National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Aniruddha Nandanwar
Student ID: x23104741

School of Computing
National College of Ireland

Supervisor: Syed Mohammed Raza Abidi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aniruddha Nandanwar
Student ID:	x23104741
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Syed Mohammed Raza Abidi
Submission Due Date:	16/09/2024
Project Title:	Configuration Manual
Word Count:	1738
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Aniruddha Nandanwar
Date:	16th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Aniruddha Nandanwar
x23104741

1 Introduction

This manual is created with stepwise implementation of how project implement worked and how model is created. In addition to it information and process of used libraries and tools required to carry out for project implementation is mentioned. Furthermore, information regarding specifications of location machine is mentioned in the manual. Each model information is mentioned in this configure manual.

2 HARDWARE CONFIGURATION

Below is the Hardware Configuration:-

A. Operating system: Windows >= 7
B. Processor: Intel i3
C. System Compatibility: 64-bit
D. Hard Disk: 1TB
E. RAM: 8/16 GB
.

3 SOFTWARE CONFIGURATIONS

3.1 Python==3.7.9

Python is a high-level, versatile programming language known for its readability and simplicity. It's favored for its ease of use and readability, making it an excellent choice for beginners and experienced developers alike. Python supports multiple programming paradigms (procedural, object-oriented, and functional), and its extensive standard library and third-party packages make it suitable for various applications, including web development, data analysis, machine learning, artificial intelligence, scientific computing, automation, and more. Its syntax emphasizes code readability and simplicity, enabling developers to write clear and concise code.

3.2 Visual Studio Code

Visual Studio Code (VS Code) is a popular, free source code editor developed by Microsoft. It's known for its lightweight nature, extensive customization options, and support for a wide range of programming languages and frameworks.

Key features of VS Code include:

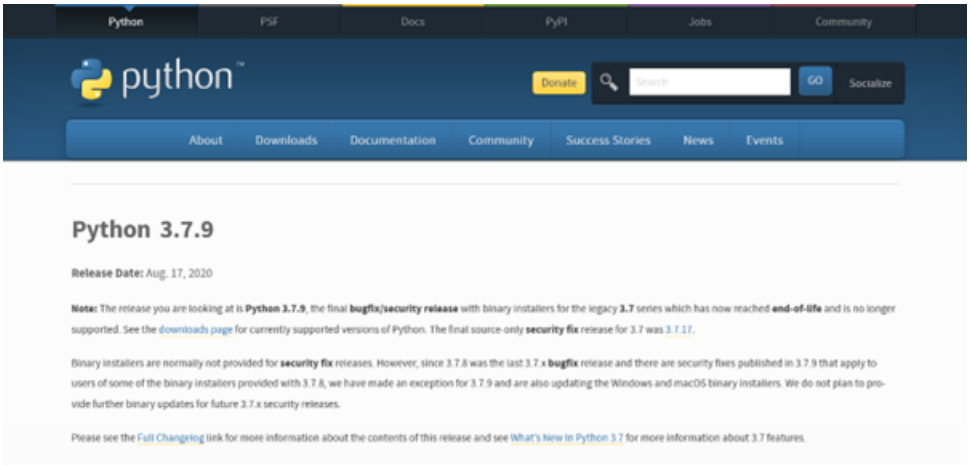


Figure 1: Python

1. Cross-platform: Available on Windows, macOS, and Linux, ensuring a consistent experience across different operating systems.

2. Extensions: Offers a rich ecosystem of extensions contributed by the community, enabling support for various languages, debugging tools, themes, and more.

3. Integrated Development Environment (IDE) features: Provides features like IntelliSense (code completion), debugging, Git integration, syntax highlighting, and code refactoring, enhancing developer productivity.

4. Customization: Highly customizable through themes, keyboard shortcuts, and extensions, allowing developers to tailor their coding environment to suit their preferences and needs.

5. Built-in Terminal: Includes a built-in terminal within the editor, enabling developers to run commands, scripts, and various tools without leaving the coding environment.

6. Performance: Known for its speed and responsiveness, even when handling large projects or multiple files simultaneously.

