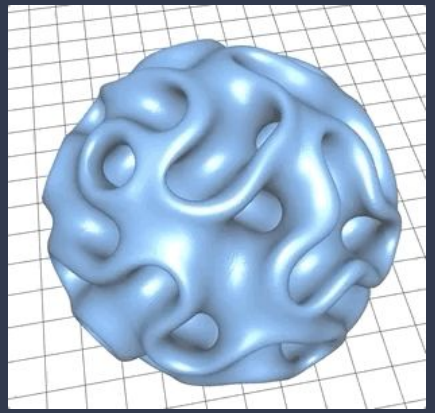
3D Mesh Mash

Final Presentation

MIDS Capstone Project Spring 2019:
Cynthia Hu, Dave Owen, Jack Workman



Solving design challenges with data science

Problem statement: How can we influence 3D forms in an effort to aid in a design process?

Customer: Design oriented professions. E.g. automotive engineers, architects, design specialists, game designers, etc.

Value Proposition: Our service will allow a user to test 3D forms to envision a new mashup object, aiding brainstorming and sparking new ideas for designers

Definition of Success: Bringing our algorithm from concept to physical reality - successfully printing an original product

***Hypothetical Customer*: Pininfarina**

Legendary Italian design shop

AUTOMOTIVE DESIGN

DESIGN PROCESS

Pininfarina's style has always been a byword for beauty, elegance and functionality.
The aesthetic research carried out by our designers is steered constantly towards achieving pure, uncluttered shapes, to perpetuate their beauty.
The activities of our Centro Stile and some of the most recent projects are described below.

CREATIVE RESEARCH

3D VIRTUAL MODELLING

3D PHYSICAL MODELLING

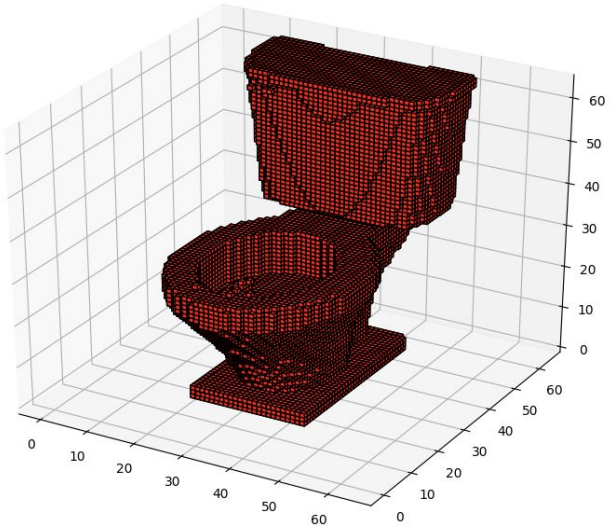
REFINEMENT

MODEL CREATION

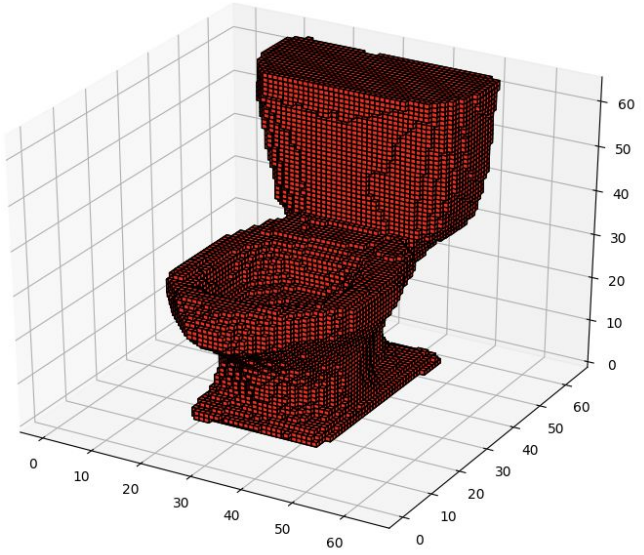
- Reduced size and 1:1 scale mock-ups
- Milling
- Manual modelling

Here we try to construct a ... toilet

Input (Original)

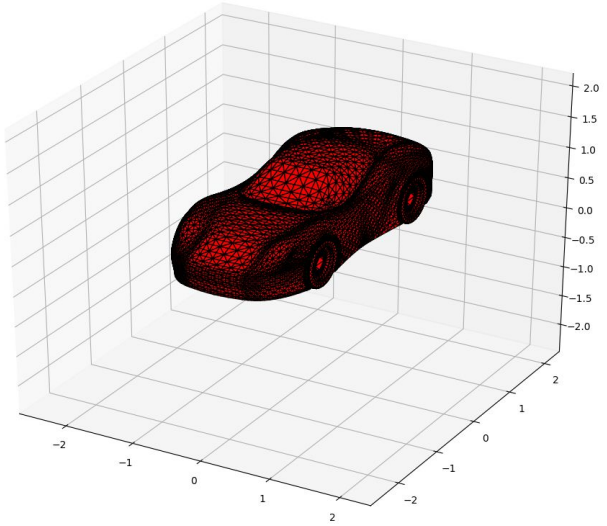


Output (Reconstruction)

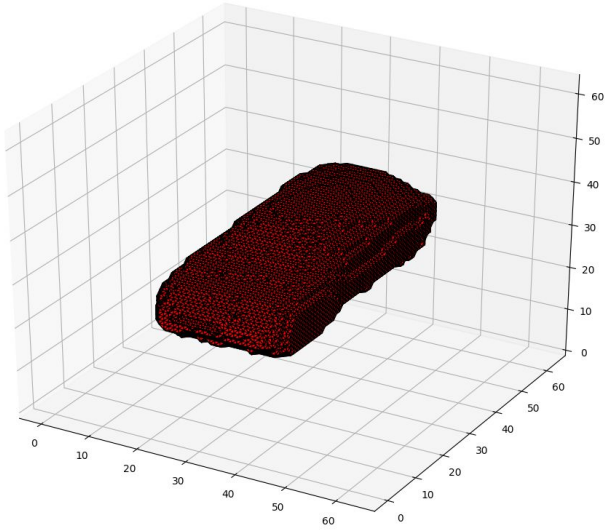


Here we try to construct a … neither car nor car category in the training data

Input (Original)



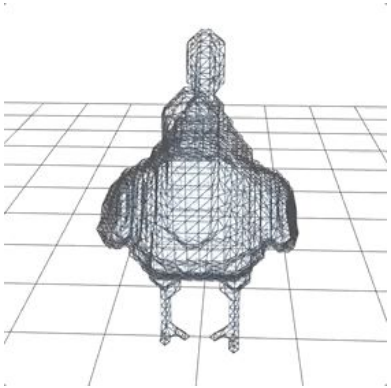
Output (Reconstruction)

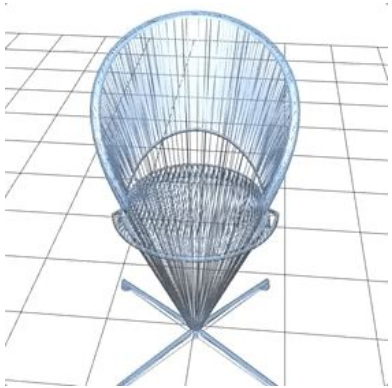


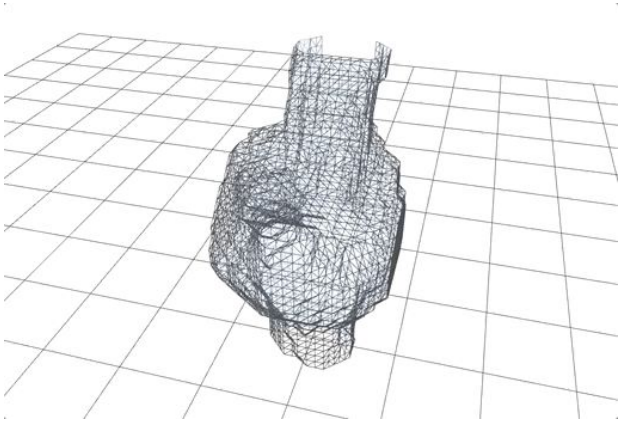
Here we try to create a combination of a chicken and a chair

Inputs (Original)

Output (Mashup)

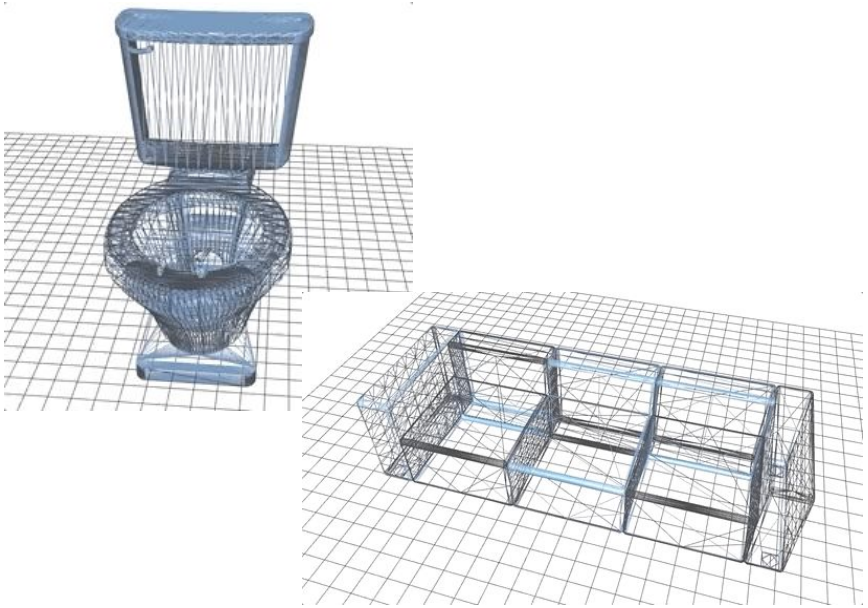




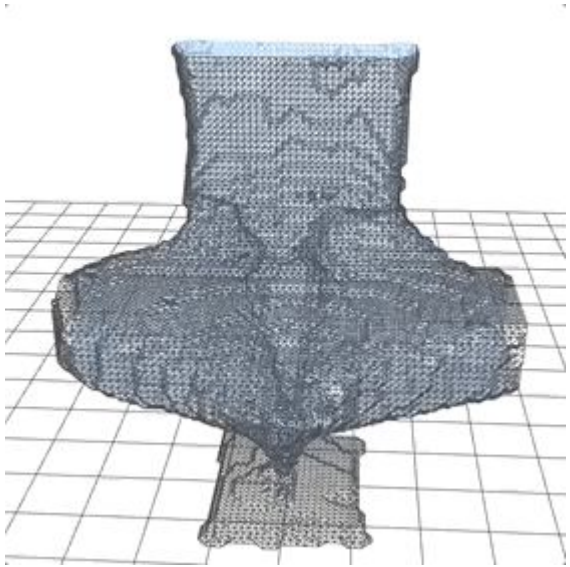


Here we try to create a combination of a sofa and a toilet... because why not?

