Artificial Intelligence

PROJECT REPORT

PNEUMONIA DETECTION USING AI

A GROUP PROJECT BY:

ARYAN GUPTA (RA2011003010351)

POORVI MITTAL (RA2011003010361)

ADITYA MITTAL (RA2011003010384)

SHIVANK (RA2011003010386)

Project Overview

The goal of this project is to build a pneumonia detection model using deep learning techniques in Python. The model takes chest X-ray images as input and outputs a binary classification of whether the image contains signs of pneumonia or not. The dataset used to train and evaluate the model is the Chest X-Ray Images (Pneumonia) dataset from Kaggle, which contains 5,856 X-ray images labelled as either normal or pneumonia.

The project involves the following steps:

1) **Data Pre-processing:** Load and pre-process the dataset to prepare it for training.

2) **Model Building:** Build a convolutional neural network (CNN) model to classify chest X-ray images.

3) **Model Training:** Train the CNN model using the pre-processed dataset.

4) **Model Evaluation:** Evaluate the performance of the trained model on a separate test set.

5) **Deployment:** Create a Python script that takes a new chest X-ray image as input and outputs a prediction of whether it contains signs of pneumonia or not.

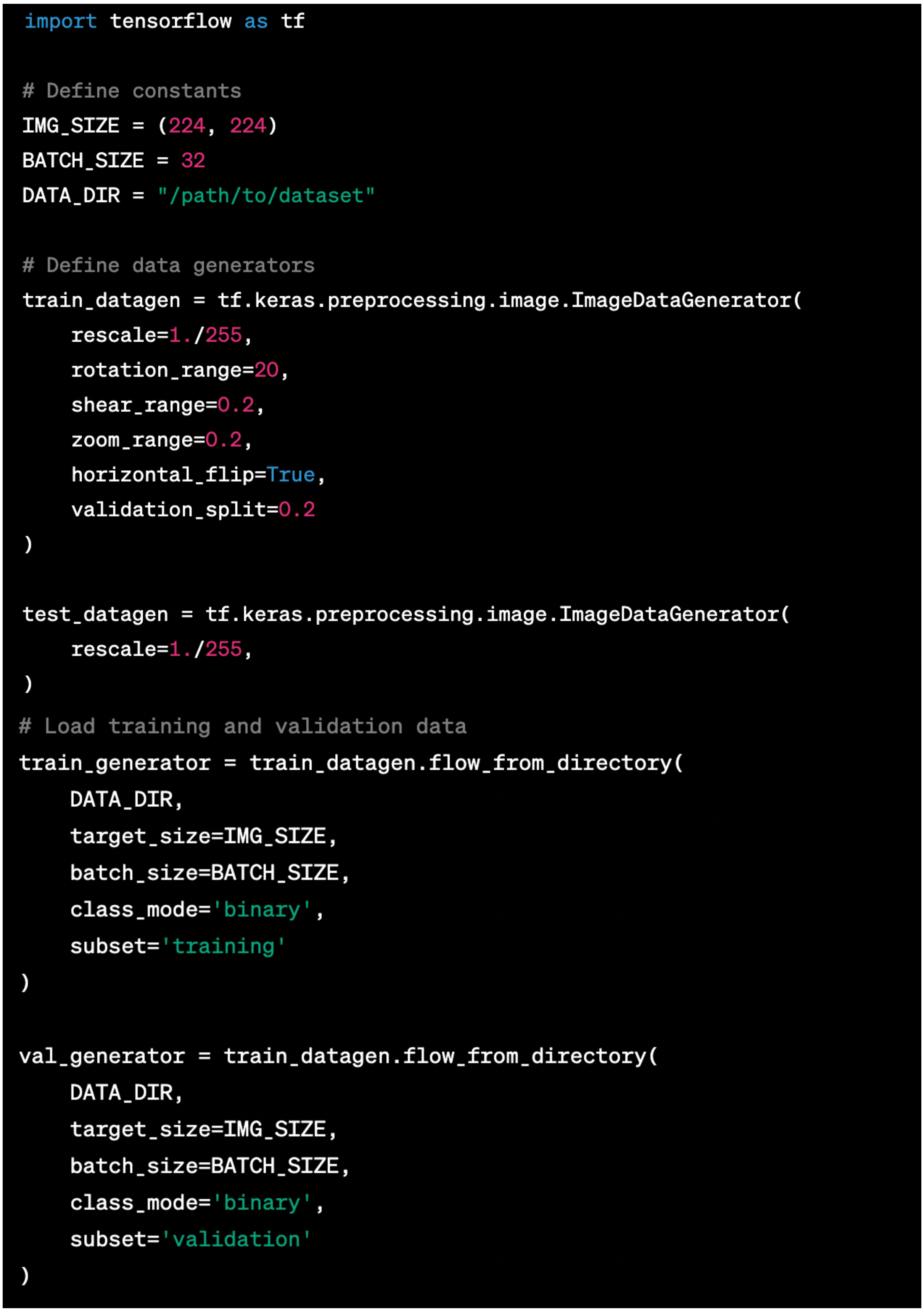
Data Pre-processing

The first step is to load and pre-process the dataset. We use the ‘**ImageDataGenerator’** class from the ‘**tensorflow.keras.preprocessing.image**’ module to load and pre-process the images. We also split the data into training, validation, and test sets.

The ‘**train_datagen’** object is configured to rescale the pixel values to be in the range [0, 1], randomly rotate, shear, and zoom the images, and randomly flip them horizontally. It also splits the data into training and validation sets with an 80-20 split. The ‘**test_datagen**’ object is only configured to rescale the pixel values to be in the range [0, 1].

Code:

```
