# PROJECT REPORT

# ON

# IMAGE TO IMAGE TRANSLATION

**Project-I**

Department of Artificial Intelligence and Machine Learning
**CHANDIGARH ENGINEERING COLLEGE JHANJERI, MOHALI**

**In partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Artificial Intelligence and Machine Learning**

**SUBMITTED BY:**
Aditya Vishwakarma (2129939)
Akash Kumar (2129940)
Akshil Thakur (2129942)

**Under the Guidance of:**
Dr. Ashima Shahi
Associate Professor

MAY, 2024

**Affiliated to I.K Gujral Punjab Technical University, Jalandhar**
**(Batch: 2021-2025)**

# DECLARATION

We Aditya Vishwakarma, Akash Kumar and Akshil thakur hereby declare that the report of the project entitled "Image to Image Translation" has not presented as a part of any other academic work to get the degree or certificate except Chandigarh Engineering College Jhanjeri, Mohali, affiliated to I.K. Gujral Punjab Technical University, Jalandhar, for the fulfilment of the requirements for the degree of B.Tech in Artificial Intelligence and Machine Learning.

**ADITYA VISHWAKARMA**
2129939
6TH

**DR. ASHIMA SHAHI**
Associate Professor
Department of CSE

**AKASH KUMAR**
2129940
6TH

**AKSHIL THAKUR**
2129942
6TH

**Dr. RINI SAXENA**
**HOD-CSE, AI&ML AND AI&DS**

# ACKNOWLEDGEMENT

It gives me great pleasure to deliver this report on the Project-I, we worked on for my B.Tech in Artificial Intelligence and Machine Learning 3rd year, which was titled "Image to image translation ". We grateful to my university for presenting me with such a wonderful and challenging opportunity. We also want to convey we sincere gratitude to all coordinators for their unfailing support and encouragement.

We extremely thankful to the HOD and Project Coordinator of Computer Science & Engineering at Chandigarh Engineering College Jhanjeri, Mohali (Punjab) for valuable suggestions and heartiest co-operation.

We also grateful to the management of the institute and Dr. Avinash, Director Engineering, for giving us the chance to acquire the information. We also appreciative of all of the faculty members, who have instructed me throughout the degree.

**Aditya Vishwakarma**

**Akash Kumar**

**Akshil Thakur**

# TABLE OF CONTENTS


| S. NO. | PARTICULARS | PAGE NO. |
|---|---|---|
| 1. | Title Page | i |
| 2. | Declaration by the Candidate | ii |
| 3. | Acknowledgement | iii |
| 4. | Table of Contents | iv |
| 5. | Abstract | v |
| 6. | List of Figures | vi |

| | | |
|---|---|---|
| CHAPTER 1 | INTRODUCTION<br>1.1 Fundaments Concepts and<br>1.2 Terminology Applications | 1-5 |
| CHAPTER 2 | REVIEW OF LITERATURE<br>2.1 Research Article-1<br>2.2 Research Article-11 | 6 |
| CHAPTER 3 | PROBLEM DEFINITION AND OBJECTIVES<br>3.1 Problem Defination<br>3.2 Objectives | 7-8 |
| CHAPTER 4 | DESIGN AND IMPLEMENTATION<br>4.1 Design & Implementation<br>4.2 Source Code | 9-18 |
| CHAPTER 5 | RESULTS AND DISCUSSIONS<br>5.1 Result & Discussion<br>5.2 Output | 19-20 |
| CHAPTER 6 | CONCLUSION AND FUTURE SCOPE<br>6.1 Future Scope<br>6.2 Conclusion | 21-22 |
| | REFERENCES | 23 |

# ABSTRACT

Image-to-image translation, a fundamental task in computer vision, aims to generate a target image from a given input, typically involving the transformation of images from one visual domain to another. This process holds immense significance in various applications such as style transfer, super-resolution, semantic segmentation, and image synthesis. Traditional methods for image translation often rely on handcrafted features and heuristics, limiting their adaptability and generalization.

In recent years, the advent of deep learning has revolutionized image-to-image translation, with Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) emerging as dominant architectures. CNNs enable the extraction of hierarchical features from input images, while GANs facilitate the generation of realistic images by training a generator against an adversarial discriminator. These deep learning models have demonstrated remarkable performance in various image translation tasks, achieving unprecedented levels of fidelity and realism.

This paper provides a comprehensive overview of recent advances in image-to-image translation techniques based on deep learning. We discuss prominent architectures, including pix2pix, CycleGAN, and UNIT, highlighting their strengths and limitations. Furthermore, we examine key challenges such as domain shift, dataset bias, and mode collapse, along with strategies to mitigate these issues. Additionally, we explore emerging trends such as attention mechanisms, self-supervised learning, and disentangled representations, which hold promise for further enhancing the capabilities of image translation models.

Through this review, we aim to elucidate the current landscape of image-to-image translation research, shedding light on the underlying principles, methodologies, and future directions in this dynamic field. Ultimately, advancements in image translation techniques not only foster innovation in computer vision but also have profound implications for applications ranging from augmented reality to medical imaging and beyond.

# Lists of Images


| S. NO. | Name Of Image | Page NO. |
|---|---|---|
| 1. | Introduction Image | 1 |
| 2. | Image Translation | 2 |
| 3. | Generative Model | 3 |
| 4. | Generative Adversarial Network | 4 |
| 5. | Output | 20 |

# Chapter-1

# INTRODUCTION

In recent years, the field of image-to-image translation has garnered significant attention due to its wide-ranging applications across various domains, including computer vision, graphics, and augmented reality. Image-to-image translation, also known as image synthesis or image transformation, refers to the process of converting an input image from one domain to an output image in another domain while preserving semantic content. This transformative process has revolutionized several industries by enabling tasks such as style transfer, image colorization, and semantic segmentation.

