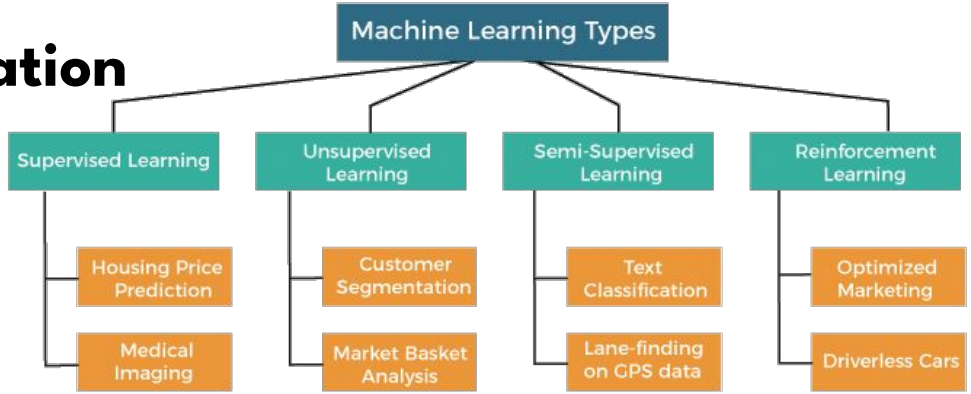
Exploring GANs and Generative Models

An overview of Generative Adversarial Networks (GANs) and generative models with a focus on their application to the CIFAR 10 dataset.

Joel Poah
P2112729
DAAA/2B/06

Introduction To Image Generation



- Deep Recurrent Attentive Networks Writers
- Variational Autoencoders
- Parametric Models like DCGANs

Why Generative Adversarial Networks?

- Generate images using labels CGAN
- Handle high dimensional data
- Produce diverse and high quality images
- Unsupervised Learning DCGAN

A discriminator and generator are both Convolutional networks. The discriminator is trained to distinguish between real and fake images while the generator is trained to fool the discriminator into thinking that the fake images are real. While training the weights of the discriminator , the generator learns from the gradient of the discriminator to generate images of a similar distribution.

```
10 classes =['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

DC-GANs EDA & Feature Engineering

- **Batch size training**

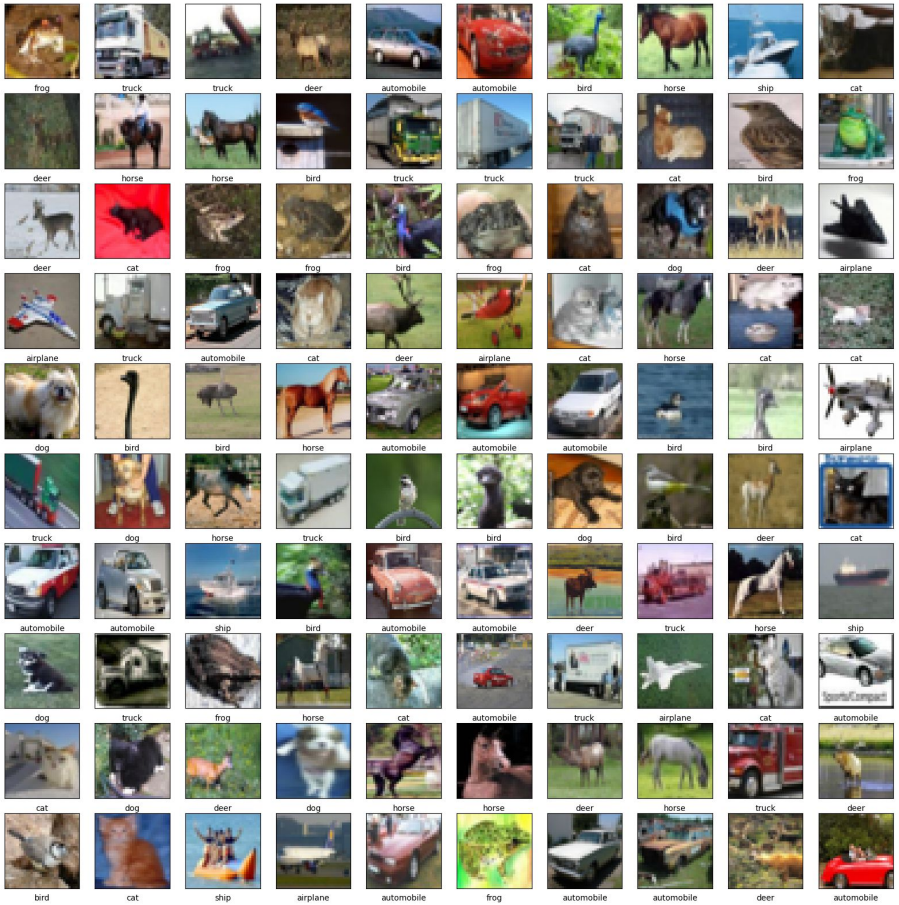
The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. Instead of having to update only after the whole dataset is trained once. It is indeed better to update after training each batch as it speeds up training time

- **3 Channels x 32 x 32 pixels**

The number of channels is the number of color channels in the image. For example, RGB images have 3 channels, while grayscale images have 1 channel. The number of channels is also known as the depth of the image.

