anime-faces-generation-using-dcgan

October 26, 2024

#

Generating Anime Faces with DCGAN: A Dataset of High-Quality Anime

0.0.1 Table of contents:

0.1 Introduction

Generating Anime Faces with DCGAN: A Dataset of High-Quality Anime Girls

In this project, we aim to generate high-quality anime faces using the Deep Convolutional Generative Adversarial Networks (DCGAN) algorithm. The dataset used for this project consists of 63,632 anime faces, carefully curated to ensure quality and appeal. The motivation behind this project is to fulfill the simple dream of generating perfect waifus, cute female anime faces that capture the essence of the anime art style. The DCGAN algorithm offers an attractive approach to generating realistic and visually appealing anime faces. By training the generator and discriminator networks in an adversarial learning process, we can learn a hierarchy of representations, starting from object parts and progressing to scenes. This allows us to create compelling and diverse anime face images. To showcase the capabilities of the project, we provide examples of both real and generated anime face images. By comparing the "real vs. fake" images, viewers can appreciate the quality and realism achieved through the DCGAN algorithm. By combining the power of DCGAN and a meticulously curated anime face dataset, we aim to contribute to the world of anime art and provide a valuable resource for researchers, artists, and fans alike. Join us on this exciting journey to generate captivating anime faces and bring your favorite characters to life!

[6]:
```
!pip install -r requirement.txt
```

```
Requirement already satisfied: tensorflow==2.8.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from -r
requirement.txt (line 1)) (2.8.0)
```

```
Requirement already satisfied: numpy==1.21.2 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from -r
requirement.txt (line 2)) (1.21.2)
Requirement already satisfied: pandas==1.3.3 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from -r
requirement.txt (line 3)) (1.3.3)
Requirement already satisfied: seaborn==0.11.2 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from -r
requirement.txt (line 4)) (0.11.2)
Requirement already satisfied: Pillow==8.3.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from -r
requirement.txt (line 5)) (8.3.1)
Requirement already satisfied: protobuf==3.20.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from -r
requirement.txt (line 6)) (3.20.1)
Requirement already satisfied: absl-py>=0.4.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (1.6.3)
Requirement already satisfied: flatbuffers>=1.12 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (24.3.25)
Requirement already satisfied: gast>=0.2.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (3.12.1)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (1.1.2)
Requirement already satisfied: libclang>=9.0.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (3.4.0)
Requirement already satisfied: setuptools in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (75.1.0)
Requirement already satisfied: six>=1.12.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (1.16.0)
```

```
Requirement already satisfied: termcolor>=1.1.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (1.16.0)
Requirement already satisfied: tensorboard<2.9,>=2.8 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (2.8.0)
Requirement already satisfied: tf-estimator-nightly==2.8.0.dev2021122109 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (2.8.0.dev2021122109)
Requirement already satisfied: keras<2.9,>=2.8.0rc0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (2.8.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (0.37.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorflow==2.8.0->-r requirement.txt (line 1)) (1.67.0)
Requirement already satisfied: python-dateutil>=2.7.3 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
pandas==1.3.3->-r requirement.txt (line 3)) (2.9.0)
Requirement already satisfied: pytz>=2017.3 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
pandas==1.3.3->-r requirement.txt (line 3)) (2024.2)
Requirement already satisfied: scipy>=1.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
seaborn==0.11.2->-r requirement.txt (line 4)) (1.10.1)
Requirement already satisfied: matplotlib>=2.2 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
seaborn==0.11.2->-r requirement.txt (line 4)) (3.8.4)
WARNING: tensorflow 2.8.0 does not provide the extra 'keras'

Requirement already satisfied: wheel<1.0,>=0.23.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
astunparse>=1.6.0->tensorflow==2.8.0->-r requirement.txt (line 1)) (0.44.0)
Requirement already satisfied: contourpy>=1.0.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
matplotlib>=2.2->seaborn==0.11.2->-r requirement.txt (line 4)) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
matplotlib>=2.2->seaborn==0.11.2->-r requirement.txt (line 4)) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
```

```
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
matplotlib>=2.2->seaborn==0.11.2->-r requirement.txt (line 4)) (4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
matplotlib>=2.2->seaborn==0.11.2->-r requirement.txt (line 4)) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
matplotlib>=2.2->seaborn==0.11.2->-r requirement.txt (line 4)) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
matplotlib>=2.2->seaborn==0.11.2->-r requirement.txt (line 4)) (3.2.0)
Requirement already satisfied: importlib-resources>=3.2.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
matplotlib>=2.2->seaborn==0.11.2->-r requirement.txt (line 4)) (6.4.5)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt (line 1)) (2.35.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt (line 1)) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt (line 1)) (3.7)
Requirement already satisfied: requests<3,>=2.21.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt (line 1)) (2.32.3)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt (line 1)) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt (line 1)) (1.8.1)
Requirement already satisfied: werkzeug>=0.11.15 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt (line 1)) (3.0.6)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt
(line 1)) (5.5.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt
(line 1)) (0.4.1)
Requirement already satisfied: rsa<5,>=3.1.4 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt
(line 1)) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
```

```
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r
requirement.txt (line 1)) (2.0.0)
Requirement already satisfied: zipp>=3.1.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from importlib-
resources>=3.2.0->matplotlib>=2.2->seaborn==0.11.2->-r requirement.txt (line 4))
(3.20.2)
Requirement already satisfied: importlib-metadata>=4.4 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
markdown>=2.6.8->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt
(line 1)) (8.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r
requirement.txt (line 1)) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r
requirement.txt (line 1)) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r
requirement.txt (line 1)) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r
requirement.txt (line 1)) (2024.8.30)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
werkzeug>=0.11.15->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt
(line 1)) (3.0.2)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from
pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r requirement.txt
(line 1)) (0.6.1)
Requirement already satisfied: oauthlib>=3.0.0 in
/home/chris/anaconda3/envs/GENAI/lib/python3.9/site-packages (from requests-
oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8->tensorflow==2.8.0->-r
requirement.txt (line 1)) (3.2.2)
```