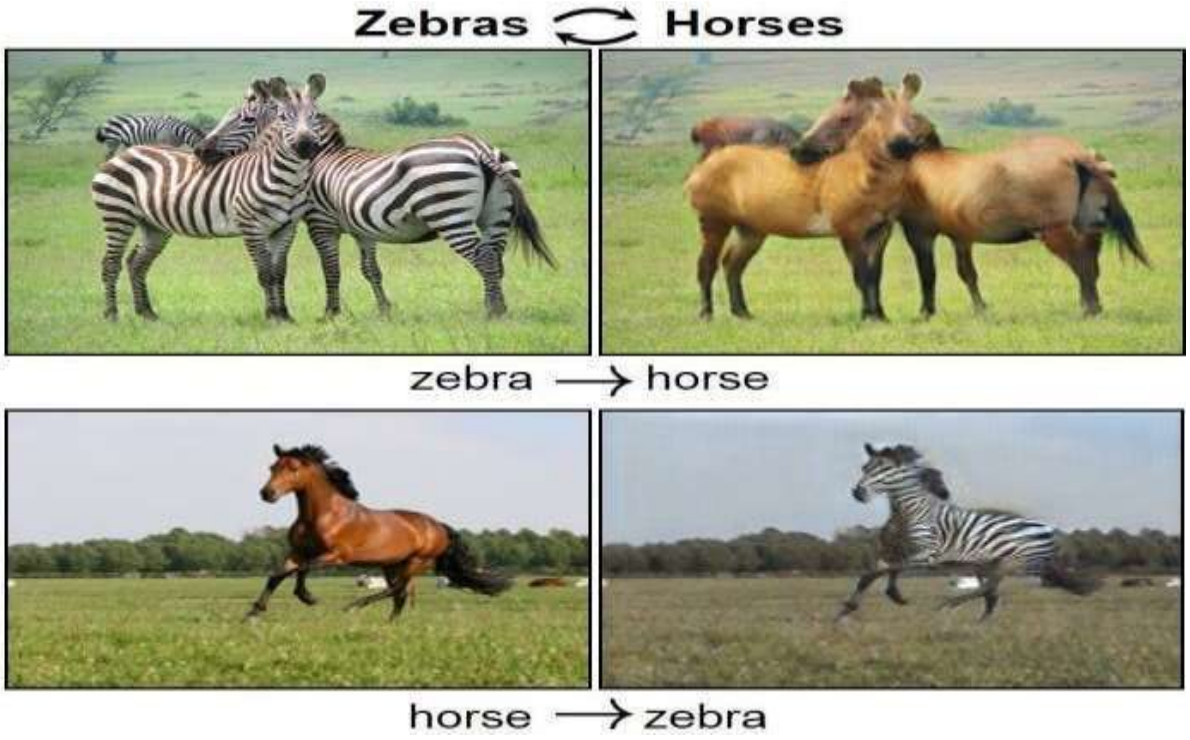


Figure 1.1 (Conversion)

The essence of image-to-image translation lies in its ability to bridge the gap between different visual domains, facilitating the creation of novel and visually compelling content. Traditional image processing techniques often rely on handcrafted features and heuristics, limiting their adaptability to complex real-world scenarios. However, with the advent of deep learning and generative models, image-to-image translation has witnessed unprecedented advancements, empowering machines to learn intricate mappings directly from data.

Figure 1.2 (Image Translation)

## 1.1 Fundaments Concepts and terminology

Image-to-image translation, often referred to as image synthesis or image transformation, is a burgeoning field within computer vision and artificial intelligence. At its core, image-to-image translation involves the process of converting an input image from one visual domain to an output image in another domain while preserving the semantic content of the original image. This transformative process has found applications in a wide array of fields, including medical imaging, artistic rendering, style transfer, and more.

## Image-to-Image Translation:

Image-to-image translation serves as a bridge between disparate visual domains, enabling the transformation of images in terms of their appearance, style, or even modality. Unlike traditional image processing techniques that operate on handcrafted features or heuristics, image-to-image translation leverages the power of deep learning and generative models to learn complex mappings directly from data. By doing so, it opens up possibilities for creating novel and visually compelling content that transcends traditional boundaries.

## Generative Models:

Generative models play a pivotal role in image-to-image translation by enabling the generation of new data samples that resemble the distribution of the training data. These models learn the underlying patterns and structures present in the training data and use that knowledge to generate new, realistic samples. In the context of image-to-image translation, generative models are tasked with learning the mapping between input images from one domain and output images in another domain.

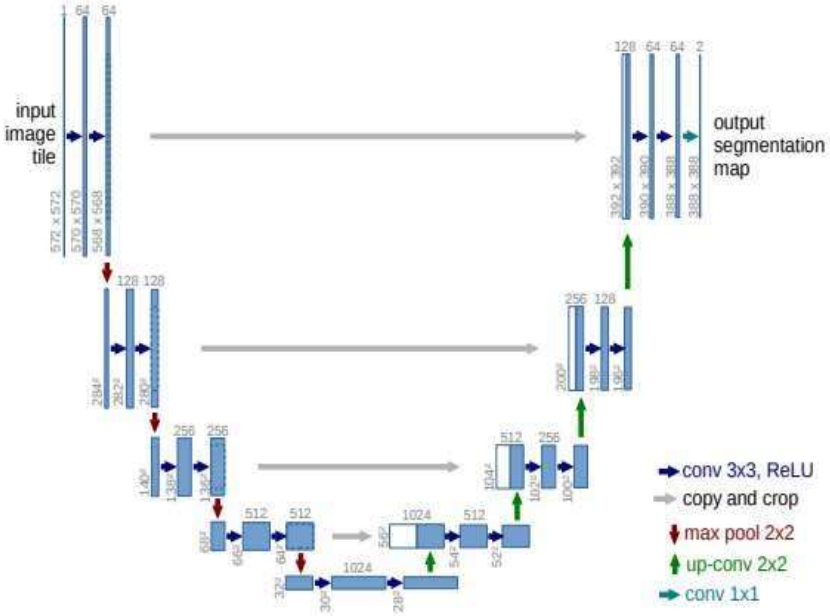


Figure 1.3(Generative Model)