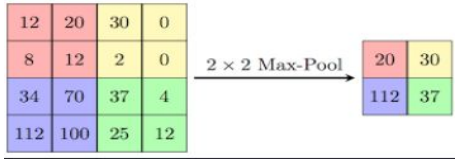
- **Pixel Scaling + Converting to Tensors**

normalization vs (between -1 and 1 AKA Centering) There is a lot of debate on which one is better and although majority has tried and concluded that zero-centered Normal distribution is better. I will still be trying both and see which one works better.I will be converting the images to tensors as it helps with batch processing and also helps with the GPU processing for faster training



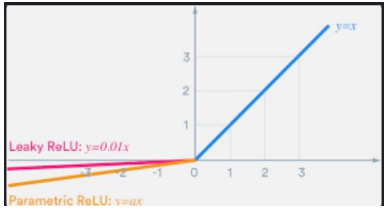
Take note before generating CIFAR 10

Don't use Max pooling



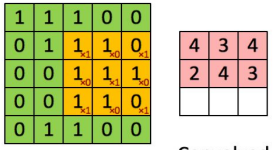
GAN researchers have figured that maxpooling layers do not help generating images as it prevents the neural network model from learning its own spatial downsampling. (Radford et al., 2015) This is the reason why most GANs example we see these days are fully strided convolutional layers 0 pooling.

Use Leaky Relu



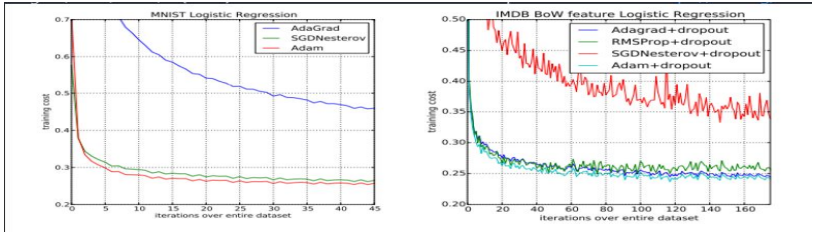
Leaky relu is used instead of relu because it prevents the dying relu problem. The dying relu problem is when the gradient of the relu function is 0 and the neural network model stops learning. Leaky relu solves this problem by having a small negative slope when the input is negative. This allows the neural network model to continue learning even when the gradient is 0.

Only Fully convolutional layers / Conv Transpose layer



the main purpose of doing this is to find features of a class which will then be used for determining if the image is real or fake. In having more convolutions, the neural network model will be able to learn more features of the image and in turn the generator will learn the features of the image and be able to generate images.

ADAM



Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. ArXiv. https://doi.org/10.48550/arXiv.1412.6980 [Accessed: 2021-01-31]

Adam is better because of its ability to be adaptive in scaling the gradient to avoid exploding or vanishing gradient yet have balancing momentum to avoid local minima for faster convergence and lesser cost

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

Training DCGAN Model 1