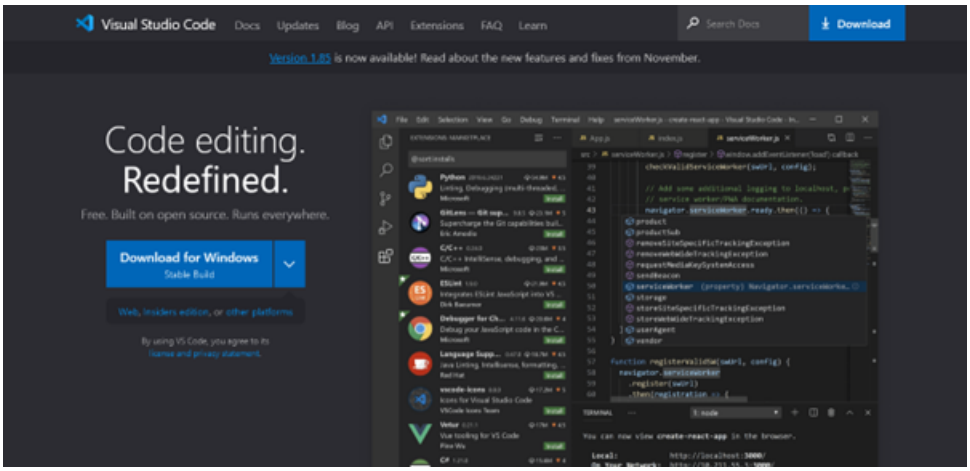


Figure 2: Visual Studio Code

3.3 Google Colab

Google Colab, short for Colaboratory, is a free cloud-based Jupyter notebook environment provided by Google. It allows users to write and execute Python code in a browser, eliminating the need to set up and configure development environments locally.

Key features of Google Colab include:

Free GPU/TPU support: Colab provides access to free GPU (Graphics Processing Unit) and TPU (Tensor Processing Unit) resources, enabling faster execution of machine learning and deep learning tasks. Jupyter Notebook Integration: It's built on top of Jupyter notebooks, allowing users to create and share documents that contain live code, equations, visualizations, and narrative text.

Easy Collaboration: Users can share their Colab notebooks just like Google Docs, allowing for real-time collaboration among multiple users.

Pre-installed Libraries: Comes with pre-installed libraries commonly used in data science and machine learning, such as NumPy, Pandas, Matplotlib, TensorFlow, and more. Cloud-Based: All computations are performed on Google's cloud servers, leveraging their computing power and storage.

Storage and Version Control: Integrates with Google Drive for easy storage and version control of notebooks.

Educational and Research Use: Widely used in academia and research for teaching, sharing code, experimenting with new ideas, and running data analysis or machine learning experiments.

Google Colab serves as an accessible and convenient platform, particularly for those interested in experimenting with machine learning models, performing data analysis, or collaborating on coding projectswithout the need for high-end hardware or complex setup procedures.

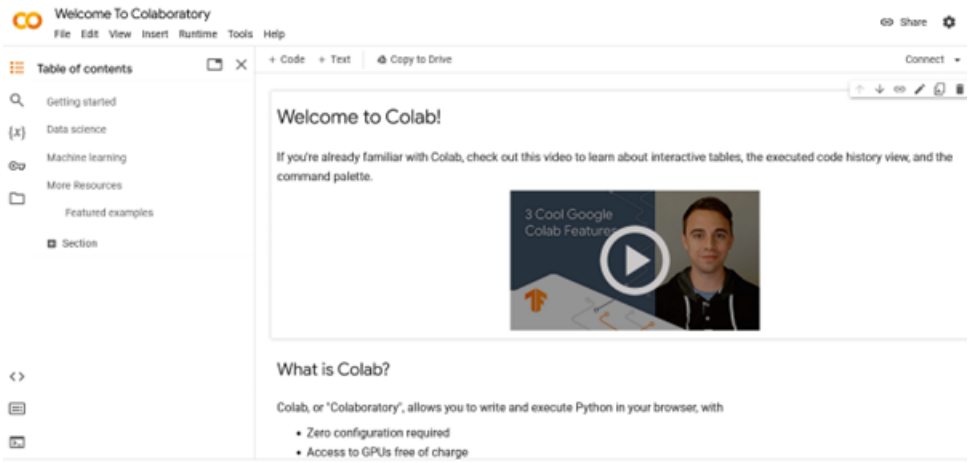


Figure 3: Google Colab

4 Libraries Configuration

i. Pandas: A powerful data manipulation and analysis library that provides easy-to-use data structures and tools for data cleaning, transformation, and analysis.