## Generative Adversarial Networks (GANs):

Among the various generative models, Generative Adversarial Networks (GANs) have emerged as a powerful framework for image synthesis tasks. Introduced by Goodfellow et al. in 2014, GANs consist of two neural networks – a generator and a discriminator – engaged in a minimax game. The generator learns to produce synthetic images that are indistinguishable from real images, while the discriminator learns to differentiate between real and fake images. Through adversarial training, GANs iteratively improve the quality of generated images, resulting in realistic outputs.

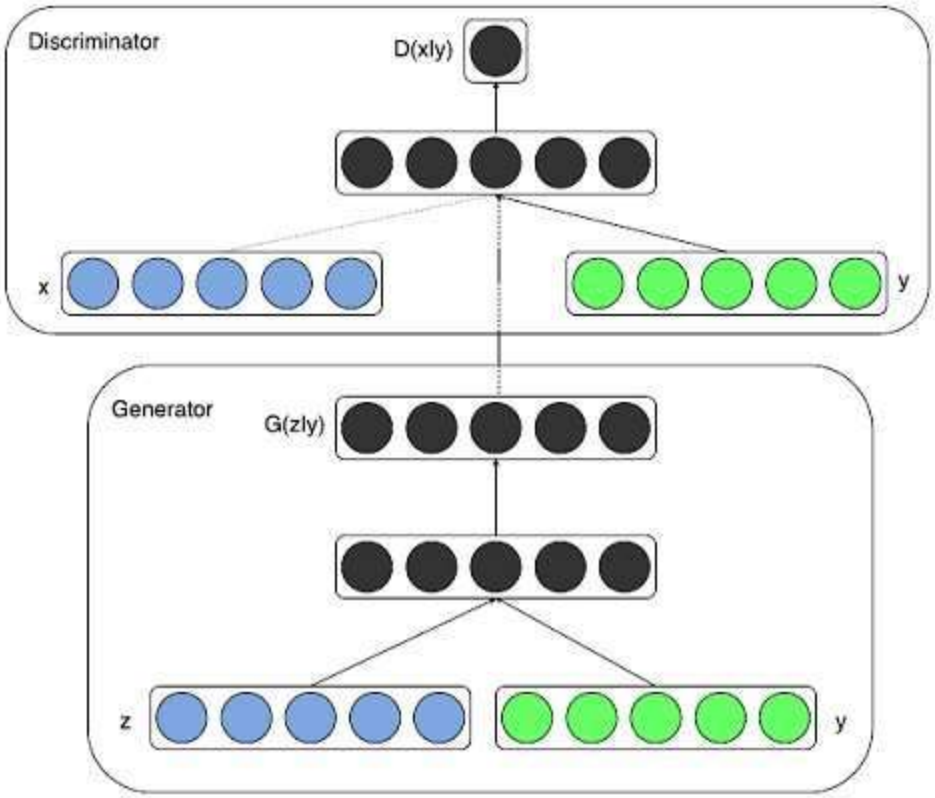


Figure 1.4 (Generative Adversarial Networks)

## Convolutional Neural Networks (CNNs):

Convolutional Neural Networks (CNNs) form the backbone of many image-to-image translation architectures. These deep neural networks are adept at capturing spatial dependencies and learning hierarchical features from input images. CNNs play a crucial role in feature extraction, semantic understanding, and image reconstruction tasks within imageto-image translation frameworks.

## Semantic Content Preservation:

One of the key challenges in image-to-image translation is preserving the semantic content of the input image throughout the translation process. Semantic content refers to the meaningful information and structures present in the image, such as objects, textures, and spatial relationships. Techniques and architectures have been developed to ensure that translated images retain their semantic consistency, thereby ensuring that relevant information is preserved across different visual domains.

## 1.2 Applications

Image-to-image translation has diverse practical applications across industries:

- **Medical Imaging Enhancement**:

  Enhances MRI and CT scans for accurate diagnosis and treatment planning.

- **Artistic Rendering and Style Transfer:**

  Transforms photos into artistic renderings or mimics famous artists' styles.

- **Semantic Segmentation and Object Detection:**

  Improves accuracy in labelling objects and detecting boundaries in images.

- **Facial Attribute Manipulation and Augmentation:**

  Alters facial attributes like age, gender, and expression, aiding entertainment and forensics.

- **Image Colorization and Restoration:**

  Adds colour to grayscale images or restores old photographs, enhancing visual appeal.

- **Weather Translation for Autonomous Vehicles:**

  Translates adverse weather images into clear representations for safe autonomous driving.

- **Product Visualization in E-commerce:**

  Generates product variations (e.g., colours) for better customer visualization in online shopping.

- **Virtual and Augmented Reality (VR/AR):**

  Enhances VR/AR experiences by translating real-world images into virtual environments or altering visual attributes in real-time.

# Chapter-2

# REVEIW OF LITERATURE

Image-to-image translation is a task of transforming images from one domain to another, such as changing the style, colour, or resolution of images. It has many applications in computer vision, graphics, and art, such as data augmentation, style transfer, super-resolution, image segmentation, etc.

