Jan. 11th, 2021

Class 4: Project

Exploring AI and Neural Nets in Design

Gia Jung
Research Associate, Lab for Design Technologies, Harvard University
Irving Innovation Fellow 2020 - 2021

Claire Djang
Lab for Design Technologies, Harvard University
Currently at Certain Measures

4.1

Training Show & Tell

Sharing and Discussion of Training Results

Quick Scripts:

Manual Seed Generation

Super-Res

4.2

Latent Space

What is Latent Space?

How can we visualize and make sense of high-dimensional space?

4.3

Visualization Workshop

Selecting Specific Latent Vectors

Interpolation Animation

Data	Model	Project
1. Collection	1. Choosing a Model	Latent Space Exploration #1 **Interpolation Animation**
2. Curation	2. Training a Model	Latent Space Exploration #2 **Interpolated Grid**
3. Processing		Latent Space Exploration #3 **Vector Arithmetic**

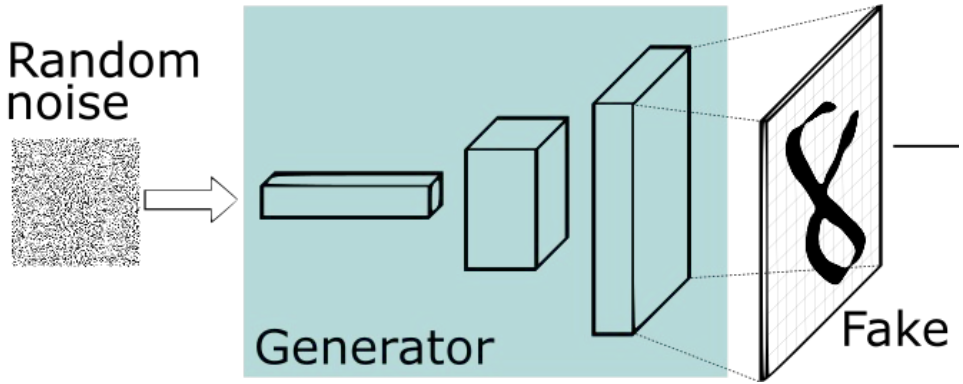
First jump into some scripts:

class 4

RANDOM SEED ITERATION AND GENERATION: *seed_generation.ipynb*
SUPER RESOLUTION: *super_res.ipynb*

What is Latent Space?

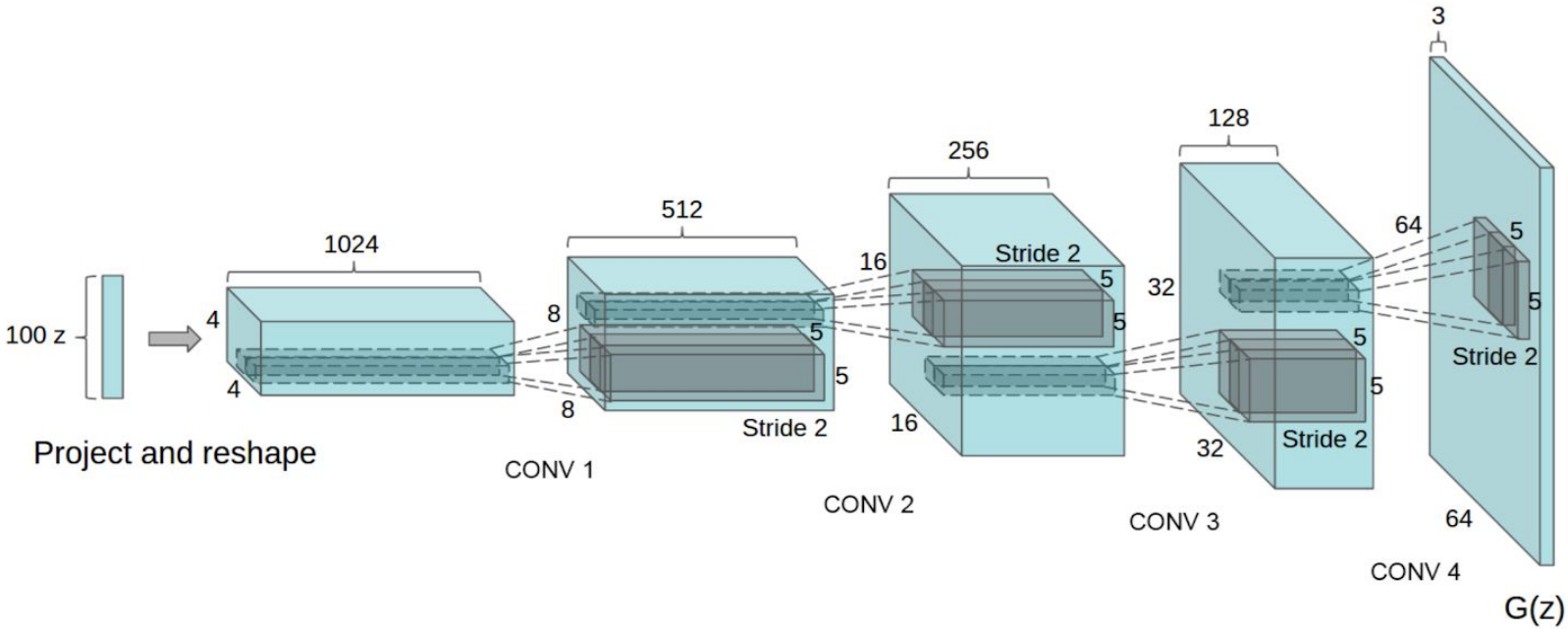
DCGAN Architecture - Generator



DCGAN Architecture (Generator)

From Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

The Generator produces a fake image from the “noise vector”

