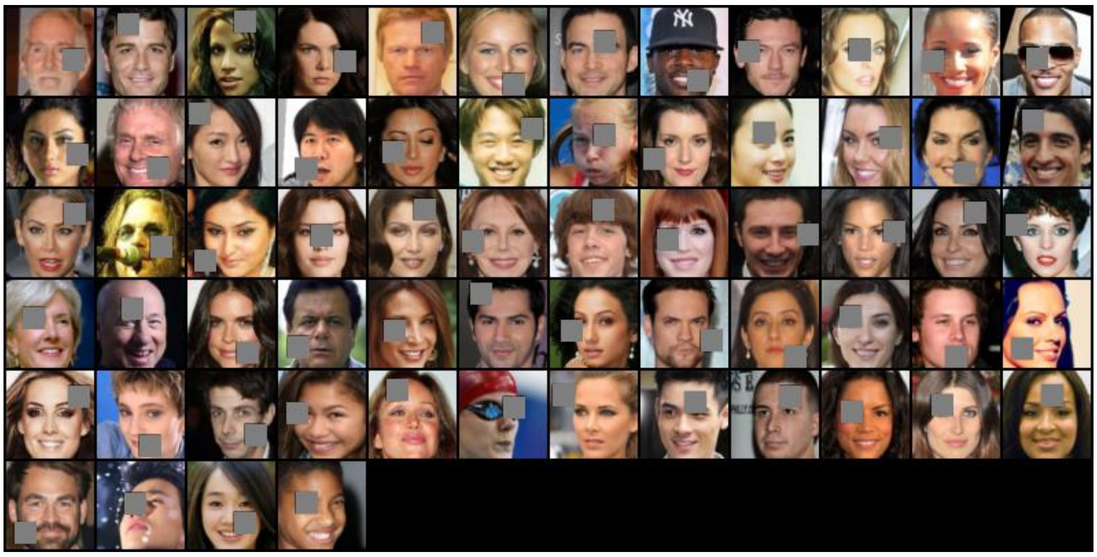
GENERATIVE INPAINTING

Training the DCGAN

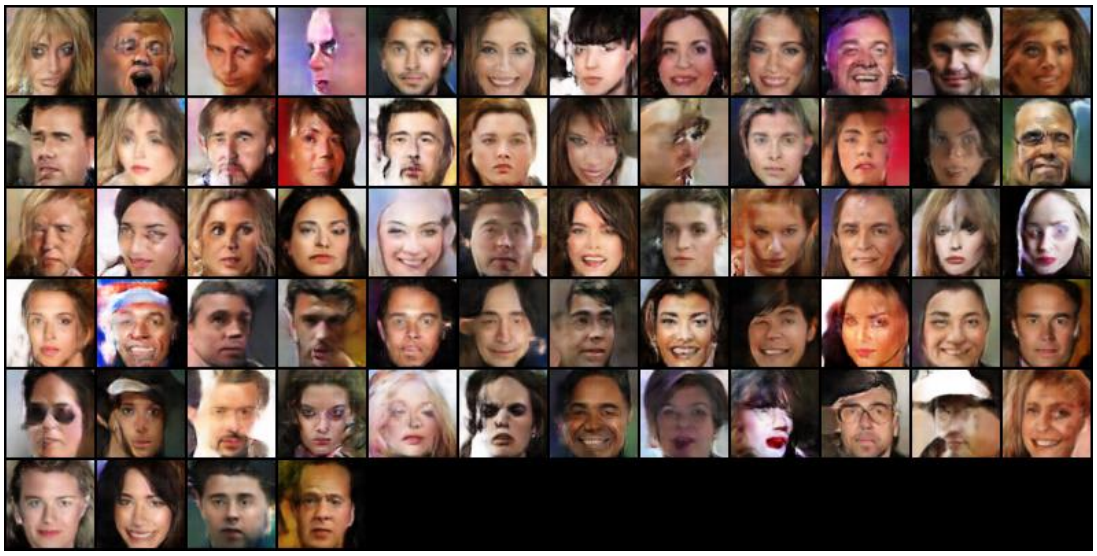
In this Assignment we worked extensively on GAN and on finding the Context Loss, Prior Loss and how to train the GAN network with a dataset.
Then we introduced a random selfie image to the network and see if the image is reconstructed or not.

