NewR22/test',
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary',
    shuffle=False
)
# Loading the pre-trained model
base_model = DenseNet169(weights='imagenet', include_top=False, input_shape=(256,
256, 3))
# Freezing the base model
base_model.trainable = False
# Create the model
model = Sequential([
    base_model,
    Flatten(),
```

```
    Dense(256, activation='relu', kernel_regularizer=l2(0.03)),
    Dropout(0.4),
    Dense(1, activation='sigmoid', kernel_regularizer=l2(0.03))
])

model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(6
    train_generator,
    steps_per_epoch=train_gen.samples // train_gen.batch_size,
    validation_data=val_generator,
    validation_steps=val_generator.samples // val_generator.batch_size,
    epochs=20
)
# Plotting the training/validation accuracy and loss curves
plt.figure(figsize=(12, 4))
# Accuracy Plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
# Loss Plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')
```

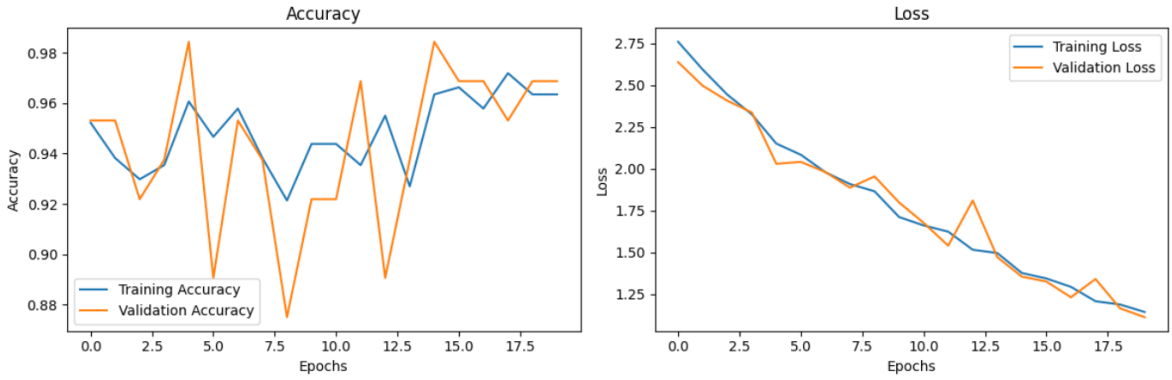
```
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()

def plot_predictions(generator, model, num_images=7):
    images, labels = next(generator)
    predictions = model.predict(images)
    predicted_classes = (predictions > 0.5).astype("int32")

    plt.figure(figsize=(15, 5))
    for i in range(num_images):
        plt.subplot(2, num_images, i + 1)
        plt.imshow(images[i])
        plt.title(f'True: {labels[i]}, Pred: {predicted_classes[i]}')
        plt.axis('off')
    plt.show()
plot_predictions(test_generator, model)
#Evaluating the model on test set
eval_results = model.evaluate(val_generator)
print(f"Test Loss: {eval_results[0]}, Test Accuracy: {eval_results[1]}")
#Classification Report
y_true = val_generator.classes
y_pred = (model.predict(val_generator) > 0.5).astype('int32').flatten()
print("\nClassification Report:")
print(classification_report(y_true,y_pred,target_names=val_generator.class_indices.keys(
)))
```

I got these result: **Test Loss: 1.2033337354660034, Test Accuracy: 0.9545454382896423**

Classification Report:

	precision	recall	f1-score	support
Defective	1.00	1.00	1.00	555
Normal	1.00	1.00	1.00	555
accuracy			1.00	555
macro avg	1.00	1.00	1.00	555
weighted avg	1.00	1.00	1.00	555

Accuracy and loss graph that I got after running these lines of code



These are some images where 0 representing Defective and 1 is representing Normal for both true and prediction: