DEEL CA2 Part A - GAN

2B04 - Raymond Ng

Abstract - Background Research

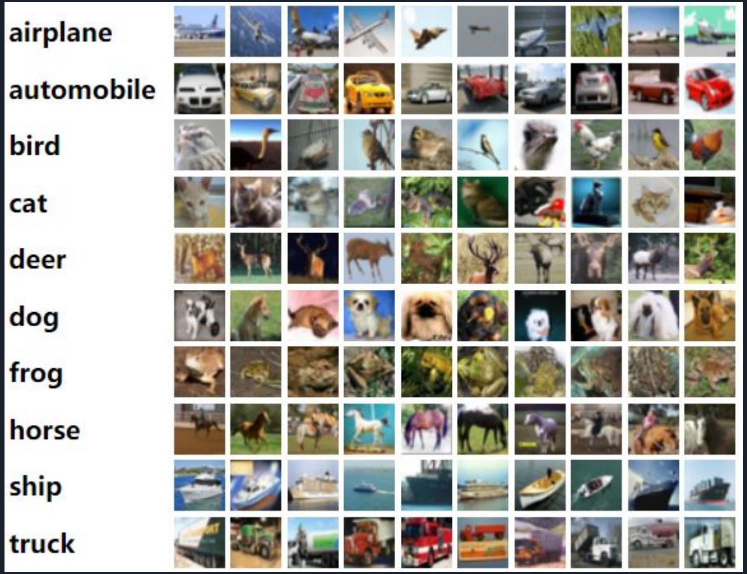
Purpose of this assignment was to create a GAN model and evaluate the performance of this network on the CIFAR10 dataset and then proceed to create 1000 small colour images.

Generative Adversarial Networks (GAN) is a machine learning model where 2 neural networks of Generator and Discriminator compete in a zero-sum game to become more accurate in predictions. GANs are difficult to train due to mode collapse and non-convergence. Models created may be unstable and take longer amounts of time to train. Thus, I attempted multiple different architectures to find the best one for this context of CIFAR-10.

Dataset

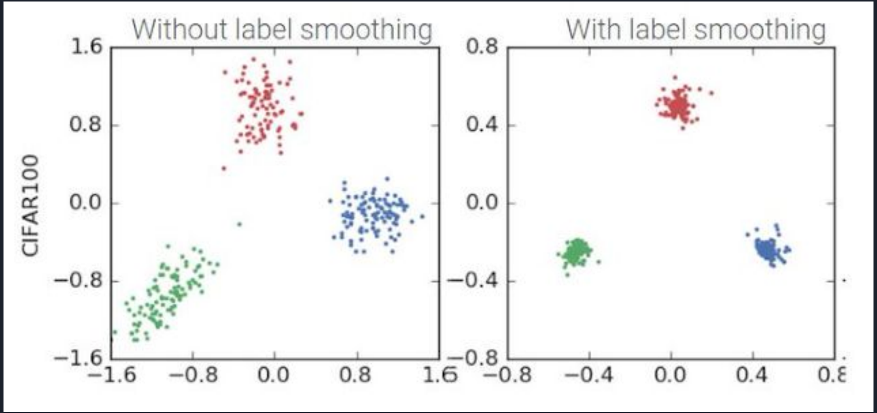
GAN image generation was done on the CIFAR-10 dataset

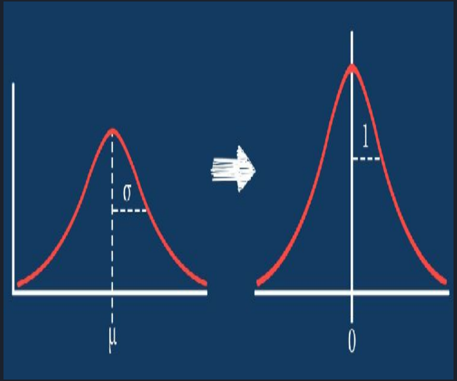
- Experimented with not just conditional GAN
- 60,000 Images of 32x32 size
- 50,000 images for training, 10,000 for validation was concatenated together
- 10 Classes with labels specified for CGANs