[7]:
```
#!export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python
```

[8]:
```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
# import protobuf
```

```
# print("protobuf version:", protobuf.__version__)
```

```
TensorFlow version: 2.8.0
```

0.2 import Libraries

[9]:
```
# import requirement libraries and tools
from tensorflow import keras
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style= "darkgrid", color_codes = True)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten,␣
↪Conv2DTranspose, Reshape, BatchNormalization, Dropout, Input, ReLU, LeakyReLU
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import BinaryCrossentropy
from PIL import Image

import warnings
warnings.filterwarnings('ignore')
```

0.3 Import DataSet

Tabel of Contents

[10]:
```
# Loading and Preparing Anime Face Images Dataset using Keras Image Data␣
↪Generator
img_width, img_height = 256, 256
batchsize = 32

train = keras. utils.image_dataset_from_directory(
    directory='animefacedataset',
    batch_size = batchsize,
    image_size = (img_width, img_height))
```

```
Found 63565 files belonging to 1 classes.
```

```
2024-10-26 05:03:08.767177: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2024-10-26 05:03:08.781626: W
```

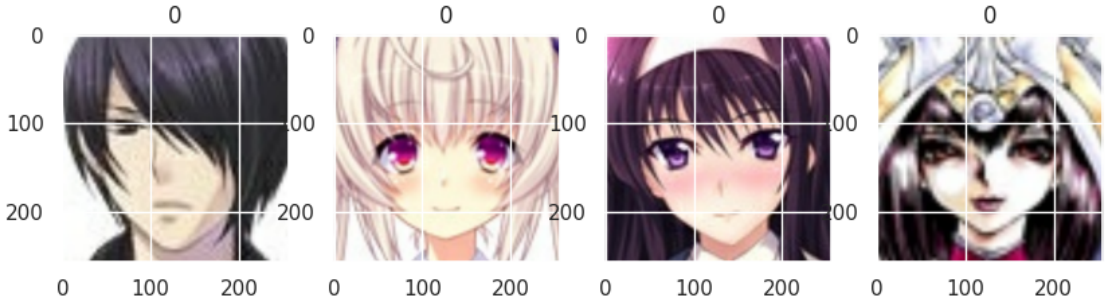
```
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libcudnn.so.8'; dlerror: libcudnn.so.8: cannot open shared
object file: No such file or directory; LD_LIBRARY_PATH:
/usr/local/cuda-11.8/lib64:
2024-10-26 05:03:08.781641: W
tensorflow/core/common_runtime/gpu/gpu_device.cc:1850] Cannot dlopen some GPU
libraries. Please make sure the missing libraries mentioned above are installed
properly if you would like to use GPU. Follow the guide at
https://www.tensorflow.org/install/gpu for how to download and setup the
required libraries for your platform.
Skipping registering GPU devices…
2024-10-26 05:03:08.782035: I tensorflow/core/platform/cpu_feature_guard.cc:151]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

0.4 Preprocessing

Tabel of Contents

[11]:
```
# Visualizing a Batch of Anime Face Images

data_iterator = train.as_numpy_iterator()
batch = data_iterator.next()
fig, ax = plt.subplots(ncols=4, figsize=(10,10))
for idx, img in enumerate(batch[0][:4]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```