ii. NumPy: Fundamental package for numerical computing in Python. It provides support for arrays, matrices, mathematical functions, and operations.

iii. Matplotlib: A popular plotting library for creating static, interactive, and animated visualizations in Python. It offers a wide variety of plots and customization options.

iv. Seaborn: Built on top of Matplotlib, Seaborn provides a high-level interface for drawing attractive and informative statistical graphics.

v. Scikit-learn: A versatile machine learning library that provides simple and efficient tools for data mining and analysis. It includes a wide range of algorithms for classification, regression, clustering, etc.

vi. Flask: A framework for backend development that can be used to build web apps. Flask is a microframework written in Python that allows developers to implement wireframes and design specs into web apps. It has a modular and flexible structure that can be customized to meet project requirements, but it lacks a large toolbox of tools, so developers may need to manually add extensions.

vii. Imbalanced-learn (Imblearn): A library used for dealing with imbalanced datasets in machine learning. It provides techniques for oversampling, undersampling, and creating balanced datasets.

viii. TensorFlow: An open-source machine learning framework developed by Google. It's widely used for building and training machine learning models, especially deep learning models.

ix. Keras: An open-source neural network library. It's high-level and user-friendly, allowing for easy prototyping and experimentation with deep neural networks. (Note: It's often integrated with TensorFlow as of its integration into TensorFlow as tf.keras.)

x. Urllib: A Python library for opening URLs. It's used for fetching URLs, making requests, and handling responses.

xi. OpenCV (opencv-python): An open-source computer vision and machine learning software library. It provides tools and functions for image and video analysis, object detection, and more.

xii. Pillow: A Python Imaging Library (PIL) fork. It adds image processing capabilities to Python and supports opening, manipulating, and saving many different image file formats.

5 PROJECT IMPLEMENTATION

```python
# Imports
import tensorflow as tf
from tensorflow.keras.layers import Input, Reshape, Dropout, Dense, Concatenate
from tensorflow.keras.layers import Flatten, BatchNormalization
from tensorflow.keras.layers import Activation, ZeroPadding2D
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.layers import UpSampling2D, Conv2D, Conv2DTranspose
from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import plot_model
from tensorflow.keras import initializers
from sklearn.metrics import mean_squared_error

import numpy as np
from PIL import Image
from tqdm import tqdm
import os
import time
import matplotlib.pyplot as plt
import pickle
```