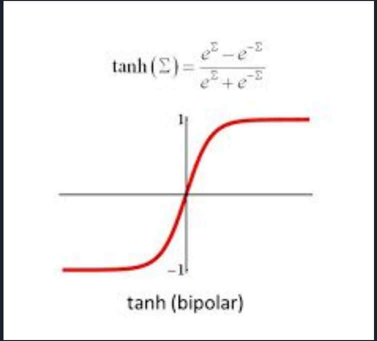


Data Preprocessing

1. Data is normalized with 0.5 values as recommended in a technical paper
2. Mean and Std values were calculated from CIFAR-10 and used later on for transforms
3. GANs don't require a training and testing set thus, datasets are concatenated
4. Pixel Normalization of between -1 and 1 is used on real images since generator output of 'tanh', this is to ensure similar data scales to generated image
5. Label smoothing was introduced to increase robustness and classification for my conditional models to prevent them from being overly confident
6. Transformed the data to tensor and put into data loader
7. Experiemented with other normalizations including spectral norm

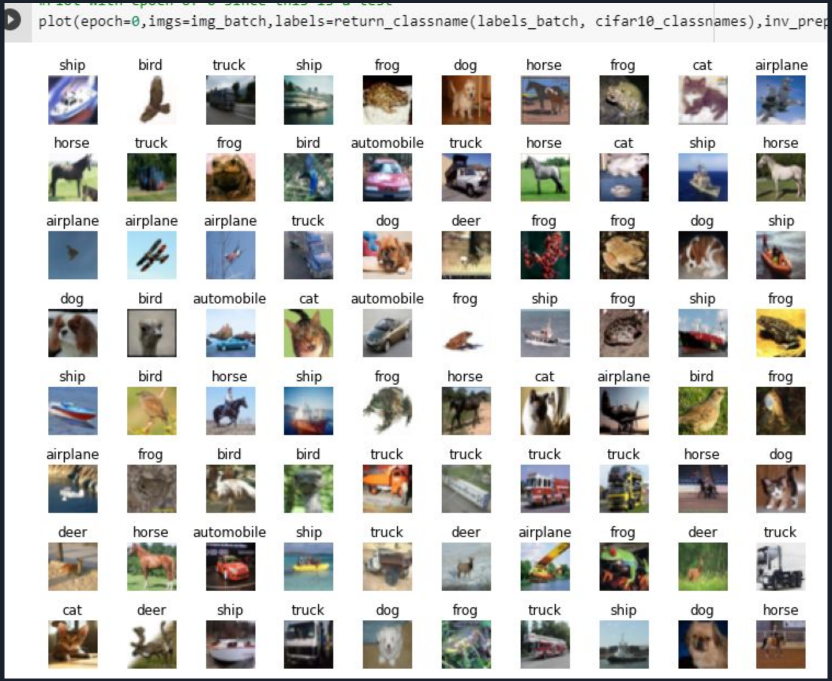






Visualizations

1. Since normalized images are used to train GAN, GAN will product images in that same range, Inverse normalize calculated to be used on the output to see the visualizations
2. Plot image in batches with their labels, this plot function is used to save image in wandb as well
3. Create grid of images and allow the batches to be saved



```
plot(epoch=0,imgs=img_batch,labels=return_classname(labels_batch, cifar10_classnames),inv_prep
```

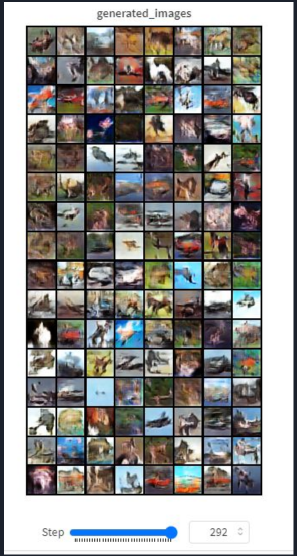
ship bird truck ship frog dog horse frog cat airplane
horse truck frog bird automobile truck horse cat ship horse
airplane airplane airplane truck dog deer frog frog dog ship
dog bird automobile cat automobile frog ship frog ship frog
ship bird horse ship frog horse cat airplane bird frog
airplane frog bird bird truck truck truck truck horse dog
deer horse automobile ship truck deer airplane frog deer truck
cat deer ship truck dog frog truck ship dog horse