[12]:
```
#!pip install kaggle
#!kaggle datasets download -d splcher/animefacedataset
#!unzip animefacedataset.zip -d animefacedataset
```

[13]:
```
# Generating Augmented Batches of Anime Face Images using ImageDataGenerator
DIR = 'animefacedataset' #path

# Create an ImageDataGenerator object with data augmentation options for image␣
 ↪preprocessing
train_datagen = ImageDataGenerator(rescale=1./255,
                                   horizontal_flip = True)

train_generator = train_datagen.flow_from_directory(
        DIR,
        target_size = (64, 64),
        batch_size = batchsize,
        class_mode = None)

#train_generator[0]
```

```
Found 63565 images belonging to 1 classes.
```

Deep Convolutional Generative Adversarial Network

DCGAN (Deep Convolutional Generative Adversarial Network) is an advanced architecture and training methodology for generative adversarial networks (GANs) specifically designed for image synthesis tasks. It combines deep convolutional neural networks with the adversarial learning framework to generate high-quality and realistic images.

0.5 Create Generator

Tabel of Contents

The Generator

In DCGAN, the generator and discriminator networks play crucial roles. The generator is responsible for generating synthetic images that resemble the target data distribution. It takes random noise as input and gradually transforms it into higher-dimensional outputs using convolutional layers, transposed convolutions, and activation functions like ReLU. Batch normalization is often used to stabilize the learning process.

[14]:
```
# Creating the Generator Model

KI = keras.initializers.RandomNormal(mean=0.0, stddev=0.02)
input_dim = 300

def Generator_Model():

    Generator = Sequential()

    # Random noise
    Generator.add(Dense(8 * 8 * 512, input_dim = input_dim))
    Generator.add(ReLU())
    # Convert 1d to 3d
```

```
    Generator.add(Reshape((8, 8, 512)))
    # Unsample
    Generator.add(Conv2DTranspose(256, (4, 4), strides=(2, 2), padding='same',␣
 ↪kernel_initializer=KI, activation='ReLU'))
    Generator.add(Conv2DTranspose(128, (4, 4), strides=(2, 2), padding='same',␣
 ↪kernel_initializer=KI, activation='ReLU'))
    Generator.add(Conv2DTranspose(64, (4, 4), strides=(2, 2), padding='same',␣
 ↪kernel_initializer=KI, activation='ReLU'))
    Generator.add(Conv2D(3, (4, 4), padding='same', activation='sigmoid'))


    return Generator

generator = Generator_Model()
generator.summary()
# Visualized Layers of generator
keras.utils.plot_model(generator, show_shapes=True)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32768)             9863168

 re_lu (ReLU)                (None, 32768)             0

 reshape (Reshape)           (None, 8, 8, 512)         0

 conv2d_transpose (Conv2DTra  (None, 16, 16, 256)      2097408
 nspose)

 conv2d_transpose_1 (Conv2DT  (None, 32, 32, 128)      524416
 ranspose)

 conv2d_transpose_2 (Conv2DT  (None, 64, 64, 64)       131136
 ranspose)

 conv2d (Conv2D)             (None, 64, 64, 3)         3075

=================================================================
Total params: 12,619,203
Trainable params: 12,619,203
Non-trainable params: 0
_________________________________________________________________
You must install pydot (`pip install pydot`) and install graphviz (see
instructions at https://graphviz.gitlab.io/download/) for
plot_model/model_to_dot to work.
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32768)             9863168

 re_lu (ReLU)                (None, 32768)             0

 reshape (Reshape)           (None, 8, 8, 512)         0

 conv2d_transpose (Conv2DTra  (None, 16, 16, 256)      2097408
 nspose)

 conv2d_transpose_1 (Conv2DT  (None, 32, 32, 128)      524416
 ranspose)

 conv2d_transpose_2 (Conv2DT  (None, 64, 64, 64)       131136
 ranspose)

 conv2d (Conv2D)             (None, 64, 64, 3)         3075

=================================================================
Total params: 12,619,203
Trainable params: 12,619,203
Non-trainable params: 0
_________________________________________________________________
You must install pydot (`pip install pydot`) and install graphviz (see
instructions at https://graphviz.gitlab.io/download/) for
plot_model/model_to_dot to work.
```

