NAAN MUDHALVAN – GENERATIVE AI COURSE

Sree Varshini S

2021503563

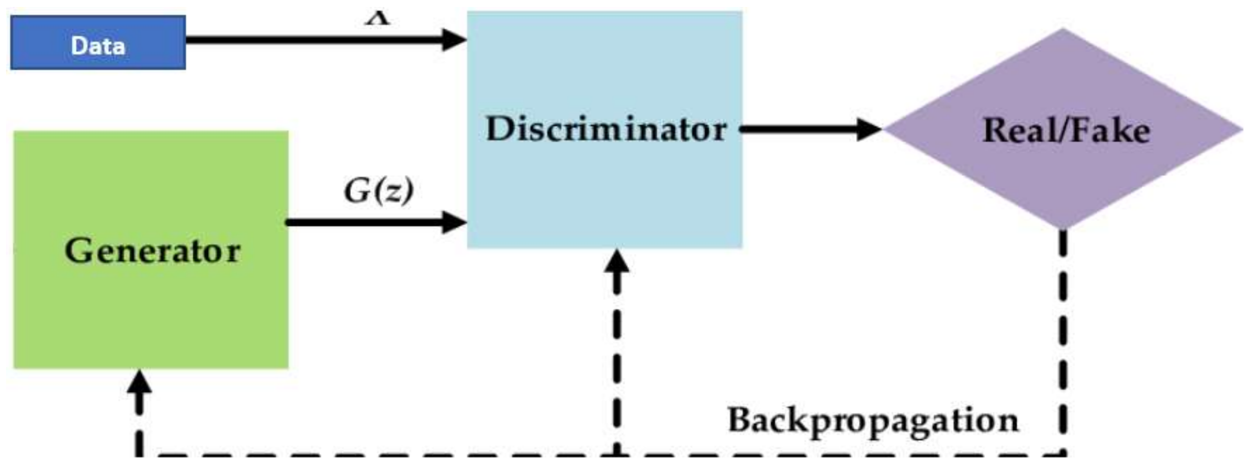
Batch 2

ABSTRACT:

In the captivating world of neural networks, Generative Adversarial Networks (GANs) introduce a unique dynamic where two networks, the Generator and the Discriminator, engage in a strategic dance. The Generator's mission is to create synthetic data that closely resembles the examples in the training dataset, while the Discriminator aims to discern between authentic and counterfeit samples. This intriguing interplay unfolds like an artistic duel, with each network striving to outsmart the other in a quest for mastery.

Our project ventures into this fascinating realm by harnessing the power of GANs to tackle the iconic MNIST dataset of handwritten digits. Through meticulous training, our network learns to generate virtual representations of these digits, mimicking their characteristics with increasing fidelity. As the Generator refines its abilities to produce convincing facsimiles, the Discriminator sharpens its discernment to differentiate between real and synthetic images. This collaborative effort results in the creation of remarkable synthetic digit renditions, showcasing the potential of GANs in generating lifelike visual content.