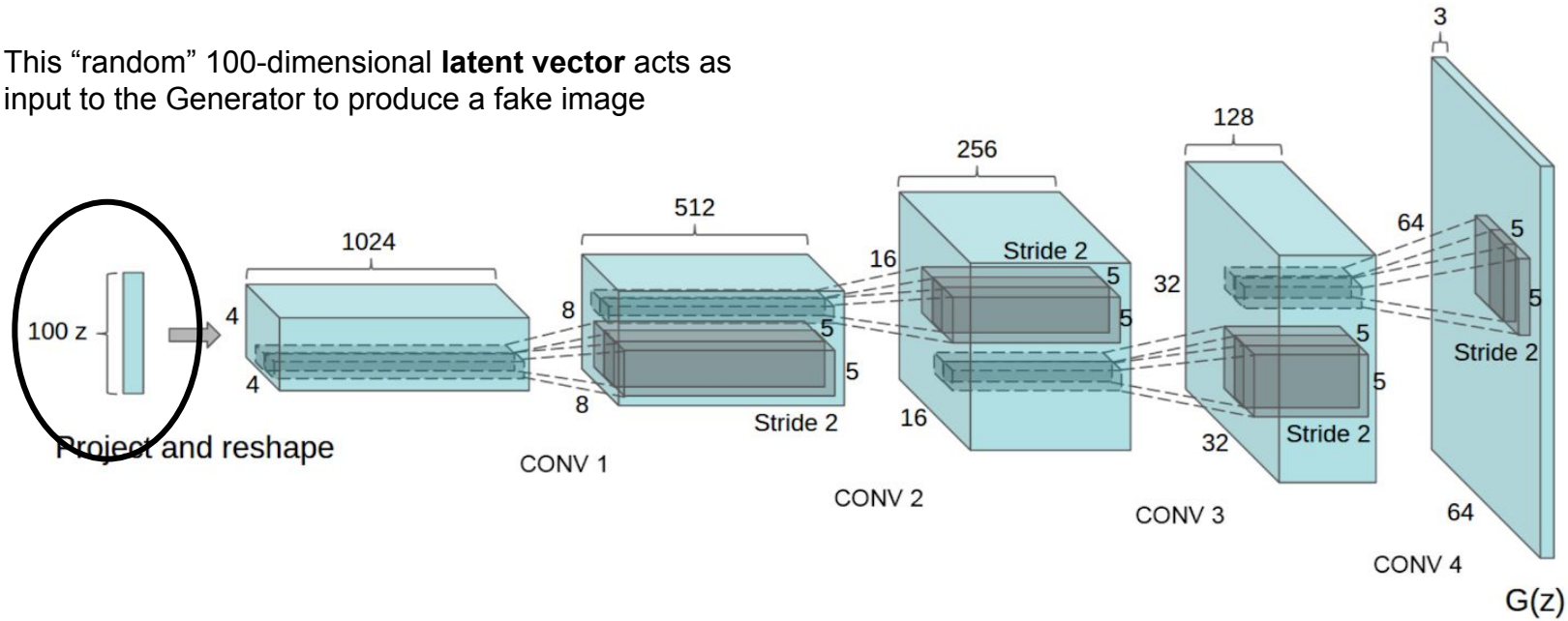


DCGAN Architecture (Generator)

From [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks]

The Generator produces a fake image from the “noise vector”

This “random” 100-dimensional **latent vector** acts as input to the Generator to produce a fake image

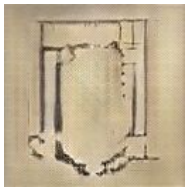


DCGAN Architecture (Generator)

From [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks]

Every unique fake image is mapped from a unique latent vector. We can regenerate the same image using the same vector.

Fake Image A:



Fake Image B:

Fake Image C:

Every unique fake image is mapped from a unique latent vector. We can regenerate the same image using the same vector.

Latent Vector A:

-0.86	0.39	1.67	1.02	-2.22	1.09	-0.99	0.21	0.10	-0.34	-0.28	1.27	2.08	0.90	1.24	0.26	-1.17	-0.71	-1.54	-2.73	0.87	0.79	1.15	1.02	-0.19
1.03	0.57	1.02	-0.42	-0.03	0.79	1.51	0.83	0.05	0.58	-0.72	1.27	0.82	-0.76	-0.09	-2.07	1.13	1.57	0.83	0.06	-1.41	1.45	0.71	0.52	-0.34
-0.41	0.91	0.80	-0.44	0.09	-0.37	-0.99	0.36	-0.66	-0.38	-0.56	0.31	1.22	-1.41	-0.28	0.38	-0.70	-0.88	0.07	0.21	-1.20	0.59	-0.45	-2.68	1.33
0.59	-0.93	0.86	0.79	0.76	0.65	0.93	0.01	1.68	0.28	-2.60	1.73	-1.25	-1.13	1.46	-1.49	-0.43	0.19	-0.37	0.52	-0.63	0.72	-0.38	1.06	0.55

Latent Vector B:

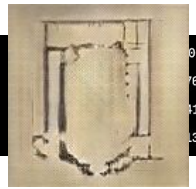
0.78	0.09	-0.06	0.35	0.26	-0.29	1.02	1.51	-0.56	-1.16	0.36	-0.56	-0.91	1.23	0.01	0.84	0.08	0.64	0.60	1.71	1.47	-0.34	0.85	-1.59	0.06
-0.68	0.58	0.72	1.09	0.47	-0.92	0.89	0.15	0.29	0.04	-0.61	-0.22	-0.51	-0.79	-0.31	0.52	-0.23	-1.12	0.30	-0.91	-0.07	-0.46	-0.18	-0.14	0.64
-1.68	0.72	0.52	0.92	-0.25	0.88	0.84	-0.38	-1.77	-0.67	0.68	-0.83	0.55	-0.05	-0.19	-0.48	-0.36	0.18	0.33	-0.16	2.52	-0.43	-0.76	-0.09	-0.91
0.87	0.26	-0.37	0.26	-0.76	1.39	-1.65	0.54	-0.25	-0.09	-1.20	-0.08	-1.60	0.68	-0.61	0.21	-1.15	0.45	0.91	-0.05	-0.15	0.07	-0.05	1.44	-0.10

