Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Gen AI	**Write a code for CYCLE GAN. USE DATASET MNIST FROM KERAS.**	
Experiment	**Date:**	**Enrolment No:92200133020**

CODE:

```
GetBot AI: Explain | Find Error | Find Resource Leaks
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, Conv2DTranspose, LeakyReLU, Activation, Concat
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

Python

```
GetBot AI: Explain | Find Error | Find Resource Leaks
def build_generator():
    input_img = Input(shape=(28, 28, 1))

    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(input_img)
    x = LeakyReLU(0.2)(x)

    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
    x = LeakyReLU(0.2)(x)

    x = Conv2DTranspose(64, kernel_size=3, strides=2, padding='same')(x)
    x = LeakyReLU(0.2)(x)

    x = Conv2DTranspose(1, kernel_size=3, strides=2, padding='same', activation='tanh')(x)

    return Model(input_img, x)
```

GetBot AI: Explain | Find Error | Find Resource Leaks

```python
def build_discriminator():
    input_img = Input(shape=(28, 28, 1))

    x = Conv2D(64, kernel_size=3, strides=2, padding='same')(input_img)
    x = LeakyReLU(0.2)(x)

    x = Conv2D(128, kernel_size=3, strides=2, padding='same')(x)
    x = LeakyReLU(0.2)(x)

    x = Conv2D(1, kernel_size=3, strides=1, padding='same')(x)
    x = Activation('sigmoid')(x)

    return Model(input_img, x)
```

Python

GetBot AI: Explain | Find Error | Find Resource Leaks

```python
# Optimizer
optimizer = Adam(0.0002, 0.5)

# Generators
G_AB = build_generator()  # MNIST → Inverted
G_BA = build_generator()  # Inverted → MNIST

# Discriminators
D_A = build_discriminator()  # Real MNIST?
D_B = build_discriminator()  # Real Inverted?

# Compile Discriminators
D_A.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
```

```
D_B = build_discriminator()  # Real inverted

# Compile Discriminators
D_A.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
D_B.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])

# Inputs
img_A = Input(shape=(28, 28, 1))  # MNIST
img_B = Input(shape=(28, 28, 1))  # Inverted MNIST

# Translate images
fake_B = G_AB(img_A)
fake_A = G_BA(img_B)

# Cycle images
reconstr_A = G_BA(fake_B)
reconstr_B = G_AB(fake_A)

# Identity mapping (optional)
img_A_id = G_BA(img_A)
img_B_id = G_AB(img_B)

# For combined model (freeze D)
D_A.trainable = False
D_B.trainable = False

valid_A = D_A(fake_A)
valid_B = D_B(fake_B)

# Combined model
cycle_gan = Model(inputs=[img_A, img_B],
                  outputs=[valid_A, valid_B, reconstr_A, reconstr_B, img_A_id, img_B_id])
```

```
GetBot AI: Explain | Find Error | Find Resource Leaks
(x_train, _), (_, _) = tf.keras.datasets.mnist.load_data()
x_train = (x_train.astype(np.float32) - 127.5) / 127.5
x_train = np.expand_dims(x_train, axis=-1)