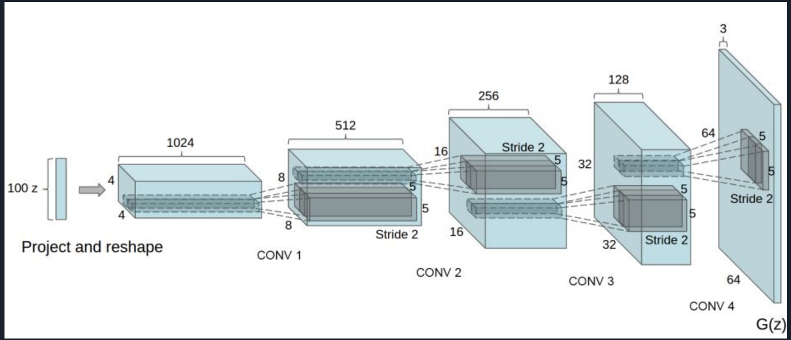
Feature Engineering or Modeling Helper Functions

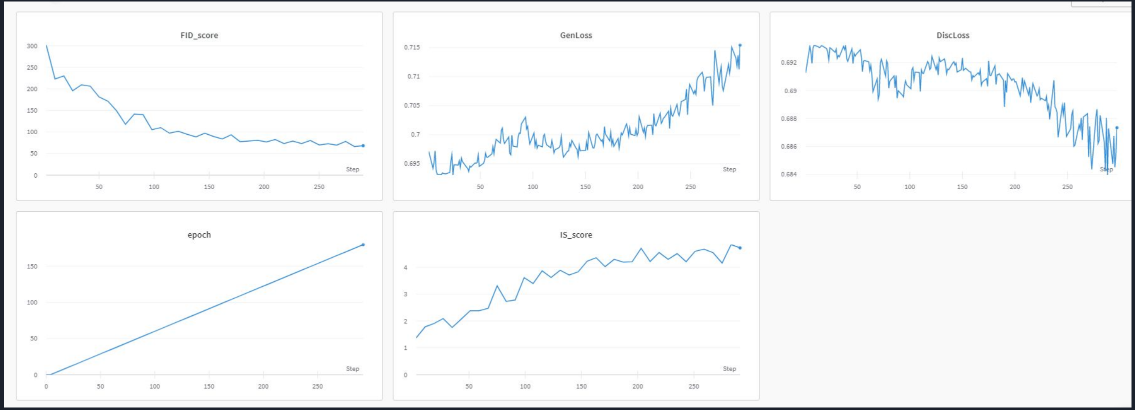
- return_classname() is to return the classnames based on label tensor
- plot() is to generate plots of image batch for visualization and saving on wandb and files
- save_gen_img() saves the image in a specific file, along with labels, generating 1000 images of 100 for each class
- create_noise() creates the noise vector
- Label_real and label_fake is to create real and fake labels, real=1, fake=0
- weights_init() is to weight initialize the model based on layers it has
- D_hinge and g_hinge is from Hinge loss from `https://github.com/POSTECH-CVLab/PyTorch-StudioGAN/blob/8c9aa5a2e9bb33eca711327c085db5ce50ff7fc0/src/utils/losses.py`
- train_batch() is to train one batch of images
- evaluate() is to evaluate with FID and IS Score and log it into wandB

DCGAN

- DCGAN was the first architecture attempted
- Code was inspired from the technical paper
- https://arxiv.org/abs/1511.06434v2
- Using convolution layers that are similar to the ones used in CA1 CNNs to make a DCGAN
- Experimented with DCGAN with best run of FID: 68 and IS:4.7
- However Gen Loss had unstable upwards trend which shouldnt be happening. In other runs, FID: 150, IS: 2.7 but good loss trends