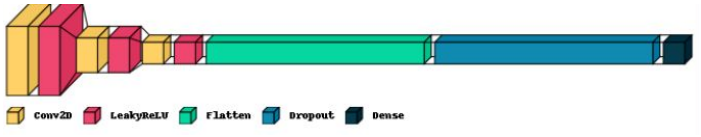
Unstable training + Mode Collapse

Mode collapse is a phenomenon that occurs in generative adversarial networks (GANs) / The generator gets stuck in generating a set of similar images over and over again . The generator loss has stayed stagnant after epoch 200 at around 0.89 and never improved . The discriminator's loss was also stagnant at 0.68 . Mode collapse can happen because of a few reasons

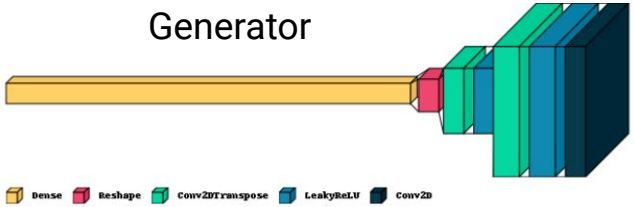
- The discriminator is stuck in a local minimum and is unable to detect the generator's fake images , the generator takes advantage of this and continues to generate similar images to fool the discriminator
- The discriminator is too powerful and can easily detect the generator's fake images so the generator keeps creating unseen images to fool the discriminator

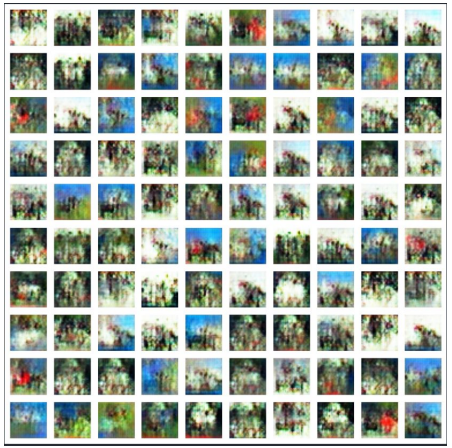
Trained using `tf.gradienttape()` which utilises automatic differentiation for faster gradient computation

Discriminator

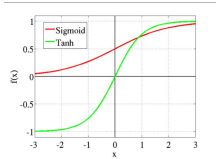


Generator



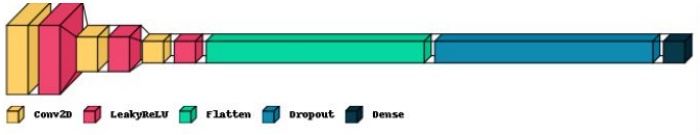


Training DCGAN Model 2

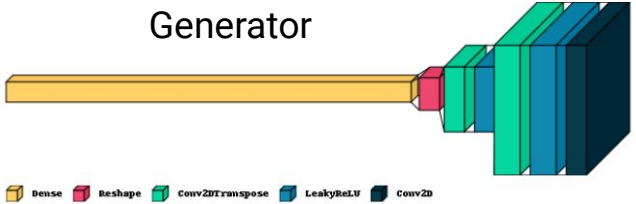


Same model architecture except tanh activation

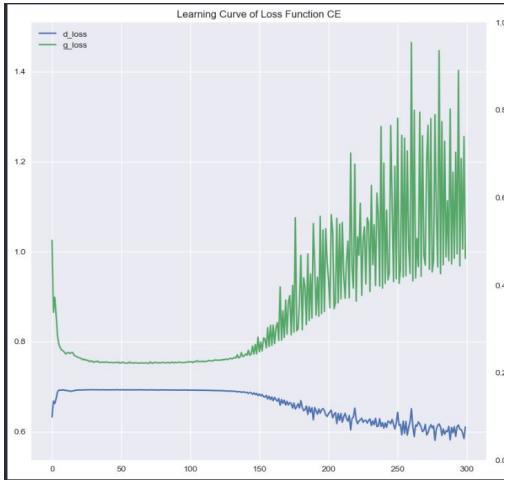
Discriminator



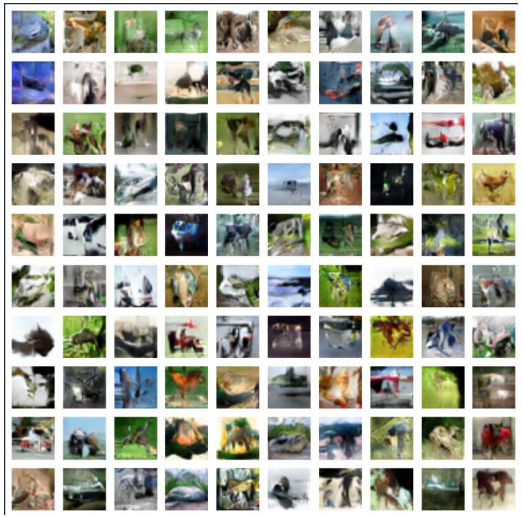
Generator



The only improvement was Scaling pixel values to [-1,1]



- Significant improvement with pixel centering
- Images are more vibrant and seemed more realistic



Practical steps for me to improve on my model

- **Spectral normalization** for discriminator
- **Batch normalization** for generator

Code credit: https://wandb.ai/ayush-thakur/gan-evaluation/reports/How-to-Evaluate-GANs-using-Frechet-Inception-Distance-FID---Vmlldzo0MTAxOTI

> Radford, A., Metz, L., Chintala, S., 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [WWW Document]. arXiv.org. URL https://arxiv.org/abs/1511.06434v2 (accessed 1.30.23).

> Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y., 2018. Spectral Normalization for Generative Adversarial Networks [WWW Document]. arXiv.org. URL https://arxiv.org/abs/1802.05957v1 (accessed 1.31.23).