Figure 1,2,3 showing a batch of datasets with one corrupted, one completed and one in-painted.

Corrupted Result

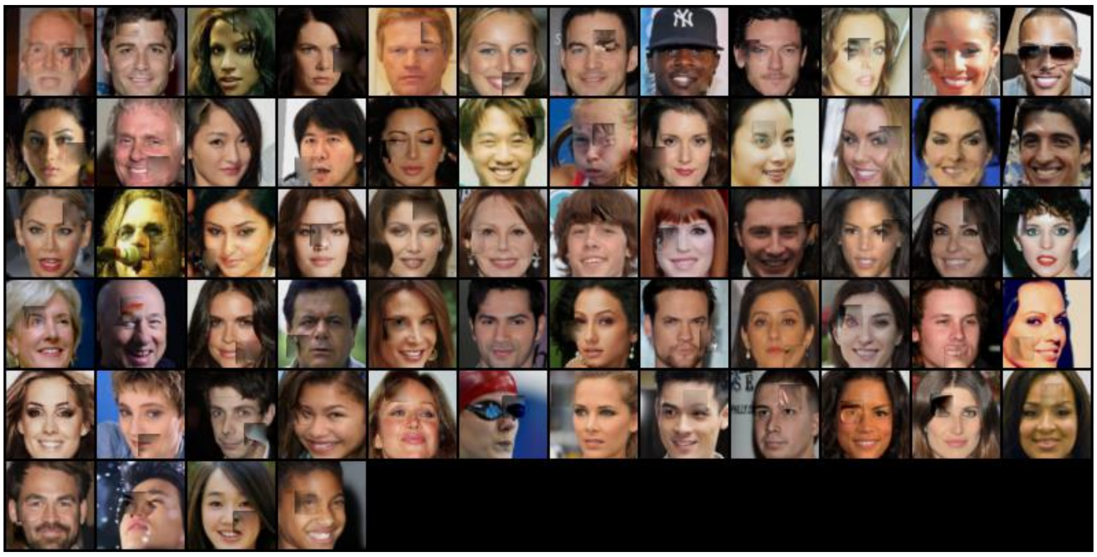


Completed Result



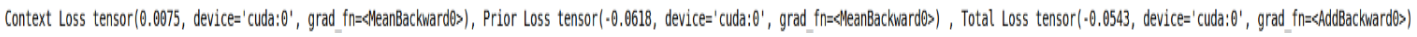
Blended Result

Although the blending was not perfect for few of the image, however I have attempted to perform few of the experiments to see how the blending can be improved.



Experiment 1) May be the images quality is detoriated because the image been reduced to a size of 64*64.?.
The answer was no. I attempted to send the original image, still the results were same.
Changing the quality of the image (loss due to the Image compression) doesn't improve the result anyway.

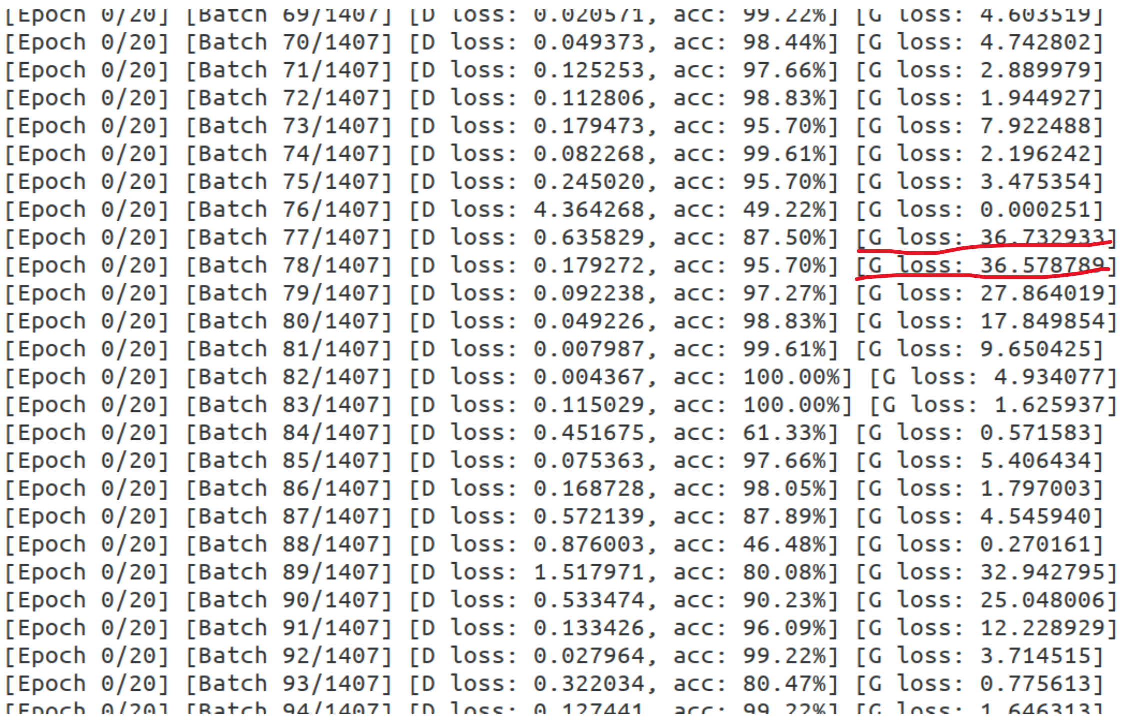
Blended Image Corrupted Image



```
Context Loss tensor(0.0075, device='cuda:0', grad_fn=<MeanBackward0>), Prior Loss tensor(-0.0618, device='cuda:0', grad_fn=<MeanBackward0>) , Total Loss tensor(-0.0543, device='cuda:0', grad_fn=<AddBackward0>)
```