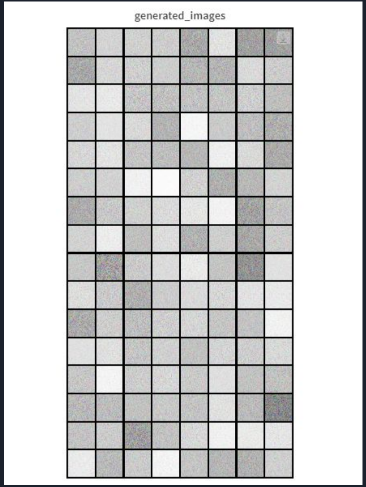


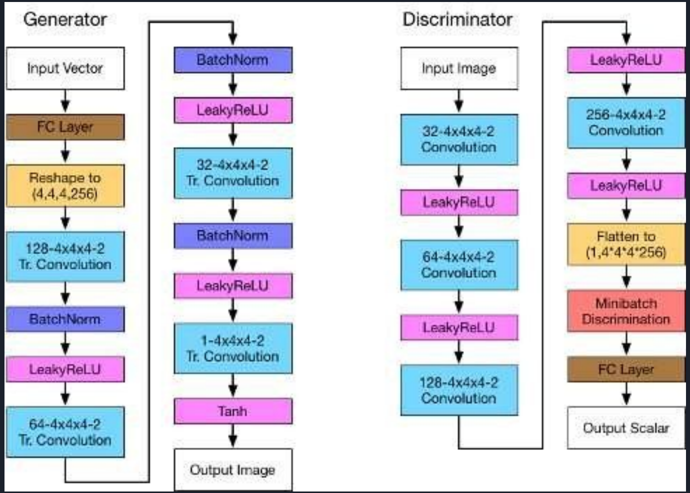


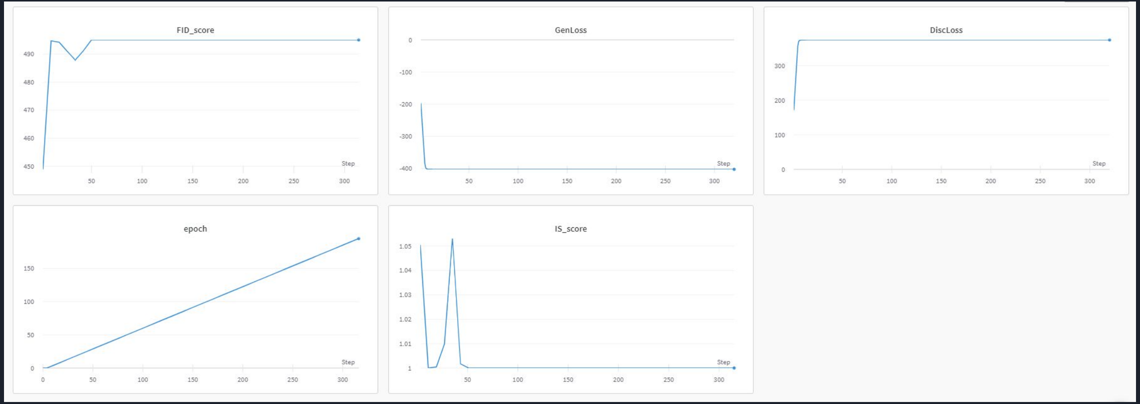


WGAN

- Wasserstein GAN minimizes Earth Movers distance rather than Jensen-Shannon divergence
- Code was inspired from the technical paper
- https://arxiv.org/abs/1701.07875v3
- WGAN Used a different loss and architecture from DCGAN
- WGAN Failed with FID Score of 496 and IS Score of 1
- Images produced were all black and this architecture was not suitable at all