# Domain A: MNIST
imgs_A = x_train

# Domain B: Inverted MNIST
imgs_B = 1.0 - imgs_A
```

Python

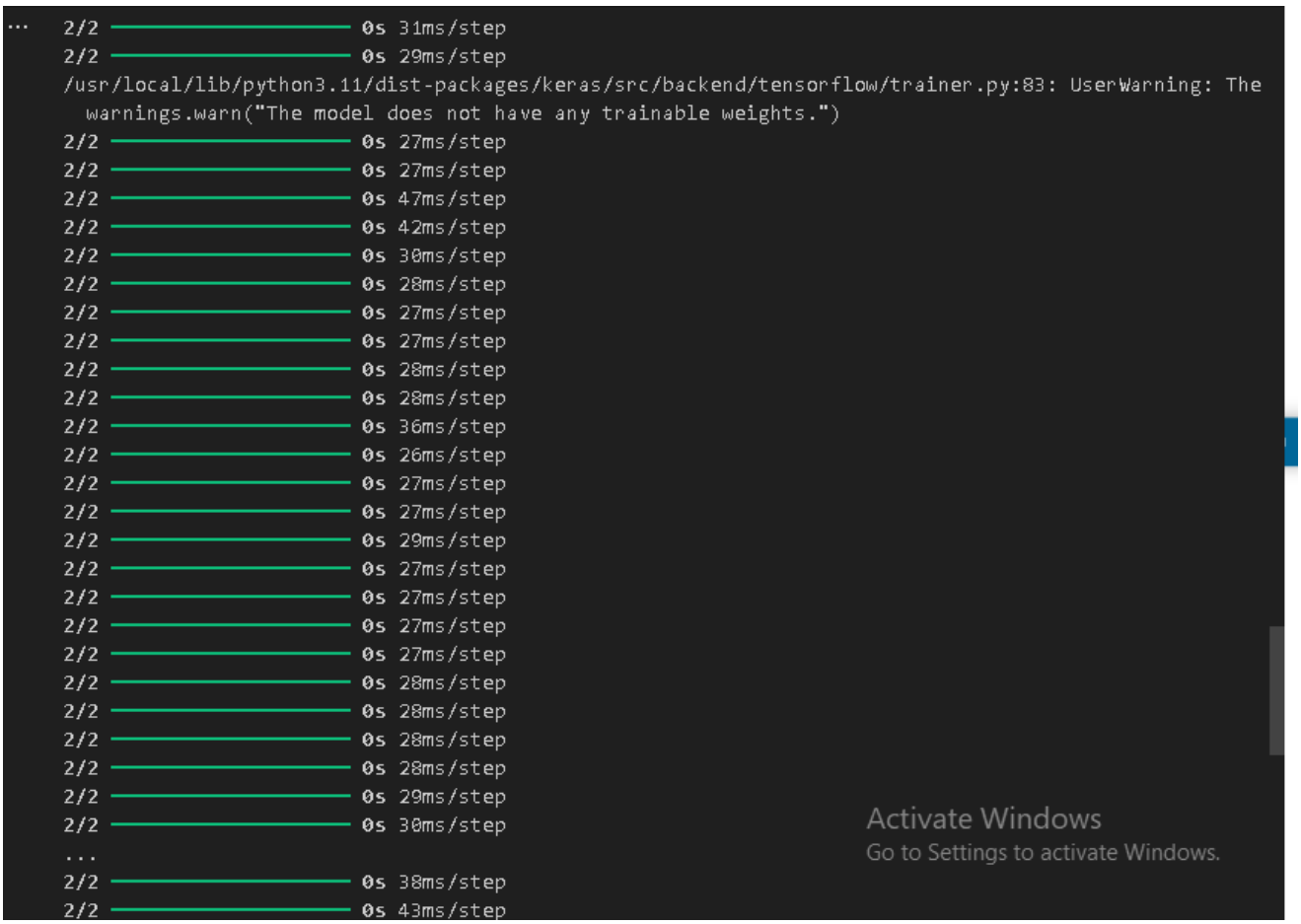
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
GetBot AI: Explain | Find Error | Find Resource Leaks
epochs = 1000
batch_size = 64
patch = D_A.output_shape[1:]

valid = np.ones((batch_size,) + patch)
fake = np.zeros((batch_size,) + patch)

for epoch in range(1, epochs + 1):
    idx = np.random.randint(0, imgs_A.shape[0], batch_size)
    real_A = imgs_A[idx]
    real_B = imgs_B[idx]

    # Translate
```



```
    real_B = imgs_B[idx]

    # Translate
    fake_B = G_AB.predict(real_A)
    fake_A = G_BA.predict(real_B)

    # Train discriminators
    dA_loss_real = D_A.train_on_batch(real_A, valid)
    dA_loss_fake = D_A.train_on_batch(fake_A, fake)
    dA_loss = 0.5 * np.add(dA_loss_real, dA_loss_fake)

    dB_loss_real = D_B.train_on_batch(real_B, valid)
    dB_loss_fake = D_B.train_on_batch(fake_B, fake)
    dB_loss = 0.5 * np.add(dB_loss_real, dB_loss_fake)

    # Total D loss
    d_loss = 0.5 * np.add(dA_loss, dB_loss)

    # Train generators (cycle + identity loss)
    g_loss = cycle_gan.train_on_batch([real_A, real_B],
                                      [valid, valid, real_A, real_B, real_A, real_B])

    if epoch % 1000 == 0:
        print(f"{epoch} [D loss: {d_loss[0]:.4f}] [G loss: {g_loss[0]:.4f}]")
```

Python


2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/trainer.py:83: UserWarning: The
warnings.warn("The model does not have any trainable weights.")
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
...
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
Activate Windows
Go to Settings to activate Windows.