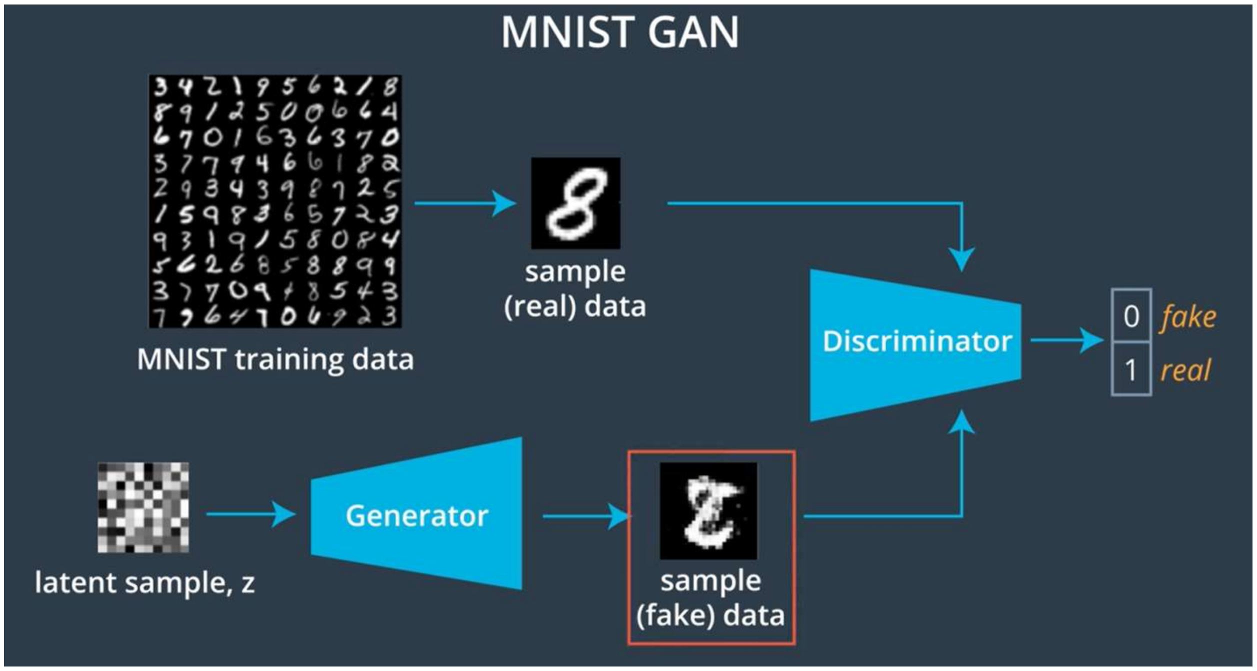
A typical GAN in shown in the image below-



DATASET DESCRIPTION:

Dataset consists of 60,000 training images and 10,000 testing images. For this task, we'll use the training dataset. Every image in the dataset will belong to one of the ten classes (digit 0 to 9). However, the image labels do not matter for this task, and we won't use them. Each image in the dataset is a 28x28 pixel grayscale image.

DESCRIPTION OF WORK IMPLEMENTATION:



1. Data Load:
- The MNIST dataset is downloaded using the torchvision dataset.
- Training dataset is wrapped in a dataloader object with a batch size of 64. The testing dataset is discarded.

2. Network Definition:
- The Discriminator network is designed as a binary classifier with LeakyReLU activation and dropout layers to prevent overfitting.
- The Generator network is designed to up-sample random noise vectors to generate fake images resembling the MNIST dataset.

3. Loss Definition:

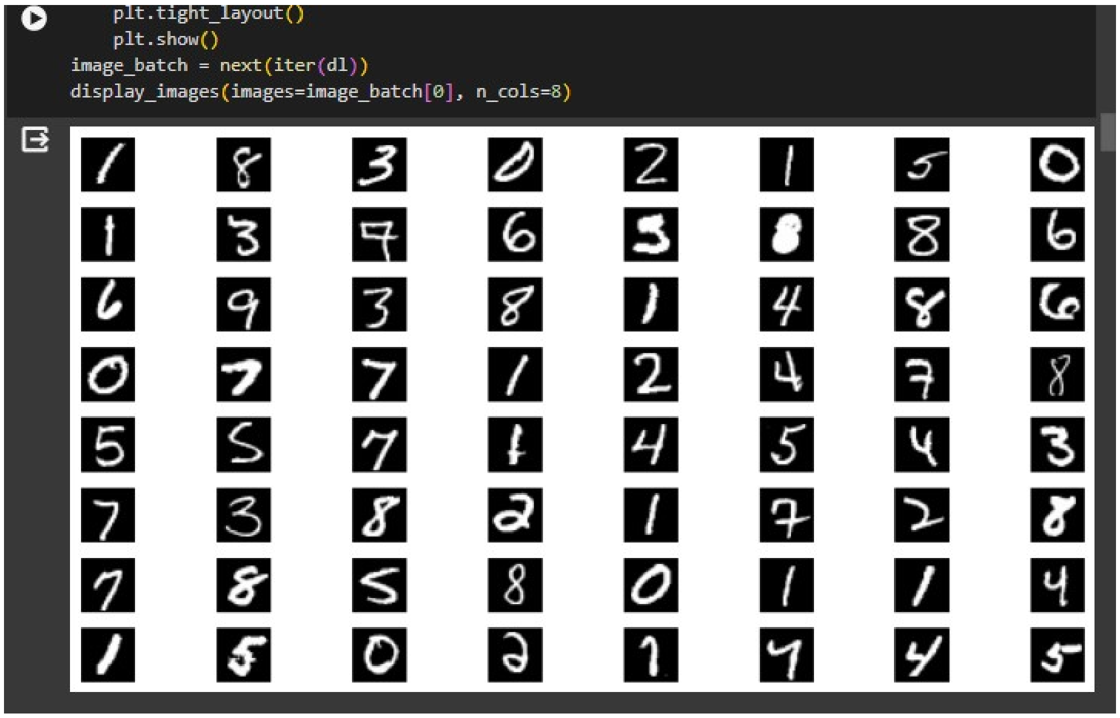
- Two separate loss functions are defined: real_loss and fake_loss, to calculate losses for the Discriminator during training.
- The real_loss function computes loss when Discriminator evaluates real images, while fake_loss computes loss when Discriminator evaluates fake images.