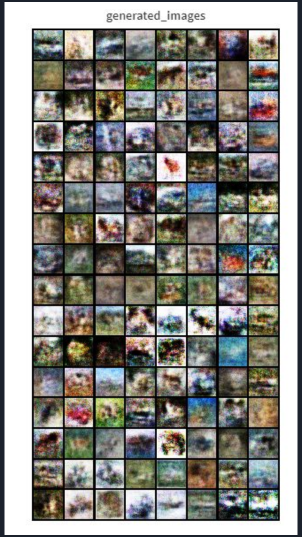


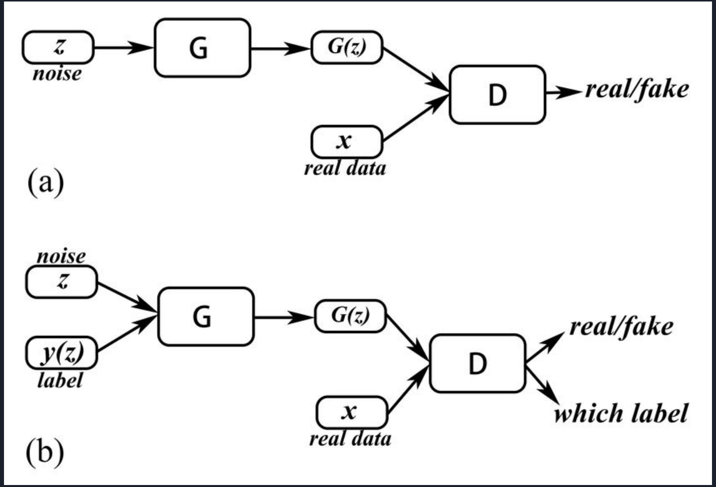


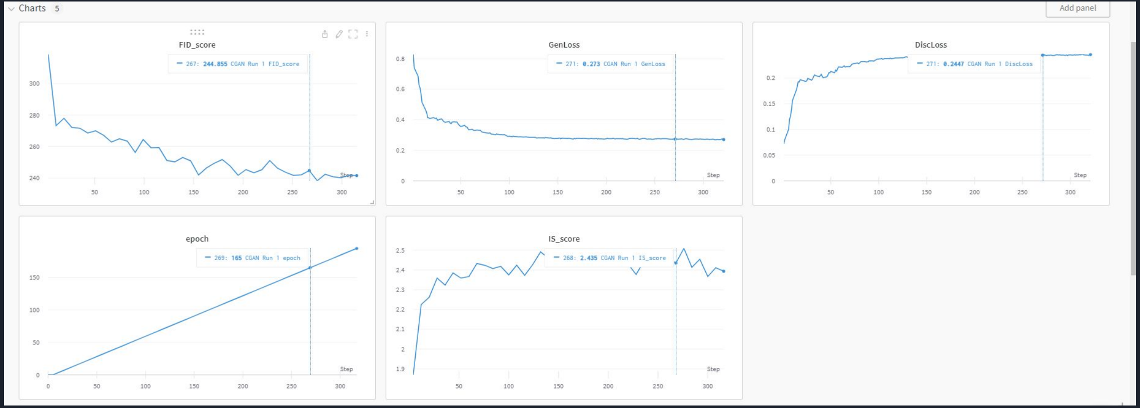


CGAN

- CGAN stands for conditional GAN, CGAN was my first conditional GAN which takes advantage of labels in training process, training framework had to be changed for this to work.
- Code was inspired from the technical paper
- https://arxiv.org/abs/1411.1784
- CGAN used a different architecture and of course required labels
- CGAN showed good promise with FID: 241 and IS: 2.4
- Images produced were not of high quality but CGAN proved to be a possible path