Figure 4: Importing all Libraries

Figure 5: Loading Image Data

Figure 6: Preprocessing

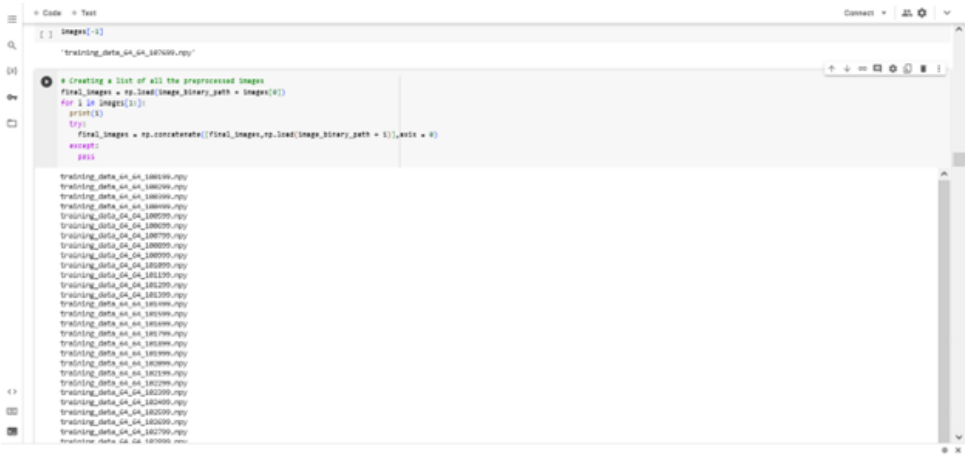


Figure 7: Saving Caption Data into a CSV File

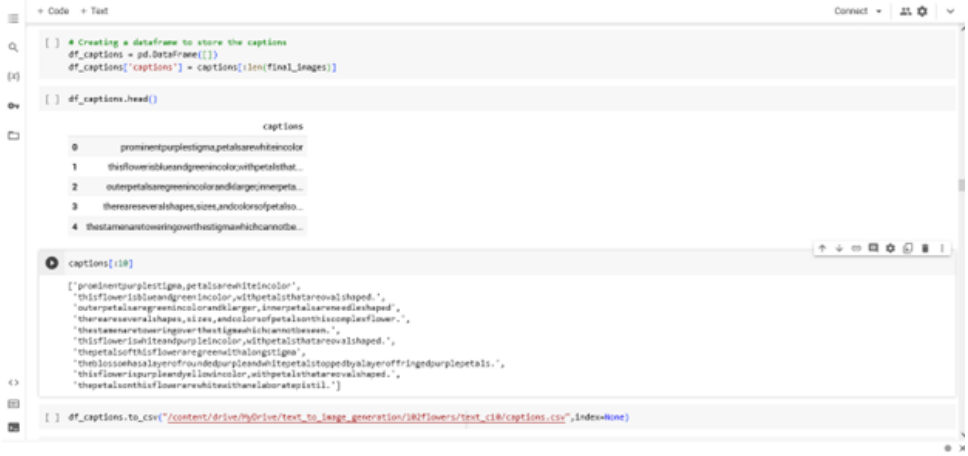


Figure 8: Generator Layer

Figure 9: Discriminator Layer

```
[ ] def build_discriminator_func(image_shape, embedding_size):
      input_shape = Input(shape=image_shape)
      input_embed = Input(shape=embedding_size)

      conv2d1 = Conv2D(32,kernel_size=4,strides=2,input_shape=image_shape,padding="same",kernel_initializer=initializers.RandomNormal(stddev=0.02))(input_shape)
      leaky1 = LeakyReLU(alpha=0.2)(conv2d1)

      drop2 = Dropout(0.25)(leaky1)
      conv2d2 = Conv2D(64, kernel_size=4, strides=2, padding="same",kernel_initializer=initializers.RandomNormal(stddev=0.02))(drop2)
      # zero2 = ZeroPadding2D(padding=((0,1),(0,1)))(conv2d2)
      batchNorm2 = BatchNormalization(momentum=0.8)(conv2d2)
      leaky2 = LeakyReLU(alpha=0.2)(batchNorm2)

      drop3 = Dropout(0.25)(leaky2)
      conv2d3 = Conv2D(128, kernel_size=4, strides=2, padding="same",kernel_initializer=initializers.RandomNormal(stddev=0.02))(drop3)
      batchNorm3 = BatchNormalization(momentum=0.8)(conv2d3)
      leaky3 = LeakyReLU(alpha=0.2)(batchNorm3)

      drop4 = Dropout(0.25)(leaky3)
      conv2d4 = Conv2D(256, kernel_size=4, strides=2, padding="same",kernel_initializer=initializers.RandomNormal(stddev=0.02))(drop4)
      batchNorm4 = BatchNormalization(momentum=0.8)(conv2d4)
      leaky4 = LeakyReLU(alpha=0.2)(batchNorm4)

      dense_embed = Dense(128,kernel_initializer=initializers.RandomNormal(stddev=0.02))(input_embed)
      leaky_embed = LeakyReLU(alpha=0.2)(dense_embed)
      reshape_embed = Reshape((4,4,8))(leaky_embed)
      merge_embed = Concatenate()([leaky4, reshape_embed])

      drop5 = Dropout(0.25)(merge_embed)
      conv2d5 = Conv2D(512, kernel_size=4,kernel_initializer=initializers.RandomNormal(stddev=0.02))(drop5)
      batchNorm5 = BatchNormalization(momentum=0.8)(conv2d5)
      leaky5 = LeakyReLU(alpha=0.2)(batchNorm5)

      drop6 = Dropout(0.25)(leaky5)
```

