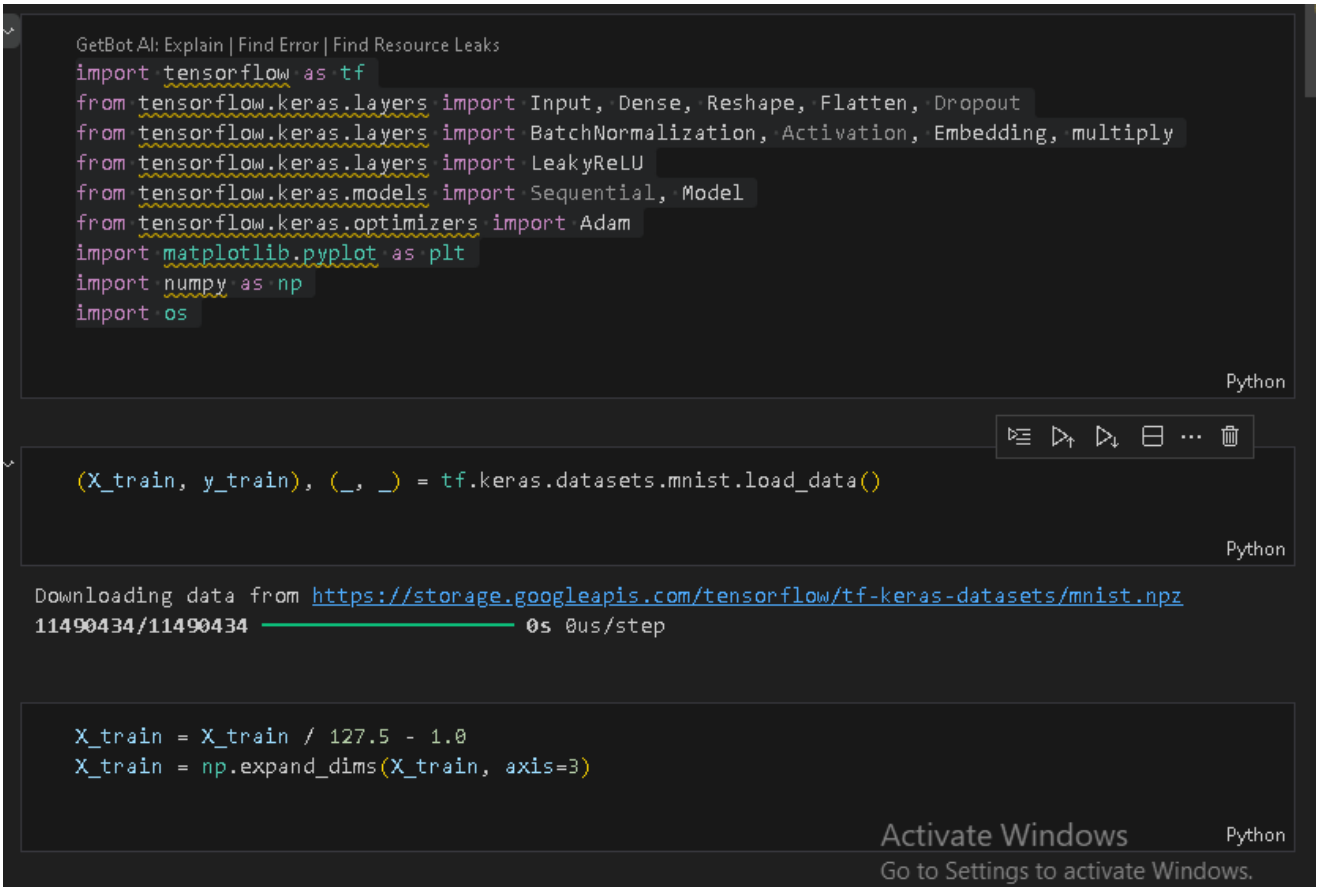
Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Gen AI	**Write a code for C_GAN. USE DATASET MNIST FROM KERAS**	
Experiment	**Date:**	**Enrolment No:92200133020**

CODE:



```python
X_train = X_train / 127.5 - 1.0
X_train = np.expand_dims(X_train, axis=3)
```