Input (Original)



Output (Mashup)

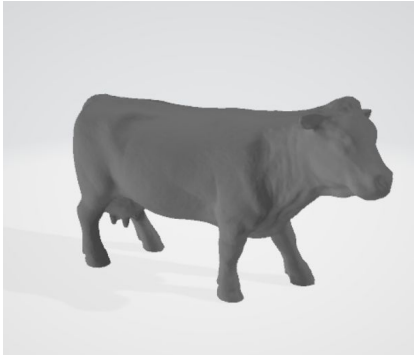


Dataset

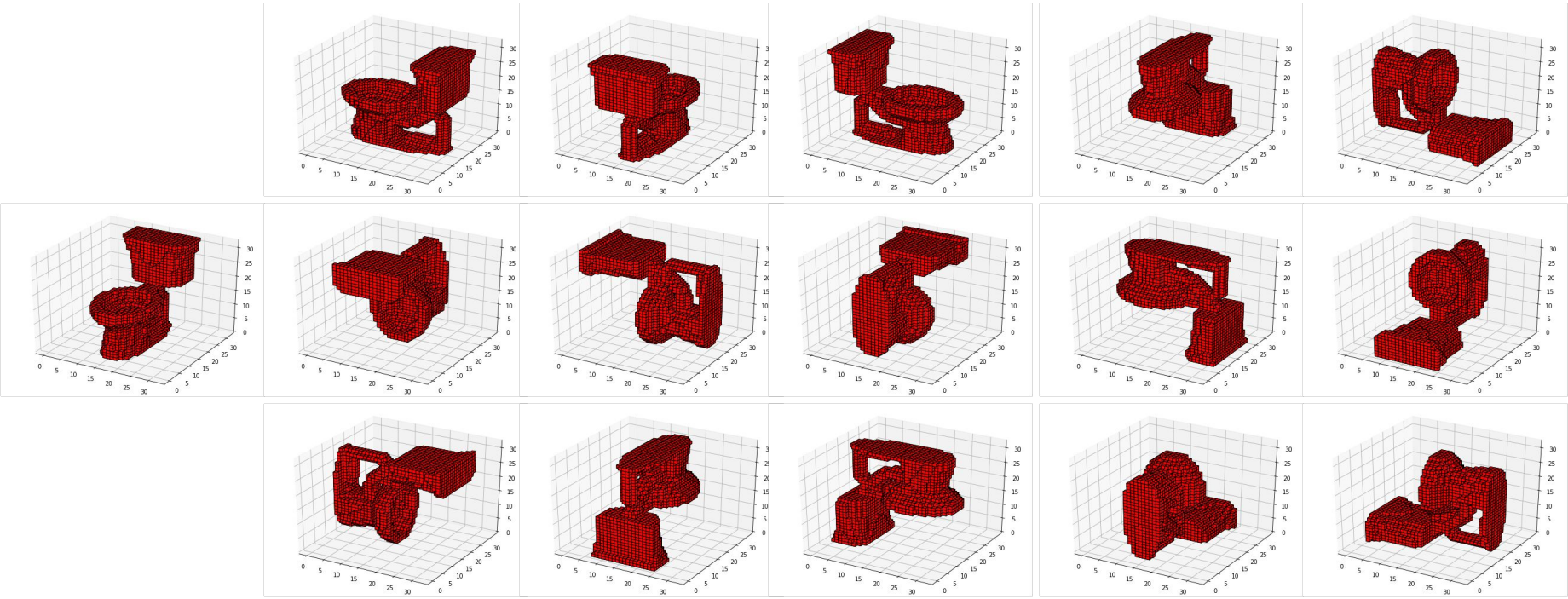
Thingi10k
- Publicly available: https://ten-thousand-models.appspot.com/
- 10,000 3D printable files
- Reviewed for quality
- Additional metadata
- Files are STL format - triangular meshes encoded as vertices

ModelNet10:
- http://modelnet.cs.princeton.edu/
- 4898 CAD models from the 10 categories
- Convert .off file to .binvox
- Add rotation versions of the file

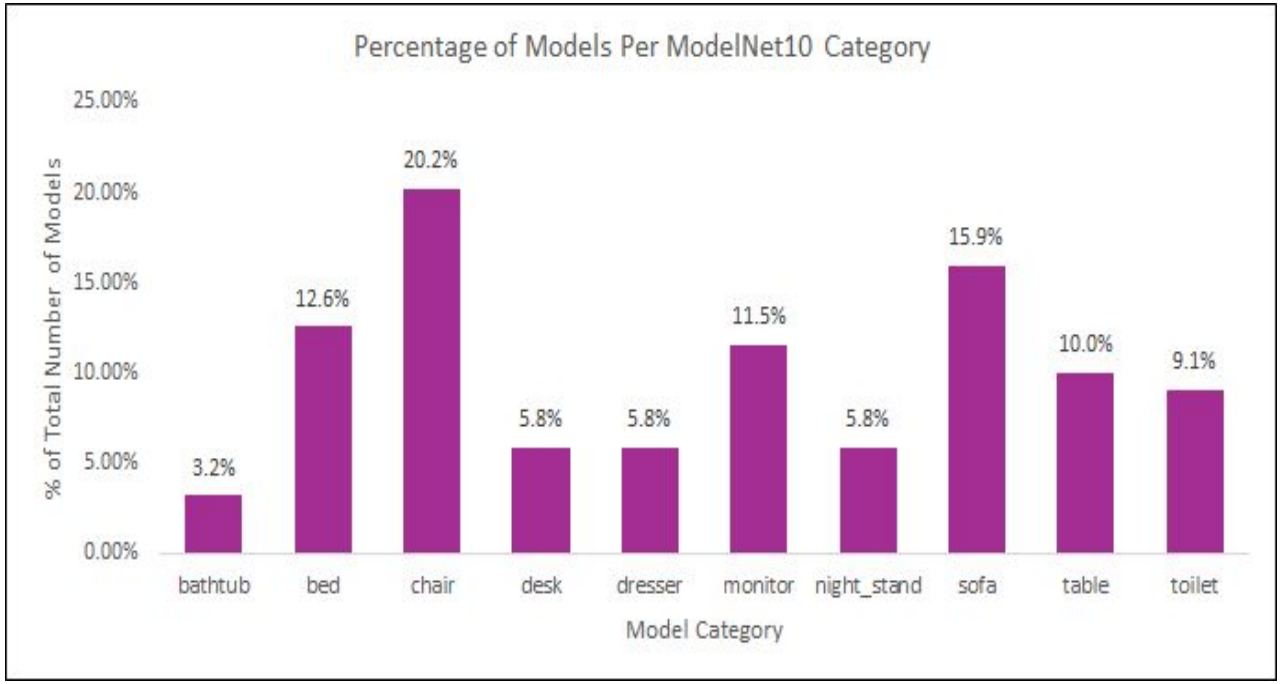
Size:
- 1 32dim binvox file = 4.0 kb
- 1 64dim binvox file = 8.0 kb
- *.stl files can be any size depending on complexity of shape (number of triangles)