Training DCGAN Model 3

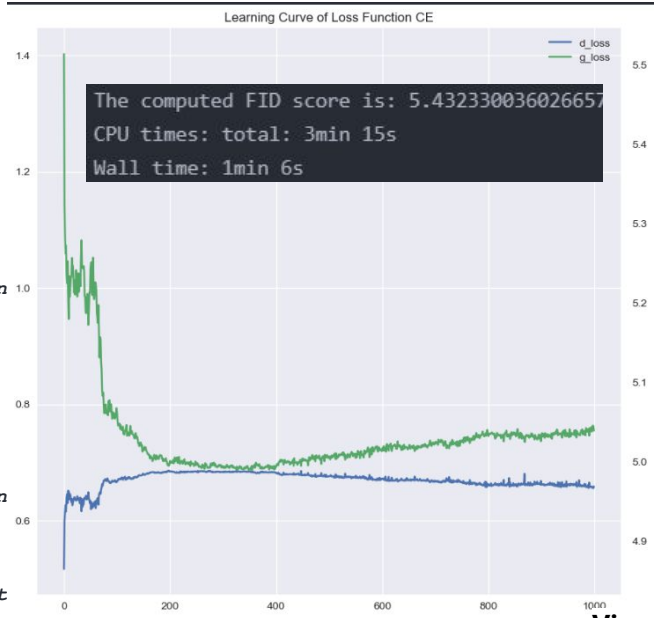
Improvements made:

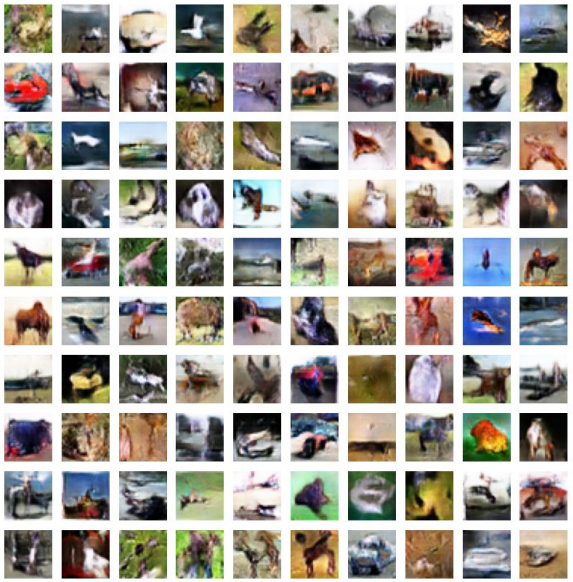
Spectral Normalization (Discriminator)

Spectral Normalization is a technique used to regularize weights of a neural network preventing overfitting. It does so by controlling Lipschitz constant also known as the maximum singular value of the discriminator function. When the convolutional layers in discriminator is training the weights, it can sometimes become too small or too large causing unstable training therefore spectral normalization is used to control the Lipschitz constant which is the magnitude of the network's learned features.

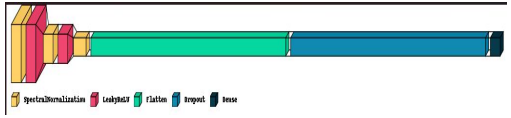
Batch Normalization (Generator)

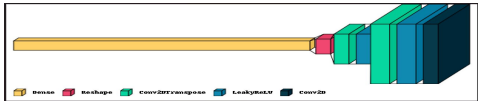
Covariate shift is not good because if the distribution keeps changing then them model will forever be trying to chase after a moving target of finding the best weights. Batch normalization helps to reduce the effect of covariate shift by normalizing the inputs to each layer and making the distribution of the inputs to each layer more stable





- Converges at 200-400
- Diverges after 400 epochs





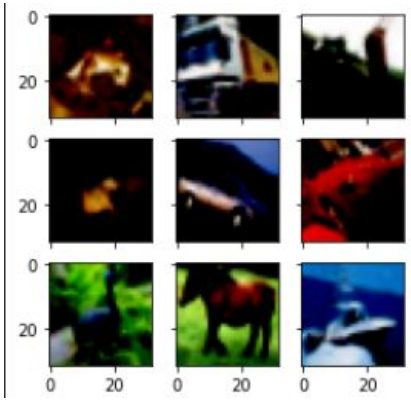
Visually its still hard to tell what images the model is trying to create but I would say the images are slightly more vibrant and clearer than the previous model as with more layers and neurons the model is able to learn more complex features

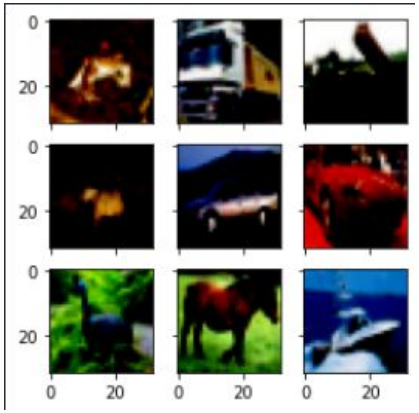
DCGAN Model 4

What I did to improve the final DCGAN model

- Augmenting by **Rotating 20 degrees**
- **Flipping** images horizontally
- Same Model architecture
- **ReLU for Generator** instead of leaky

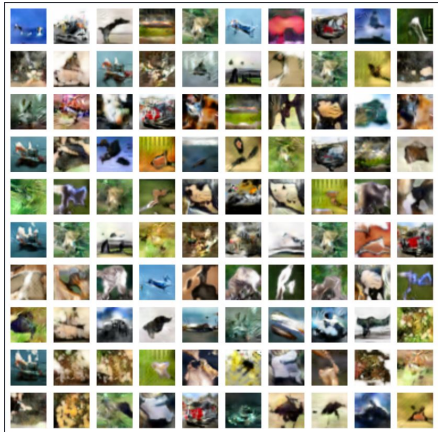
using a bounded activation allowed the model to learn more quickly to saturate and cover the color space of the training distribution.(Radford et al., 2015)





Conclusion for DCGAN

- **I still think the images are not realistic as it is still very blurry with glitches and I hope to see better results with the next few models I will be trying out**
