Problem 2

a.

We used a DCGAN implementation from PyTorch. They used the celeba dataset to train their DCGAN and it was trained with these following hyperparameters:

	Params for DCGAN Paper
Resolution	64x64x3
Latent Space Dim	100
Epochs	5
Learning Rate	0.0002
Beta1 (Adam)	0.5

The paper for DCGAN suggested the hyperparameters of learning rate = 0.0002 and $beta_1 = 0.5$ so the only hyperparameter that was changed from the source code was the number of epochs. This depended on how many datapoints (images) were in the dataset. The celeba dataset has about 200k images so 5 epochs were sufficient for them. Our dataset had TODO and thus the following hyperparameters:

	Params for our DCGAN
Resolution	64x64x3
Latent Space Dim	100
Epochs	TODO
Learning Rate	0.0002
Beta1 (Adam)	0.5

Issue 1 - Resolution:

I tried changing the dimensions of the data to a higher resolution but I couldn't resolve some error that popped up relating to the dimensions of some vector and I wasn't able to fix it. So I decided to keep the resolution at 64x64.

Issue 2 - Seeding:

Heres the order of operations:

$$\text{Seeding} \rightarrow \text{Training dataset augmentation and creation} \rightarrow \text{Fixed noise} \rightarrow \dots$$

This fixed noise was used to keep track of how the legos turned out over time and since the dataset batching and augmentation happened before the fixed noise, when trying to load the trained model, the fixed noise would look different without the training dataset. The easy fix would be to change the order at which it happens:

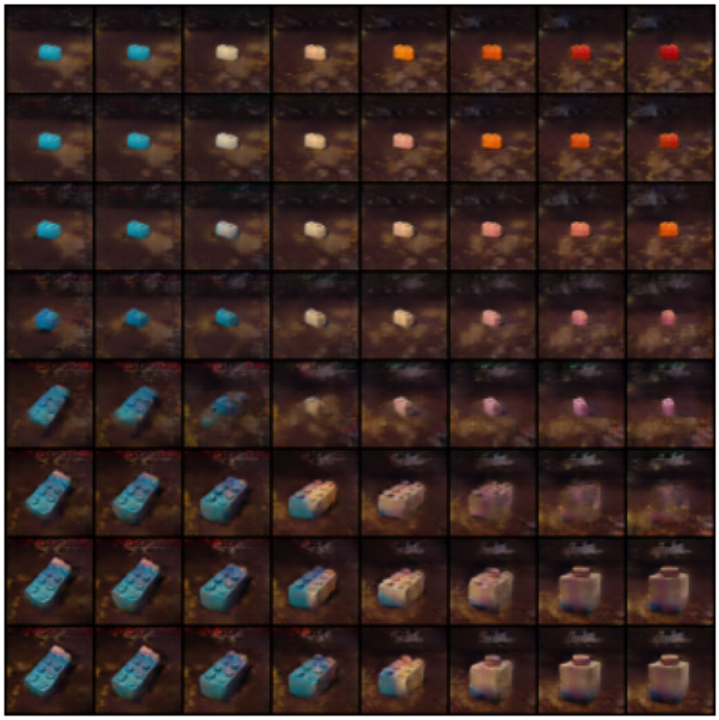
$$\text{Seeding} \rightarrow \text{Fixed noise} \rightarrow \text{Training dataset augmentation and creation} \rightarrow \dots$$

b.

TODO

c.

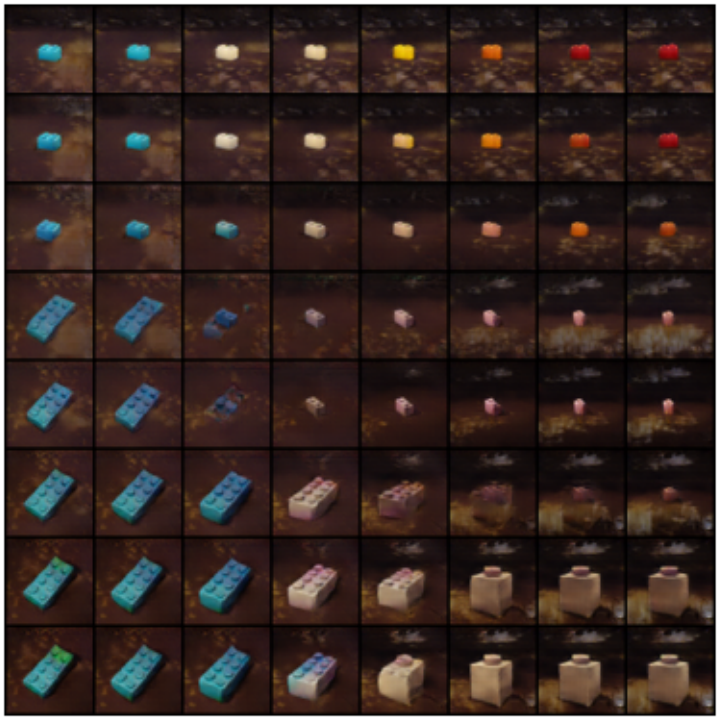
TODO



Issue 3 - Lego Inconsistency:

Interpolating between these 4 legos:

The interpolation gave us legos that weren't the same as the input ones (see the bottom right is green tipped instead of pink tipped):



The network takes a B (batch) sized of vector of latents and converts it to batch of images

$$(B \times d_z \times 1 \times 1) \rightarrow_G (B \times 3 \times 64 \times 64)$$

The fix was to individually send each image through the network and combine the images afterwards:

$$(1 \times d_z \times 1 \times 1) \rightarrow_G (1 \times 3 \times 64 \times 64)$$