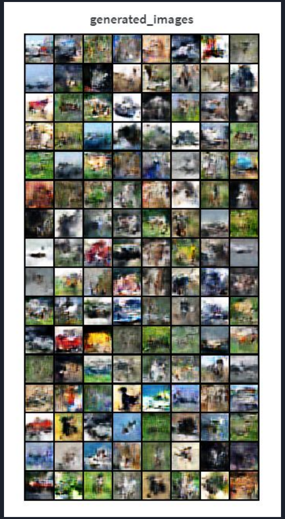


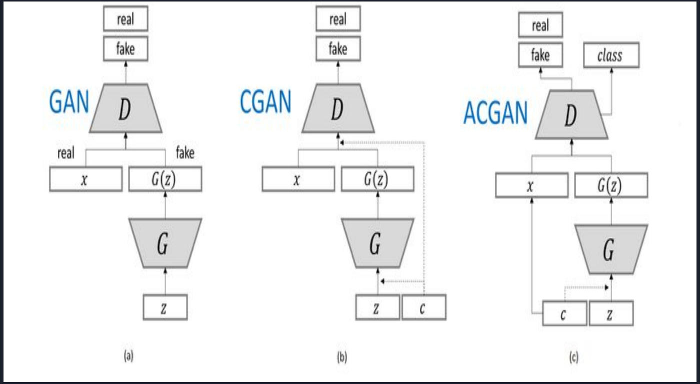


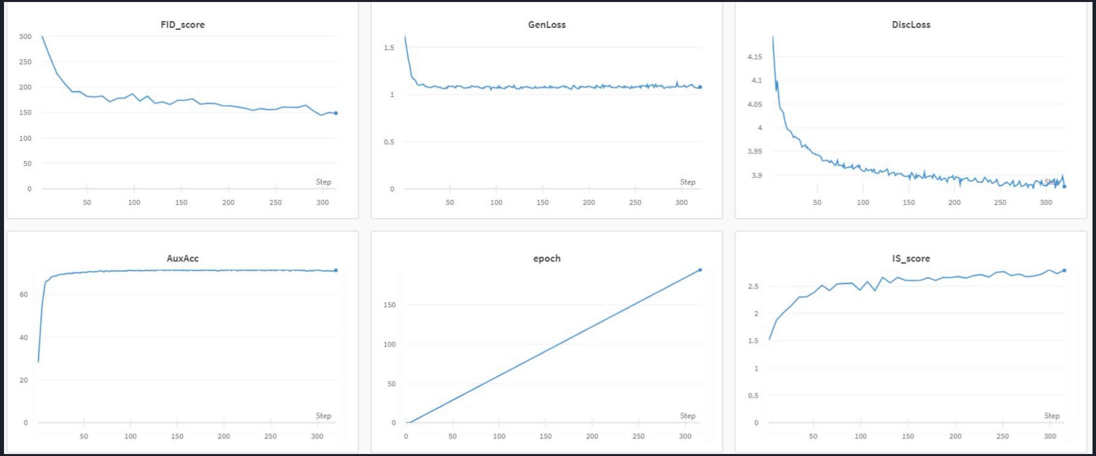


ACGAN

- ACGAN stands for auxiliary classifier from CGAN with softmax cross entropy loss.
- Code was inspired from the technical paper
- https://arxiv.org/abs/2111.01118
- AGAN used a different architecture and still required labels
- ACGAN FID Score of 148 and IS Score of 2.8
- Images produced were better than CGAN as expected and used Aux Acc



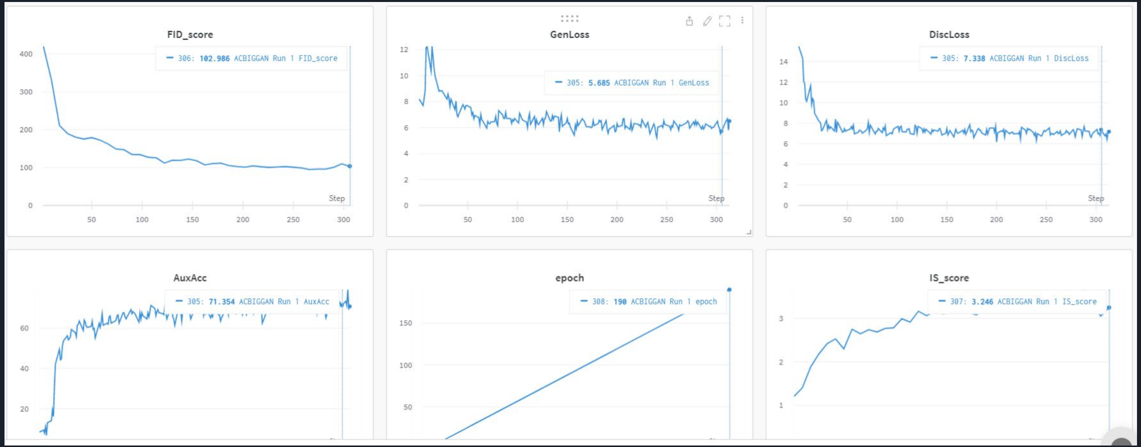




ACGAN Improved with BIGGAN

- ACGAN was improved with some inspired architecture from BIGGAN
- Used Calculate gradient penalty and BIGGAN discriminator
- ACBIGGAN has FID Score of 102 and IS Score of 3.2
- Images produced were also of better quality proving that BIGGAN architecture did help

generated_images



BIGGAN

- BIGGAN is the most complex architecture used in this experiment
- Code inspired from
- https://arxiv.org/abs/1809.11096
- Architecture was more complex and used BIGGANConditionalBatchNorm2d and a lot of gen and disc blocks
- BIGGAN has FID Score of 46 and IS Score of 6.2
- Images produced were of better quality proving that BIGGAN architecture is the best