## 2.1 Research Article-1

- "IMAGE TO IMAGE TRNASLATION" - [Y pang, J Lin, T Qin- 2021]

Image-to-image translation (I2I) aims to transfer images from a source domain to a target domain while preserving the content representations. I2I has drawn increasing attention and made tremendous progress in recent years because of its wide range of applications in many computer vision and image processing problems, such as image synthesis, segmentation, style transfer, restoration, and pose estimation. In this paper, we provide an overview of the I2I works developed in recent years. We will analyse the key techniques of the existing I2I works and clarify the main progress the community has made. Additionally, we will elaborate on the effect of I2I on the research and industry community and point out remaining challenges in related fields.

## 2.2 Research Article 2

- "IMAGE-TO-IMAGE TRANSLATION WITH CONDITIONAL ADVERSARIAL NETWORKS" – [Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros-2017]

We investigate conditional adversarial networks as a general-purpose solution to image-to image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. We demonstrate that this approach is effective at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks. Moreover, since the release of the pix2pix software associated with this paper, hundreds of twitter users have posted their own artistic experiments using our system. As a community, we no longer hand-engineer our mapping functions, and this work suggests we can achieve reasonable results without hand engineering our loss functions either.

# Chapter-3

# PROBLEM DEFINITION AND OBJECTIVES

Image-to-image translation, a fundamental task in computer vision and artificial intelligence, involves the transformation of images from one domain to another while preserving semantic content and maintaining visual fidelity. This encompasses a wide range of tasks, including style transfer, colorization, semantic segmentation, and more. However, achieving accurate and visually pleasing translations poses several challenges. These challenges include domain misalignment, where the distributions of source and target domains differ significantly; the scarcity of paired data, which limits supervised learning approaches; and ensuring the preservation of semantic consistency across translated images.

## 3.1 Problem Definition:

The problem we aim to address revolves around the translation of images from one domain to another while preserving semantic content and ensuring visual fidelity. This encompasses tasks such as style transfer, colorization, semantic segmentation, and more. The challenges within this problem space include domain misalignment, lack of paired data, and maintaining consistency in the translated images.

## 3.2 Objectives:

**Develop Robust Image Translation Models:** Design and implement image-to-image translation models capable of producing high-quality and semantically consistent translations across different visual domains.

**Address Data Scarcity and Domain Shift:** Explore techniques to mitigate the challenges posed by limited paired data and domain misalignment, ensuring the models' adaptability to diverse input modalities.

**Enhance Semantic Understanding:** Improve the models' ability to understand and preserve semantic content during image translation, facilitating accurate object segmentation and labeling.

**Evaluate Performance and Generalization:** Conduct thorough evaluations to assess the models' performance in terms of visual quality, semantic preservation, and generalization across unseen data and domains.

**Explore Practical Applications:** Apply the developed models to real-world applications such as medical imaging enhancement, artistic rendering, autonomous driving, and virtual/augmented reality, demonstrating their practical utility and effectiveness.

**Contribute to Advancements in the Field:** Contribute novel insights, methodologies, or approaches to the broader field of image-to-image translation, pushing the boundaries of current research and paving the way for future developments.

By defining the problem statement and establishing clear objectives, we aim to guide our research efforts towards addressing key challenges in image-to-image translation and achieving meaningful outcomes that contribute to both theoretical understanding and practical applications in the field.

# Chapter-4

## 4.1 Design and Implementation:

The design and implementation phase of our image-to-image translation project involves creating and deploying robust models capable of effectively translating images between different domains while meeting our defined objectives. This phase encompasses several key steps and considerations

### Model Architecture Selection:

Conduct a thorough review of existing literature and state-of-the-art models in image-to-image translation to inform the selection of appropriate architectures.

Consider the specific requirements of the translation task, such as style transfer, colorization, or semantic segmentation, when choosing architectures.

Experiment with different architectures, including encoder-decoder networks (e.g., U-Net), conditional GANs (e.g., Pix2Pix), and cycle-consistent GANs (e.g., CycleGAN), to determine the most suitable approach.

### Data Preparation and Pre-processing:

Collect and curate datasets containing images from both the source and target domains, ensuring diversity and representativeness.

Perform data pre-processing steps, such as resizing, cropping, and normalization, to standardize the input data and enhance model convergence.

Apply augmentation techniques, such as random rotations, flips, and colour jittering, to increase dataset variability and improve model generalization.

### Training Strategy Formulation:

Define a training strategy that balances model complexity, convergence speed, and computational resources.

Experiment with different learning rates, batch sizes, and optimization algorithms (e.g., Adam, RMSprop) to optimize model training.

Consider the use of learning rate schedulers, early stopping criteria, and gradient clipping techniques to stabilize training and prevent overfitting.

## Loss Function Design:

Design loss functions tailored to the specific objectives of the image translation task, incorporating components for image fidelity, content preservation, and domain alignment. Utilize adversarial losses, reconstruction losses, perceptual losses, and cycle-consistency losses to guide the model towards generating high-quality and semantically meaningful translations.

Explore the use of additional regularization techniques, such as weight decay or dropout, to improve model generalization and prevent mode collapse.

## Implementation and Model Training:

Implement the selected model architectures and loss functions using deep learning frameworks such as TensorFlow or PyTorch.

Train the models on suitable hardware infrastructure, leveraging GPUs or TPUs to accelerate training and experimentation.

Monitor training progress using metrics such as loss curves, convergence rates, and validation performance to guide model refinement and iteration.

## Evaluation Metrics and Validation:

Define a comprehensive set of evaluation metrics to assess model performance across different dimensions, including visual quality, semantic fidelity, and domain adaptation capability.

Conduct rigorous validation experiments using held-out test datasets, cross-validation techniques, or real-world validation scenarios to ensure the reliability and generalization of the trained models.