ModelNet10 Rotations



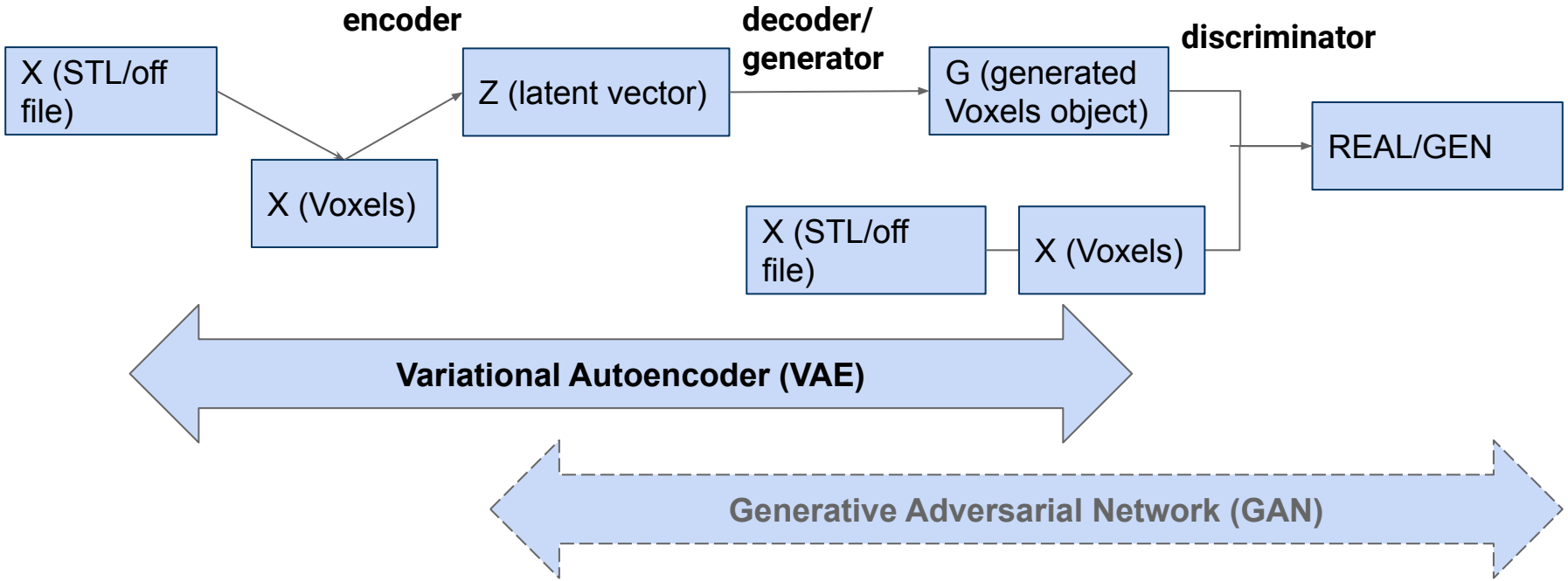
ModelNet10 Categories



Training Set: 81%

Test Set: 19%

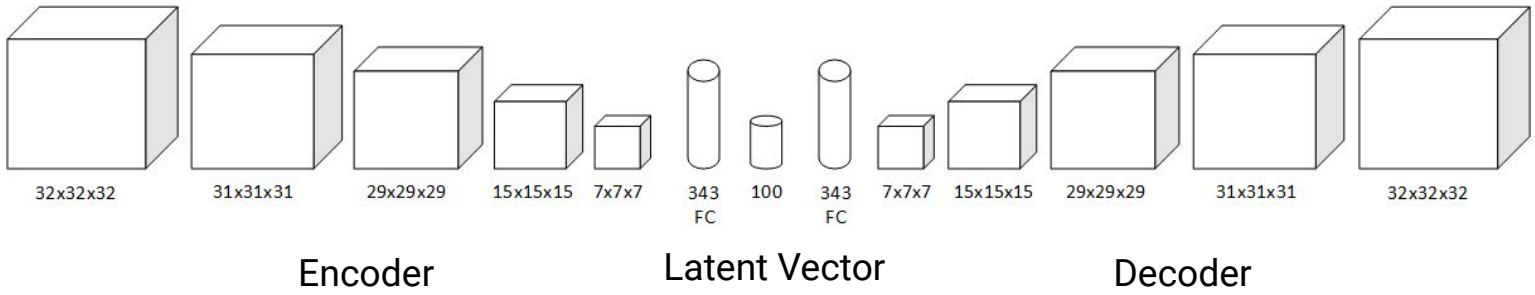
Solution Architecture



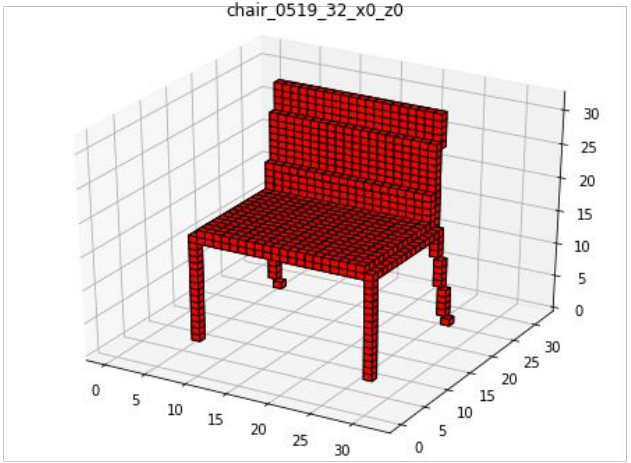
VAE 1.0 Architecture

VAE 1.0

- Input & Output: 32x32x32 voxel shape
- Latent vector dimension: 100
- Number of layers: 10
- Loss function: $W_{recon}L_{recon} + W_{recon}L_{kl}$



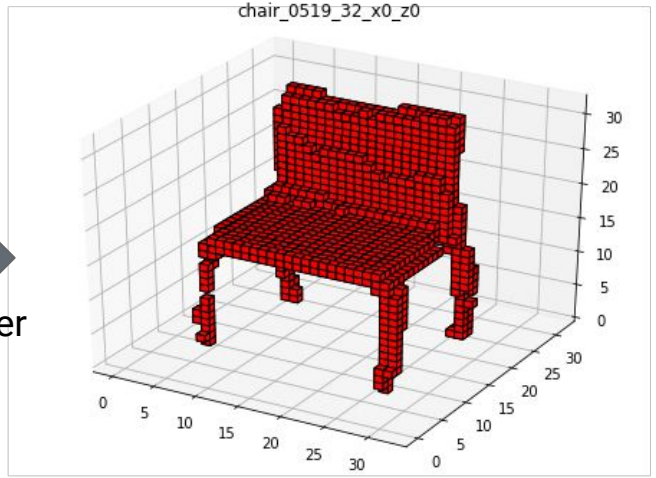
VAE 1.0 Architecture



Encoder

x_1
x_2
...
x_n

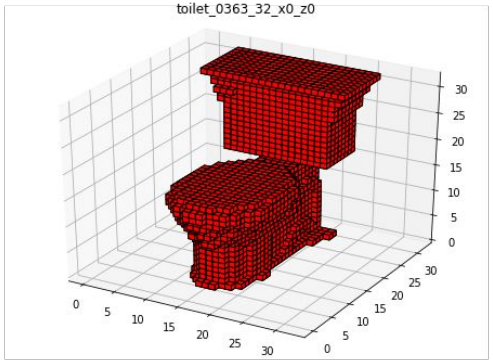
Decoder

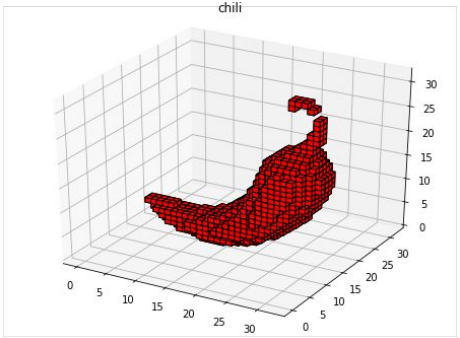


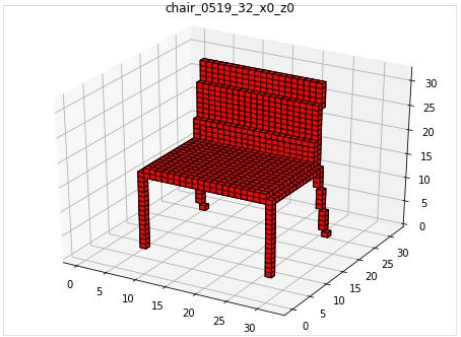
n = 100
Latent Vector

VAE 1.0 Results

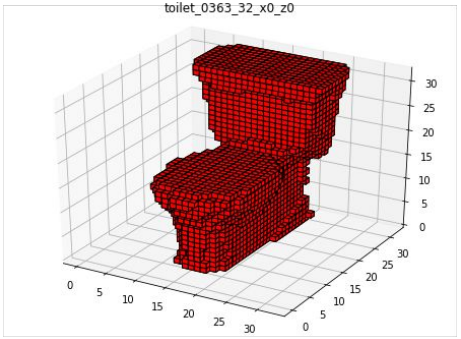
Original

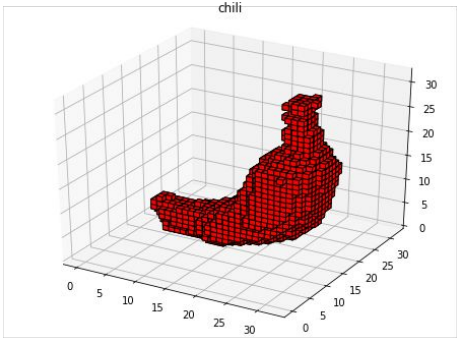


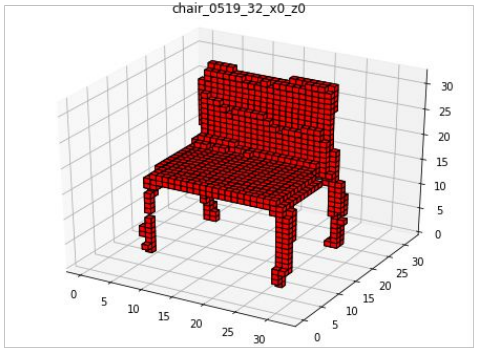




Reconstruction



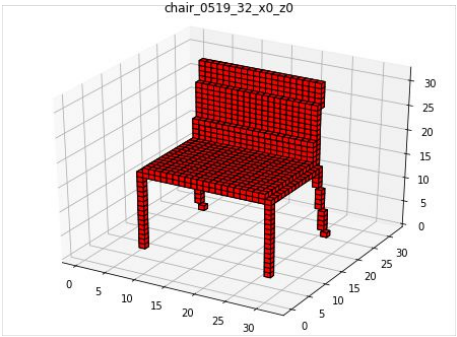




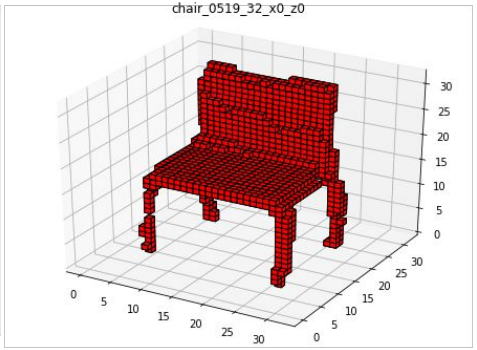
VAE 1.0 Limitations

1. VAEs tend to produce “blurry” output… A problem fixed by GANs

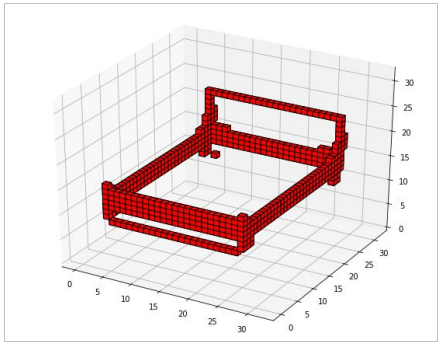
2. Input shape of 32x32x32 fails to capture important features



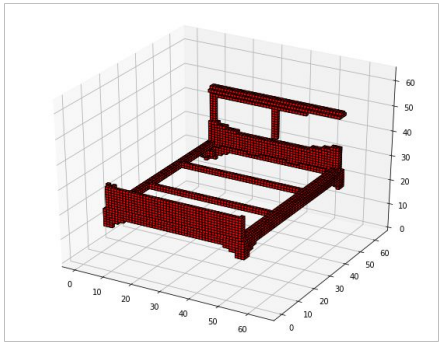
Straight Legs



“Blurry” Legs



32x32x32



64x64x64

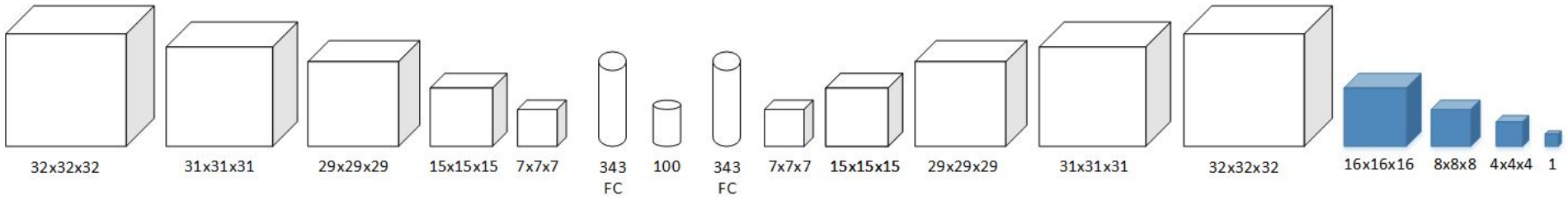
Solution: Add a GAN

Solution: Increase to 64x

So we tried a VAE-GAN...