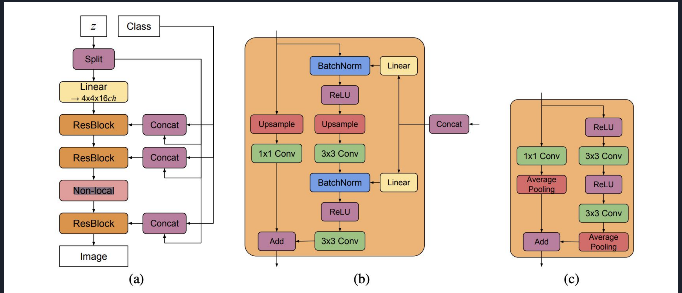
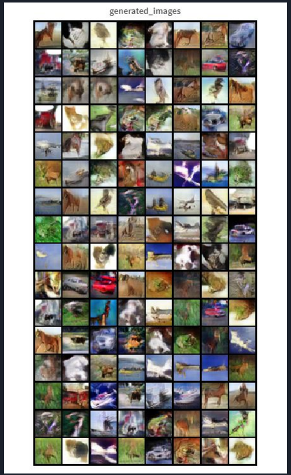
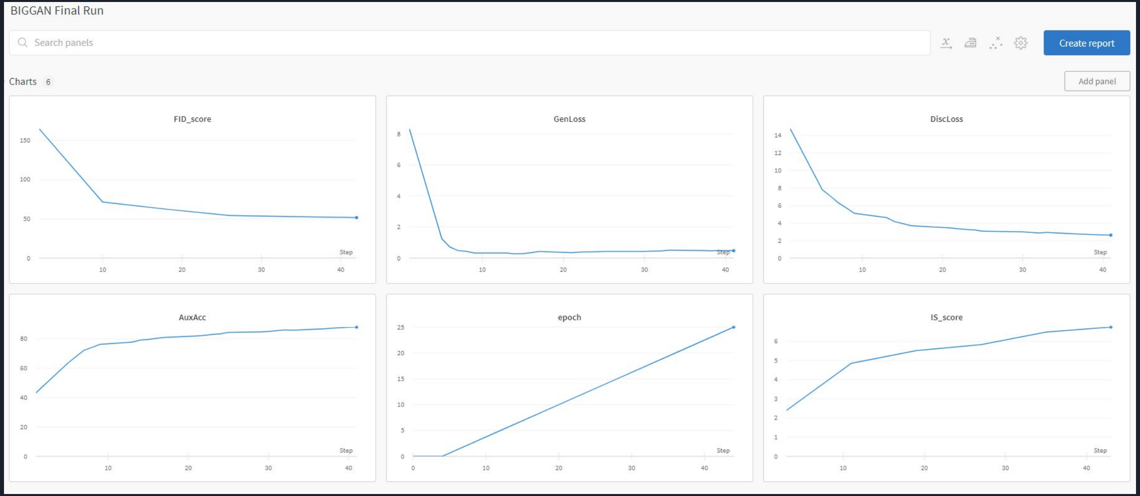


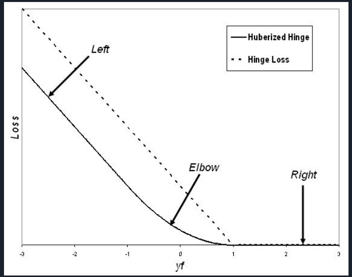
Figure 15: (a) A typical architectural layout for BigGAN's G; details are in the following tables. (b) A Residual Block (*ResBlock up*) in BigGAN's G. (c) A Residual Block (*ResBlock down*) in BigGAN's D.

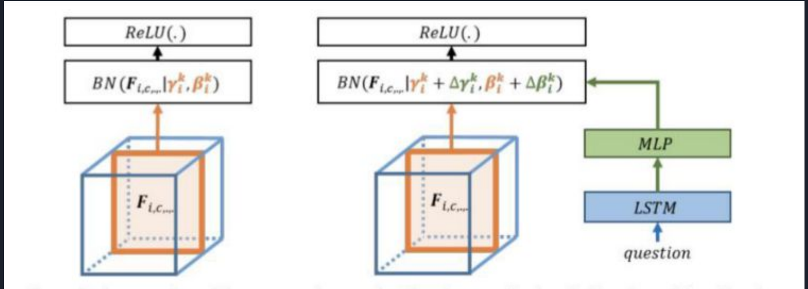



BIGGAN Final Run

Fine-Tuning

- Shared Embedding since it is not as efficient to have different embedding layer transform categorical labels into features which stores class information
- Conditional Batch-Norm was used in BIGGAN as bigganconditionalbatchnorm2d
- Hinge Loss was introduced from `https://github.com/POSTECH-CVLab/PyTorch-StudioGAN/blob/8c9aa5a2e9bb33eca711327c085db5ce50ff7fc0/src/utils/losses.py`
- Spectral Normalization was used to stabilize training of discriminators by rescaling weight tensor quoted from: https://arxiv.org/abs/1802.05957
- Learning Rate scheduler StepLR from pytorch
- Label smoothing is introduced as well for classification, CGAN problems to increase generalization and learning speed by making the model not overconfident from: https://arxiv.org/abs/1906.02629




