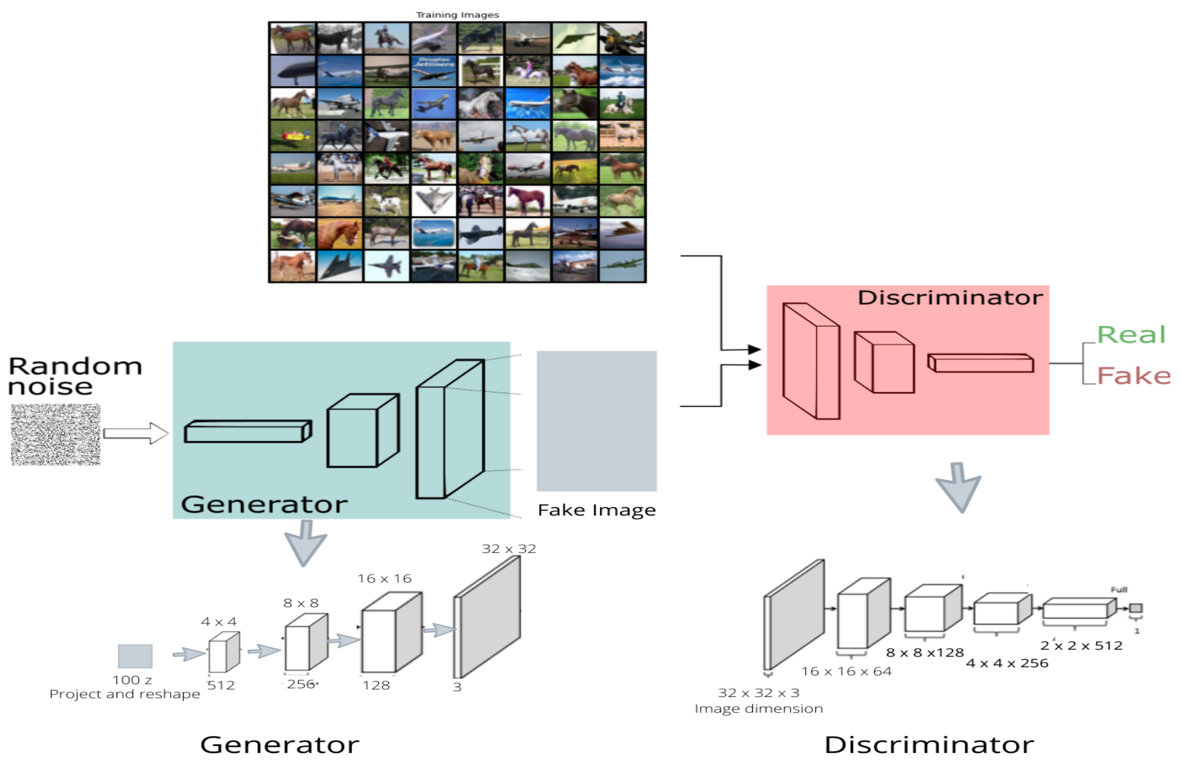
<u>**Deep Learning Coursework- bpxd72**</u>

Architectural design:

The system used the Deep Convolutional Generative Adversarial Network (DCGAN) [1] to generate the image of Pegasus. The implementation of the code was based on the tutorial from [2]. DCGAN is the improve version of Generative Adversarial Network (GAN) since DCGAN implements transposed convolution technique to perform uses the transposed convolution technique to perform up-sampling of 2D image size while GAN only consists of fully connected layers. Generally, GAN is a generative model that comes up with a way of matching their generated distribution to a real data distribution. Both GAN and DCGAN has two models, a generator and a discriminator. The generator creates a fake data in order to trick the discriminator. The discriminator checks whether the image generated from the generator is fake or real. The generator will create better fake images while the discriminator will improve its evaluation whether the images are fake or real as the training progresses further.