VAE-GAN

- Base: 32x32x32 VAE 1.0 model
- Input & Output: 32x32x32 voxel shape
- Number of layers: 10 -> 14
- Loss function: GAN



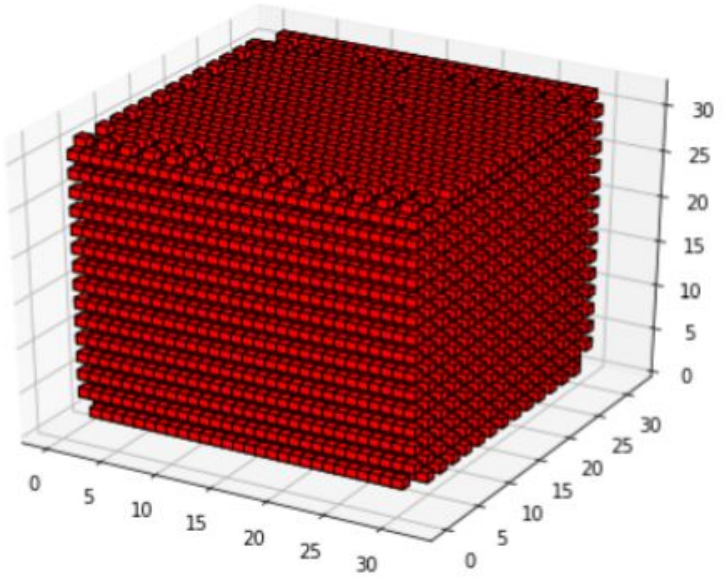
VAE-GAN Results

Training time: 3x slower than 32x32x32 VAE 1.0 Model

Training impacted by

- Non-convergence
- Mode collapse

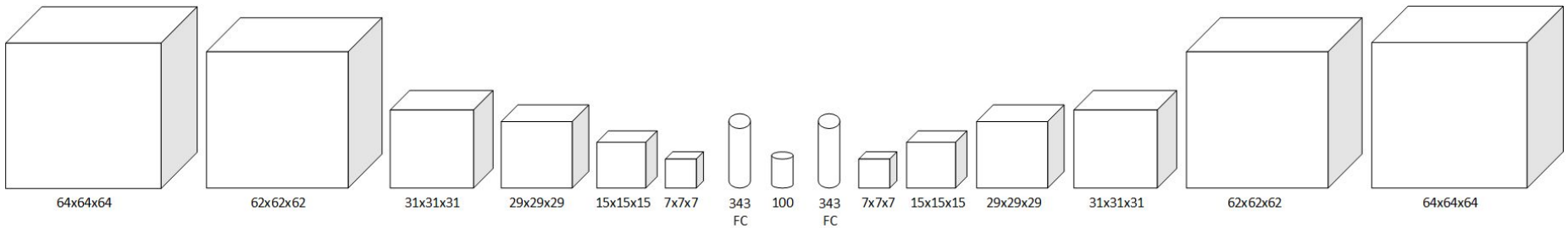
Ultimately, our efforts were unsuccessful, so we pursued option 2 increasing the dimensions to 64 and...



Introducing VAE 2.0!

VAE 2.0

- Input & Output: 64x64x64 voxel shape
- Number of layers: 10 -> 12
- Same loss function
- Parameters: 2,374,0958 -> 2,396,2526



VAE 2.0 Training Challenges

More layers and larger input drastically slows down training time…

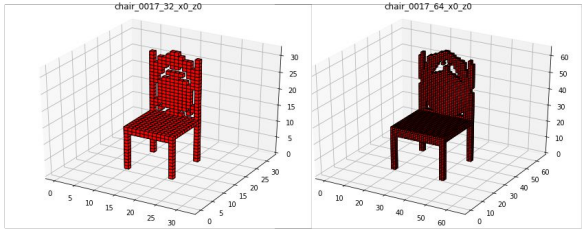
- **32x Model at 200 epochs on CPU = 4.5 days**
- **64x Model at 200 epochs on CPU ~= 53 days!**

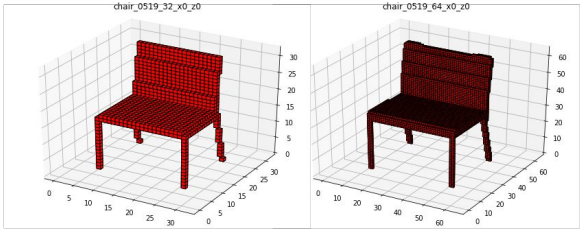
Our training system's final configuration:

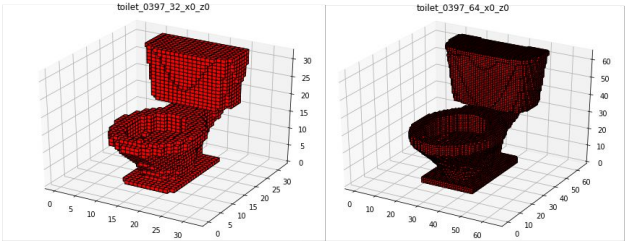
- Memory: 64 GB
- Processor: Intel Core i9-7900X CPU @ 3.30GHz x 20
- Disk: 1.0 TB
- OS: Ubuntu 16.04
- NVIDIA GeForce GTX 1060 6GB (added for VAE 2.0)

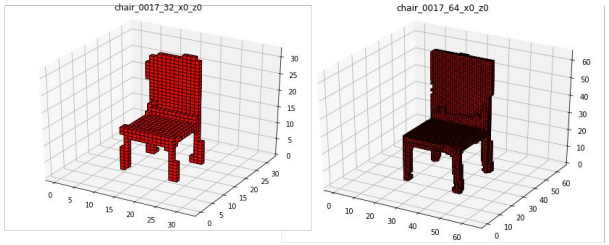
NVIDIA GPU Saves the Day: 14x Speedup

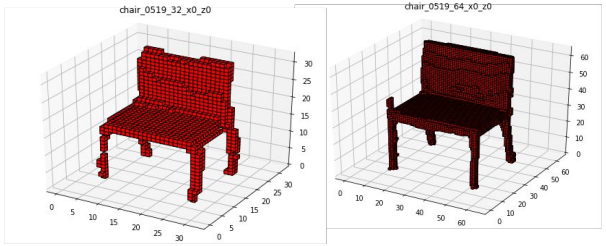
VAE 2.0 Results

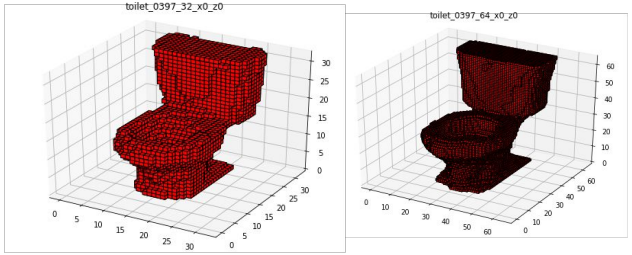




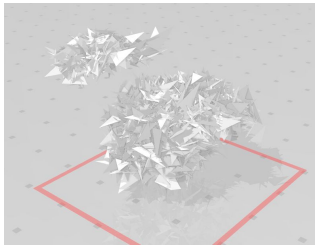


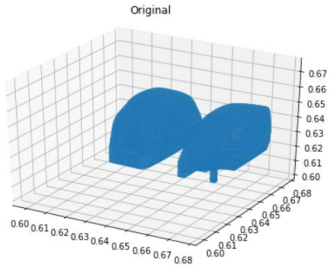


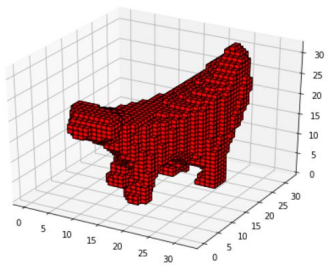


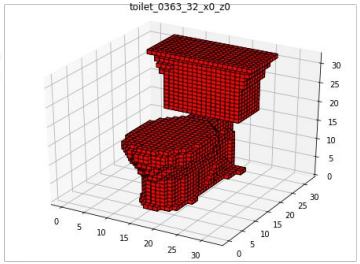


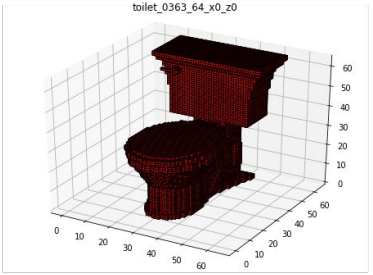
A Look Back: How Our Model Evolved

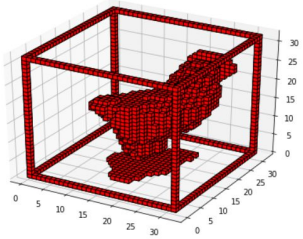


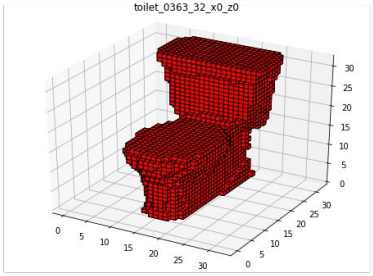


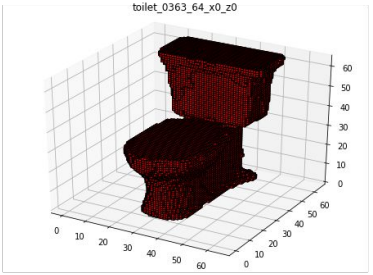












Week 4 | Week 6 | Week 8 | Week 12 | Week 14

STL VAE Tuning | Mesh -> Voxels | ModelNet10/VAE-GAN | 64x64x64

The Challenge of 3D Data

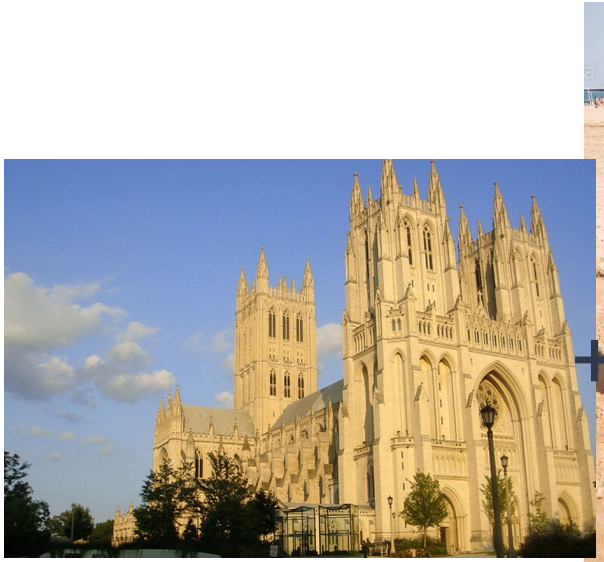
Moving to 3D from 2D is not as simple as adding another dimension

- 2D images are abundant; 3D objects are not
- Voxel vs Mesh
- Voxel grids do not capture scale well (airplane is same size as an apple)
- Higher dimensionality means higher quality but slower training - GPU is a must

Total project training time across all experiments: 22.26 days

- 5.4 days total for the VAE-GAN
- 16.83 days total for VAE 1.0/2.0

And we imagine enhancing to solve some meaningful long run problems...