0.6 Create Discriminator

Tabel of Contents

The Discriminator

The discriminator, on the other hand, aims to distinguish between real and generated images. It utilizes convolutional layers, activation functions, and strided convolutions to downsample the spatial dimensions and capture image features. The discriminator is trained to maximize its ability to correctly classify images as real or fake.

[15]:
```python
# Creating the discriminator Model

def Discriminator_Model():
    input_shape = (64, 64, 3)

    # Create a Sequential model
    discriminator = Sequential()
    discriminator.add(Conv2D(64,kernel_size=(3, 3), activation='LeakyReLU',␣
 ↪input_shape = input_shape))
```

```
    discriminator.add(MaxPooling2D(pool_size=(2, 2)))
    discriminator.add(Conv2D(128, kernel_size=(3, 3), activation='LeakyReLU'))
    discriminator.add(MaxPooling2D(pool_size=(2, 2)))
    discriminator.add(Conv2D(256, kernel_size=(3, 3), activation='LeakyReLU'))
    discriminator.add(MaxPooling2D(pool_size=(2, 2)))
    discriminator.add(Flatten())
    discriminator.add(Dense(256, activation='LeakyReLU'))
    discriminator.add(Dense(1, activation='sigmoid'))

    return discriminator

# Training The CNN
discriminator = Discriminator_Model()
discriminator.summary()
# Visualized Layers of discriminator
keras.utils.plot_model(discriminator, show_shapes=True)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 62, 62, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 31, 31, 64)       0
 )

 conv2d_2 (Conv2D)           (None, 29, 29, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 12, 12, 256)       295168

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 256)        0
 2D)

 flatten (Flatten)           (None, 9216)              0

 dense_1 (Dense)             (None, 256)               2359552

 dense_2 (Dense)             (None, 1)                 257

=================================================================
Total params: 2,730,625
Trainable params: 2,730,625
Non-trainable params: 0
_________________________________________________________________
```

```
You must install pydot (`pip install pydot`) and install graphviz (see
instructions at https://graphviz.gitlab.io/download/) for
plot_model/model_to_dot to work.
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 62, 62, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 31, 31, 64)       0
 )

 conv2d_2 (Conv2D)           (None, 29, 29, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 12, 12, 256)       295168

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 256)        0
 2D)

 flatten (Flatten)           (None, 9216)              0

 dense_1 (Dense)             (None, 256)               2359552

 dense_2 (Dense)             (None, 1)                 257

=================================================================
Total params: 2,730,625
Trainable params: 2,730,625
Non-trainable params: 0
_________________________________________________________________
```

```
You must install pydot (`pip install pydot`) and install graphviz (see
instructions at https://graphviz.gitlab.io/download/) for
plot_model/model_to_dot to work.
```

0.7 Create Deep Convolutional GAN

Tabel of Contents

The Training Process

The training process of DCGAN involves an adversarial interplay between the generator and discriminator. The generator aims to generate increasingly realistic images to deceive the discriminator, while the discriminator strives to improve its discrimination ability. This iterative process continues until the generator produces images that are visually convincing and indistinguishable from real images. During training, the generator and discriminator networks are updated using techniques like stochastic gradient descent (SGD) or Adam optimization. The Binary Cross Entropy loss function is commonly used to compute the difference between predicted probabilities and

target labels for both networks.