The introduction of Deep Convolutional networks in DCGAN make DCGAN is more suitable for images; hence this is the reason why DCGAN is chosen as the main method in the system. Its simplicity compared to other methods such as StyleGAN and BigGAN while still generating a good output [1] than GAN. Hence, DCGAN is ideal to be implemented in the system. DCGAN's generator consists of convolutional-transpose layers, batch norm layers, and LeakyReLU activations. Its discriminator consists of convolution layers, batch norm layers, and LeakyReLU activations. The final architecture of the system is shown as below:



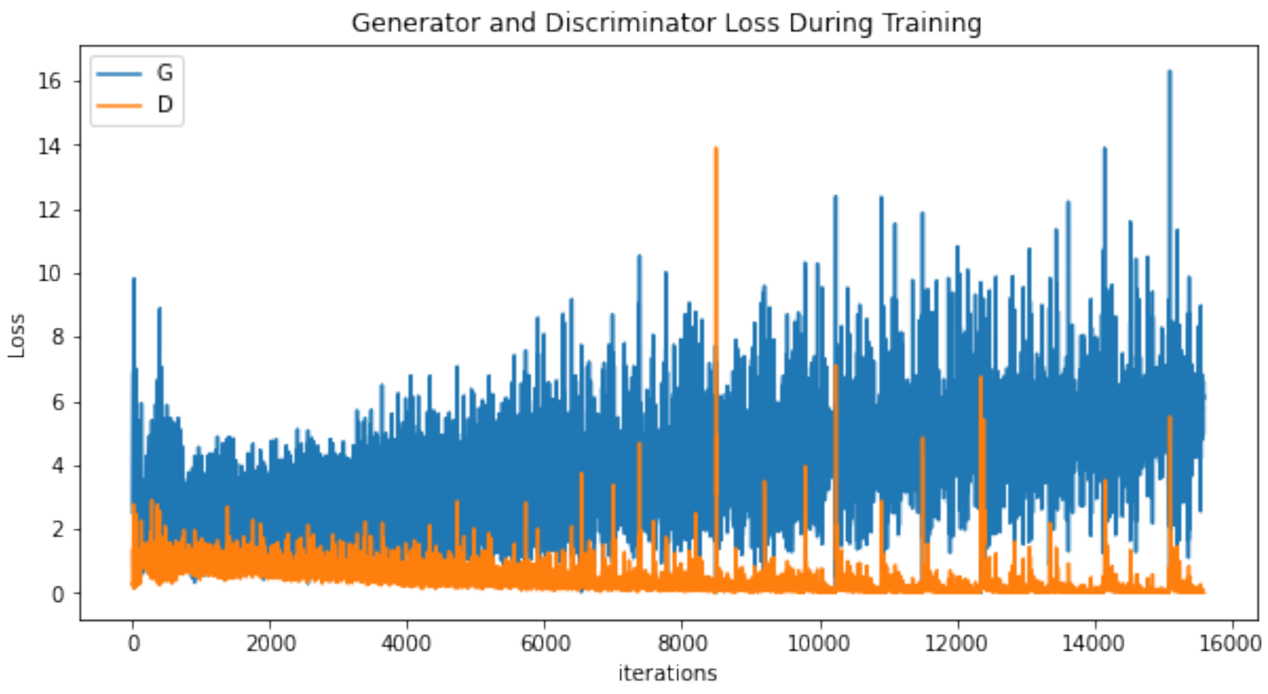
Architecture of DGGAN Implemented In The System

Discussion Of Design

The system followed the implementation created by N. Inkawhich [1]. The size of z latent vector that was used to create the fake images is 100, and both sizes of feature map in generator and discriminator followed the same setting like the referred implementation, which was 64. For weight initialization in Generator and Discriminator, weights were randomly initialized from a Normal distribution with mean=0, stdev=0.02. The optimizer used was Adam optimizer with 0.5 beta1 hyperparameter.

In order to produce a good result, the best number of epochs was needed to be identified. The design used 100 epochs since the loss of generator and discriminator converged beyond 100 epochs. The images generated from 100 epochs were also great. Learning rate for the optimizer was chosen accordingly value proposed by the author of DCGAN [2] which was 0.0002. Further experiments were conducted to check whether the different value of the learning rate would produce a greater result compared to the original value.