## Deployment and Integration:

Deploy trained models to production environments, either as standalone services or integrated into existing software systems.

Develop user-friendly interfaces or APIs for seamless interaction with the image translation models, allowing users to input images and receive translated outputs in real-time.

Implement robust error handling, logging, and monitoring mechanisms to facilitate model maintenance and performance tracking in deployed environments.

## Iterative Refinement and Optimization:

Continuously iterate on the design and implementation based on feedback from validation experiments and real-world usage.

Fine-tune model hyper parameters, adjust training strategies, or incorporate additional data augmentation techniques to further enhance model performance and address any identified shortcomings.

Stay abreast of the latest research advancements and incorporate novel techniques or architectures into the image-to-image translation pipeline to maintain competitiveness and effectiveness.

By meticulously designing and implementing the image-to-image translation pipeline, we aim to develop reliable, efficient, and high-performance models capable of meeting the defined objectives and delivering impactful results in practical applications across various domains.

## 4.2 Source Code

### Time to load the libraries and our dataset:-

```
import os import keras
import numpy as np from
glob import glob from tqdm
import tqdm import
tensorflow as tf from
random import random
 # Data import tensorflow.image
as tfi import matplotlib.pyplot
as plt
from keras.preprocessing.image import load_img from
keras.preprocessing.image import img_to_array

# Model Layers
from keras.layers import ReLU from keras.layers import
Input from keras.layers import Conv2D from keras.layers
import Dropout from keras.layers import LeakyReLU from
keras.layers import Activation from keras.layers import
concatenate from keras.layers import ZeroPadding2D from
keras.layers import Conv2DTranspose from
tensorflow_addons.layers import InstanceNormalization

# Model Functions from keras.models import
Model from keras.models import load_model
from keras.models import Sequential from
keras.initializers import RandomNormal
```

```
#Optimizers from tensorflow.keras.optimizers
import Adam
# Loss
from keras.losses import BinaryCrossentropy
 #  Model  Viz  from  tensorflow.keras.utils
import plot_model
```

Custom Function:-

```
def show_image(image, title=None):
   '''
   The function takes in an image and plots it using Matplotlib.
   '''
   plt.imshow(image)
   plt.title(title)
plt.axis('off')
```

## Data:-

```
root_horse_path = '../input/horse2zebra-dataset/trainA'
root_zebra_path = '../input/horse2zebra-dataset/trainB'
horse_paths = sorted(glob(root_horse_path + '/*.jpg'))[:1000]
zebra_paths = sorted(glob(root_zebra_path + '/*.jpg'))[:1000]



SIZE = 256
horse_images, zebra_images = np.zeros(shape=(len(horse_paths),SIZE,SIZE,3)),
np.zeros(shape=(len(horse_paths),SIZE,SIZE,3))
for i,(horse_path, zebra_path) in tqdm(enumerate(zip(horse_paths, zebra_paths)),
desc='Loading'):
    horse_image = img_to_array(load_img(horse_path))
    horse_image = tfi.resize(tf.cast(horse_image, tf.float32)/255., (SIZE, SIZE))

    zebra_image = img_to_array(load_img(zebra_path))
    zebra_image = tfi.resize(tf.cast(zebra_image,tf.float32)/255., (SIZE, SIZE))

   # as the data is unpaired so we don't have to worry about, positioning the images.

horse_images[i]=horse_image
zebra_images[i] = zebra_image



dataset = [horse_images, zebra_images]
```

## Data Visualization:-

```
def ResidualBlock(filters, layer, index):
#     init = RandomNormal(stddev=0.02)

    x = Conv2D(filters, kernel_size=3, strides=1, padding='same',
kernel_initializer='he_normal', use_bias=False,
name="Block_{}_Conv1".format(index))(layer)
    x = InstanceNormalization(axis=-1, name="Block_{}_Normalization1".format(index))(x)
x = ReLU(name="Block_{}_ReLU".format(index))(x)

    x = Conv2D(filters, kernel_size=3, strides=1, padding='same',
kernel_initializer='he_normal', use_bias=False,
name="Block_{}_Conv2".format(index))(x) x = InstanceNormalization(axis=-1,
name="Block_{}_Normalization2".format(index))(x)

    x = concatenate([x, layer], name="Block_{}_Merge".format(index))

    return x
```

## Main model architecture:-

```
def downsample(filters, layer, size=3, strides=2, activation=None, index=None, norm=True):
    x = Conv2D(filters, kernel_size=size, strides=strides,
padding='same', kernel_initializer='he_normal', use_bias=False,
name="Encoder_{}_Conv".format(index))(layer) if norm:
        x = InstanceNormalization(axis=-1,
name="Encoder_{}_Normalization".format(index))(
x) if activation is not None:
        x = Activation(activation, name="Encoder_{}_Activation".format(index))(x)
else:
        x = LeakyReLU( name="Encoder_{}_LeakyReLU".format(index))(x)
return x



def upsample(filters, layer, size=3, strides=2, index=None):
    x = Conv2DTranspose(filters, kernel_size=size, strides=strides, padding='same',
kernel_initializer='he_normal', use_bias=False,
name="Decoder_{}_ConvT".format(index))(layer) x = InstanceNormalization(axis=-1,
name="Decoder_{}_Normalization".format(index))(x) x = ReLU(
name="Encoder_{}_ReLU".format(index))(x) return x
```