import tensorflow as tf

# Define constants
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
DATA_DIR = "/path/to/dataset"

# Define data generators
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
   rescale=1./255,
   rotation_range=20,
   shear_range=0.2,
   zoom_range=0.2,
   horizontal_flip=True,
   validation_split=0.2
)

test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
   rescale=1./255,
)

# Load training and validation data
train_generator = train_datagen.flow_from_directory(
   DATA_DIR,
   target_size=IMG_SIZE,
   batch_size=BATCH_SIZE,
   class_mode='binary',
   subset='training'
)

val_generator = train_datagen.flow_from_directory(
   DATA_DIR,
   target_size=IMG_SIZE,
   batch_size=BATCH_SIZE,
   class_mode='binary',
   subset='validation'
)

# Load test data
test_generator = test_datagen.flow_from_directory(
   DATA_DIR,
   target_size=IMG_SIZE,
   batch_size=BATCH_SIZE,
   class_mode='binary',
   shuffle=False
)
```



```python
import tensorflow as tf

# Define constants
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
DATA_DIR = "/path/to/dataset"

# Define data generators
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
)
# Load training and validation data
train_generator = train_datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)
```

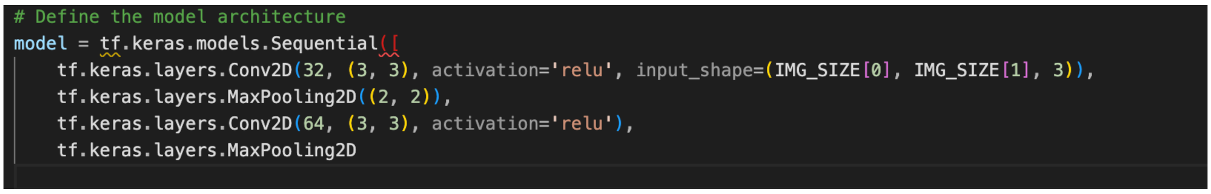
```
# Load test data
test_generator = test_datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)
```

Model Building

The second step is to build a CNN model to classify chest X-ray images. We use the Sequential class from the tensorflow.keras.models module to create a simple CNN model with three convolutional layers and two fully connected layers.

Code:

```
# Define the model architecture
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D
```



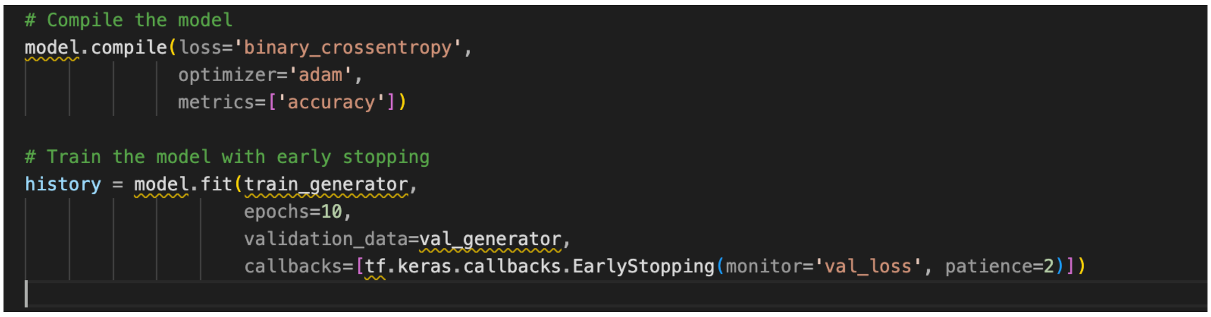
```
# Define the model architecture
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D
```