Figure 10: Loss and Optimizer

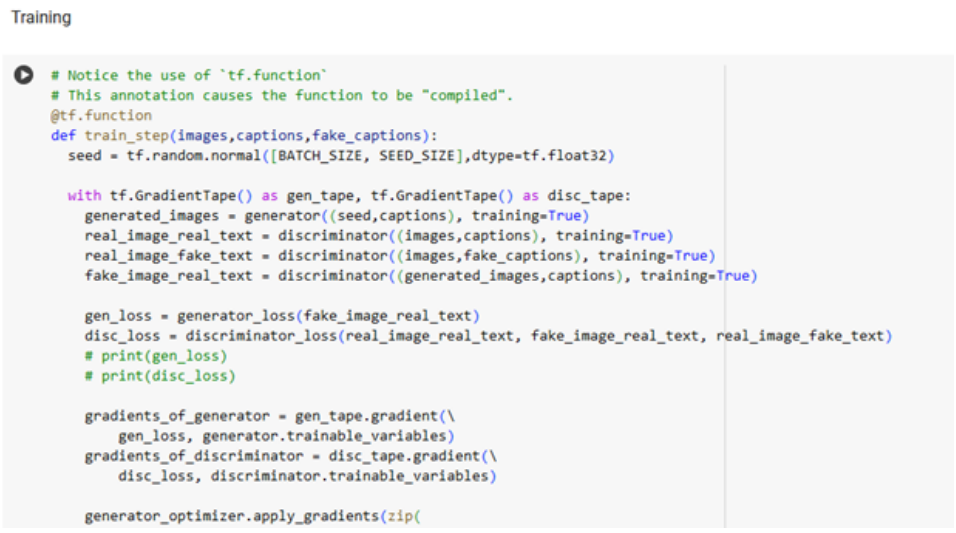
```
[ ]   filename = os.path.join(output_path,f"train-{cnt}.png")
      im = Image.fromarray(image_array)
      im.save(filename)
```

```
# This method returns a helper function to compute cross entropy loss
cross_entropy = tf.keras.losses.BinaryCrossentropy()

def discriminator_loss(real_image_real_text, fake_image_real_text, real_image_fake_text):
    real_loss = cross_entropy(tf.random.uniform(real_image_real_text.shape,0.8,1.0), real_image_real_text)
    fake_loss = (cross_entropy(tf.random.uniform(fake_image_real_text.shape,0.0,0.2), fake_image_real_text) +
                 cross_entropy(tf.random.uniform(real_image_fake_text.shape,0.0,0.2), real_image_fake_text))/2

    total_loss = real_loss + fake_loss
    return total_loss

def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)
```

+ Code + Text

```
[ ] # lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    #     initial_learning_rate=2e-4,
    #     decay_steps=100,
    #     decay_rate=0.5)
    generator_optimizer = tf.keras.optimizers.Adam(learning_rate=2.0e-4,beta_1 = 0.5)
    discriminator_optimizer = tf.keras.optimizers.Adam(learning_rate=2.0e-4,beta_1 = 0.5)
```

Figure 11: Model Training



Training

```
# Notice the use of `tf.function`
# This annotation causes the function to be "compiled".
@tf.function
def train_step(images,captions,fake_captions):
  seed = tf.random.normal([BATCH_SIZE, SEED_SIZE],dtype=tf.float32)

  with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
    generated_images = generator((seed,captions), training=True)
    real_image_real_text = discriminator((images,captions), training=True)
    real_image_fake_text = discriminator((images,fake_captions), training=True)
    fake_image_real_text = discriminator((generated_images,captions), training=True)

    gen_loss = generator_loss(fake_image_real_text)
    disc_loss = discriminator_loss(real_image_real_text, fake_image_real_text, real_image_fake_text)
    # print(gen_loss)
    # print(disc_loss)

    gradients_of_generator = gen_tape.gradient(\
        gen_loss, generator.trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(\
        disc_loss, discriminator.trainable_variables)

    generator_optimizer.apply_gradients(zip(
```