+

=

… and some a little more fun.

And we take it forward by enhancements

Next Steps

- [ ] Improve Generative Capabilities / Finer Control in Manipulation
- [ ] Further tuning 64x64x64 models on GPU
- [ ] Explore more data sets
- [ ] Research non-linear mashup
- [ ] Combine 3 or more objects

Questions?

Check out our website:
http://people.ischool.berkeley.edu/~dave.owen/

Appendix - Resources

- Github link
- Presentation link
- Project Website

datascience@berkeley

3D Mesh Mash

Reimaging design and aiding creativity through algorithm-driven 3D form manipulation.

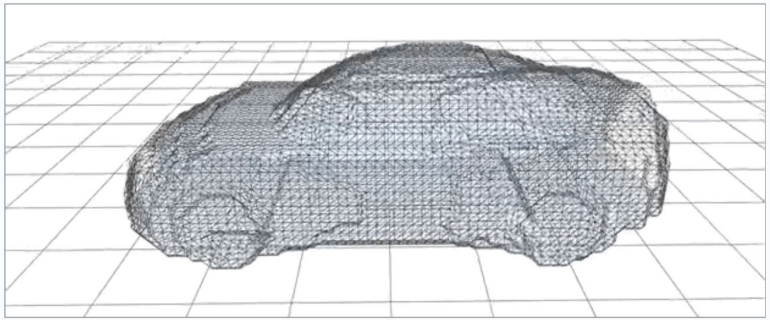
Purpose

Practice

Theory

Path Forward

Team



3D Mesh Mash Combination of Elise and R8 Car STL Models

Appendix - Reference

Papers

- Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (2016) [Paper]
- Neural 3D Mesh Renderer (2017) [Paper] [Code]
- Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling: http://3dgan.csail.mit.edu/
- Generating 3D-objects using neural networks: http://www.diva-portal.org/smash/get/diva2:1218064/FULLTEXT01.pdf
- Analogy-Driven 3D Style Transfer (2014) [Paper]
- Functionality Preserving Shape Style Transfer (2016) [Paper] [Code]
- Generating 3D Mesh Models from Single RGB Images (2018) [Paper]
- https://arxiv.org/pdf/1512.09300.pdf

Appendix - Reference

Blogs/Github

- GANs: https://skymind.ai/wiki/generative-adversarial-network-gan
  https://wiseodd.github.io/techblog/2016/09/17/gan-tensorflow/
- Autoencoders:http://kvfrans.com/variational-autoencoders-explained/
  https://jmetzen.github.io/2015-11-27/vae.html
  Math-heavy tutorial: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
  YouTube video [github] [paper]
  https://github.com/tayden/VAE-Latent-Space-Explorer/blob/master/scripts/VAE.ipynb
- VAE-GAN:
  https://github.com/timsainb/Tensorflow-MultiGPU-VAE-GAN
  https://github.com/Spartey/3D-VAE-GAN-Deep-Learning-Project/blob/master/3D-VAE-WGAN/model.py
- Training
  https://github.com/anitan0925/vaegan/blob/master/examples/train.py
  https://github.com/jlindsey15/VAEGAN/blob/master/main.py
- Style transfer (2D):
  https://github.com/lengstrom/fast-style-transfer
  https://harishnarayanan.org/writing/artistic-style-transfer/
  https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/15_Style_Transfer.ipynb

A Full Design Pipeline

Production Path

1st Design Selection

2nd Design Selection

Smoothing Engine

- Final prep for print
- Using commercial smoothing functions

3D Publish or Print

- Send STL file to print and produce final form

Training & Stylization Path

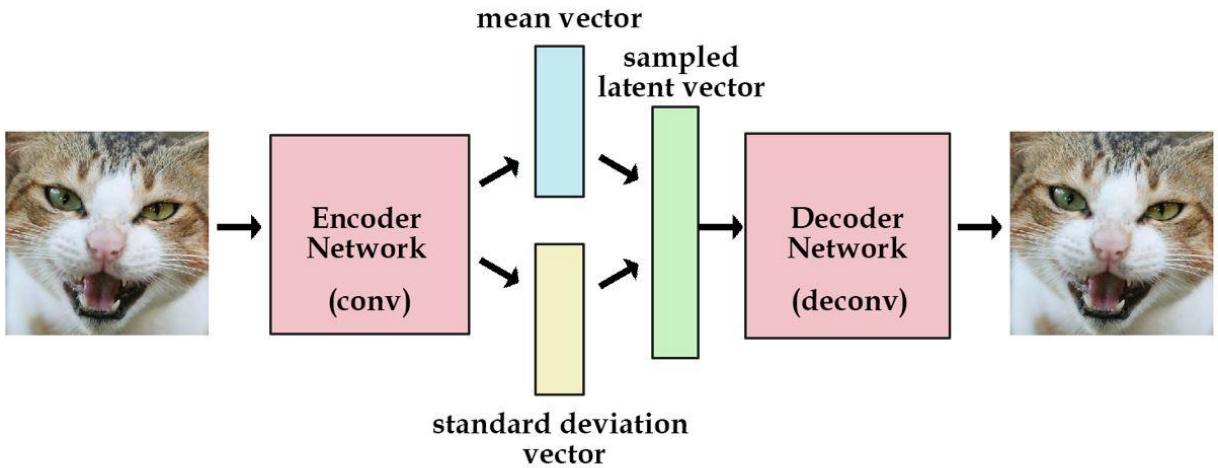
Generator Engine

- Generates new form
- Influenced by design selection

Discriminator Engine

- Is object realistic?
- Trained on large corpus of existing 3D designs

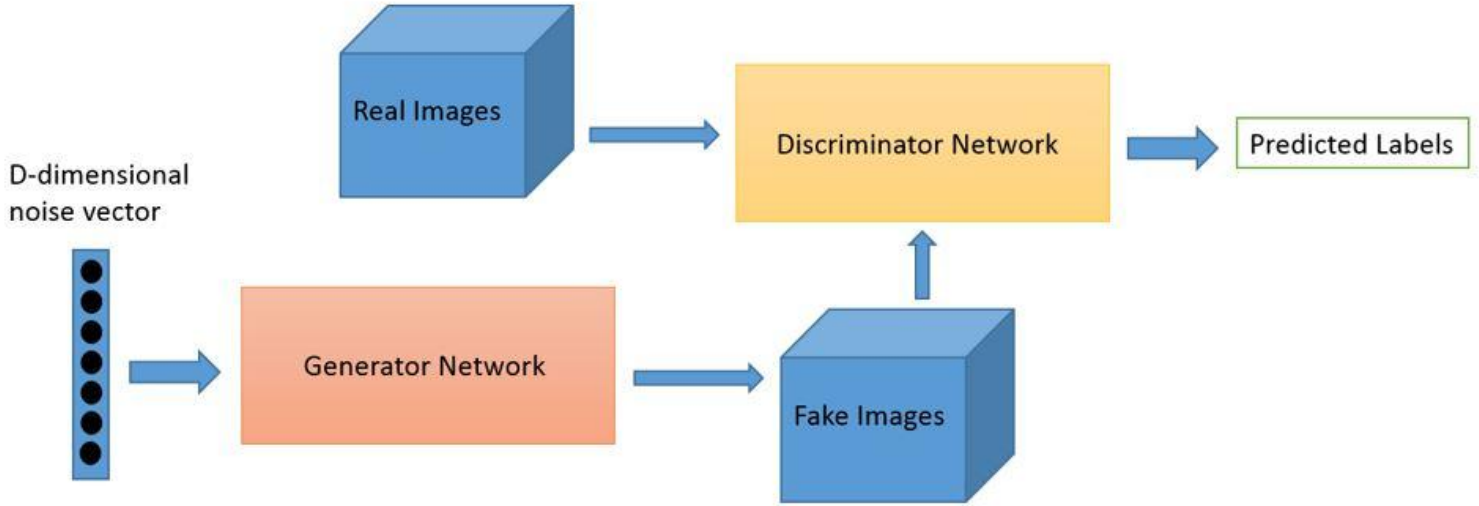
Appendix - Variational Autoencoder (VAE)




```
generation_loss = mean(square(generated_image - real_image))
latent_loss = KL-Divergence(latent_variable, unit_gaussian)
loss = generation_loss + latent_loss
```

Source: http://kvfrans.com/variational-autoencoders-explained/

Appendix -Generative Adversarial Network (GAN)



Credit: O'Reilly

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))], \quad (1)$$

VAE 1.0 Loss Function

$$Loss = W_{recon}L_{recon} + W_{recon}L_{kl}$$

Loss Function

- **Reconstruction Loss**: weighted binary cross entropy to penalize false negatives
- **KL Divergence**: ensures that the encoded latent vector is an efficient descriptor of the input
- **Recon weight**: 10^{5}
- **KL divergence weight**: 10^{-1}

VAE 2.0 Encoder Model Layers

Layer	Filters	Strides	Padding	Activation	Shape
Enc_Conv1	8	[1, 1, 1]	Valid	ReLu	[62, 62, 62, 8]
Enc_Conv2	16	[2, 2, 2]	Same	ReLu	[31, 31, 31, 16]
Enc_Conv3	32	[1, 1, 1]	Valid	ReLu	[29, 29, 29, 32]
Enc_Conv4	64	[2, 2, 2]	Same	ReLu	[15, 15, 15, 64]
Enc_Conv5	64	[2, 2, 2] (1,1,1)	Valid	ReLu	[7, 7, 7, 64]