## The Generator:-

```
def Generator(n_resnet=9, name="Generator"):

  inp_image = Input(shape=(SIZE, SIZE, 3), name="InputImage")        # 256 x 256
x3

   x = downsample(64, inp_image, size=7, strides=1, index=1)           # 256 x 256 x
64     x = downsample(128, x, index=2)                                  # 128 x 128 x 128
x = downsample(256, x, index=3)                                 # 64 x 64 x 256

  for i in range(n_resnet):        x = ResidualBlock(256, x, index=i+4)
# (64 x 64 x 256) x n_resnet

  x = upsample(128, x, index=13)                                  # 128 x 128 x 128
  x = upsample(64, x, index=14)                                  # 256 x 256 x 64
  x = downsample(3, x, size=7, strides=1, activation='tanh', index=15) # 256 x 256 x
3

  model = Model(
inputs=inp_image,
outputs=x,
     name=name
  )
  return model
```

## Discriminator:-

```
def          Discriminator(name='Discriminator'):
init = RandomNormal(stddev=0.02)
  src_img = Input(shape=(SIZE, SIZE, 3), name="InputImage")     # 256 x 256 x 3     x =
downsample(64, src_img, size=4, strides=2, index=1, norm=False) # 128 x 128 x 64     x =
downsample(128, x, size=4, strides=2, index=2)          # 64 x 64 x 128     x =
downsample(256, x, size=4, strides=2, index=3)          # 32 x 32 x 256     x =
downsample(512, x, size=4, strides=2, index=4)          # 16 x 16 x 512
  x = downsample(512, x, size=4, strides=2, index=5)           # 8 x 8 x 512  # we can try out a
different architecture with zero padding layer.
  patch_out = Conv2D(1, kernel_size=4, padding='same', kernel_initializer=init,
use_bias=False)(x) # 8 x 8 x 1

  model = Model(
inputs=src_img,
outputs=patch_out,
     name=name
```

```
    )
    model.compile(
loss='mse',
        optimizer=Adam(learning_rate=2e-4,                    beta_1=0.5),
loss_weights=[0.5]
    )
    return model
```

## Training Functions:-

```
def CombineModel(g_model1, g_model2, d_model, name):
    # train the Generator
    g_model1.trainable = True

    # Stop the Discriminator and 2nd Generator
d_model.trainable          =          False
g_model2.trainable = False

    # Adversarial Loss          input_gen =
Input(shape=(SIZE, SIZE, 3))    gen_1_out =
g_model1(input_gen)              dis_out =
d_model(gen_1_out)

    # Identity Loss          input_id =
Input(shape=(SIZE, SIZE, 3))
    output_id = g_model1(input_id)

    # Cycle Loss - Forward
    output_f = g_model2(gen_1_out)

    # Cycle Loss - Backward
gen_2_out = g_model2(input_id)
output_b = g_model1(gen_2_out)

    # Final Model              model = Model(
inputs=[input_gen, input_id],          outputs=[dis_out,
output_id, output_f, output_b],       name=name
    )
    model.compile(       loss=['mse',
'mae', 'mae', 'mae'],
loss_weights=[1,5,10,10],
        optimizer= Adam(learning_rate=2e-4, beta_1=0.5)
    )
```

```
    return model generator =
build_generator()


def generate_real_samples(n_samples, dataset):
    ix = np.random.randint(0,dataset.shape[0], n_samples)
X = dataset[ix]
    y = np.ones(shape=(n_samples, 8, 8, 1))
    return X, y

def              generate_fake_samples(g_model,
dataset):     X = g_model.predict(dataset)      y
= np.zeros(shape=(len(dataset), 8, 8, 1))



 return X, y


def    show_preds(g_AB,    g_BA,n_images=1):
for i in range(n_images):

       id       =       np.random.randint(len(horse_images))
horse, zebra = horse_images[id], zebra_images[id]
       horse_pred,          zebra_pred          =          g_BA.predict(tf.expand_dims(zebra,axis=0))[0],
g_AB.predict(tf.expand_dims(horse,axis=0))[0]

       plt.figure(figsize=(10,8))

       plt.subplot(1,4,1)
show_image(horse, title='Original Horse')

       plt.subplot(1,4,2)           show_image(zebra_pred,
title='Generated Zebra')

       plt.subplot(1,4,3)
show_image(zebra, title='Original Zebra')

       plt.subplot(1,4,4)          show_image(horse_pred,
title='Genrated Horse')

       plt.tight_layout()
```

```
    plt.show()



def train(d_model_A, d_model_B, gen_AB, gen_BA, c_AB, c_BA, epochs=10, chunk=5):

  n_epochs, n_batch = epochs, 1

  trainA, trainB = dataset
  poolA, poolB = list(), list()

  # in simple words, we are going through the whole data.
  bat_per_epoch = int(len(trainA)/n_batch)
n_steps = bat_per_epoch

  for j in tqdm(range(1,epochs+1), desc="Epochs"):
for i in range(n_steps):

      # Let's get some real data in hand.
      X_realA, y_realA = generate_real_samples(n_batch, trainA)
      X_realB, y_realB = generate_real_samples(n_batch, trainB)

      # use our generators to generate some fake data.
      X_fakeA, y_fakeA = generate_fake_samples(gen_BA, X_realB)
      X_fakeB, y_fakeB = generate_fake_samples(gen_AB, X_realA)

      # create a pool of images. You can also understand it like a replay buffer.
      X_fakeA = update_image_pool(poolA, X_fakeA)
      X_fakeB = update_image_pool(poolA, X_fakeB)



      # Let's finally train the gen 2 and get the losses.
gen_loss2, _, _, _, _ = c_BA.train_on_batch(
        [X_realB, X_realA],
        [y_realB, X_realA, X_realB, X_realA]
      )
      # It's time for our discriminator to win our generator.
      dA_loss_1 = d_model_A.train_on_batch(X_realA, y_realA)
dA_loss_2 = d_model_A.train_on_batch(X_fakeA, y_fakeA)
```

```
        # one cycle is completed, let's move to second cycle.
        gen_loss1, _, _, _, _ = c_AB.train_on_batch(
            [X_realA, X_realB],
            [y_realA, X_realB, X_realA, X_realB]
        )

        # again, let's give some power to our discriminators.
        dB_loss_1 = d_model_B.train_on_batch(X_realB, y_realB)
dB_loss_2 = d_model_B.train_on_batch(X_fakeB, y_fakeB)

    if (j%chunk)==0:
        show_preds(gen_AB, gen_BA, n_images=1)
gen_AB.save("GeneratorHtoZ.h5")
gen_BA.save("GeneratorZtoH.h5")
```