Latent Vector C:

1.01	0.01	-0.30	-0.90	-0.58	-0.96	0.41	1.28	0.94	0.37	0.77	-0.69	-0.06	1.06	-0.03	-0.26	-1.87	0.90	-0.56	0.33	0.67	0.04	-0.44	-0.03	0.48
0.70	-0.07	-0.19	0.39	0.19	-0.74	0.07	-1.01	1.20	0.77	0.00	-0.32	-0.89	0.08	0.10	0.44	0.32	-0.68	-0.91	-1.09	-0.09	0.35	0.12	-0.92	-0.16
0.90	-2.21	-1.03	-1.08	0.68	1.28	-0.11	-0.27	-0.67	-0.24	0.66	-0.87	-0.70	0.41	1.15	0.52	0.25	0.81	0.05	0.27	0.30	0.70	-0.46	0.66	-1.08
0.15	0.17	0.16	-0.61	-0.01	-0.40	-0.47	0.63	0.81	0.04	-0.47	-0.83	-0.13	-1.04	0.91	-1.64	0.98	-0.23	0.45	0.43	-0.51	0.37	-1.07	-0.45	-0.43

Every unique fake image is mapped from a unique latent vector. We can regenerate the same image using the same vector.

-0.86	0.39	1.67	1.02	-2.22	1.09	-0.99	0.21	0.10	-0.34	-0.28
1.03	0.57	1.02	-0.42	-0.03	0.79	1.51	0.83	0.05	0.58	-0.72
-0.41	0.91	0.80	-0.44	0.09	-0.37	-0.99	0.36	-0.66	-0.38	-0.56
0.59	-0.93	0.86	0.79	0.76	0.65	0.93	0.01	1.68	0.28	-2.60



0	1.24	0.26	-1.17	-0.71	-1.54	-2.73	0.87	0.79	1.15	1.02	-0.19
76	-0.09	-2.07	1.13	1.57	0.83	0.06	-1.41	1.45	0.71	0.52	-0.34
41	-0.28	0.38	-0.70	-0.88	0.07	0.21	-1.20	0.59	-0.45	-2.68	1.33
13	1.46	-1.49	-0.43	0.19	-0.37	0.52	-0.63	0.72	-0.38	1.06	0.55

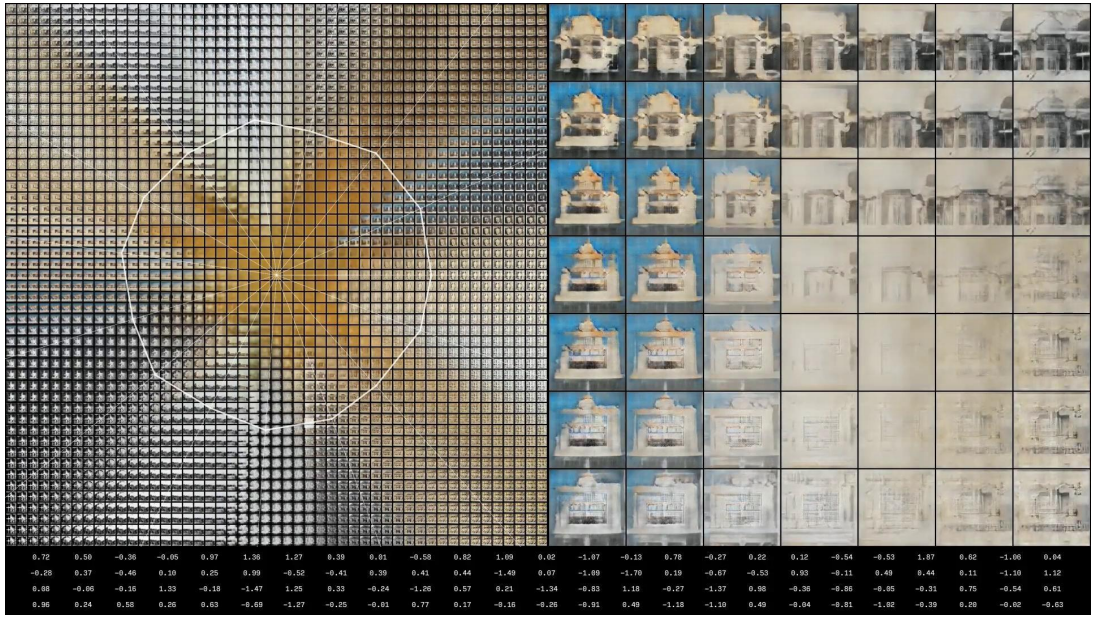
0.78	0.09	-0.06	0.35	0.26	-0.29	1.02	1.51	-0.56	-1.16	0.36
-0.68	0.58	0.72	1.09	0.47	-0.92	0.89	0.15	0.29	0.04	-0.61
-1.68	0.72	0.52	0.92	-0.25	0.88	0.84	-0.38	-1.77	-0.67	0.68
0.87	0.26	-0.37	0.26	-0.76	1.39	-1.65	0.54	-0.25	-0.09	-1.20

3	0.01	0.84	0.08	0.64	0.60	1.71	1.47	-0.34	0.85	-1.59	0.06
79	-0.31	0.52	-0.23	-1.12	0.30	-0.91	-0.07	-0.46	-0.18	-0.14	0.64
05	-0.19	-0.48	-0.36	0.18	0.33	-0.16	2.52	-0.43	-0.76	-0.09	-0.91
8	-0.61	0.21	-1.15	0.45	0.91	-0.05	-0.15	0.07	-0.05	1.44	-0.10