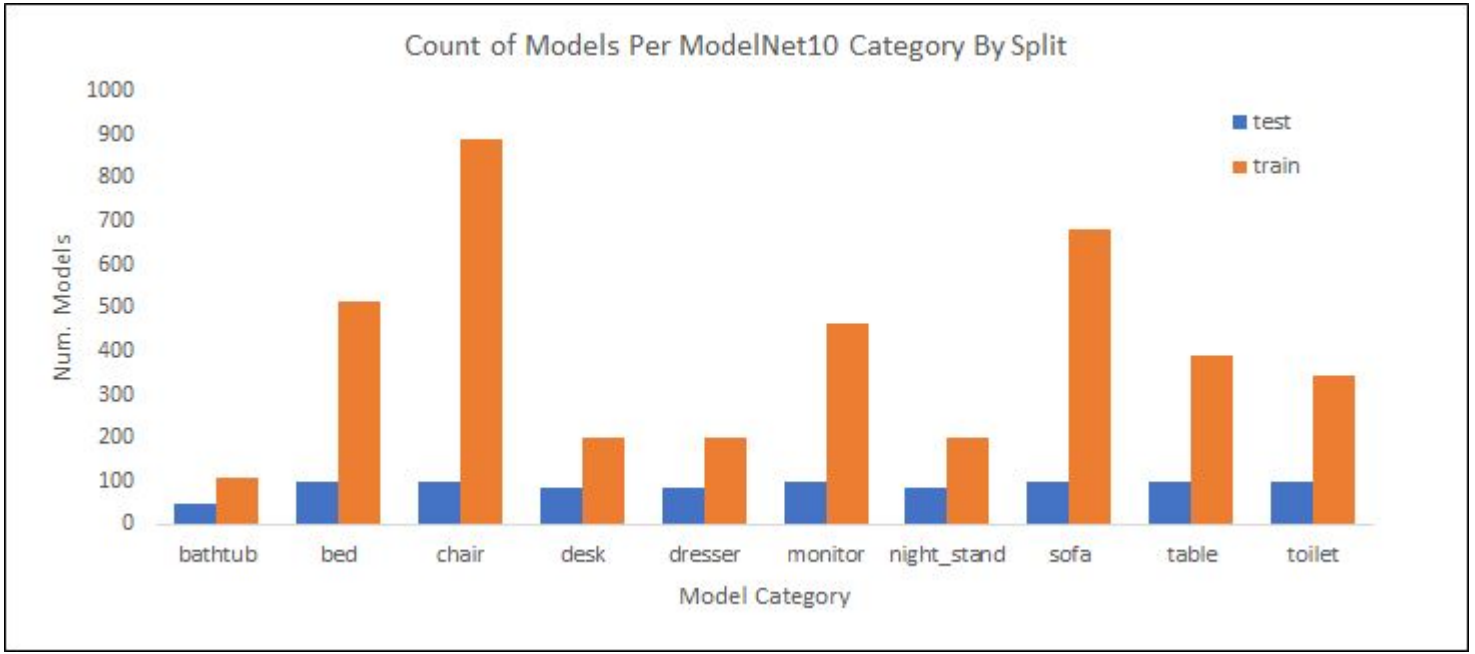
VAE 2.0 Decoder Model Layers

Layer	Filters	Strides	Padding	Activation	Shape
Dec_Conv1	64	[1, 1, 1]	Same	ReLu	[7, 7, 7, 64]
Dec_Conv2	64	[1, 1, 1]	Same	ReLu	[15, 15, 15, 32]
Dec_Conv3	32	[2, 2, 2]	Valid	ReLu	[31, 31, 31, 32]
Dec_Conv4	16	[1, 1, 1]	Same	ReLu	[31, 31, 31, 16]
Dec_Conv5	8	[2, 2, 2]	Valid	ReLu	[64, 64, 64, 8]
Dec_Conv6	1	[1, 1, 1]	Same	Linear	[64, 64, 64, 1]

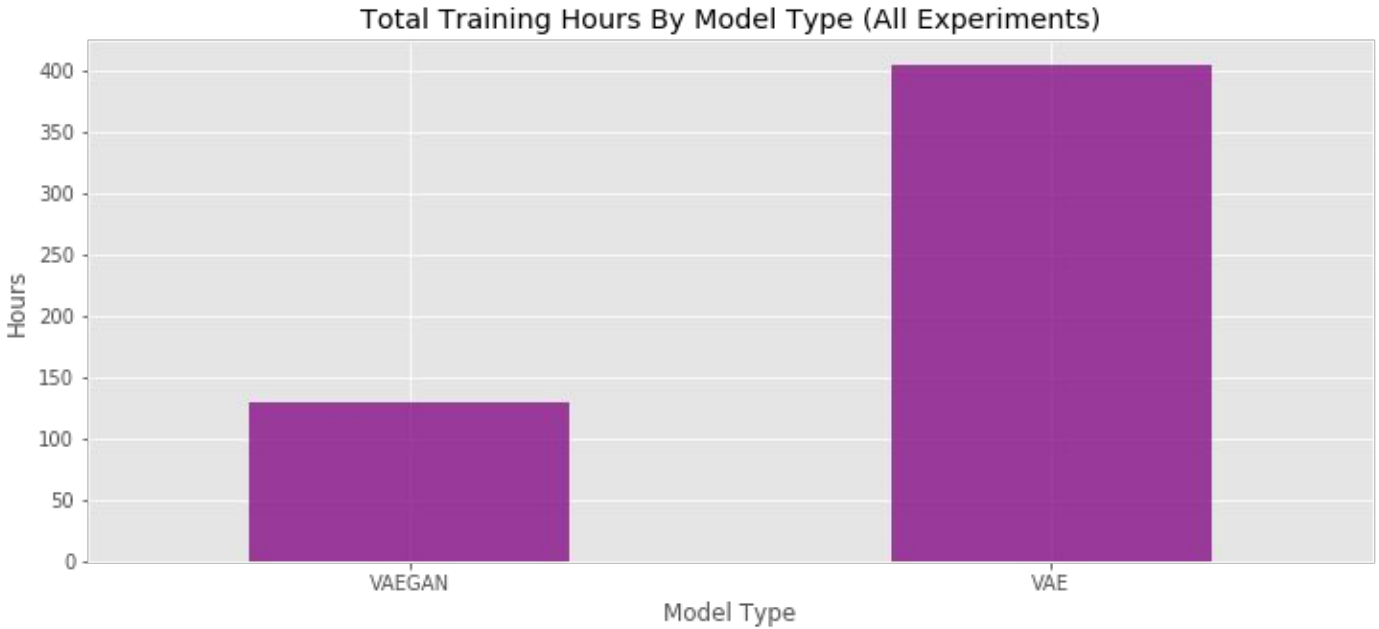
VAE-GAN Discriminator Layers

Layer	Filters	Strides	Padding	Activation	Shape
Dis_Conv1	128	[2, 2, 2]	Same	Leaky_ReLu	[7, 7, 7, 64]
Dis_Conv2	256	[2, 2, 2]	Same	Leaky_ReLu	[15, 15, 15, 32]
Dis_Conv3	512	[2, 2, 2]	Same	Leaky_ReLu	[15, 15, 15, 64]
Dis_Conv4	1024	[1, 1, 1]	Valid	Linear	[31, 31, 31, 32]

ModelNet10 Test/Train Split



Training



Total Project Training Time: **22.26 days = 534.17 hours**

Model Training

Model	VAE 1.0	VAE 2.0
System	CPU	GPU
Dimension	32	64
Epochs	200	200
Batch Size	128	32
Time per batch (mins)	0.05	0.01
Time per epoch (mins)	31.85	30.95
Training Time	4.45 days	4.29 days

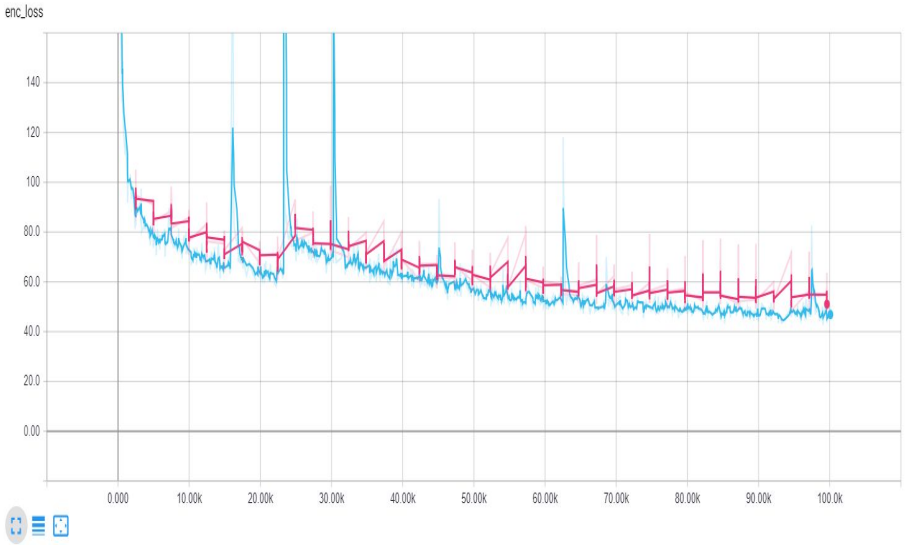
VAE 1.0 Training

Training Parameters

- Epochs: 200 (13+ million input examples)
- Batch size: 128
- Optimizer: Adam
- Learning rate: 0.0001