Figure 12: Model Training

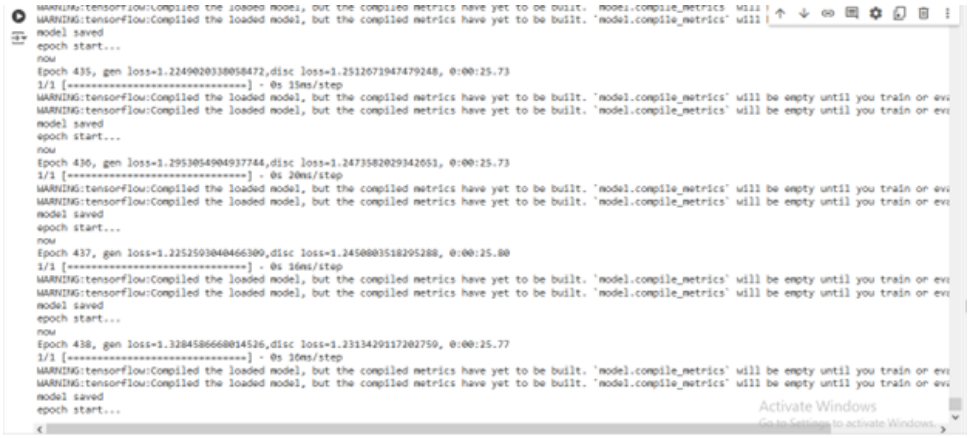
```python
# Function that executes trianing process
def train(train_dataset, epochs):
  fixed_seed = np.random.normal(0, 1, (PREVIEW_ROWS * PREVIEW_COLS,
                                       SEED_SIZE))
  fixed_embed = save_images_embeddings

  start = time.time()

  for epoch in range(epochs):
    print("epoch start...")
    epoch_start = time.time()

    gen_loss_list = []
    disc_loss_list = []

    for batch in train_dataset[:-1]:
      # train_batch = training_data[BATCH_SIZE*image_batch : BATCH_SIZE*image_batch + BATCH_SIZE]
      # caption_batch = captions[BATCH_SIZE*image_batch : BATCH_SIZE*image_batch + BATCH_SIZE]
      train_batch = batch['images']
      caption_batch = batch['embeddings']

      fake_caption_batch = np.copy(caption_batch)
      np.random.shuffle(fake_caption_batch)

      t = train_step(train_batch,caption_batch,fake_caption_batch)
      # print(t)
      gen_loss_list.append(t[0])
      disc_loss_list.append(t[1])
      # if image_batch%50 == 0:
      #   print(image_batch)
```

Figure 13: Model Training Output

```
#training
train(list(train_dataset.as_numpy_iterator()), EPOCHS)
```

```
epoch start...
now
Epoch 1, gen loss=0.7801750302314758,disc loss=1.552404522895813, 0:00:38.40
1/1 [==============================] - 1s 670ms/step
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
epoch start...
now
Epoch 2, gen loss=0.7527354359626770,disc loss=1.4824262857437134, 0:00:25.55
1/1 [==============================] - 0s 20ms/step
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
epoch start...
now
Epoch 3, gen loss=0.7659760117530823,disc loss=1.4517611265182495, 0:00:26.17
1/1 [==============================] - 0s 16ms/step
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
epoch start...
now
Epoch 4, gen loss=0.7343311309814453,disc loss=1.435768961906433, 0:00:25.67
1/1 [==============================] - 0s 16ms/step
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
```

Figure 14: Model Training Output



```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
epoch start...
now
Epoch 435, gen loss=1.2249020338058472,disc loss=1.2512671947479248, 0:00:25.73
1/1 [==============================] - 0s 15ms/step
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
epoch start...
now
Epoch 436, gen loss=1.2953054904937744,disc loss=1.2473582029342651, 0:00:25.73
1/1 [==============================] - 0s 20ms/step
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
epoch start...
now
Epoch 437, gen loss=1.2252593040466309,disc loss=1.2450803518295288, 0:00:25.80
1/1 [==============================] - 0s 16ms/step
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
epoch start...
now
Epoch 438, gen loss=1.3284586668014526,disc loss=1.2313429117202759, 0:00:25.77
1/1 [==============================] - 0s 16ms/step
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
model saved
epoch start...
```