1.01	0.01	-0.30	-0.90	-0.58	-0.96	0.41	1.28	0.94	0.37	0.77
0.70	-0.07	-0.19	0.39	0.19	-0.74	0.07	-1.01	1.20	0.77	0.00
0.90	-2.21	-1.03	-1.08	0.68	1.28	-0.11	-0.27	-0.67	-0.24	0.66
0.15	0.17	0.16	-0.61	-0.01	-0.40	-0.47	0.63	0.81	0.04	-0.47

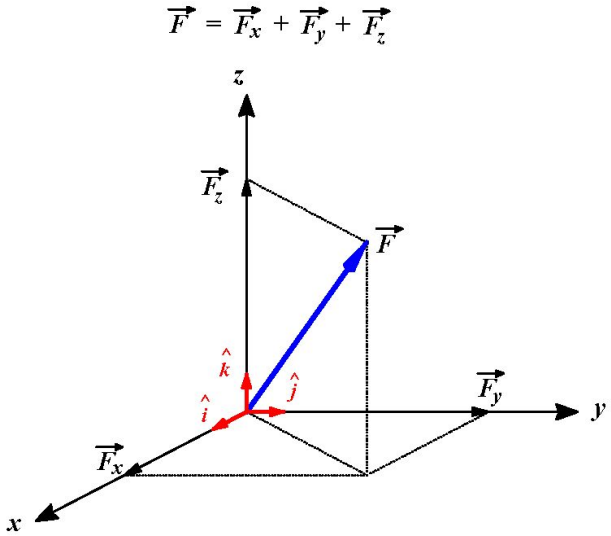
6	-0.03	-0.26	-1.87	0.90	-0.56	0.33	0.67	0.04	-0.44	-0.03	0.48
8	0.10	0.44	0.32	-0.68	-0.91	-1.09	-0.09	0.35	0.12	-0.92	-0.16
1	1.15	0.52	0.25	0.81	0.05	0.27	0.30	0.70	-0.46	0.66	-1.08
04	0.91	-1.64	0.98	-0.23	0.45	0.43	-0.51	0.37	-1.07	-0.45	-0.43

Every unique fake image is mapped from a unique latent vector. We can regenerate the same image using the same vector.



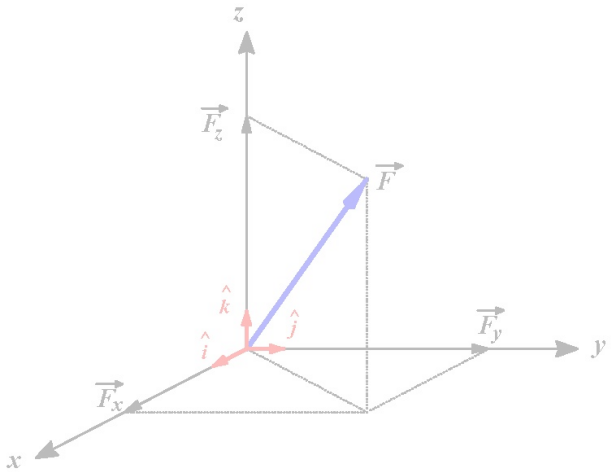
“Beaux-art Latent Walk” (2020)
Lab for Design Technologies
Prof. Andrew Witt, Gia Jung, Claire Djang

Think of Latent Space as a 100-dimensional Vector Space where Latent Vectors live



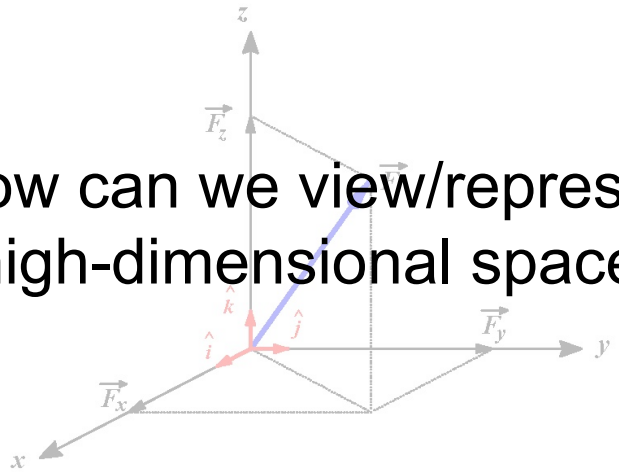
Think of Latent Space as a 100-dimensional Vector Space where Latent Vectors live

$$\text{vector} = (v_1, v_2, v_3, \ldots, v_{99}, v_{100})$$



Think of Latent Space as a 100-dimensional Vector Space where Latent Vectors live

$$\text{vector} = (v_1, v_2, v_3, \ldots, v_{99}, v_{100})$$



How can we view/represent high-dimensional space?

Related concept: projecting from higher dimensions

https://projector.tensorflow.org/

What do the latent vectors look like and how do their values vary?