ReLU(.)
$BN(F_{i,c,..}|\gamma_i^k,\beta_i^k)$
$F_{i,c,..}$
ReLU(.)
$BN(F_{i,c,..}|\gamma_i^k+\Delta\gamma_i^k,\beta_i^k+\Delta\beta_i^k)$
$F_{i,c,..}$
MLP
LSTM
question
Figure 2: An overview of the computation graph of batch normalization (left) and conditional batch normalization (right). Best viewed in color.

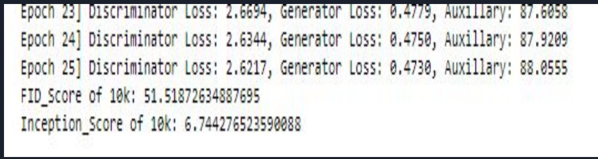
Evaluation

Comparing all the architectures, BIGGAN achieved the best FID and IS Score but requires the longest training times due to the complex network, we will still be going with BIGGAN for the final model.

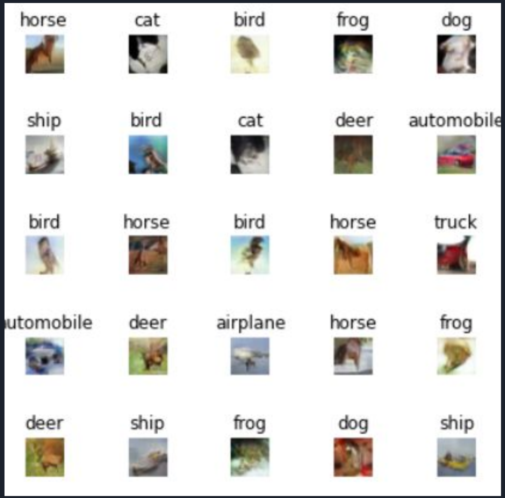
Evaluating these models we use 2 main metrics: FID Score and IS Score

1. Inception Score (IS) - Inception score is a metric for automatically evaluating the quality of image generative models (Salimans et al., 2016). Metric was shown to correlate well with hu- man scoring of the realism of generated images from the CIFAR-10 dataset
2. Frechet Inception Distance (FID) - FID Score is a metric for evaluating the quality of images as well, lower scores correlating to higher quality images while vice-versa for IS Score.

Overall the model managed to generate images that has a decent amount of quality and represents its label’s correctly, example cars can be seen clearly from shape and color. However some mode-colapsing happened where cars generated seem to be the same model.



```
Epoch 23] Discriminator Loss: 2.6694, Generator Loss: 0.4779, Auxillary: 87.6058
Epoch 24] Discriminator Loss: 2.6344, Generator Loss: 0.4750, Auxillary: 87.9209
Epoch 25] Discriminator Loss: 2.6217, Generator Loss: 0.4730, Auxillary: 88.0555
FID_Score of 10k: 51.51872634887695
Inception_Score of 10k: 6.744276523590088
```



Conclusion

In Conclusion, I experimented with 5 architectures of DCGAN, WGAN, CGAN, ACGAN and BIGGAN to find that the architecture which performed the best is BIGGAN managing to achieve the FID-Score of 51 and IS-Score of 6.7. The generator loss dropped from 8.3 to 0.47 and the discriminator loss dropped from 14.7 to 2.6, which is good as generator loss should be lower than discriminator loss. The Adversarial Accuracy increased from 43 to 88. With a total of 11 runs comparing all architectures recorded on WandB and a total training time of 49 hours ran on the Colab GPU not including non-listed experimental runs. I have learnt alot about GAN and experimenting with simple architectures such as DCGAN to complex ones such as BIGGAN and all this is done on PyTorch again where I have learnt much from my CA1 to be able to convert my self-taught Pytorch skills here.