Figure 15: Performance Evaluation

Figure 16: Output

Figure 17: Output

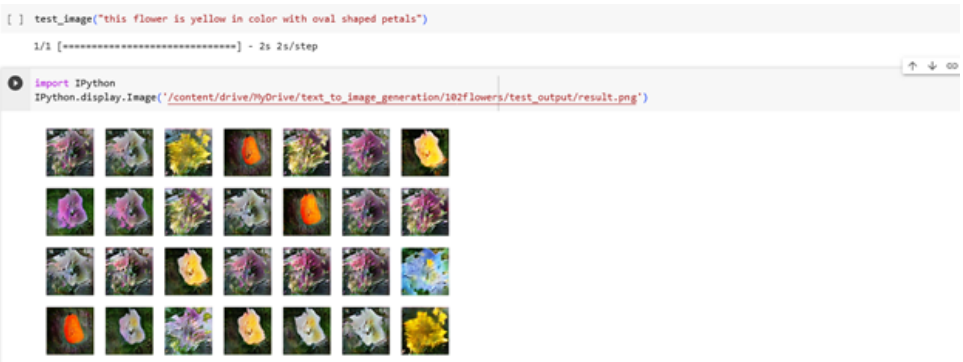


Figure 18: GUI Flask Screen

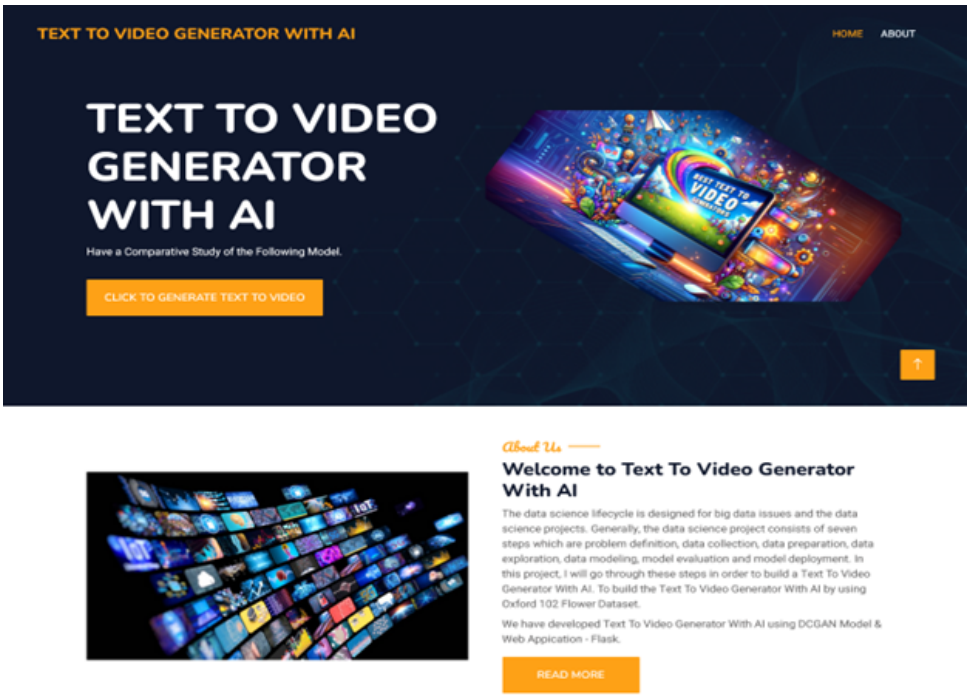


Figure 19: GUI Flask Screen Output