4. Network Training:

- Discriminator and Generator networks are trained simultaneously using Adam optimizers with a learning rate of 0.002.
- The Discriminator is trained to distinguish between real and fake images, while the Generator is trained to generate images that deceive the Discriminator.
- After each epoch, fake images are generated from a fixed noise vector to visualize the training progress and quality of generated images.

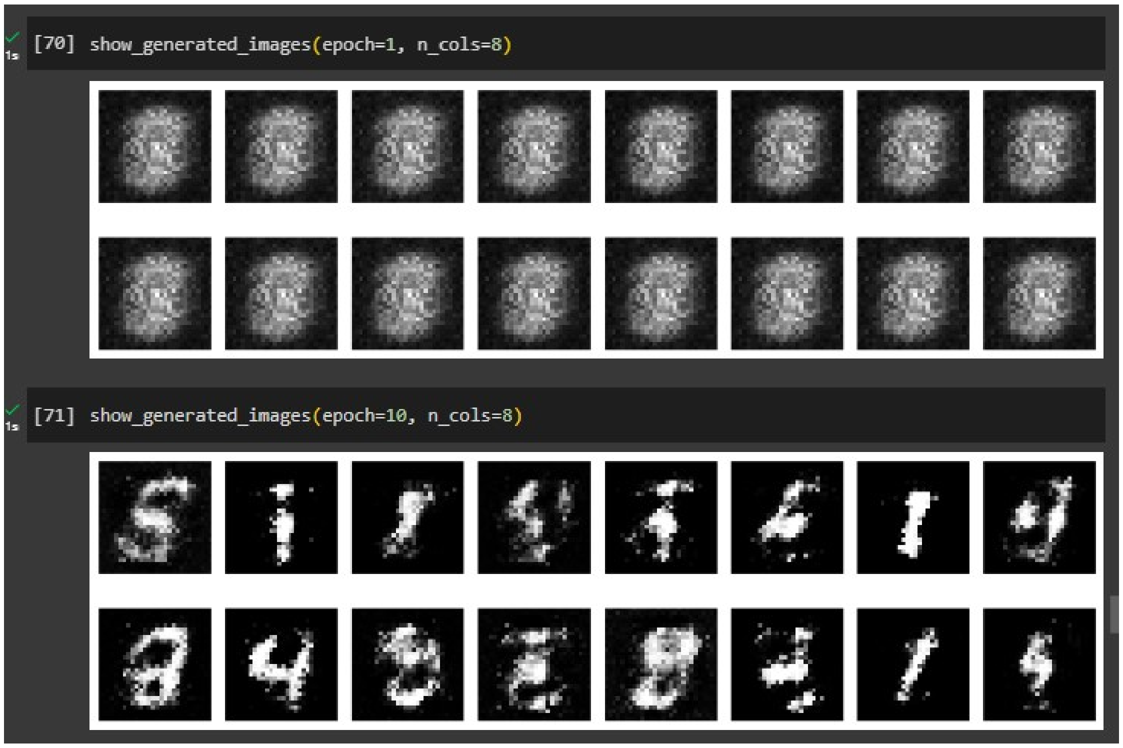
OUTPUT SCREENSHOTS:

Sample batch chosen



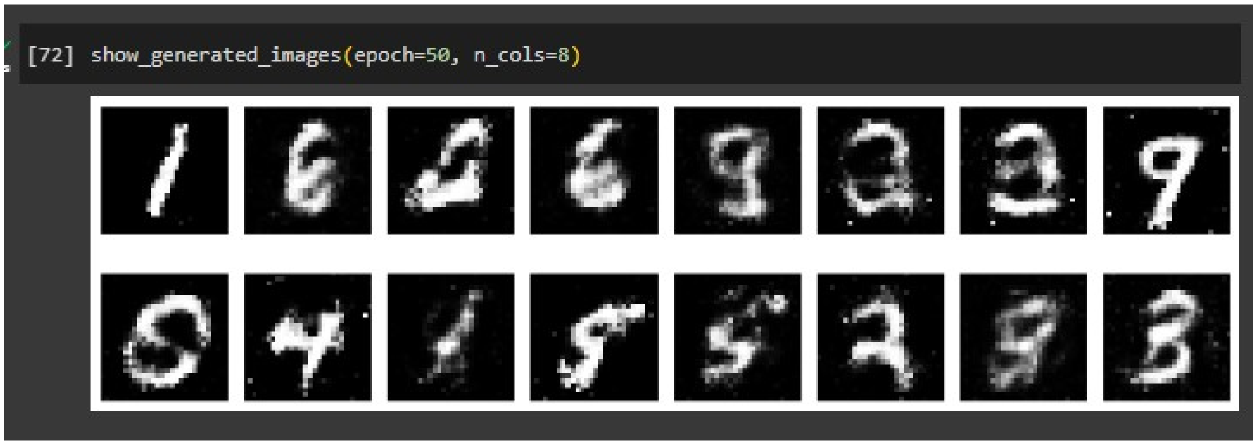
```
    plt.tight_layout()
    plt.show()
image_batch = next(iter(dl))
display_images(images=image_batch[0], n_cols=8)
```

Visualizing training



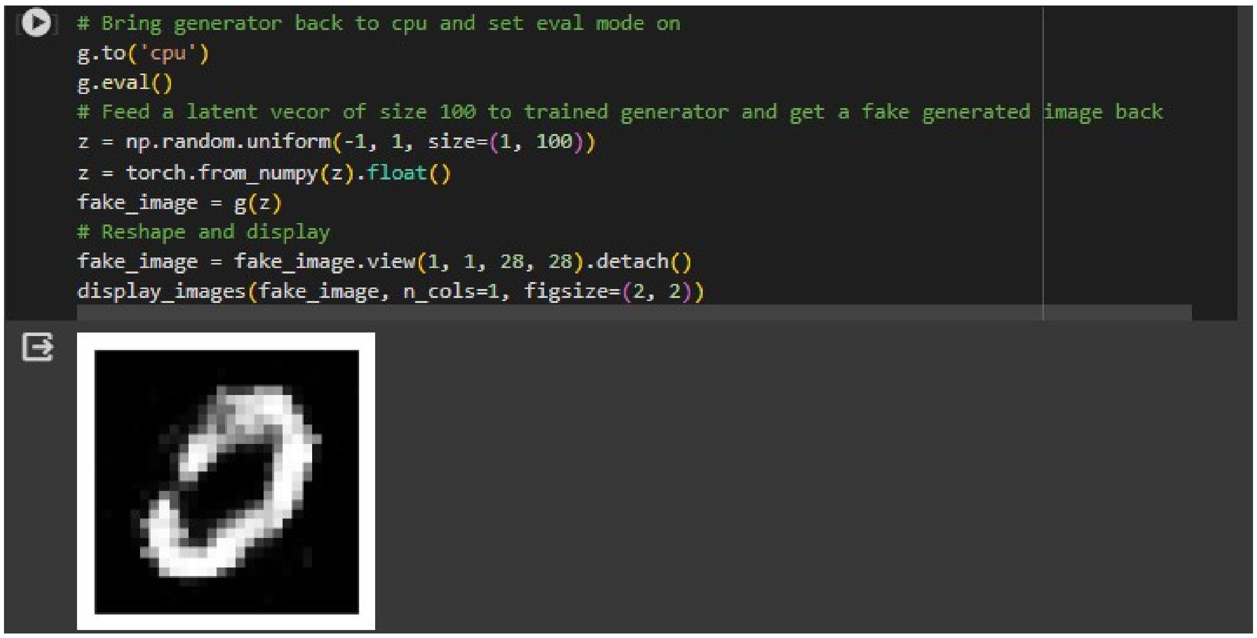
```
[70] show_generated_images(epoch=1, n_cols=8)
```

```
[71] show_generated_images(epoch=10, n_cols=8)
```