Experiment 2) Removing the batch normalization detoriated the output results.

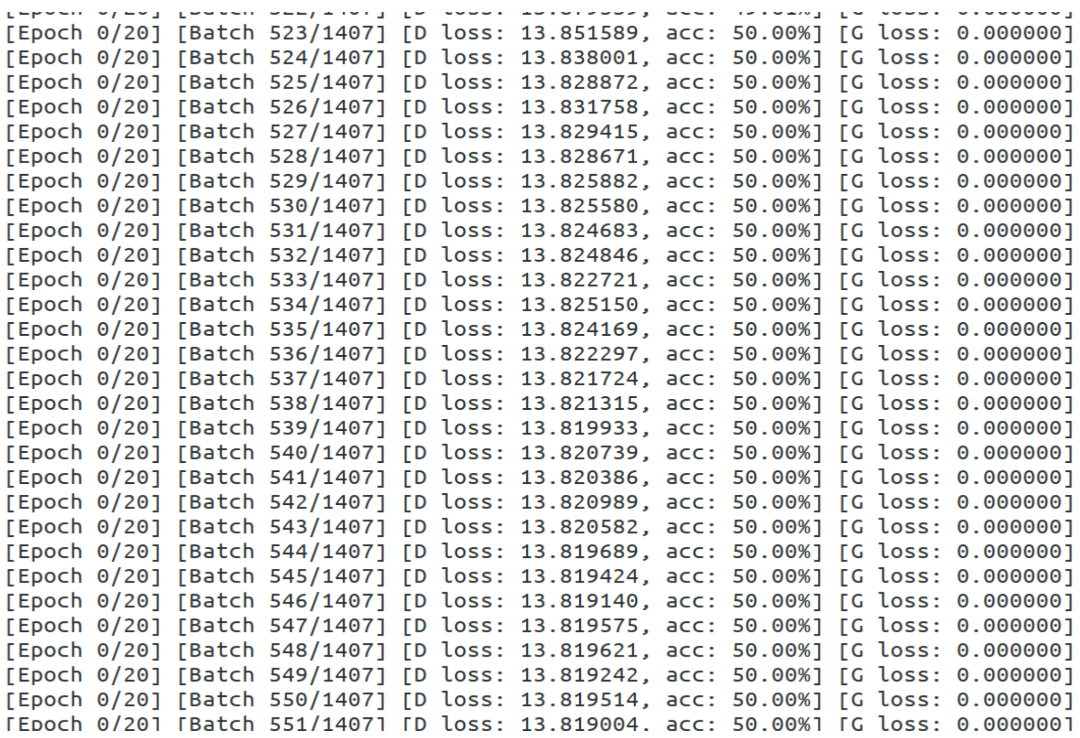
The aim of this experiment is to find out how batch normalisations stabilises the training of the GAN.



```
[Epoch 0/20] [Batch 69/1407] [D loss: 0.020571, acc: 99.22%] [G loss: 4.603519]
[Epoch 0/20] [Batch 70/1407] [D loss: 0.049373, acc: 98.44%] [G loss: 4.742802]
[Epoch 0/20] [Batch 71/1407] [D loss: 0.125253, acc: 97.66%] [G loss: 2.889979]
[Epoch 0/20] [Batch 72/1407] [D loss: 0.112806, acc: 98.83%] [G loss: 1.944927]
[Epoch 0/20] [Batch 73/1407] [D loss: 0.179473, acc: 95.70%] [G loss: 7.922488]
[Epoch 0/20] [Batch 74/1407] [D loss: 0.082268, acc: 99.61%] [G loss: 2.196242]
[Epoch 0/20] [Batch 75/1407] [D loss: 0.245020, acc: 95.70%] [G loss: 3.475354]
[Epoch 0/20] [Batch 76/1407] [D loss: 4.364268, acc: 49.22%] [G loss: 0.000251]
[Epoch 0/20] [Batch 77/1407] [D loss: 0.635829, acc: 87.50%] [G loss: 36.732933]
[Epoch 0/20] [Batch 78/1407] [D loss: 0.179272, acc: 95.70%] [G loss: 36.578789]
[Epoch 0/20] [Batch 79/1407] [D loss: 0.092238, acc: 97.27%] [G loss: 27.864019]
[Epoch 0/20] [Batch 80/1407] [D loss: 0.049226, acc: 98.83%] [G loss: 17.849854]
[Epoch 0/20] [Batch 81/1407] [D loss: 0.007987, acc: 99.61%] [G loss: 9.650425]