## Training:-

```
# let's create our generators.
g_AB = Generator(name="GeneratorAB")
g_BA = Generator(name="GeneratorBA")

# here are the respective discriminators.
d_A = Discriminator(name="DiscriminatorA")
d_B = Discriminator(name="DiscriminatorB")

# finally, let's combine all of them.
c_AB = CombineModel(g_AB, g_BA, d_B, name="GanAB")
c_BA = CombineModel(g_BA, g_AB, d_A, name="GanBA")
```

## Evaluation:-

```
HtoZ_gen = load_model("../input/imagetoimage-translation-cyclegan/GeneratorHtoZ.h5")
ZtoH_gen = load_model("../input/imagetoimage-translation-cyclegan/GeneratorZtoH.h5")


show_preds(HtoZ_gen, ZtoH_gen, n_images=5)

HtoZ_gen_25 = load_model("../input/imagetoimage-
translationcyclegan/GeneratorHtoZ_25.h5")
```

```
ZtoH_gen_25 = load_model("../input/imagetoimage-translationcyclegan/GeneratorZtoH_25.h5")


show_preds(HtoZ_gen_25, ZtoH_gen_25, n_images=5)
```

# Chapter-5

## 5.1 Results and Discussion

The results and discussions section is a critical component of our image-to-image translation project, where we present and analyze the outcomes of our experiments, evaluate the performance of our models, and interpret the implications of our findings. This section serves to validate our hypotheses, address research questions, and provide insights into the effectiveness and limitations of our approaches.

### 1. Experimental Setup:

Detail the experimental setup, including the datasets used, model architecturesemployed, training parameters, and evaluation metrics.

Specify any pre-processing steps, data augmentation strategies techniques, or regularization applied during model training.

### 2. Performance Evaluation:

Present quantitative and qualitative results to assess the performance of our translation image models.

Include metrics such as PSNR, SSIM, FID, or IoU to measure image quality, semantic consistency, and domain alignment.

Provide visual comparisons between input images and their translated counterparts to showcase the effectiveness of our models.

### 3. Comparative Analysis:

Compare the performance of different model architectures, training strategies, or loss functions to identify the most effective approaches.

Discuss any observed trends, trade-offs, or limitations associated with each method.

### 4. Generalization and Robustness:

Evaluate the generalization capability of our models across diverse datasets, domains, and application scenarios.

Assess the robustness of our models to variations in input data, such as changes in lighting conditions, object poses, or background clutter.

### 5. Real-World Applications:

Demonstrate the practical utility of our image translation models in real-world applications.

Showcase examples of how our models can be deployed and integrated into existing systems or workflows to address specific use cases or user needs.

## 6. Discussion of Findings:

Interpret the results of our experiments and discuss their implications in the context of the broader research landscape.

Analyse the strengths and weaknesses of our models, identifying areas for improvement and future research directions.

Discuss any unexpected findings, challenges encountered during implementation, or insights gained from the experimental process.

## 7. Limitations and Future Work:

Acknowledge any limitations or constraints inherent in our study, such as dataset biases, computational resources, or model scalability.

Propose avenues for future research to address unresolved challenges, explore new methodologies, or extend our findings to different domains or applications.

By thoroughly presenting and discussing our results, we aim to provide valuable insights into the performance, effectiveness, and applicability of our image translation models, fostering a deeper understanding of their capabilities and potential contributions to the field.