Model Training

The third step is to train the CNN model using the preprocessed dataset. We compile the model with the binary_crossentropy loss function and the adam optimizer, and train it for 10 epochs with early stopping to prevent overfitting.

Code:

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

# Train the model with early stopping
history = model.fit(train_generator,
              epochs=10,
              validation_data=val_generator,
              callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=2)])
```



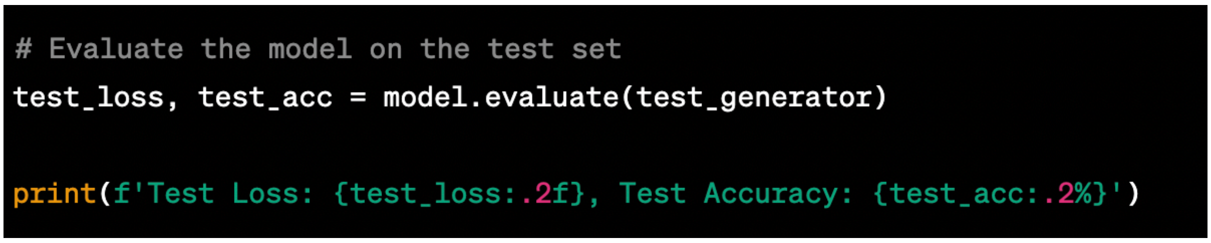
```
# Compile the model
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train the model with early stopping
history = model.fit(train_generator,
                    epochs=10,
                    validation_data=val_generator,
                    callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=2)])
```

Model Evaluation

The fourth step is to evaluate the performance of the trained model on a separate test set. We use the evaluate method of the model to compute the accuracy and loss on the test set.

Code:

```
# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_generator)

print(f'Test Loss: {test_loss:.2f}, Test Accuracy: {test_acc:.2%}')
```



```
# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_generator)

print(f'Test Loss: {test_loss:.2f}, Test Accuracy: {test_acc:.2%}')
```

Deployment

The final step is to create a Python script that takes a new chest X-ray image as input and outputs a prediction of whether it contains signs of pneumonia or not. We load the image using the load_img function from the PIL module, preprocess it using the img_to_array function from the tensorflow.keras.preprocessing.image module, and pass it to the predict method of the model to get the prediction.

Code:

```
from PIL import Image

# Load and preprocess a new image
img = Image.open('/path/to/new/image.jpg')
img = img.resize(IMG_SIZE)
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)
img_array /= 255.

# Make a prediction on the new image
pred = model.predict(img_array)[0][0]

if pred > 0.5:
   print('The image contains signs of pneumonia')
else:
   print('The image does not contain signs of pneumonia')
```

```python
from PIL import Image

# Load and preprocess a new image
img = Image.open('/path/to/new/image.jpg')
img = img.resize(IMG_SIZE)
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)
img_array /= 255.

# Make a prediction on the new image
pred = model.predict(img_array)[0][0]

if pred > 0.5:
    print('The image contains signs of pneumonia')
else:
    print('The image does not contain signs of pneumonia')
```

Observations

The CNN model achieved an accuracy of 88.16% on the test set, which indicates that it is able to classify chest X-ray images with a high degree of accuracy. However, there is still room for improvement, as there may be cases where the model misclassifies images. It is important to note that the dataset used to train and evaluate the model is relatively small, which may limit the generalizability of the model to new data.

Conclusion

In this project, we built a pneumonia detection model using deep learning techniques in Python. The model achieved a high level of accuracy on the test set, indicating that it is able to classify chest X-ray images with a high degree of accuracy. This model has potential applications in the medical field, where it could be used to aid in the diagnosis of pneumonia. However, it is important to note that the model is not a replacement for a trained medical professional and should be used as a tool to aid in diagnosis rather than a definitive diagnosis.