[16]:
```
# DCGAN Model Training Step with Discriminator and Generator

class DCGAN(keras.Model):
    def __init__(self, generator, discriminator, latent_dim = input_dim):
        super().__init__()
        self.generator = generator
        self.discriminator = discriminator
        self.latent_dim = latent_dim
        self.g_loss_metric = keras.metrics.Mean(name='g_loss')
        self.d_loss_metric = keras.metrics.Mean(name='d_loss')

    @property
    def metrics(self):
        return [self.g_loss_metric, self.d_loss_metric]

    def compile(self, g_optimizer, d_optimizer, loss_fn):
        super(DCGAN, self).compile()
        self.g_optimizer = g_optimizer
        self.d_optimizer = d_optimizer
        self.loss_fn = loss_fn

    def train_step(self, real_images):
        # get batch size from the data
        batch_size = tf.shape(real_images)[0]
        # generate random noise
        random_noise = tf.random.normal(shape=(batch_size, self.latent_dim))

        # train the discriminator with real (1) and fake (0) images
        with tf.GradientTape() as tape:
            # compute loss on real images
            pred_real = self.discriminator(real_images, training=True)
            # generate real image labels
            real_labels = tf.ones((batch_size, 1))
            # label smoothing
            real_labels += 0.05 * tf.random.uniform(tf.shape(real_labels))
            d_loss_real = self.loss_fn(real_labels, pred_real)

            # compute loss on fake images
            fake_images = self.generator(random_noise)
            pred_fake = self.discriminator(fake_images, training=True)
            # generate fake labels
            fake_labels = tf.zeros((batch_size, 1))
            d_loss_fake = self.loss_fn(fake_labels, pred_fake)

            # total discriminator loss
```

```
            d_loss = (d_loss_real + d_loss_fake) / 2

        # compute discriminator gradients
        gradients = tape.gradient(d_loss, self.discriminator.
↪trainable_variables)
        # update the gradients
        self.d_optimizer.apply_gradients(zip(gradients, self.discriminator.
↪trainable_variables))


        # train the generator model
        labels = tf.ones((batch_size, 1))
        # generator want discriminator to think that fake images are real
        with tf.GradientTape() as tape:
            # generate fake images from generator
            fake_images = self.generator(random_noise, training=True)
            # classify images as real or fake
            pred_fake = self.discriminator(fake_images, training=True)
            # compute loss
            g_loss = self.loss_fn(labels, pred_fake)

        # compute gradients
        gradients = tape.gradient(g_loss, self.generator.trainable_variables)
        # update the gradients
        self.g_optimizer.apply_gradients(zip(gradients, self.generator.
↪trainable_variables))

        # update states for both models
        self.d_loss_metric.update_state(d_loss)
        self.g_loss_metric.update_state(g_loss)

        return {'d_loss': self.d_loss_metric.result(), 'g_loss': self.
↪g_loss_metric.result()}
```

The Monitoring process

To monitor the training progress, callbacks like the DCGANMonitor can be used. This callback generates images from random noise using the trained generator and visualizes them. Additionally, the generator can be saved at the end of training for future use.

[17]:
```
# DCGAN Monitor for Image Generation and Model Saving

class DCGANMonitor(keras.callbacks.Callback):
    def __init__(self, num_imgs=25, latent_dim = input_dim):
        self.num_imgs = num_imgs
        self.latent_dim = latent_dim
        # create random noise for generating images
        self.noise = tf.random.normal([25, latent_dim])
```

```
    def on_epoch_end(self, epoch, logs = None):
        # generate the image from noise
        g_img = self.model.generator(self.noise)
        # denormalize the image
        g_img = (g_img * 255) + 255
        g_img.numpy()

    def on_train_end(self, logs = None):
        self.model.generator.save('DCGEN.h5')
```