Training Time

- Time per batch: 0.05 mins
- Time per epoch: 31.85 mins
- Total time: 4.5 days



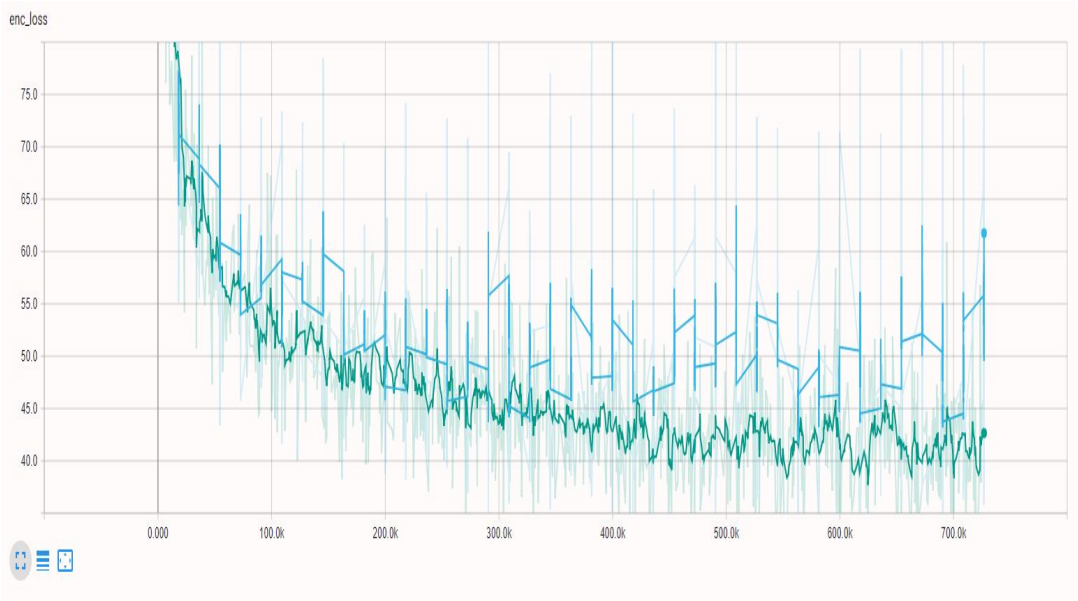
VAE 2.0 Training

Training Parameters

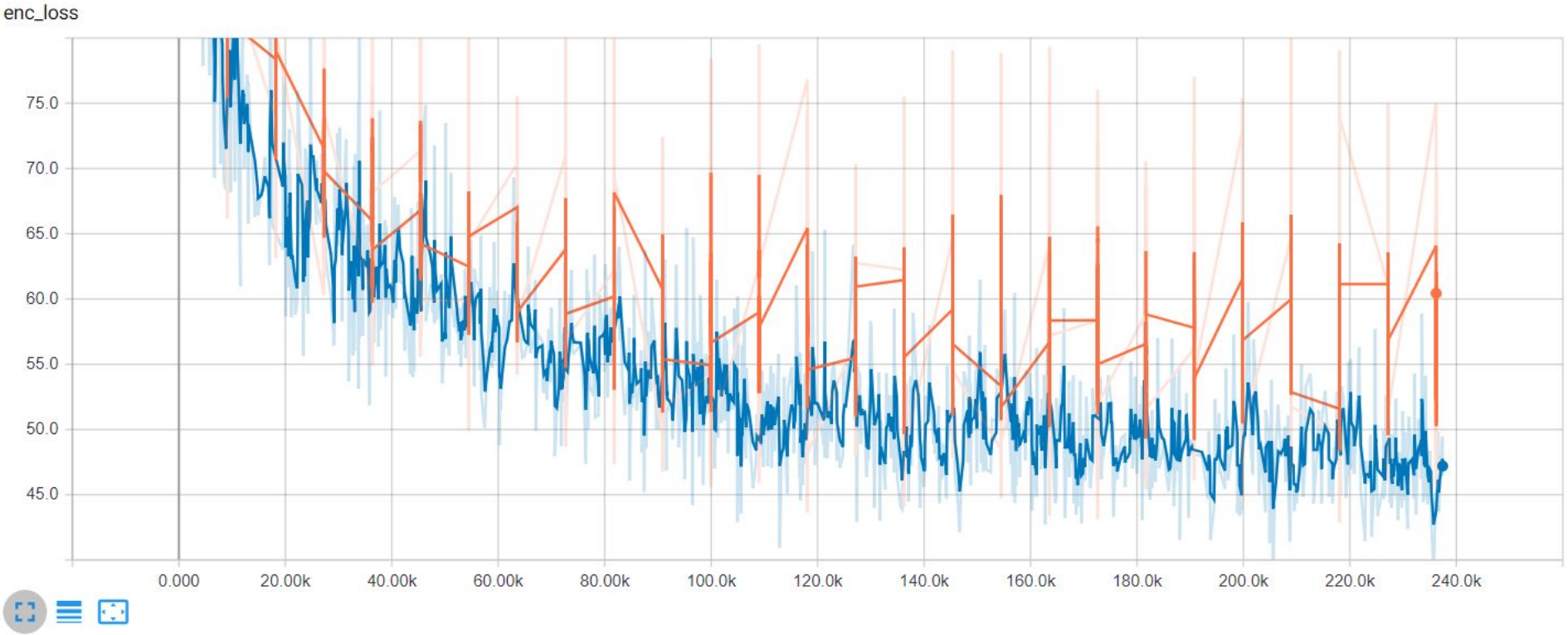
- Epochs: 200
- Batch size: 128 -> 32
- Optimizer: Adam
- Learning rate: 0.0001

Training Time

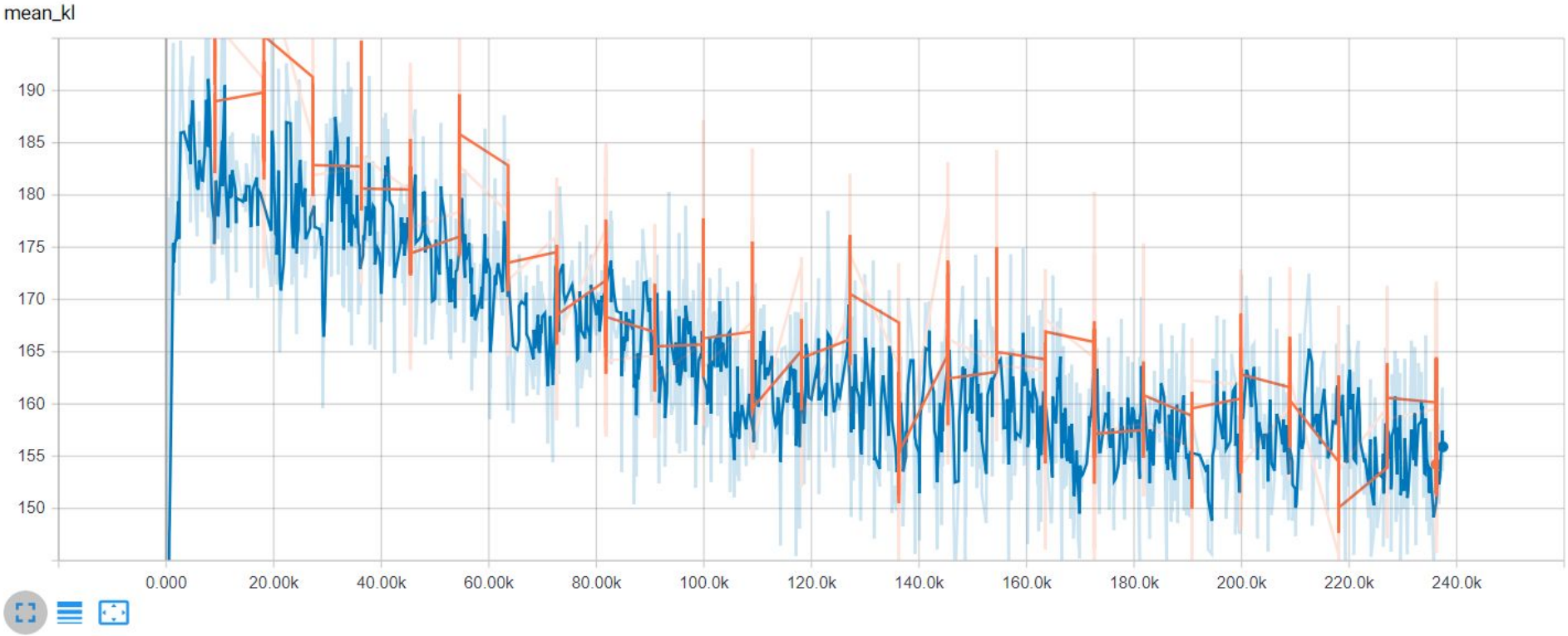
- Time per batch: 0.01 mins
- Time per epoch: 30.95 mins
- Total time: 4.29 days



