**Chapter 5**

# Testing and Evaluation

## 5.1 Introduction

In this chapter, I will discuss the testing of the software. I will mainly be using unit testing to test the different component of my GAN implementation. The test cases used in my unit testing will be further used as part of the Continuous Integration Pipeline (CI pipeline). This pipeline will be implemented in GitHub using GitHub actions.

In addition to that, the performance of the GAN model during the training phase will be evaluate using the FID score computed for a random batch at each training epoch.

## 5.2 Testing

Testing is an essential part of any software development process. It ensures that the software is running properly and checks if there are any bugs or errors that need to be fixed. I used Continuous Integration pipeline to automate necessary tests at a particular event or in a regular time intervals. Since I am using GitHub in my development, I run such automated tests when merging to the main branch. The main branch is very critical in GitHub and one must make sure that the code pushed or merges to such a branch is not faulty. My GitHub repo consists of two branches: Main branch and development branch. The development branch is used to edit necessary code before finalizing the implementation. The main branch, on the other hand is used to show the final version code. Hance, before merging into the main branch, we need to make sure that different components in the software system are working properly. In GitHub, there is a feature called GitHub actions that enables the tests to be run automatically once we push into the main branch using a pull request. If there is any error during the testing phase, we can cancel the pull request and make necessary fixes and changes.

A picture containing diagram

Description automatically generated

Figure : CI pipeline

### 5.2.1 Unit Testing

Unit testing is a form of software testing that focuses on testing individual units or components of a program, in isolation from the rest of the system. The goal is to verify that each unit behaves as expected and that the system as a whole works correctly. For my GAN system, there are 2 main unit tests that mainly checks the input and output shapes for both discriminator and generator. Hence I have 2 unit test that have been implemented using pytest. In my testing implementation I instantiated one generator and one discriminator and then test the input and output shapes. As mentioned earlier, the generator takes as input a vector of 100 inputs generated based on a random normal distribution and the output is an images of 128x128x3 pixels. There is an extra dimension in tensorflow that corresponds to image order in the minibatch. For the discriminator, the input is an image of size 128x128x3 pixels and the output s a single value that corresponds to the probability of the image being real or fake.

Text

Description automatically generated

Figure : Implementation of the unit tests in test.py

### 5.2.2 Continuous Integration with GitHub Actions

As mentioned earlier in this chapter, Continuous Integration runs necessary integration and testing steps at a regular time intervals or once an event occurs. I implemented the CI pipeline using GitHub actions. This will run the unit tests once we push or merge into the main branch. The CI workflow is defined using a yaml file. The code for the workflow file is shown below.

Text

Description automatically generated

Figure : CI workflow.

The first line defines the name of the workflow. The second section defines the type of events that will trigger the workflow. In my case, these events are when one pushes ore merges into the main branch. The second block set up the python version to run the tests which in my case is python 3.8. The next block will install necessary libraries defined in the requirements.txt file. This includes the tensorflow library. The last block will basically run the unit tests defined in test.py file.

## 5.3 Performance Evaluation

Besides unit testing it is very important to evaluate the GAN training performance using some well-established metrics from the literature. I used the FID score to evaluate the generator performance after each training epoch. The FID score, will compute the activations of the fake and real images based on a pretrained deep learning model (Inception-v3). After that it compares the distribution of values and returns the similarity between the fake images activations and the real ones. The lower the FID score the better the generator model is performing. Below I show the FID score for each training epoch where it can be clearly noted that later epochs have significantly lower FID score.

Chart, scatter chart

Description automatically generated

Figure : FID score for each training epoch.