The result of the loss for both generator and discriminator can be shown below:

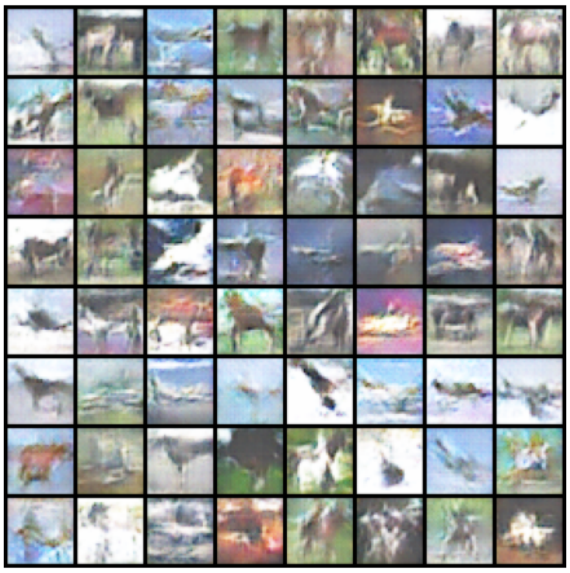


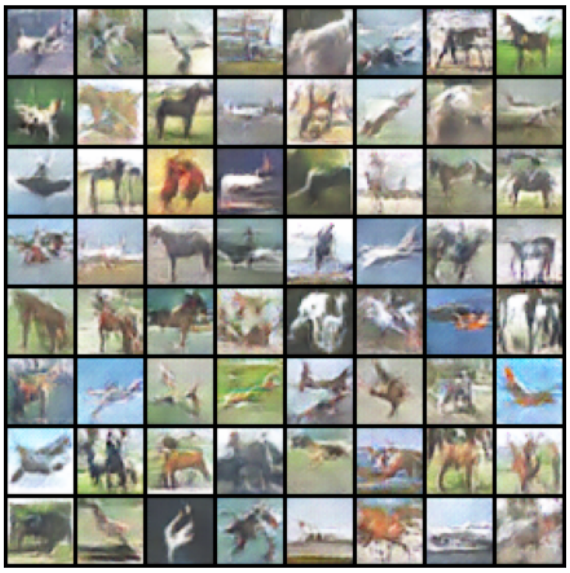
Graph of Generator Loss and Discriminator Loss

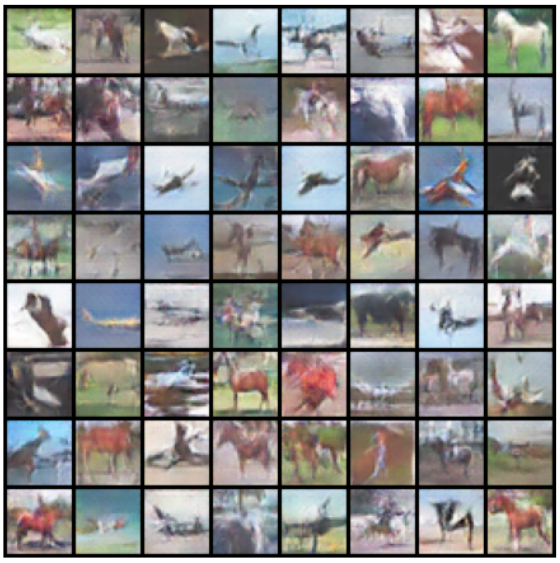
Based on the result, the discriminator loss decreased and converged at a point where it almost reached 0. In theory, it showed that the discriminator could guess the fake images accurately. However, for generator, the loss decreased below 2000 iterations, but it started to increase beyond 2000 iterations and stabilized beyond 14000 iterations. The random fluctuations which mean higher variance can also be observed on the generator loss. This scenario showed that the generator could not trick the discriminator anymore; hence why the generator loss is high. In order to improve the result of the loss, few experiments were conducted accordingly to the guides provided by [1]

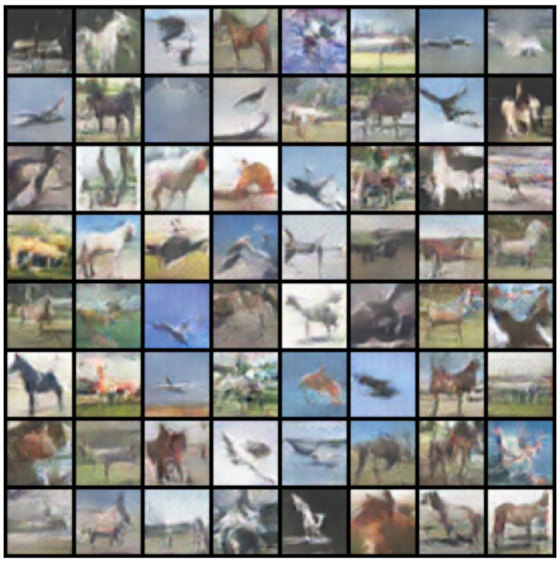
Experiments:

Number of epochs. An experiment has been conducted to check the best number of epochs for the training process. In order to evaluate the result, the quality of batch images created were observed. The value of epochs selected were 25, 50 and 100,150. Based on the result, the quality of the batch images created by epochs 100 and 150 were similar; hence we selected the least epoch, which was 100.

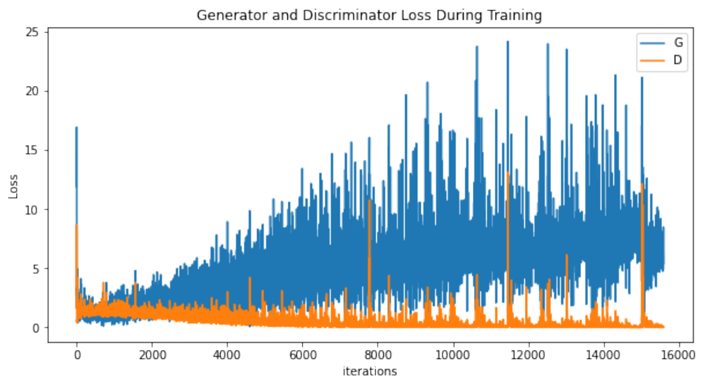

25 Epochs


50 Epochs

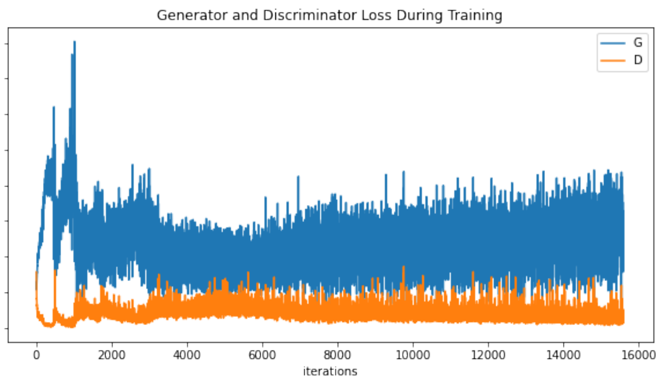

100 Epochs


150 Epochs

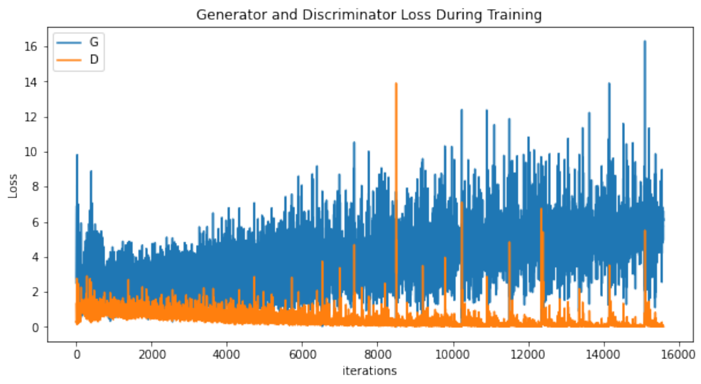
Value of learning rate. An experiment has been conducted to check the best value of the learning rate (LR) for the optimizer during the training process. In order to evaluate the result, the loss of generator and discriminator were evaluated. As showed in the result below, 0.002 was the best learning rate for the implementation.