Standard Normal Distribution

TORCH.RANDN

`torch.randn(*size, *, out=None, dtype=None, layout=torch.strided, device=None, requires_grad=False)` → Tensor

Returns a tensor filled with random numbers from a normal distribution with mean *0* and variance *1* (also called the standard normal distribution).

$$\text{out}_i \sim \mathcal{N}(0, 1)$$

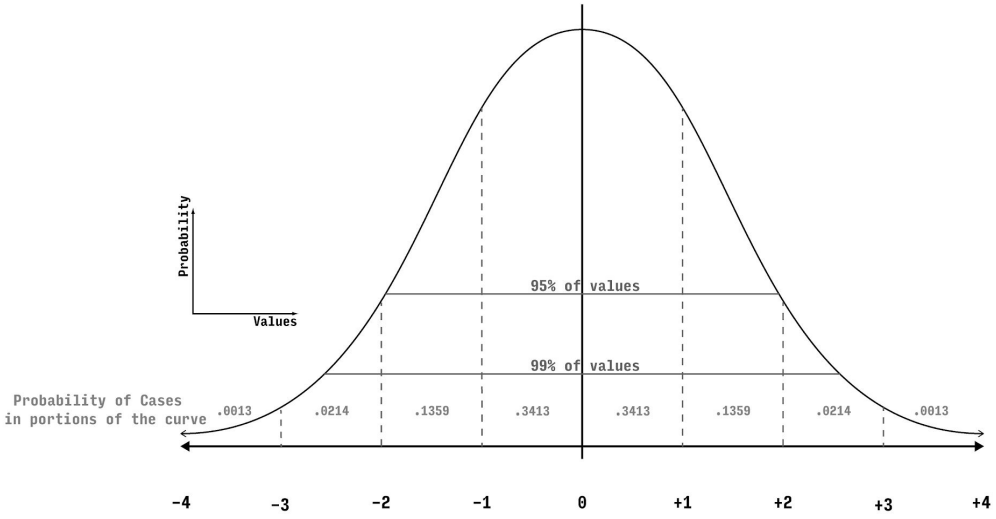
The shape of the tensor is defined by the variable argument `size`.

Parameters

size (*int...*) – a sequence of integers defining the shape of the output tensor. Can be a variable number of arguments or a collection like a list or tuple.

https://pytorch.org/docs/stable/generated/torch.randn.html

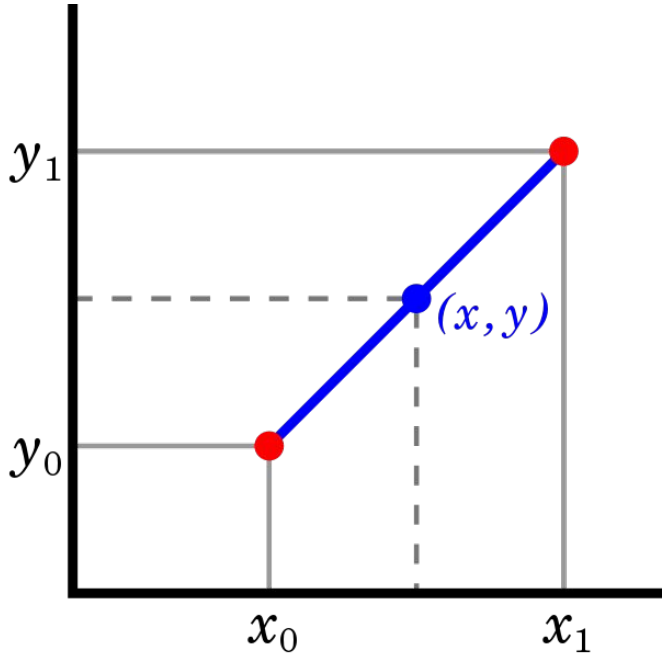
Standard Normal Distribution



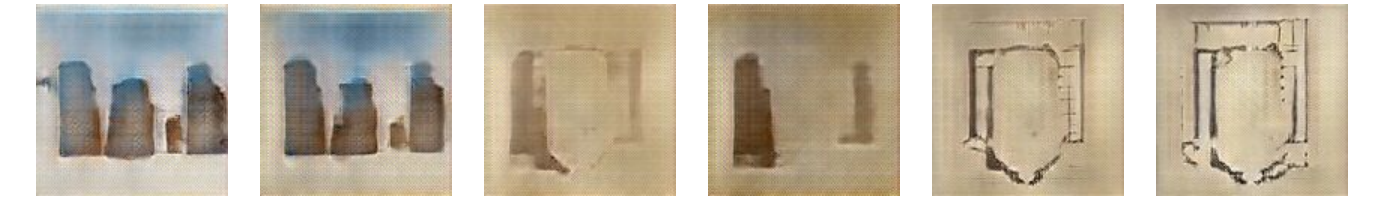
-0.86	0.39	1.67	1.02	-2.22	1.09	-0.99	0.21	0.10	-0.34	-0.28	1.27	2.08	0.90	1.24	0.26	-1.17	-0.71	-1.54	-2.73	0.87	0.79	1.15	1.02	-0.19
1.03	0.57	1.02	-0.42	-0.03	0.79	1.51	0.83	0.05	0.58	-0.72	1.27	0.82	-0.76	-0.09	-2.07	1.13	1.57	0.83	0.06	-1.41	1.45	0.71	0.52	-0.34
-0.41	0.91	0.80	-0.44	0.09	-0.37	-0.99	0.36	-0.66	-0.38	-0.56	0.31	1.22	-1.41	-0.28	0.38	-0.70	-0.88	0.07	0.21	-1.20	0.59	-0.45	-2.68	1.33
0.59	-0.93	0.86	0.79	0.76	0.65	0.93	0.01	1.68	0.28	-2.60	1.73	-1.25	-1.13	1.46	-1.49	-0.43	0.19	-0.37	0.52	-0.63	0.72	-0.38	1.06	0.55

One possible way to explore the latent space:
linear interpolation

Linear Interpolation



Linear Interpolation



Curating Individual Latent Vectors (Beaux-Arts Example)