[Epoch 0/20] [Batch 82/1407] [D loss: 0.004367, acc: 100.00%] [G loss: 4.934077]
[Epoch 0/20] [Batch 83/1407] [D loss: 0.115029, acc: 100.00%] [G loss: 1.625937]
[Epoch 0/20] [Batch 84/1407] [D loss: 0.451675, acc: 61.33%] [G loss: 0.571583]
[Epoch 0/20] [Batch 85/1407] [D loss: 0.075363, acc: 97.66%] [G loss: 5.406434]
[Epoch 0/20] [Batch 86/1407] [D loss: 0.168728, acc: 98.05%] [G loss: 1.797003]
[Epoch 0/20] [Batch 87/1407] [D loss: 0.572139, acc: 87.89%] [G loss: 4.545940]
[Epoch 0/20] [Batch 88/1407] [D loss: 0.876003, acc: 46.48%] [G loss: 0.270161]
[Epoch 0/20] [Batch 89/1407] [D loss: 1.517971, acc: 80.08%] [G loss: 32.942795]
[Epoch 0/20] [Batch 90/1407] [D loss: 0.533474, acc: 90.23%] [G loss: 25.048006]
[Epoch 0/20] [Batch 91/1407] [D loss: 0.133426, acc: 96.09%] [G loss: 12.228929]
[Epoch 0/20] [Batch 92/1407] [D loss: 0.027964, acc: 99.22%] [G loss: 3.714515]
[Epoch 0/20] [Batch 93/1407] [D loss: 0.322034, acc: 80.47%] [G loss: 0.775613]
[Epoch 0/20] [Batch 94/1407] [D loss: 0.127441, acc: 99.22%] [G loss: 1.646313]
```

As we could see that the generated loss drops because there was no Batch Normalization been performed. **This clearly indicates that a batch normalization is necessary for a DCGAN.**