- **Model did not converge but am fine with that since it is quite difficult for convergence**
- **FID score is quite good and I am satisfied with all the improvements method that were recommended by researchers and scholars**



```
The computed FID score is: 2.3096476762356404
CPU times: total: 3min 33s
Wall time: 1min 13s
```

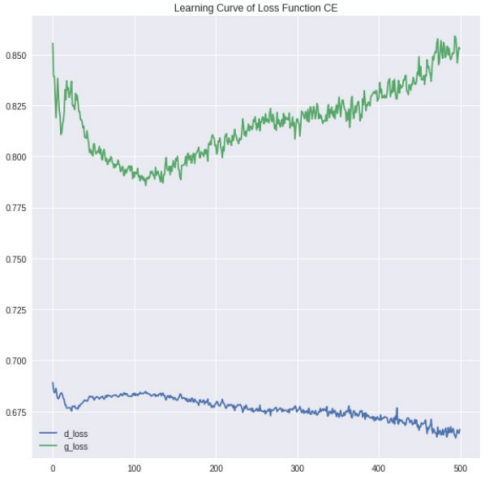
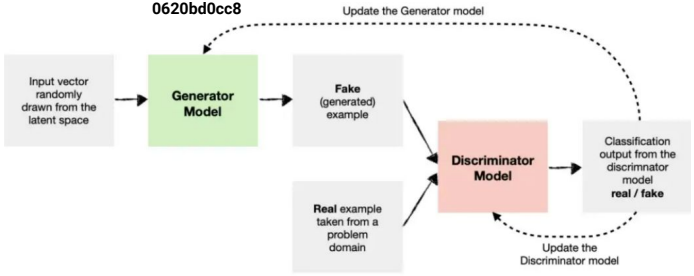
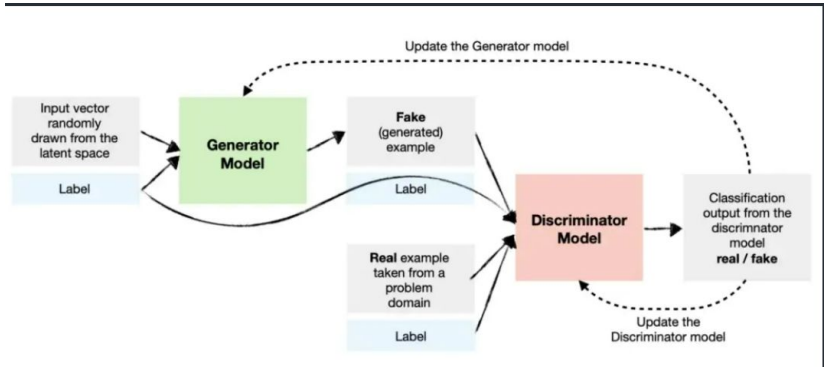


image by Saul Dobilas
https://towardsdatascience.com/cgan-conditional-generative-adversarial-network-how-to-gain-control-over-gan-outputs-b30620bd0cc8





A GAN contains a generator and a discriminator. A discriminator can be like a Convolutional Neural Network that tries to determine whether an image is real or fake. While doing so, the discriminator leaves behind a gradient that the generator can use to improve its own output. The generator competes with the discriminator and generates images. However when exploring simple GANs and looking at the output I noticed that the images was difficult to interpret or it seemed like the images was a little bit of an airplane and a little bit of a car.

> CGAN is different from GAN where not only the pixels of real images are passed into the model , labels of each image is also passed in . As such the generator model generates images with a specific label

CDCGAN contains additional Convolutional transpose layers in the generator and Convolutional layers in the Discriminator

We try to add the latent dimensions and the number of classes in the image so that we can condition the generator to be able to generate images based on classes. This is also true for the discriminator such that the discriminator that distinguish whether an image is fake and also the class of it.

```
generator_in_channels = latent_dim + num_classes
discriminator_in_channels = num_channels + num_classes
print(generator_in_channels, discriminator_in_channels)
```

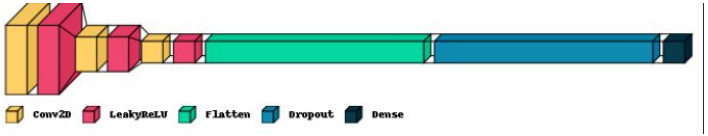
```
138 13
```

```
random_latent_vectors = tf.random.normal(shape=(batch_size, self.latent_dim))
random_vector_labels = tf.concat(
    [random_latent_vectors, one_hot_labels], axis=1
)