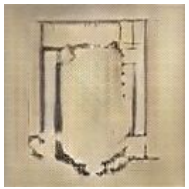
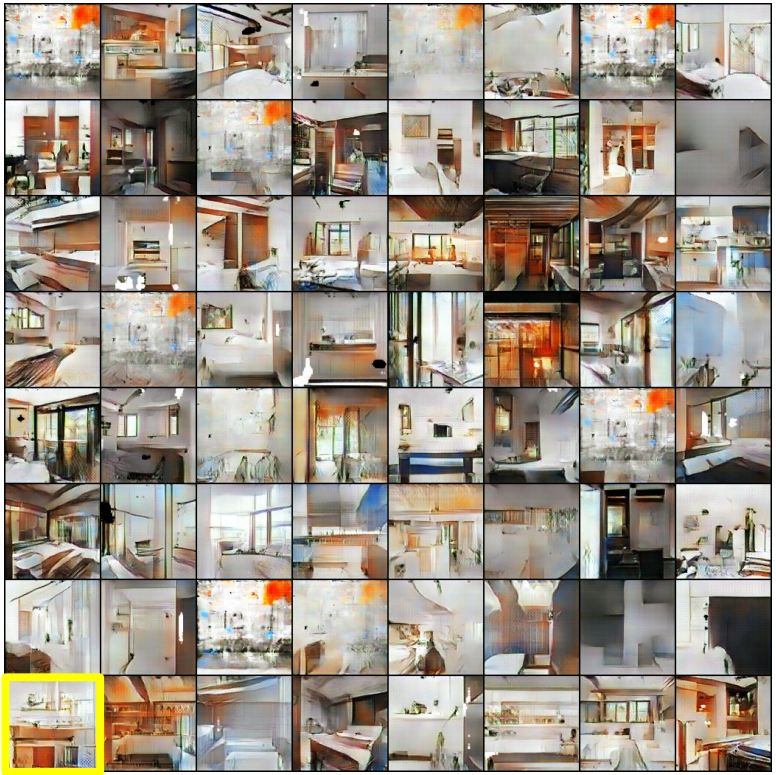
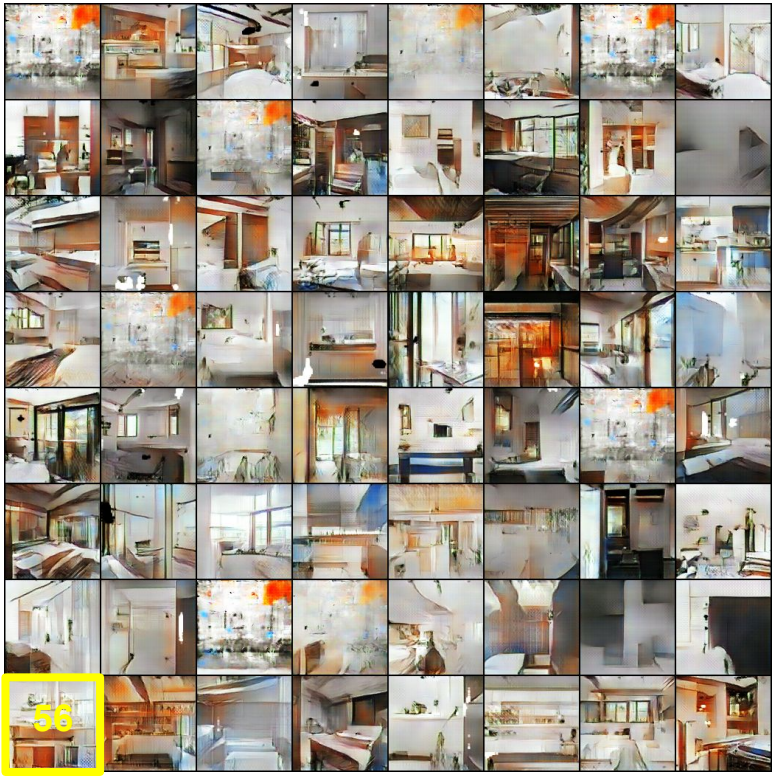
Image					
Random Seed	0	15	73	80	98
Image Index	52	46	33	32	60
Vector Index	0	1	2	3	4

Image Index in PyTorch Grid

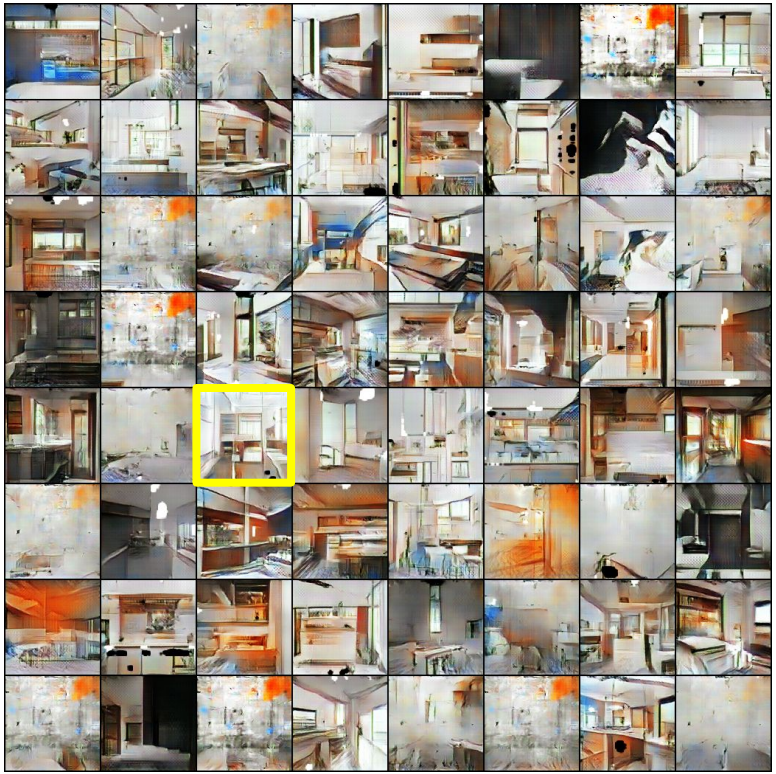
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63