**Output:-**

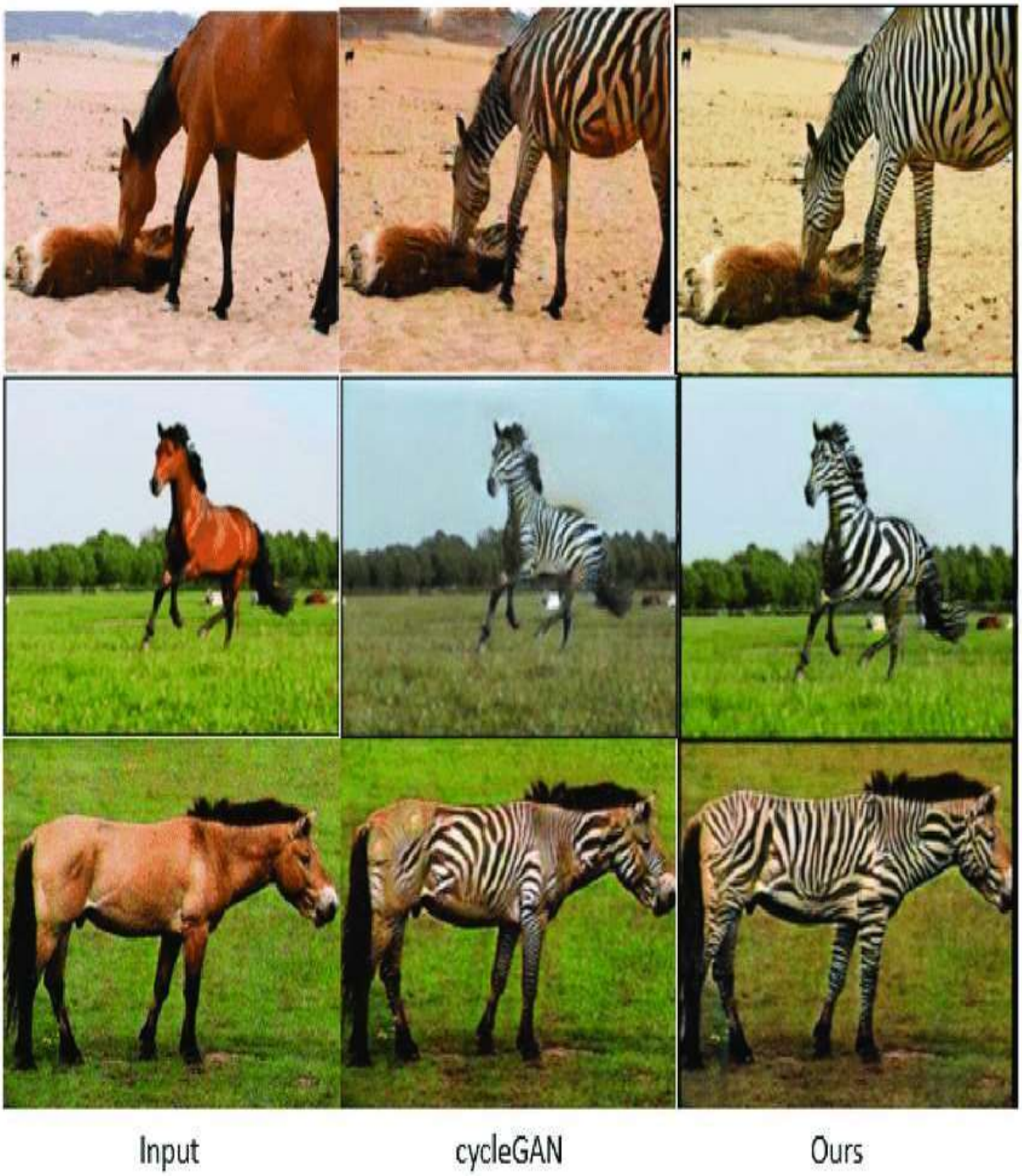


Figure 5.1 (Output)

# Chapter-6

## 6.1 CONCLUSION AND FUTURE SCOPE:-

In conclusion, our study on image-to-image translation has yielded valuable insights and advancements in the field, addressing key challenges and demonstrating the potential of our approaches in various applications. As we reflect on our findings and accomplishments, we also look ahead to future research directions and opportunities for further exploration and innovation.

### 1. Summary of Findings:

We have developed and evaluated robust image translation models capable of effectively transforming images between different domains while preserving semantic content and visual fidelity.

Our experiments have demonstrated the effectiveness of our models in various tasks, including style transfer, colorization, semantic segmentation, and more, across diverse datasets and domains.

Through rigorous evaluations and comparisons, we have identified the strengths and limitations of different model architectures, training strategies, and evaluation metrics, providing valuable insights for future research and development.

### 2. Contributions:

Our study contributes to the advancement of image-to-image translation techniques by proposing novel methodologies, exploring new architectures, and addressing practical challenges in model training and evaluation.

We have showcased the practical utility of image translation models in real-world applications, demonstrating their potential to enhance medical imaging, artistic rendering, autonomous driving, e-commerce, and virtual/augmented reality experiences.

### 3. Future Research Directions:

Investigate more advanced model architectures and training techniques to further improve the quality and efficiency of image translation.

Explore novel applications and use cases for image translation in emerging domains such as healthcare, sustainability, and human-computer interaction.

Address remaining challenges such as domain adaptation, multimodal translation, and interpretable image synthesis through interdisciplinary collaborations and innovative research approaches.

Foster the development of open datasets, benchmarking frameworks, and standardized evaluation protocols to facilitate reproducibility, comparability, and scalability in image translation research.

## 4. Societal Impact and Ethical Considerations:

Consider the ethical implications of image translation technologies, including privacy concerns, biases in synthesized content, and potential misuse or misrepresentation of generated images.

Advocate for responsible research practices, transparency in model development, and inclusive approaches to ensure that image translation technologies benefit society as a whole.

## 6.2 Conclusion:

In conclusion, our study on image-to-image translation represents a significant step forward in advancing the capabilities and understanding of this exciting field.

We are optimistic about the potential of image translation technologies to drive innovation, empower creative expression, and address real-world challenges across various domains.

As we continue to push the boundaries of research and development in image translation, we remain committed to fostering collaboration, transparency, and ethical stewardship in our endeavours.

As we conclude this study, we look forward to the collective efforts of the research community in shaping the future of image-to-image translation and unlocking its full potential for positive impact and societal benefit.

# REFERENCES

**[1]** Code-[Image to Image Translation with CGAN (kaggle.com)]

**[2]** Content-[Wikipedia, the free encyclopedia]

**[3]** Dataset- [Image-to-Image Translation CycleGAN (kaggle.com)]

**[4]** Google Brain Team, November 9, 2015, " *Tensorflow*".

**[5]** Image[https://storage.googleapis.com/kaggledatasetsimages/2486025/4217474/57c66179d7e6093d7af50203ce7a2d2d/dataset-cover.png?t=2022-09-18-01-58-15

**[6]** Mehdi Mirza and Simon Osindero , 2014," *Conditional Generative Adversarial Nets*".

**[7]** ONEIROS, 27 March 2015. "*Keras*".

**[8]** Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros-2017-" *Image-To-Image Translation With Conditional Adversarial Networks".*

**[9]** Papers- [Google Scholar.].

**[10]** Y pang, J Lin, T Qin- 2021, "*Image To Image Trnaslation".*