As the training approaches we could see that the Generated Loss drops down to be 'o' eventually.



```
[Epoch 0/20] [Batch 523/1407] [D loss: 13.851589, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 524/1407] [D loss: 13.838001, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 525/1407] [D loss: 13.828872, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 526/1407] [D loss: 13.831758, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 527/1407] [D loss: 13.829415, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 528/1407] [D loss: 13.828671, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 529/1407] [D loss: 13.825882, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 530/1407] [D loss: 13.825580, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 531/1407] [D loss: 13.824683, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 532/1407] [D loss: 13.824846, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 533/1407] [D loss: 13.822721, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 534/1407] [D loss: 13.825150, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 535/1407] [D loss: 13.824169, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 536/1407] [D loss: 13.822297, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 537/1407] [D loss: 13.821724, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 538/1407] [D loss: 13.821315, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 539/1407] [D loss: 13.819933, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 540/1407] [D loss: 13.820739, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 541/1407] [D loss: 13.820386, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 542/1407] [D loss: 13.820989, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 543/1407] [D loss: 13.820582, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 544/1407] [D loss: 13.819689, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 545/1407] [D loss: 13.819424, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 546/1407] [D loss: 13.819140, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 547/1407] [D loss: 13.819575, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 548/1407] [D loss: 13.819621, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 549/1407] [D loss: 13.819242, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 550/1407] [D loss: 13.819514, acc: 50.00%] [G loss: 0.000000]
[Epoch 0/20] [Batch 551/1407] [D loss: 13.819004, acc: 50.00%] [G loss: 0.000000]
```

Experiment 3) May be a change of the loss function do the trick?

This is a subject that needs to be explored more with more amount of time and can find out which loss results better. This would ideally be my future work.

For the selfie I created a corrupted image and then created_mask .npy file that contains the mask.
I import both the image and the mask file under inpaint.py file and then perform the inpainting operation.

The SELFIE Experiments

Experiment 1) Results with batch size 1

This experimentation was based on the number of batches that the selfie images are obtained and see for the results as such.

The results are as below

Single batch of images in-painted

Blended	Corrupted	Comments
		Increased the mask size and tried inpainting, where the blending was imperfect.
		Changing the mask size gave me better results as we could tha the blending is fine.
		Blended correctly.

Experiment 2) Multiple batch size of 4 images

Blended	Corrupted	Comments
		Multiple batches of inpainting provides the same result of inpainting.
		The quality of the inpainting is better and needs much improvement.

Implementation: -

- ✓ The corrupted image has been created and fed to the best trained model. We could partially recover the corrupted image via inpainting.
- ✓ The GAN model tried to find the best fitted inpainting for the corrupted image.
- ✓ Few of the batched images above shows that the blending goes smooth to the corrupted area, but still lacks a complete convergence. GAN may not be a better approach for inpainting.
- ✓ The quality of the recovered image overall looks fine for a 64*64 patched image.

Advantages:-

The usage of latent vectors is easier in GAN. We randomly create a latent vector and introduce it to the GAN for training.

Disadvantages :-

GAN's are very hard to train. We could see the training for the dataset provided a lot of time to train.

There is always a unstable state between the generator and the discriminator.

Future Work:-

- GAN's are very useful for semi-supervised training. But only because of the training, we find it harder enough to use GAN's. Identifying the way on how the training of the generator and the discriminator can be made simple to be identified.

- Since the discriminator has no purpose after the training, we could find ways to identify and make the discriminator more useful.

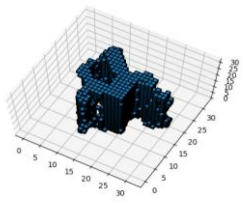
- As mentioned above, identifying a better loss will be a future work.

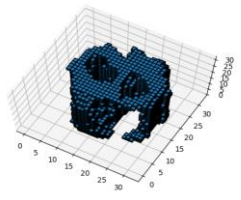
3DGAN – TASKS AND IMPLEMENTATION

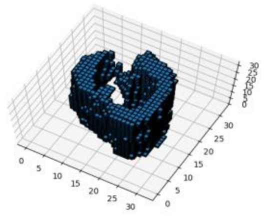
In this assignment we learnt the voxelization of images and synthesizing the images for a better 3D view.

Experiment 1) **Learning Rate did the trick - Changing the learning rate improved the 3DGAN results** learning rate – 0.00004 gave better results than 0.00002

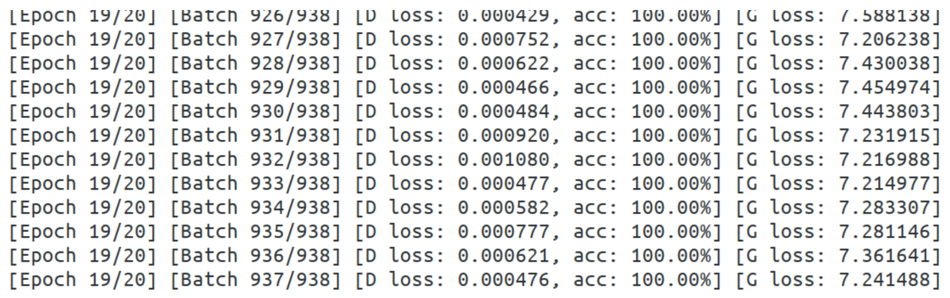
Results for Learning Rate 0.00002 shows the results are not good.