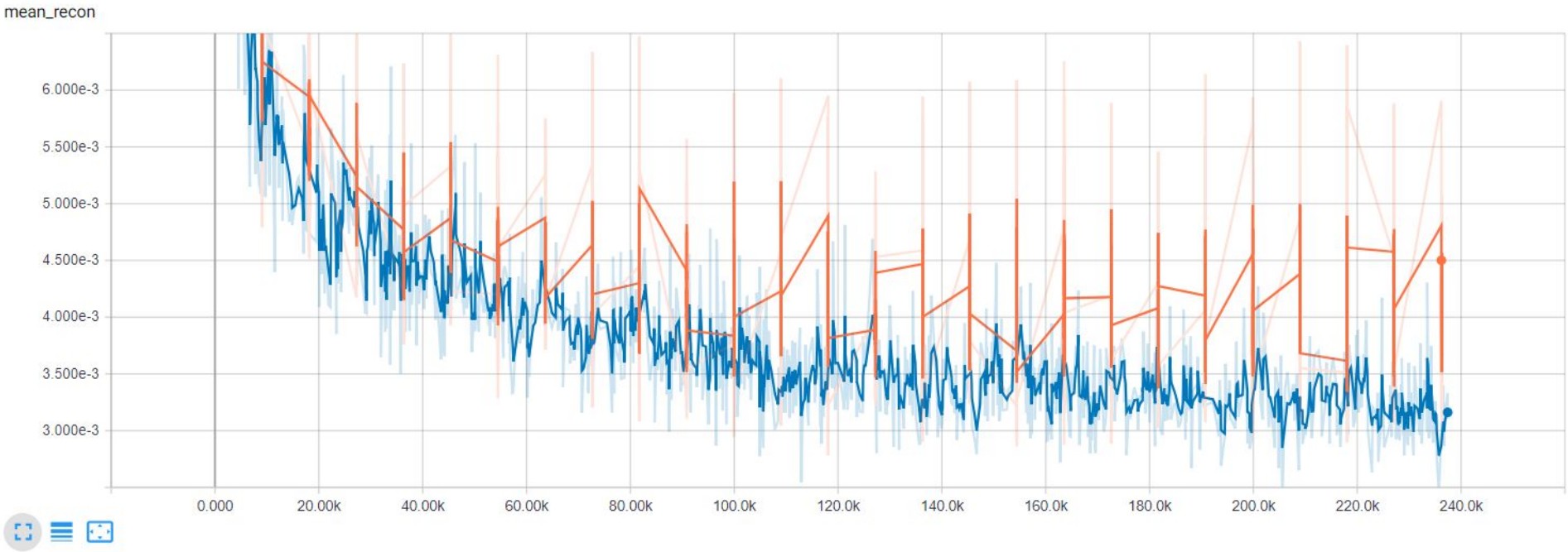
VAE 2.0 Encoder Loss



VAE 2.0 KL Divergence

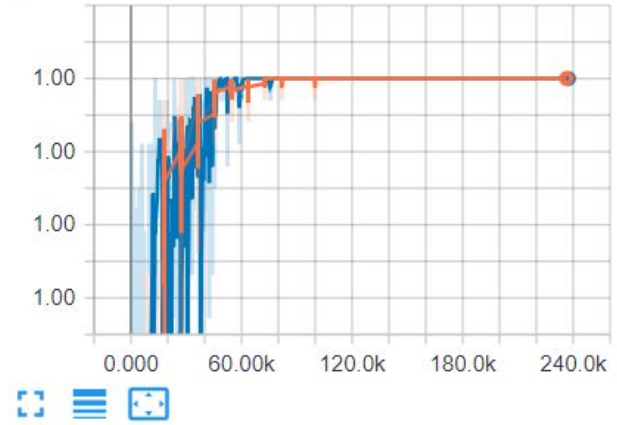


VAE 2.0 Reconstruction Loss

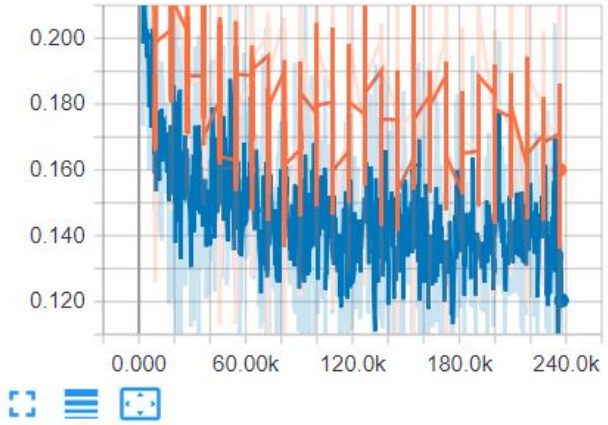


VAE 2.0 Decoder Metrics

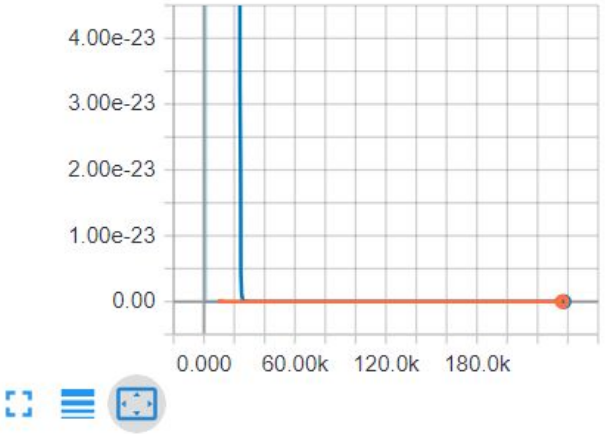
decoder_vae/decoder/decoder_vae/max_decoded_output



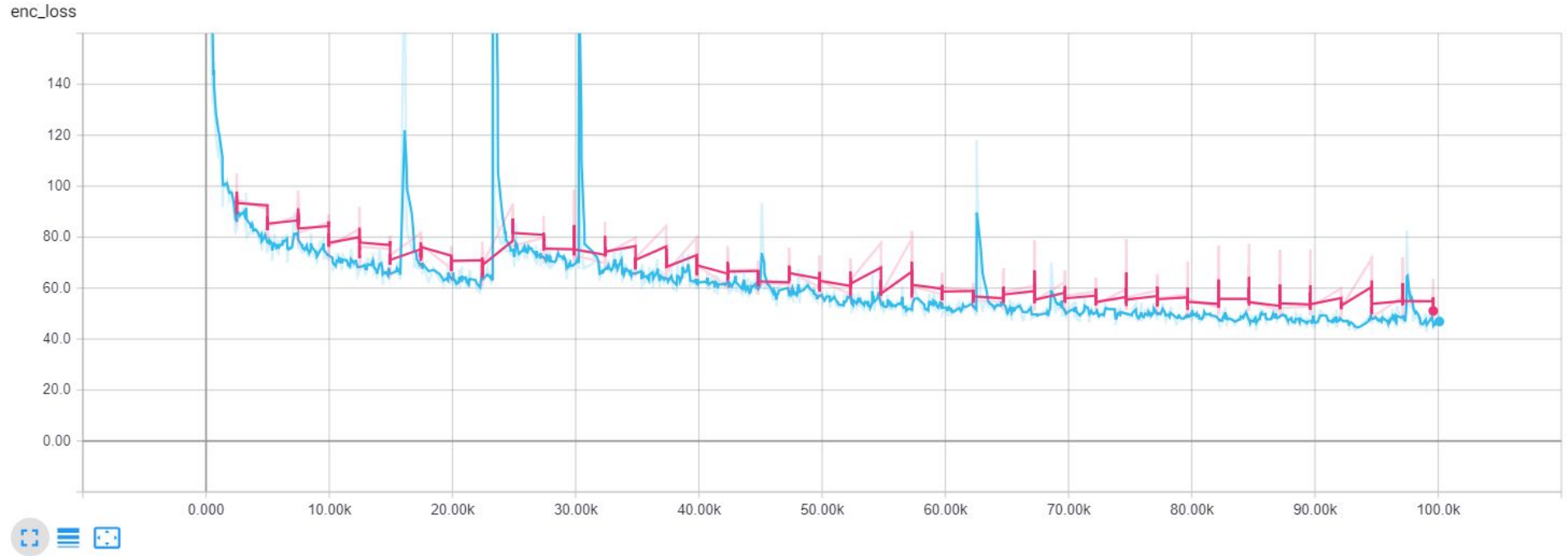
decoder_vae/decoder/decoder_vae/mean_decoded_output



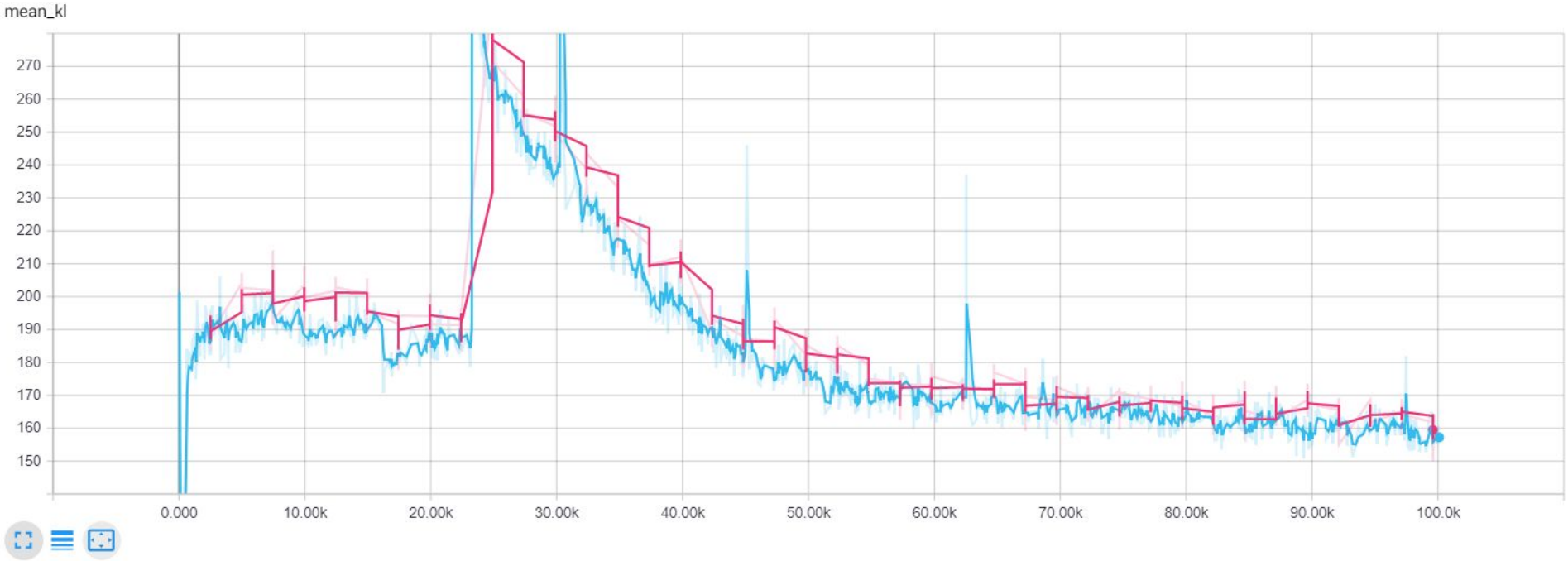
decoder_vae/decoder/decoder_vae/min_decoded_output



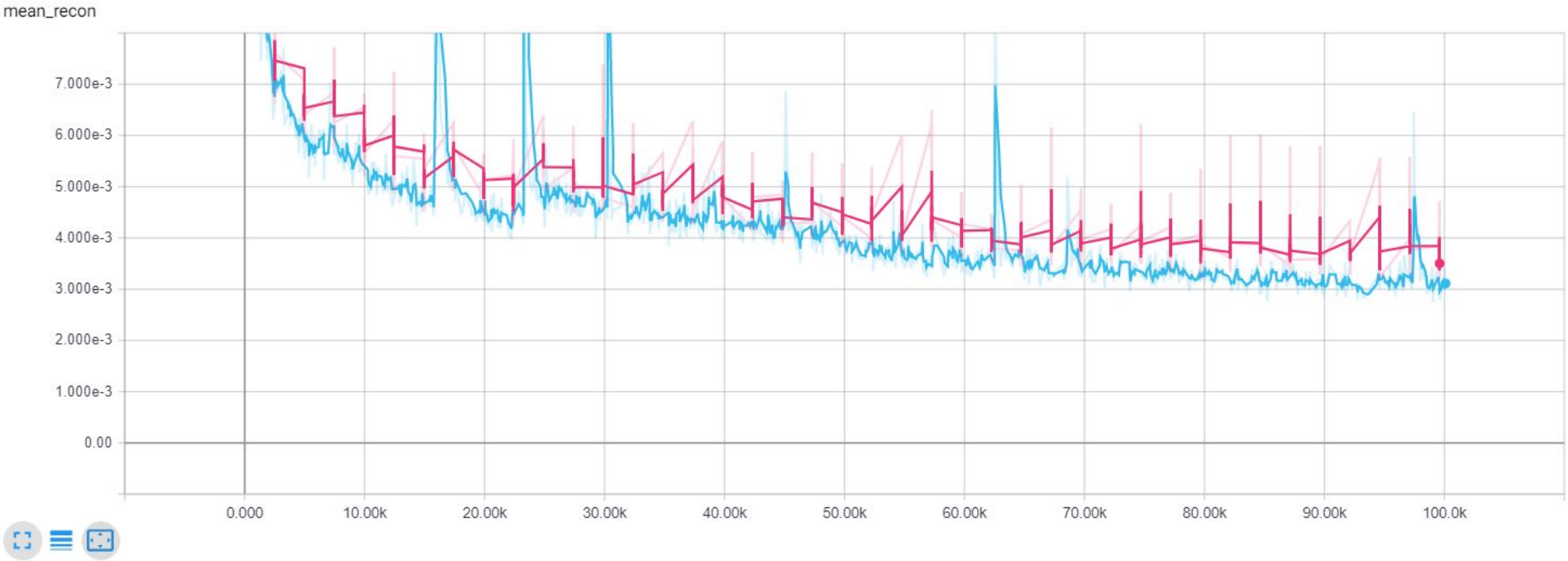
VAE 1.0 Encoder Loss



VAE 1.0 KL Divergence



VAE 1.0 Reconstruction Loss



VAE 1.0 Decoder Metrics