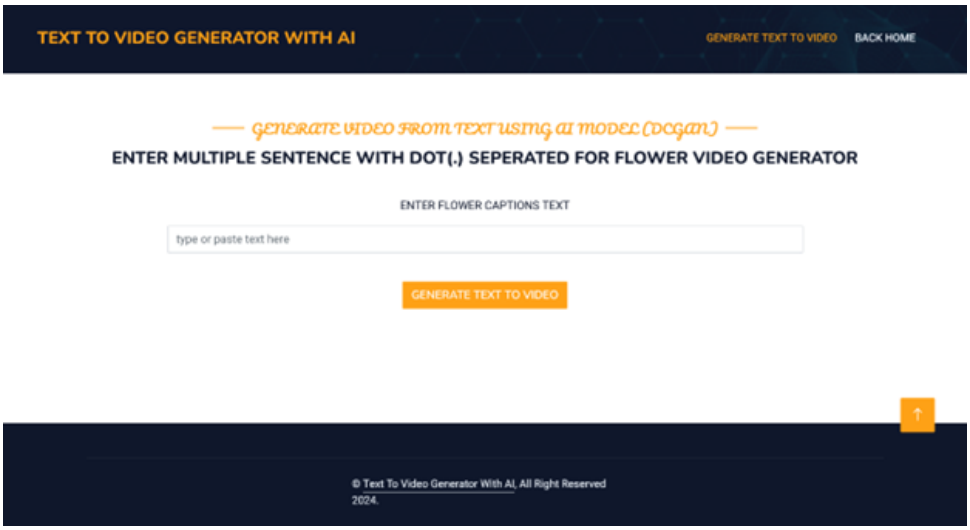


Figure 20: GUI Flask Screen Output

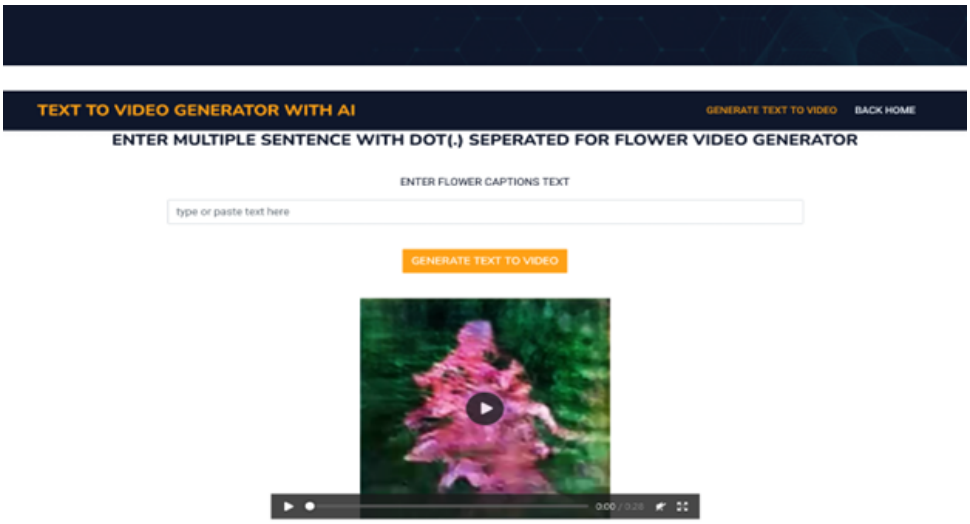


Figure 21: GUI Flask Screen Output

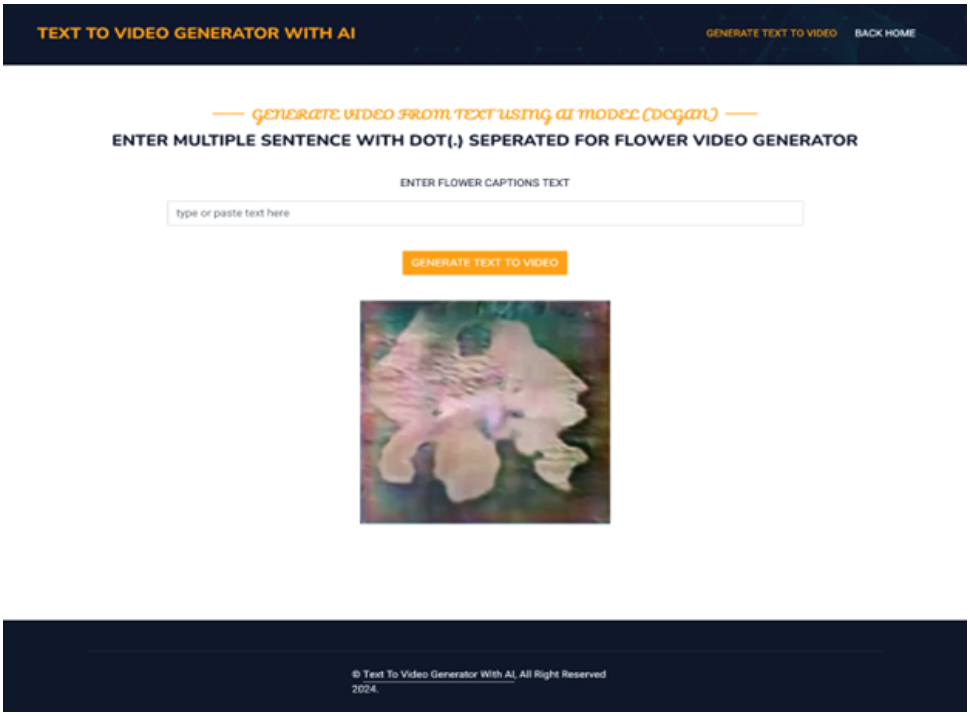


Figure 22: GUI Flask Screen Output