Python

```python
num_classes = 10
optimizer = Adam(0.0002, 0.5)
```

Python

GetBot AI: Explain | Find Error | Find Resource Leaks

```python
def build_generator():
    noise = Input(shape=(100,))
    label = Input(shape=(1,), dtype='int32')
    label_embedding = Flatten()(Embedding(num_classes, 100)(label))
    model_input = multiply([noise, label_embedding])

    x = Dense(256, activation="relu")(model_input)
    x = BatchNormalization(momentum=0.8)(x)
    x = Dense(512, activation="relu")(x)
    x = BatchNormalization(momentum=0.8)(x)
    x = Dense(1024, activation="relu")(x)
    x = BatchNormalization(momentum=0.8)(x)
    x = Dense(np.prod((28, 28, 1)), activation='tanh')(x)
    img = Reshape((28, 28, 1))(x)

    return Model([noise, label], img)
```


GetBot AI: Explain | Find Error | Find Resource Leaks

```python
def build_discriminator():
    img = Input(shape=(28, 28, 1))
    label = Input(shape=(1,), dtype='int32')

    label_embedding = Flatten()(Embedding(num_classes, np.prod((28, 28, 1)))(label))
    flat_img = Flatten()(img)
    model_input = multiply([flat_img, label_embedding])

    x = Dense(512)(model_input)
    x = LeakyReLU(alpha=0.2)(x)
    x = Dense(512)(x)
    x = LeakyReLU(alpha=0.2)(x)
    x = Dense(1, activation='sigmoid')(x)

    return Model([img, label], x)
```

Python

GetBot AI: Explain | Find Error | Find Resource Leaks

```python
generator = build_generator()
discriminator = build_discriminator()

discriminator.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])

noise = Input(shape=(100,))
label = Input(shape=(1,))
img = generator([noise, label])
discriminator.trainable = False
```

```
    noise = Input(shape=(100,))
    label = Input(shape=(1,))
    img = generator([noise, label])
    discriminator.trainable = False
    valid = discriminator([img, label])
    combined = Model([noise, label], valid)
    combined.compile(loss='binary_crossentropy', optimizer=optimizer)
```

Python

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/activations/leaky_relu.py:41: UserWarning:
  warnings.warn(
```

GetBot AI: Explain | Find Error | Find Resource Leaks

```
def sample_images(epoch):
    r, c = 2, 5
    noise = np.random.normal(0, 1, (r * c, 100))
    sampled_labels = np.arange(0, 10).reshape(-1, 1)

    gen_imgs = generator.predict([noise, sampled_labels])
    gen_imgs = 0.5 * gen_imgs + 0.5

    fig, axs = plt.subplots(r, c)
    cnt = 0
    for i in range(r):
        for j in range(c):
            axs[i,j].imshow(gen_imgs[cnt, :, :, 0], cmap='gray')
            axs[i,j].set_title(f"Digit: {sampled_labels[cnt][0]}")
```

```
                axs[i,j].set_title(f"Digit: {sampled_labels[cnt][0]}")
                axs[i,j].axis('off')
                cnt += 1
    os.makedirs("images", exist_ok=True)
    fig.savefig(f"images/{epoch}.png")
    plt.close()
```

Python

GetBot AI: Explain | Find Error | Find Resource Leaks

```
def train(epochs, batch_size=128, sample_interval=200):
    half_batch = int(batch_size / 2)

    for epoch in range(epochs):

        # Train Discriminator
        idx = np.random.randint(0, X_train.shape[0], half_batch)
        imgs, labels = X_train[idx], y_train[idx]
        noise = np.random.normal(0, 1, (half_batch, 100))
        gen_labels = np.random.randint(0, num_classes, half_batch).reshape(-1, 1)
        gen_imgs = generator.predict([noise, gen_labels])

        d_loss_real = discriminator.train_on_batch([imgs, labels], np.ones((half_batch, 1)))
        d_loss_fake = discriminator.train_on_batch([gen_imgs, gen_labels], np.zeros((half_batch,
        d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

        # Train Generator
        noise = np.random.normal(0, 1, (batch_size, 100))
        sampled_labels = np.random.randint(0, num_classes, batch_size).reshape(-1, 1)
        g_loss = combined.train_on_batch([noise, sampled_labels], np.ones((batch_size, 1)))
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
0 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
2 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
3 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
4 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
5 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
6 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
7 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
8 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5303]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
9 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5302]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
10 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5302]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
11 [D loss: 0.7878, acc: 41.88%] [G loss: 0.5302]
...
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
998 [D loss: 0.7926, acc: 41.84%] [G loss: 0.5232]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
999 [D loss: 0.7926, acc: 41.84%] [G loss: 0.5232]
```


Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...