LR: 0.002

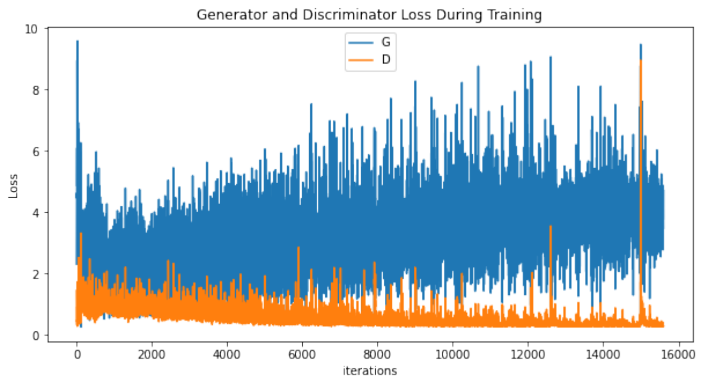


LR: 0.00002

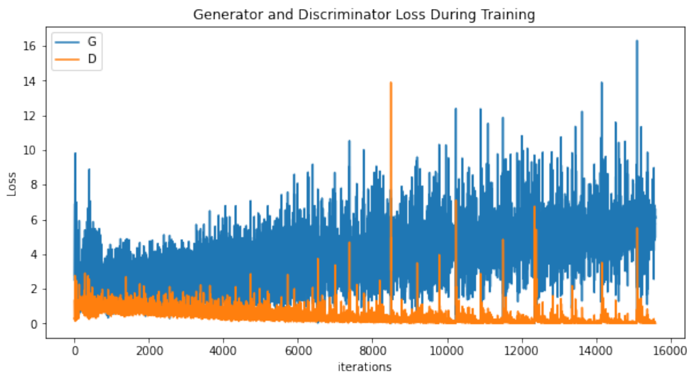


LR: 0.0002

Label smoothing. Based on the approach proposed by [3], label smoothing may increase the performance of GAN. Rather than labelling 0 for fake images and 1 for real images, a random number between 0.7 and 1.2 was selected for real images, and if it is a fake sample, replace it with 0.0 and 0.3. The expected outcome would be the loss of the generator would be decreasing. However, the proposed method could not produce a better result; hence the initial was selected.



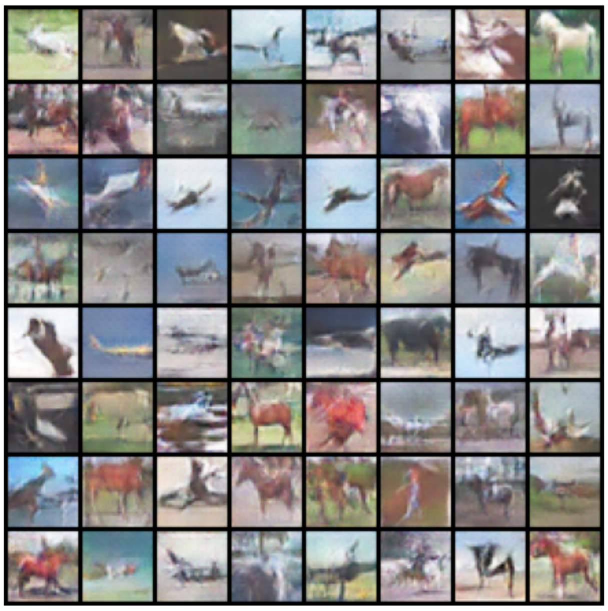
Loss With Smoothing



Loss Without Smoothing

Generating a Pegasus:

CIFAR-10 datasets was used In order to generate a pegasus. Rather than using all the ten classes for training, horse and airplane class were chosen to undergo the training. Horse class was chosen since Pegasus has the main features of a horse (head, body, legs, tail). Airplane class has an important feature of Pegasus, which was the wing. Bird class did not include in the training since most of the images were stationary, and it was very difficult to generate a wing using the images of the bird. The combination of bird and horse also did not produce a great result compared to the combination of horse and airplane


A batch of 64 samples from DCGAN

Best result:

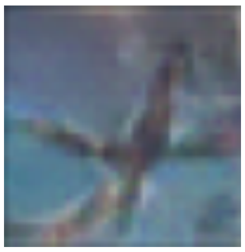


Image of Pegasus

Reference

[1] A. Radford, L. Metz and S. Chintala, UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS, arXiv preprint arXiv:1511.06434, 2016.

[2] N. Inkawhich, DCGAN TUTORIAL, 2017. [Online]. Available: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html. [Accessed: 28- Mar- 2020].

[3] T. Salimans, I. Goodfellow, W. Zaremba, V.Cheung, A. Radford, X.chen, IMPROVED TECHNIQUES FOR TRAINING GANS, arXiv preprint arXiv:1606.03498v1 [cs.LG], 2016