# Decode the noise (guided by labels) to fake images.
generated_images = self.generator(random_vector_labels)
```

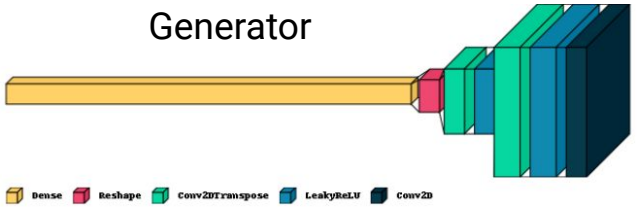
Training CGAN Model 1

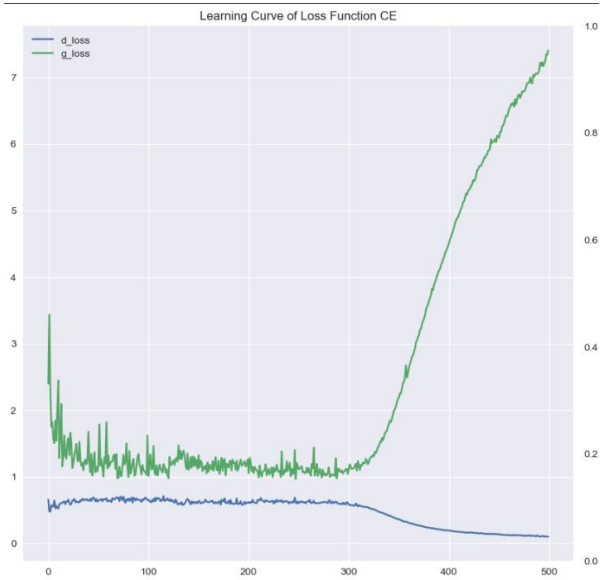
- **Scaling pixel values to [-1,1]**
- **Leaky ReLU for both** (testing hypothesis
- Same Model architecture
- **Spectral Normalization**

Discriminator



Generator



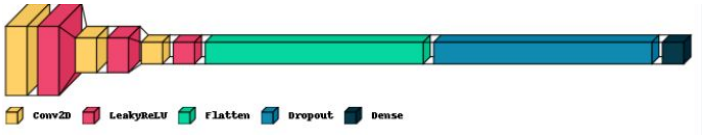


- Model trained for 500 epochs
- Model didn't really converge to a good result and the images generated is better than DCGAN perhaps due to the fact that the generator is conditioned on the class labels

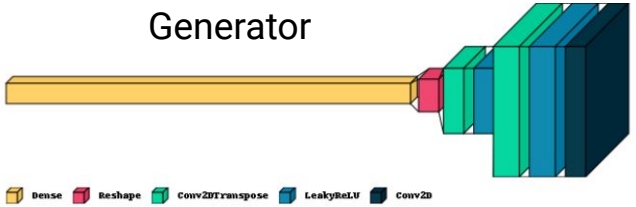
Training CGAN Model 2

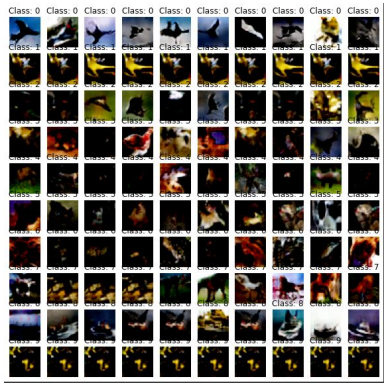
- Augmenting by Rotating 20 degrees
- Flipping images horizontally
- Add label smoothing

Discriminator



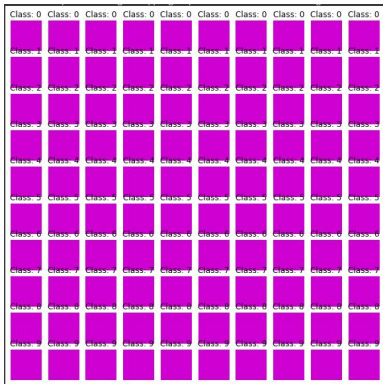
Generator





200 epochs

Mode collapse :(

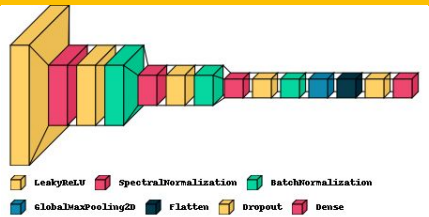


1000 epochs

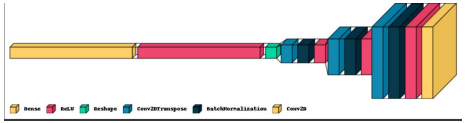
Training SNCGAN

- Reduced 850k Params for Generator
- Reduced 400k Params for Discriminator
