Problem 2: GANs

a. Architecture and Hyperparamters

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
Batch Size	128

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 also had 200k images but with augmentation it doubled to 400k.

Issue 1 - Epochs was too low

We thought that 3 epochs would be enough from the ratio of images to epochs. However, it was not enough to train the model as seen in the follow image:

We assume that this is due to the variation between the legos is higher than the faces in celeba.

Issue 2 - Augmentation

Realized that the diffusion model has its own Dataloader inside the pip package and thus we were not able to apply the augmentations directly to the tensors when they were loaded in. Thus we decided to not to use the augmentations due to not enough time to rewrite the entire package in the code.
Due to the two issues above, we have 200k images and decided to raise the number of epochs to 40 and thus our hyperparameters are:

	Params for our DCGAN
Resolution	64x64x3
Latent Space Dim	100
Epochs	40
Learning Rate	0.0002
Beta1 (Adam)	0.5
Batch Size	128

Issue 3 - 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 4 - 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:

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

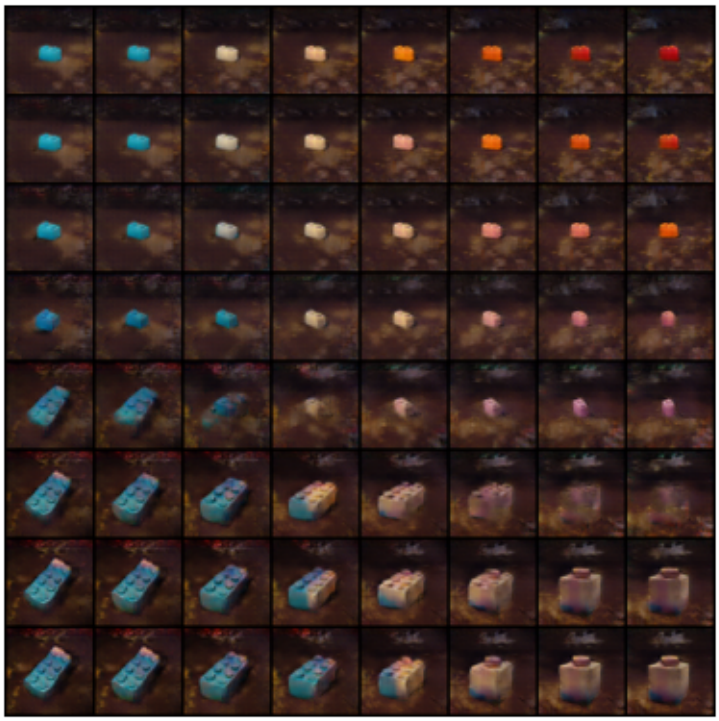
b. Example Images

TODO

We also decided to train a DCGAN on the VAE dataset: TODO

c1. Interpolation

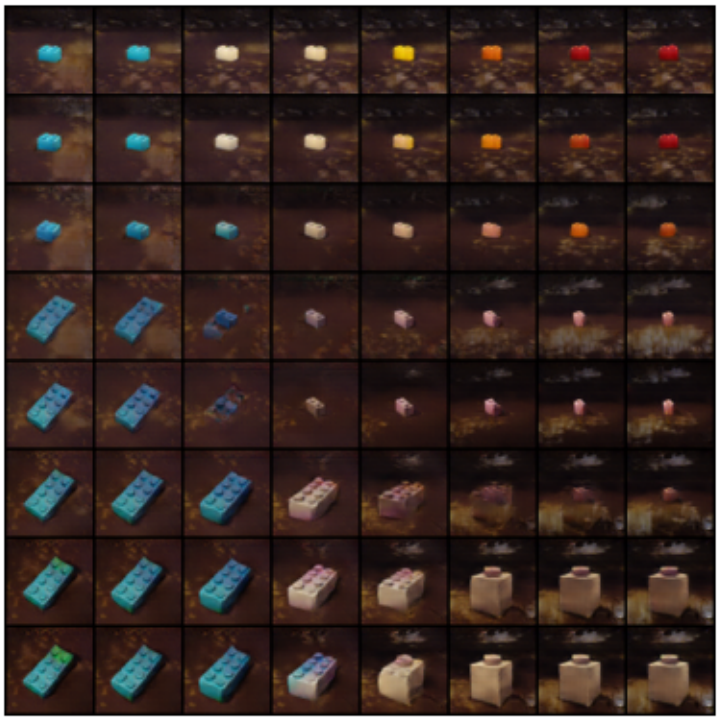
Here is an interpolation between 4 different latent vectors:



Issue 5 - 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:

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

c2. Interpolation Video

Generating many different latent vectors and interpolating between them in a video. See Box or Github for the mp4 file. TODO LINK

c3. Adding Constants to Latents

TODO

Problem 3: Diffusion Models

a. Architecture and Hyperparamters

We used a Denoising Diffusion Probabilistic Model (DDPM). The original hyperparameters for the DDPM repo that we used were:

	Params for repo DDPM
Resolution	128x128x3
Time Steps	1000
Training Steps	700k
Learning Rate	8e-5
Ema Decay	0.995
Gradient Accumulate Every	2

We decided to keep the resolution the same as the GAN to keep comparisons more fair. The only hyperparameter that we changed was the training steps since 700k was going to take way too long and 10k gave good results.

	Params for repo DDPM
Resolution	64x64x3
Time Steps	1000
Training Steps	20k
Learning Rate	8e-5
Ema Decay	0.995
Gradient Accumulate Every	2

Issue 6: Taking to long to train

Started training the DDPM and it was 2k/700k steps and it had taken 5 hours. The reason why cause it was doing FID Evaluations and they would take forever since each image had 250 timesteps and there were 1500 evaluations. I calculated that would it would take approx 60 days to train with evaluations. The fix was to just turn off the evaluations by setting it to False.

b. Example Images

TODO

Issue 7: VRAM Issues

I was running out of VRAM when generating more than 5 images at a time. The fix was to generate one image at a time, bring it to CPU/RAM and clear the cache in the GPU.

c1. Interpolation

Here is an interpolation between two different latent spaces: TODO Compared to the GAN and VAE, the diffusion interpolation happens very suddenly.

c2. Interpolation Video

As with the GAN we created a video of the interpolation between many different latent spaces on Box and Github. TODO LINK and IMages