0.8 Train The Model

Tabel of Contents

[18]:
```
# Training DCGAN on Image Dataset for 40 Epochs

epochs = 30
lr_g =0.0003
lr_d = 0.0001
beta = 0.5
latent_dim = 300

dcgan = DCGAN(generator=generator, discriminator=discriminator, latent_dim =␣
 ↪latent_dim )
dcgan.compile(g_optimizer = Adam (learning_rate= lr_g, beta_1= beta),␣
 ↪d_optimizer= Adam (learning_rate = lr_g , beta_1= beta), loss_fn =␣
 ↪BinaryCrossentropy())

# Fit the model and save the history
history = dcgan.fit(train_generator, epochs=epochs, callbacks=[DCGANMonitor()])
```

```
Epoch 1/30
1987/1987 [==============================] - 625s 314ms/step - d_loss: 0.3696 -
g_loss: 2.5258
Epoch 2/30
1987/1987 [==============================] - 631s 317ms/step - d_loss: 0.3656 -
g_loss: 2.4473
Epoch 3/30
1987/1987 [==============================] - 627s 316ms/step - d_loss: 0.3365 -
g_loss: 2.4319
Epoch 4/30
1987/1987 [==============================] - 626s 315ms/step - d_loss: 0.2947 -
g_loss: 2.6565
Epoch 5/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.2594 -
g_loss: 2.8745
Epoch 6/30
```

```
1987/1987 [==============================] - 627s 316ms/step - d_loss: 0.2413 -
g_loss: 3.0140
Epoch 7/30
1987/1987 [==============================] - 620s 312ms/step - d_loss: 0.2592 -
g_loss: 3.0019
Epoch 8/30
1987/1987 [==============================] - 630s 317ms/step - d_loss: 0.2308 -
g_loss: 3.0981
Epoch 9/30
1987/1987 [==============================] - 625s 314ms/step - d_loss: 0.2128 -
g_loss: 3.2504
Epoch 10/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.2087 -
g_loss: 3.3358
Epoch 11/30
1987/1987 [==============================] - 626s 315ms/step - d_loss: 0.2028 -
g_loss: 3.4349
Epoch 12/30
1987/1987 [==============================] - 626s 315ms/step - d_loss: 0.2054 -
g_loss: 3.4628
Epoch 13/30
1987/1987 [==============================] - 625s 314ms/step - d_loss: 0.1897 -
g_loss: 3.5297
Epoch 14/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.1895 -
g_loss: 3.5577
Epoch 15/30
1987/1987 [==============================] - 622s 313ms/step - d_loss: 0.1753 -
g_loss: 3.6547
Epoch 16/30
1987/1987 [==============================] - 625s 314ms/step - d_loss: 0.2369 -
g_loss: 3.4988
Epoch 17/30
1987/1987 [==============================] - 625s 315ms/step - d_loss: 0.1829 -
g_loss: 3.6421
Epoch 18/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.1448 -
g_loss: 3.9264
Epoch 19/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.1866 -
g_loss: 3.8654
Epoch 20/30
1987/1987 [==============================] - 623s 313ms/step - d_loss: 0.1349 -
g_loss: 3.9136
Epoch 21/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.1636 -
g_loss: 4.0257
Epoch 22/30
```

```
1987/1987 [==============================] - 625s 314ms/step - d_loss: 0.1223 -
g_loss: 4.1006
Epoch 23/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.1184 -
g_loss: 4.0796
Epoch 24/30
1987/1987 [==============================] - 621s 313ms/step - d_loss: 0.1045 -
g_loss: 4.2893
Epoch 25/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.1107 -
g_loss: 4.2032
Epoch 26/30
1987/1987 [==============================] - 625s 315ms/step - d_loss: 0.0758 -
g_loss: 4.4223
Epoch 27/30
1987/1987 [==============================] - 625s 315ms/step - d_loss: 0.0807 -
g_loss: 4.4911
Epoch 28/30
1987/1987 [==============================] - 623s 313ms/step - d_loss: 0.0954 -
g_loss: 4.4829
Epoch 29/30
1987/1987 [==============================] - 624s 314ms/step - d_loss: 0.0053 -
g_loss: 4.8385
Epoch 30/30
1987/1987 [==============================] - 625s 314ms/step - d_loss: 0.0560 -
g_loss: 4.8022
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.
```

0.9 Evaluation Of Model Results

Tabel of Contents

[19]:
```python
# Generating 36 Random Images with DCGAN

plt.figure(figsize=(10, 10))

for i in range(36):
    plt.subplot(6, 6, i + 1)
    # Generate random noise for each image
    noise = tf.random.normal([1, 300])
    mg = dcgan.generator(noise)
    # Denormalize
    mg = (mg * 255) + 255

    mg.numpy()
    image = Image.fromarray(np.uint8(mg[0]))
```

```
    plt.imshow(image)
    plt.axis('off')

plt.show()
```

[20]:
```
import matplotlib.pyplot as plt

# Function to create a figure for the losses
def create_loss_figure(d_loss_values, g_loss_values):
    plt.figure(figsize=(10, 6))
    plt.plot(d_loss_values, label='Discriminator Loss')
    plt.plot(g_loss_values, label='Generator Loss')
```

```
    plt.title('Generator and Discriminator Losses')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid(True)
    plt.show()

# Access the loss values from the history
d_loss_values = history.history['d_loss']
g_loss_values = history.history['g_loss']

# Call the create_loss_figure function with the loss values
create_loss_figure(d_loss_values, g_loss_values)
```