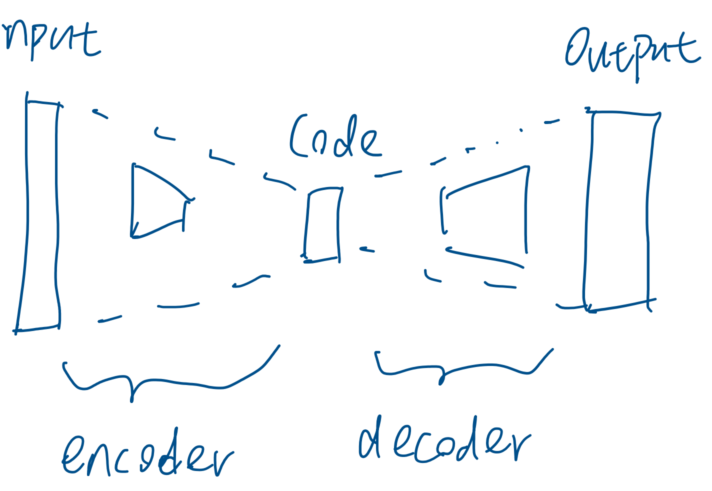
Notes on DeepSDF

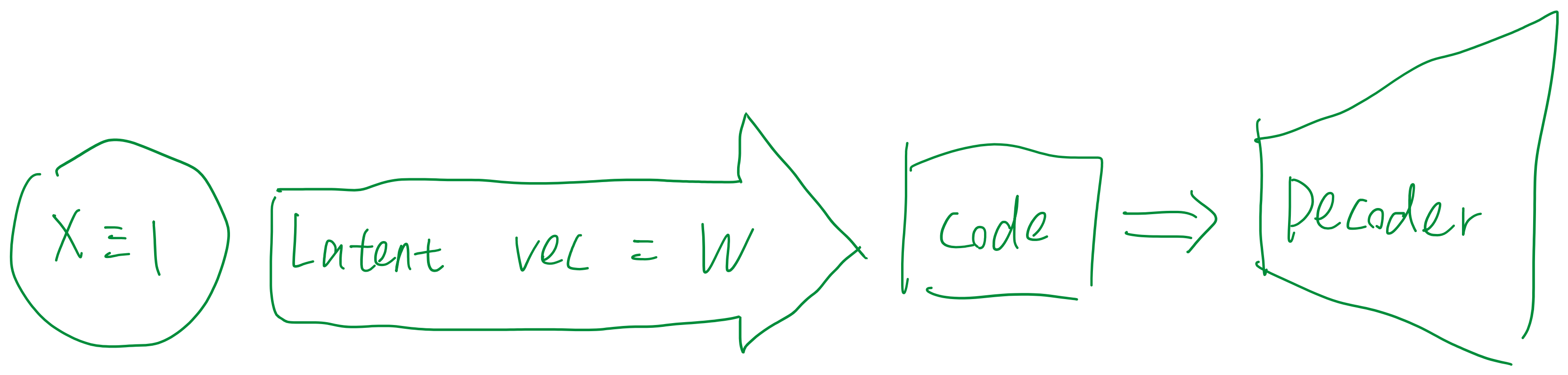
Monday, December 28, 2020 10:30 AM

Autodecoder

Standard autoencoder



Autodecoder



$X \equiv 1$ is at layer 0 } the weights of layer 0 to Code layer W_0
Code layer is linear } is the latent vector

- layer 0 always has 1 as input.
- Code layer and (x, y, z) are fed to the decoder.
- decoder outputs SDF

Training:

- for each Model i, initialize an independent latent vector V_i
- for each $((x,y,z), sdf)^i$ sample, use vector V_i as W_0: $W^i_{(0 \to Code)} = V_i^T$
- train $W^i_{(0 \to code)}$ + decoder