  - Dropout to regularize Discriminator
- Batch Normalization for both Generator and Discriminator with 0.8 momentum and 0.00001 epsilon

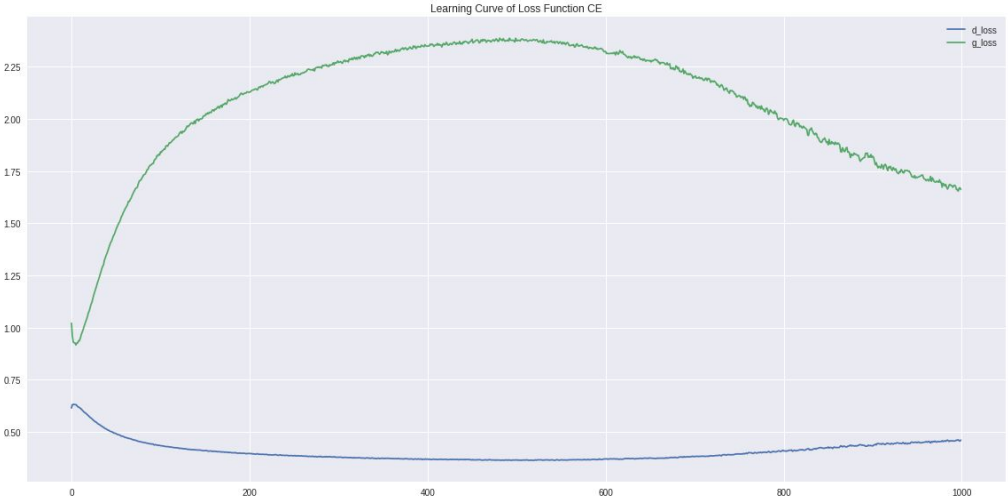
Discriminator

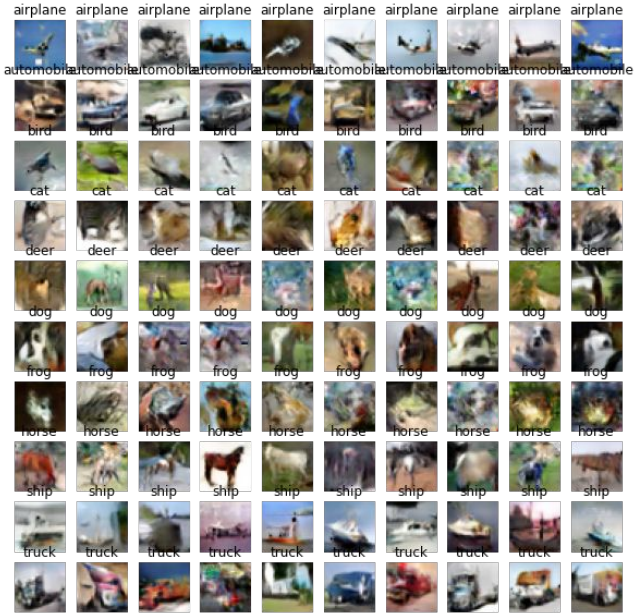


Generator



```
The computed FID score is: 2.2241562222058104
CPU times: user 3min 22s, sys: 36.6 s, total: 3min 58s
Wall time: 42.1 s
```





WGAN Concept

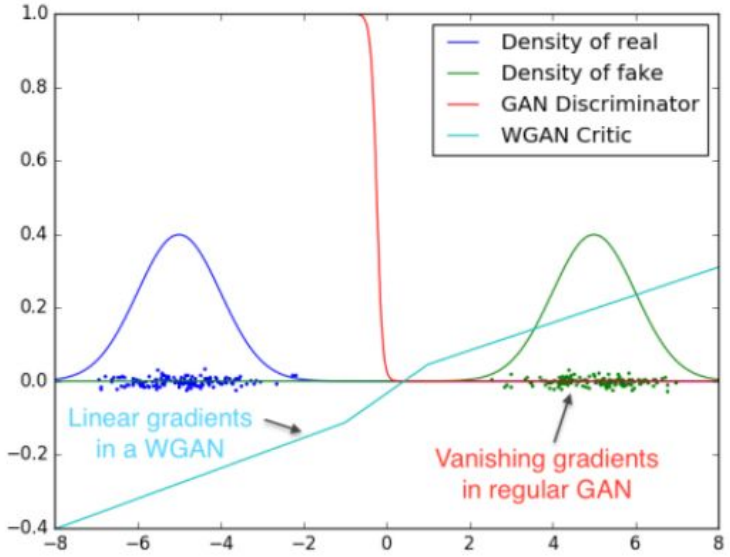
Hui, J., 2021. GAN — Wasserstein GAN & WGAN-GP [WWW Document]. Medium. URL https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490 (accessed 1.31.23).

Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein GAN [WWW Document]. arXiv.org. URL https://arxiv.org/abs/1701.07875v3 (accessed 2.1.23).

$$D_{KL}(P||Q) = \sum_{x=1}^{N} P(x) \log \frac{P(x)}{Q(x)}$$

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q||\frac{p+q}{2})$$

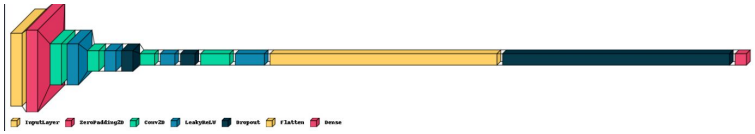
WGAN first looked at KL and JS divergence above and realised that if the 2nd Gaussian distribution Q becomes too large P(x)/Q(x) will give a really small number , gradient of divergency diminishes and generator learns nothing.



