# Lab 9 - Generative Adversarial Network

IT21184444

GitHub Link- <https://github.com/hansika99u/DL-Lab09.git>

## Part 01

### Question 01

**Modifying Latent Space Size**

* **Smaller Latent Space (e.g., 50)**: With a smaller latent space, the generator has less room to encode variations, which could limit the diversity of the generated images. The images generated may start to look more like each other, potentially reducing overall quality.
* **Larger Latent Space (e.g., 200)**: A larger latent space provides more capacity for encoding diverse features, which may lead to a broader variety of images. However, this can also make training more challenging as the generator has to learn to navigate a larger space effectively.

Overall, increasing the latent dimension tends to increase diversity but may also require more training time or a larger dataset to achieve optimal results.

### Question 02

**Training the GAN for 10,000 Epochs**

* Initial Epochs (0 - 1000): The generator usually produces noisy or blurry images as it begins learning to mimic the data distribution.
* Towards 10,000 Epochs: With sufficient training, you should see more defined and clear images. If training continues past the point of convergence, there's a risk of mode collapse, where the generator produces highly similar images, losing diversity.

Visualizing images at regular intervals (e.g., every 1000 epochs) can help capture progress, which can be compiled into a GIF to observe how the GAN learns over time.

### Question 03

**Changing the Optimizer (Adam to RMSprop)**

* **Adam**: Adam is generally the preferred optimizer for GANs due to its adaptive learning rate and stability. It often results in smoother and more consistent training, producing good quality images with fewer training artifacts.
* **RMSprop**: RMSprop can also perform well but might be slightly less stable than Adam. It’s useful for reducing oscillations in training, but it might require more tuning (like adjusting the learning rate) to match Adam’s performance.

### Question 04

**Experiment with different batch sizes**

Smaller batch sizes (batch size = 32) can introduce more noise into the training process due to higher variance in the gradients. This added noise can sometimes be beneficial for GANs, as it might help prevent the discriminator from overpowering the generator, thereby improving diversity in the generated images.

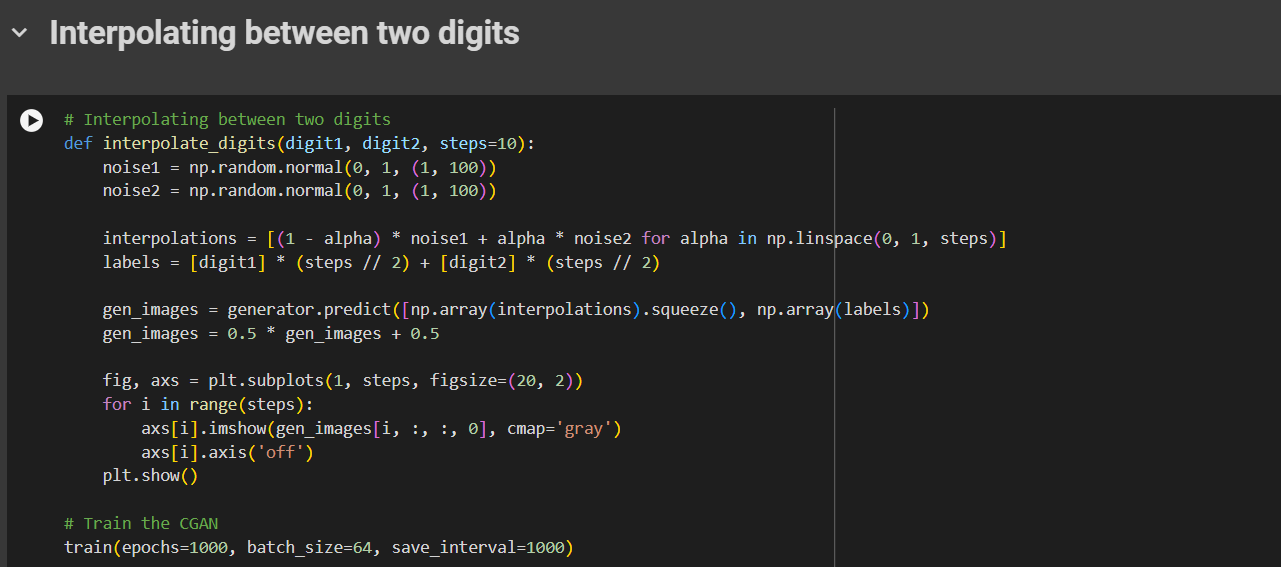
Large batch size (batch size = 64) is commonly used batch size for GANs as it often strikes a balance between stability and noise. It usually provides relatively stable updates with sufficient diversity in each batch, helping the model converge more smoothly.

## Part 2

**Q2 -** Implement label smoothing by replacing real labels of 1 with random values between 0.9 and 1 during training. Analyze how label smoothing affects the training process and the quality of generated images.



**Q3 -** Create noise vectors corresponding to two different digits (e.g., '3' and '8'). Perform interpolation between the two noise vectors and visualize how the generated images morph from one digit to the other.



A group of squares with a white background

Description automatically generated

A group of squares with a white background

Description automatically generated

A group of squares with black dots

Description automatically generated

## Part 3

**Q2 -** Apply data augmentation (random cropping, flipping, etc.) to the real CIFAR-10 images during discriminator training. Evaluate if augmentation improves the discriminator's performance and results in better-generated images

A computer screen shot of a program

Description automatically generated

**Q 3 -** Implement the Inception Score to evaluate the quality of generated images. Train the DCGAN and compute the IS at various points (e.g., 2000, 5000, 10000 epochs). Compare the IS values with the visual quality of generated images

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Q4 -** Add more layers to the generator and discriminator to create deeper models. Train the deeper DCGAN and observe if this complexity improves the quality of generated images.