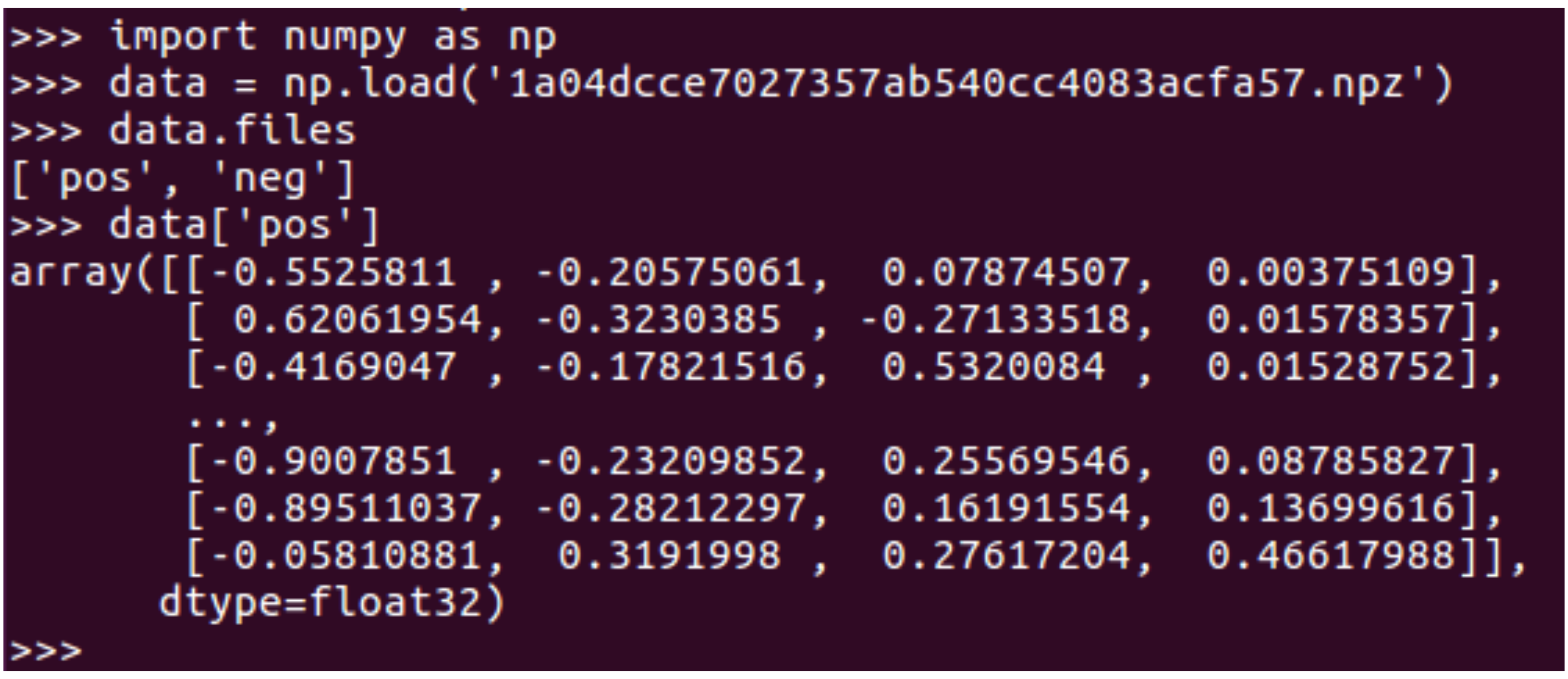
- after training, each Sample trains for its own latent vector, and the decoder learns the sdf at (x, y, z), for each latent vector.

During inference, decoder is fixed, i.e. $(v, (x, y, z)) \to$ sdf value the same, only train new $V_t^T = W^t_{(0 \to Code)}$, t is new data.

DeepSDF is using the finite SDF input, to provide an infinite SDF function.

Notes on the DeepSDF code.

- preprocess_data.py generates the SDFs for training, from the ShapeNet V2 dataset.
- generated SDF is in npz file, used by numpy.
  - npz has "pos" and "neg", are the sampled points inside/outside the mesh.
  - each data has: x, y, z, sdf value.



```
>>> import numpy as np
>>> data = np.load('1a04dcce7027357ab540cc4083acfa57.npz')
>>> data.files
['pos', 'neg']
>>> data['pos']
array([[-0.5525811 , -0.20575061,  0.07874507,  0.00375109],
       [ 0.62061954, -0.3230385 , -0.27133518,  0.01578357],
       [-0.4169047 , -0.17821516,  0.5320084 ,  0.01528752],
       ...,
       [-0.9007851 , -0.23209852,  0.25569546,  0.08785827],
       [-0.89511037, -0.28212297,  0.16191554,  0.13699616],
       [-0.05810881,  0.3191998 ,  0.27617204,  0.46617988]],
      dtype=float32)
>>>
```