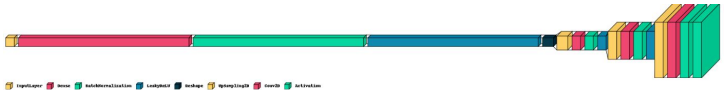
The figure above from the original WGAN paper (Arjovsky et al., 2017) shows an optimal discriminator trying to differentiate two Gaussians (aka 2 types of images) using the original loss function Binary Cross entropy = abs(y_pred - y_true). However as it trains , the discriminator draws the boundaries but also result in a vanishing gradient. This is a huge problem because without the critic's gradient being fed to the generator , training process stops. However, the Wasserstein loss appears to provide a clean gradient for the generator to continue learning

Experimenting WGAN

Discriminator

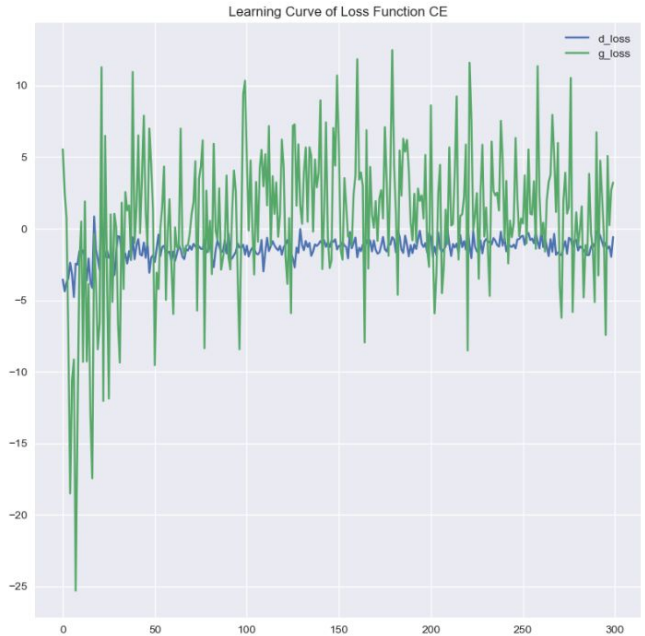


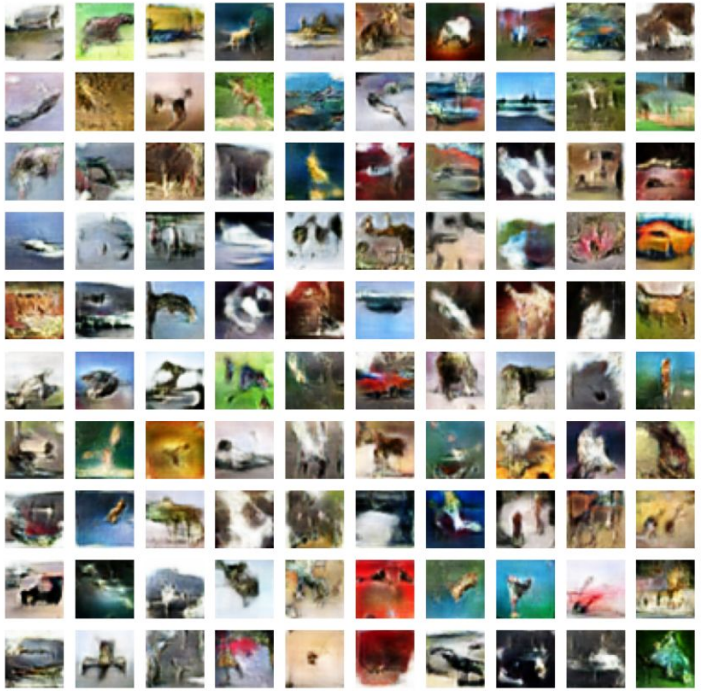
Generator



Scaling pixel values to [-1,1]

WGAN is a better GAN than GAN because it has a better loss function and it is more stable. However, the images are not superb and I would say the images are kind of the same or maybe just slightly better than the original DCGAN images. I would say it is too computationally expensive to continue exploring WGAN and even trying out Data augmentation which will take a lot longer than the current Spectral Normalization CGAN that i did before this. Therefore I will submit the images generated by Spectral Normalization CGAN as my final submission.





Conclusion 😄

- Pixel Scaling (-1 to 1) though no heuristics to prove but it is very effective
- Don’t use Maxpooling , use Conv layers to allow the NN to learn features
- leakyRelu for Discriminator and ReLU for Generator
- Use ADAM to continue testing hypothesis
- Data augmentation did not enhance the quality of the image generated by a significant margin
- Training GANs with labels did improve the images alot
- Spectral Normalization helps stabilize training by controlling how much the weights can stretch
- WGAN solves the vanishing gradient problem and helps with mode collapse
- WGAN may not be the best solution all the time depending on situation
- I decided that WGAN is too computationally expensive but it may be different for others.
- SN-CGAN was my best model that I used to generate 1000 images