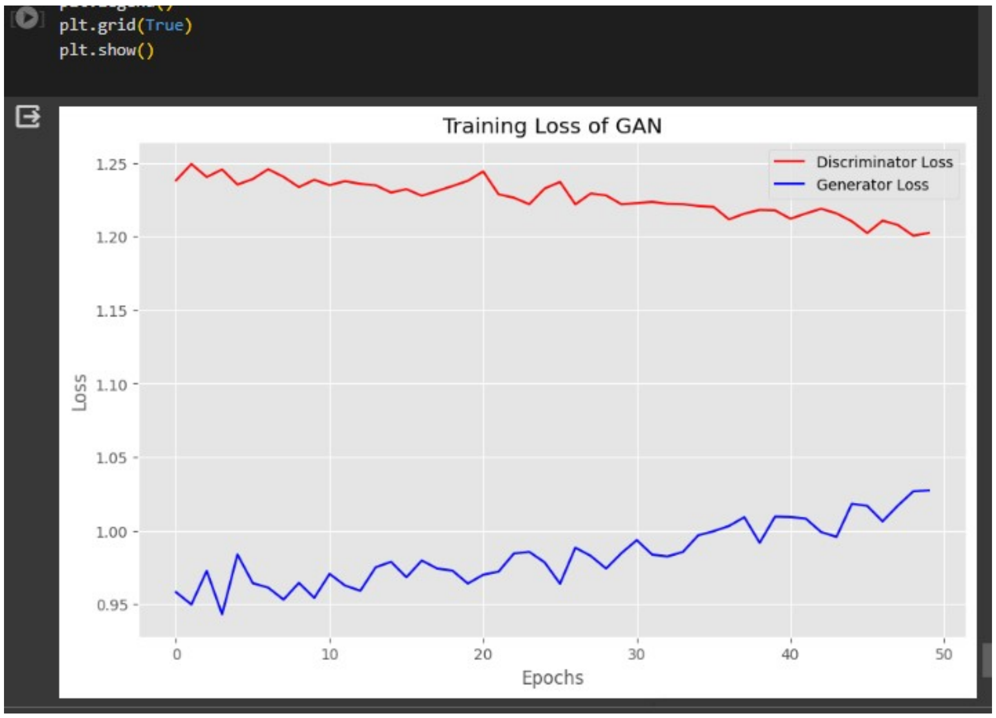
```
[72] show_generated_images(epoch=50, n_cols=8)
```

A randomly generated fake image using the above code is shown below; we can see that the generated fake image looks very similar to the actual number 0 –