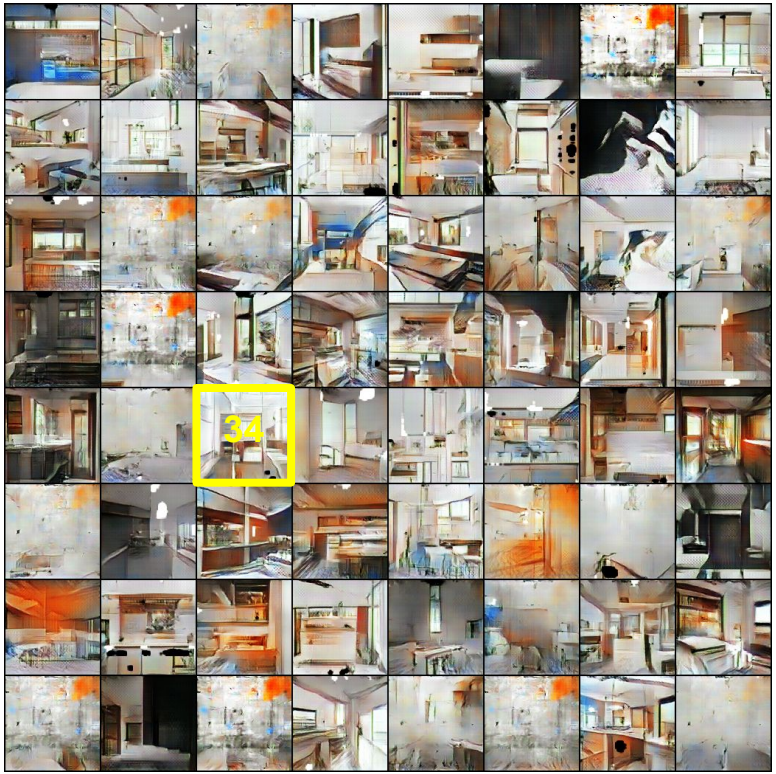
seed00002.png



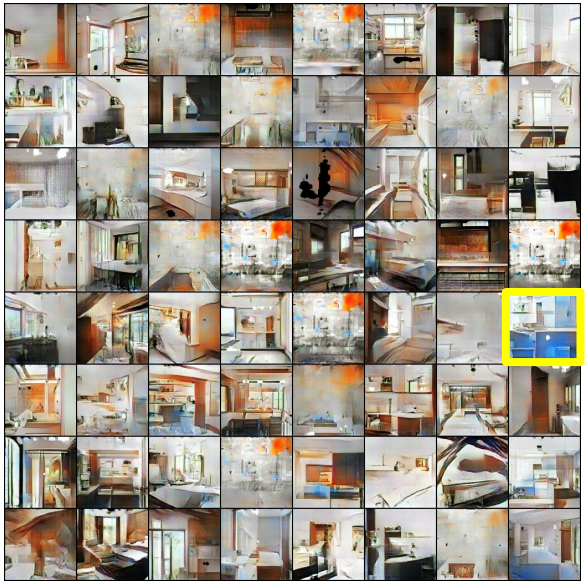
seed000**02**.png



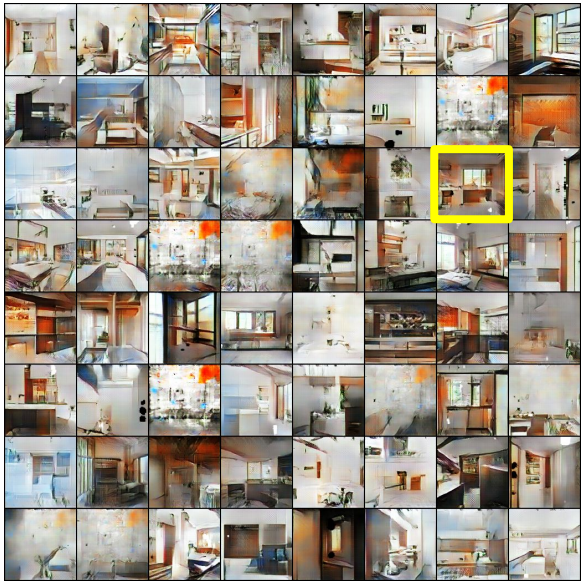
seed00010.png



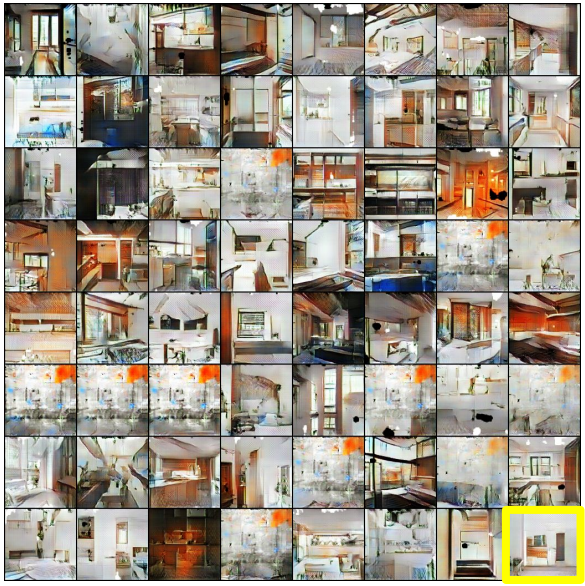
seed00010.png



seed00018.png



seed00021.png



seed00043.png

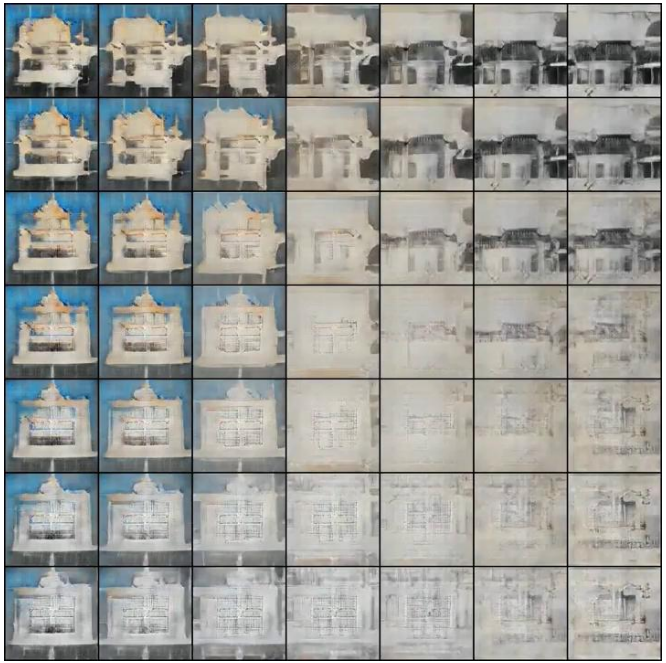
Selecting Individual Latent Vectors (Dwell Example)

Image					
Random Seed	2	10	18	21	43
Image Index	56	34	39	22	63
Vector Index	0	1	2	3	4

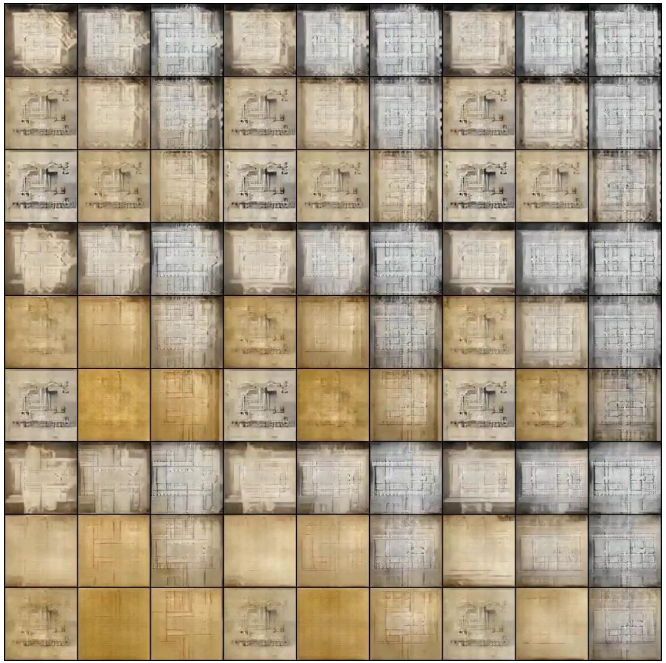
Linear Interpolation

More Advanced Visualizations
(Coming soon… on Wednesday workshop!)

Linear Grid Interpolation



Multidimensional Iterations



Vector Arithmetic

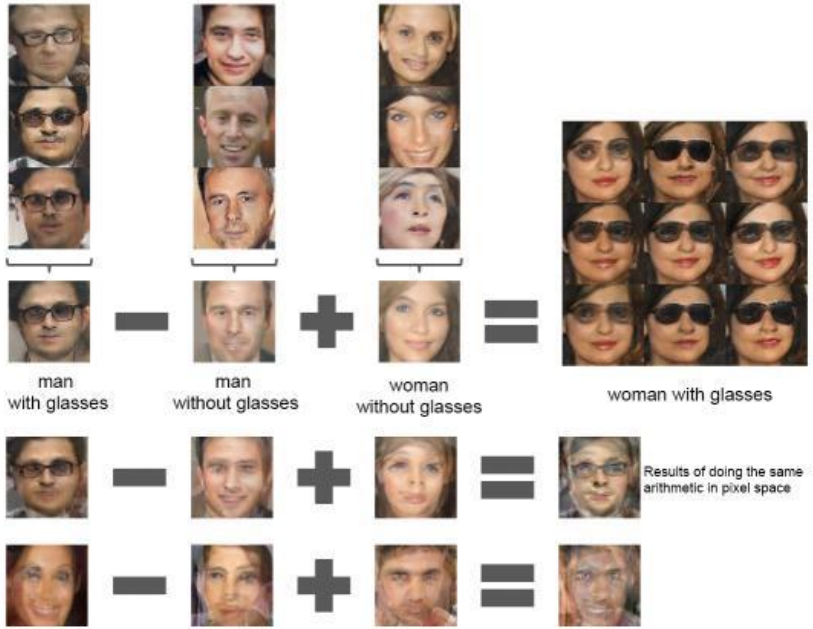


Figure 7: Vector arithmetic for visual concepts. For each column, the Z vectors of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector Y. The center sample on the right hand side is produce by feeding Y as input to the generator. To demonstrate the interpolation capabilities of the generator, uniform noise sampled with scale +-0.25 was added to Y to produce the 8 other samples. Applying arithmetic in the input space (bottom two examples) results in noisy overlap due to misalignment.

From [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks]

HW 4

For next class on Wednesday we will begin with each group/individual presenting their final project proposal

HW 4

After learning about the latent space visualization tools and scripts we have provided (or will provide Wednesday), begin working towards your final project by proposing a visualization idea specific to your dataset. What have you learned from your data / training results and how can you show this through visualization? The possibilities for the final project are open-ended.

The first step is usually *isolating the specific latent vectors you'd like to work with*. Prepare a quick presentation that includes these individual chosen vectors, what you'd like to accomplish with them, and any process/progress/challenges.

Due: Wednesday January 13th