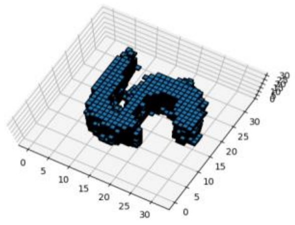
Training Loss:

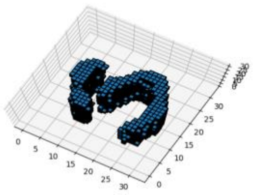


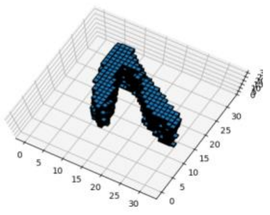
```
[Epoch 19/20] [Batch 926/938] [D loss: 0.000429, acc: 100.00%] [G loss: 7.588138]
[Epoch 19/20] [Batch 927/938] [D loss: 0.000752, acc: 100.00%] [G loss: 7.206238]
[Epoch 19/20] [Batch 928/938] [D loss: 0.000622, acc: 100.00%] [G loss: 7.430038]
[Epoch 19/20] [Batch 929/938] [D loss: 0.000466, acc: 100.00%] [G loss: 7.454974]
[Epoch 19/20] [Batch 930/938] [D loss: 0.000484, acc: 100.00%] [G loss: 7.443803]
[Epoch 19/20] [Batch 931/938] [D loss: 0.000920, acc: 100.00%] [G loss: 7.231915]
[Epoch 19/20] [Batch 932/938] [D loss: 0.001080, acc: 100.00%] [G loss: 7.216988]
[Epoch 19/20] [Batch 933/938] [D loss: 0.000477, acc: 100.00%] [G loss: 7.214977]
[Epoch 19/20] [Batch 934/938] [D loss: 0.000582, acc: 100.00%] [G loss: 7.283307]
[Epoch 19/20] [Batch 935/938] [D loss: 0.000777, acc: 100.00%] [G loss: 7.281146]
[Epoch 19/20] [Batch 936/938] [D loss: 0.000621, acc: 100.00%] [G loss: 7.361641]
[Epoch 19/20] [Batch 937/938] [D loss: 0.000476, acc: 100.00%] [G loss: 7.241488]
```

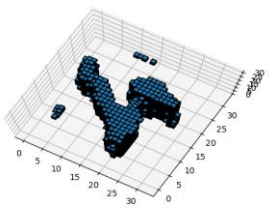
Experiment 2) Plotted only the points with the threshold value > 0.5 in the latent vector

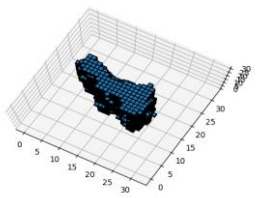
Results for Learning rate 0.00004

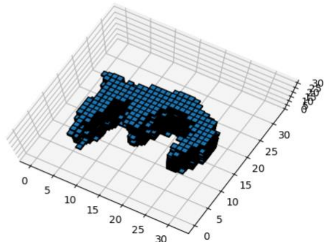


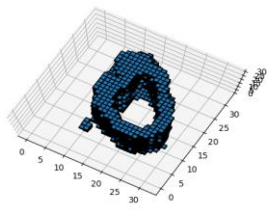
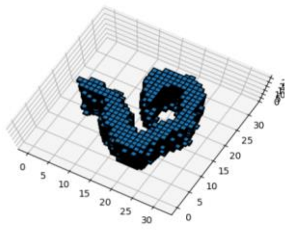
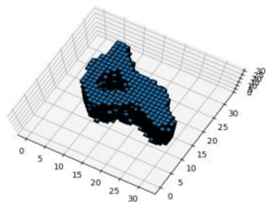




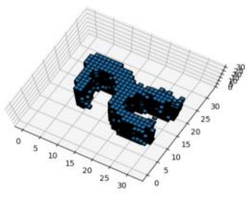
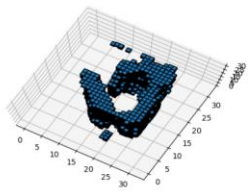