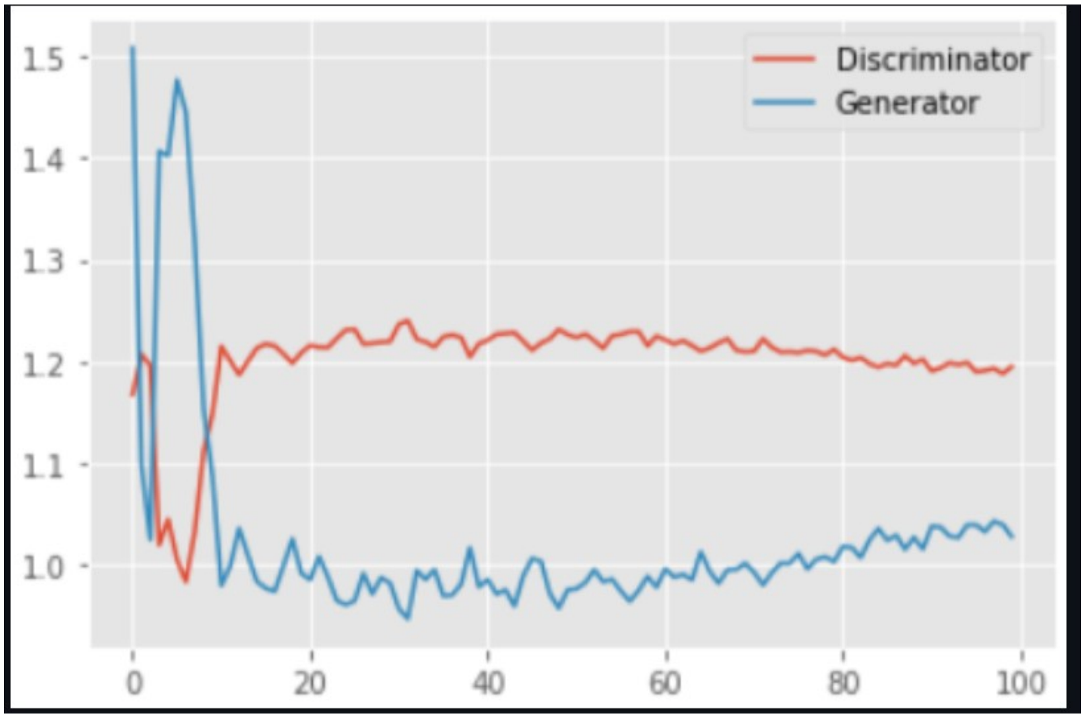
```
# Bring generator back to cpu and set eval mode on
g.to('cpu')
g.eval()
# Feed a latent vecor of size 100 to trained generator and get a fake generated image back
z = np.random.uniform(-1, 1, size=(1, 100))
z = torch.from_numpy(z).float()
fake_image = g(z)
# Reshape and display
fake_image = fake_image.view(1, 1, 28, 28).detach()
display_images(fake_image, n_cols=1, figsize=(2, 2))
```

Discriminator and Generator loss plot



```
plt.grid(True)
plt.show()
```

At the end of the 100th epoch, the discriminator loss (in red) seems to be going down, and generator loss (in blue) increases.



It's possible that if we train our network for even more epochs, losses may converge, indicating an equilibrium between the competing networks. Equilibrium is the fundamental idea behind GAN, suggests that competing rationale agents will ultimately reach an equilibrium where they can't improve anymore.

CONCLUSION:

In conclusion, the Generative Adversarial Network (GAN) framework presents a powerful approach to generating synthetic data that closely resembles real-world datasets. By training the Discriminator and Generator networks in tandem, the GAN learns to produce convincing fake images, in this case, handwritten digits from the MNIST dataset. Through meticulous training and optimization, the Generator refines its ability to generate realistic images, while the Discriminator enhances its discernment to distinguish between real and fake samples. The success of the GAN in generating high-quality synthetic data underscores its potential applications in various domains, from image generation to data augmentation. Moving forward, further research and experimentation could explore advanced GAN architectures and techniques to push the boundaries of synthetic data generation even further.