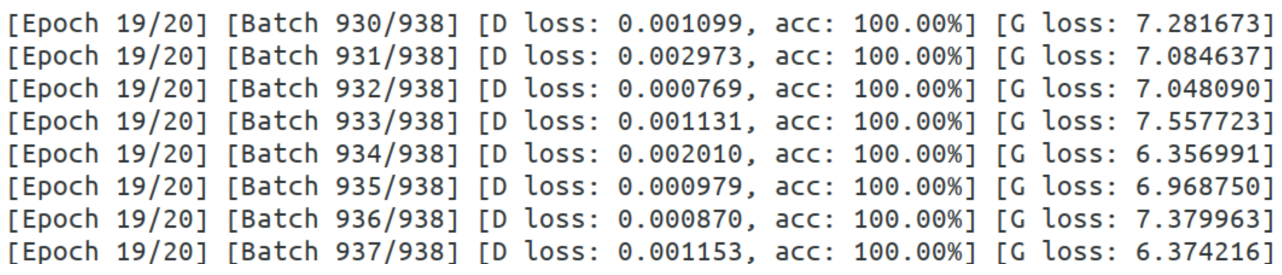


Tough Decisions (Eg’s digit 2, 8, digit 6)

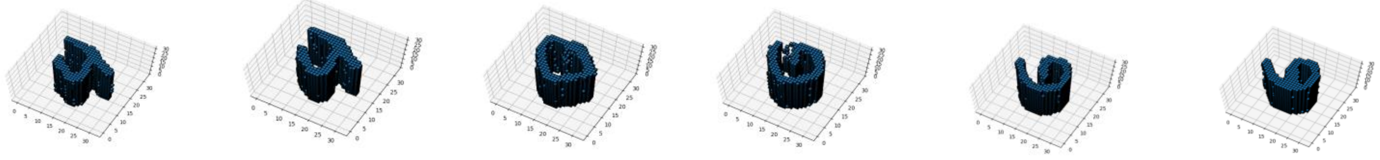
Training Loss



```
[Epoch 19/20] [Batch 930/938] [D loss: 0.001099, acc: 100.00%] [G loss: 7.281673]
[Epoch 19/20] [Batch 931/938] [D loss: 0.002973, acc: 100.00%] [G loss: 7.084637]
[Epoch 19/20] [Batch 932/938] [D loss: 0.000769, acc: 100.00%] [G loss: 7.048090]
[Epoch 19/20] [Batch 933/938] [D loss: 0.001131, acc: 100.00%] [G loss: 7.557723]
[Epoch 19/20] [Batch 934/938] [D loss: 0.002010, acc: 100.00%] [G loss: 6.356991]
[Epoch 19/20] [Batch 935/938] [D loss: 0.000979, acc: 100.00%] [G loss: 6.968750]
[Epoch 19/20] [Batch 936/938] [D loss: 0.000870, acc: 100.00%] [G loss: 7.379963]
[Epoch 19/20] [Batch 937/938] [D loss: 0.001153, acc: 100.00%] [G loss: 6.374216]
```

Interpolation Results

The Interpolation is shown from the digit 4 to 6



Advantages

- The Voxelised GAN captures very realistic images of any object in 3D such as numbers, cars, chairs, etc enabling the user to view it in 3D. This would particularly be very useful in the current world, because everywhere everything is turned into 3 dimensional space nowadays which would be the future.

- The trained generator can identify any 3D voxels and it's easy to interpolate from any latent space to the others. This gives the independence of viewing any object in the latent space.

Disadvantages

- **T**raining a GAN was hard. To train a higher dataset, we may require much more pulling capacity of GPU's to train it in 3D.

- Viewing the 3 dimensional or more would be a problem.

- Training the network would be very time consuming.

- It was very had enough to view the objects in 3D. For example, in our assignment the MNIST datasets with anything more than 32*32*32 dimension was be a problem. So, the dimensional constraint is still a concern.

Future Works

- Even though the Generator loss decreases over the training, still the amount of loss is not acceptable. So, we can identify how to decrease the Generator loss and achieve a minimal loss all the time.

- The 3D GAN can be implemented to various objects and it will be very useful for any industry who wants to synthesize the object. So, learning to synthesize any objects in a 3D would be great.