OpticSiren status

Ka Ming Tsui

kaming.tsui@ipmu.jp

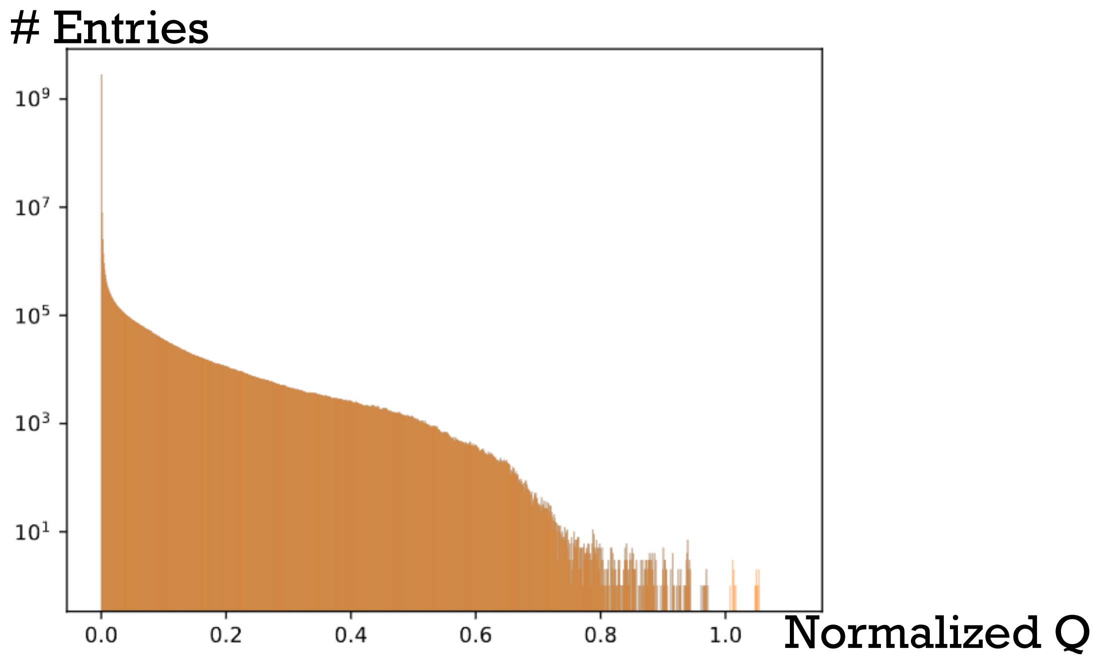
Project name: first_wcte_posdir_500000
Cylinder geometry
R: 0.0 => 1271.96
Z: -1379.118 => 1379.118
Bin in Direction: 1
Gap space: 200.0
Gap angle: 10.0
Starting n phi: 4
Sampling points: 2268
Sampling directions: 684
Sampling configs: 1551312
Photons per config: 500000

Production summary

- Total number of configs (voxels): 1551312
- Storage
  - /sdf/data/neutrino/deperio/wcprod/first_wcte_posdir_500000/
  - Total size ~ 7.4 TB
  - h5 size ~ 141 GB
- Further reduction of h5
  - Convert to fixed length array (nPMT=1843) of charge and time per voxel
  - Size ~ 22 GB

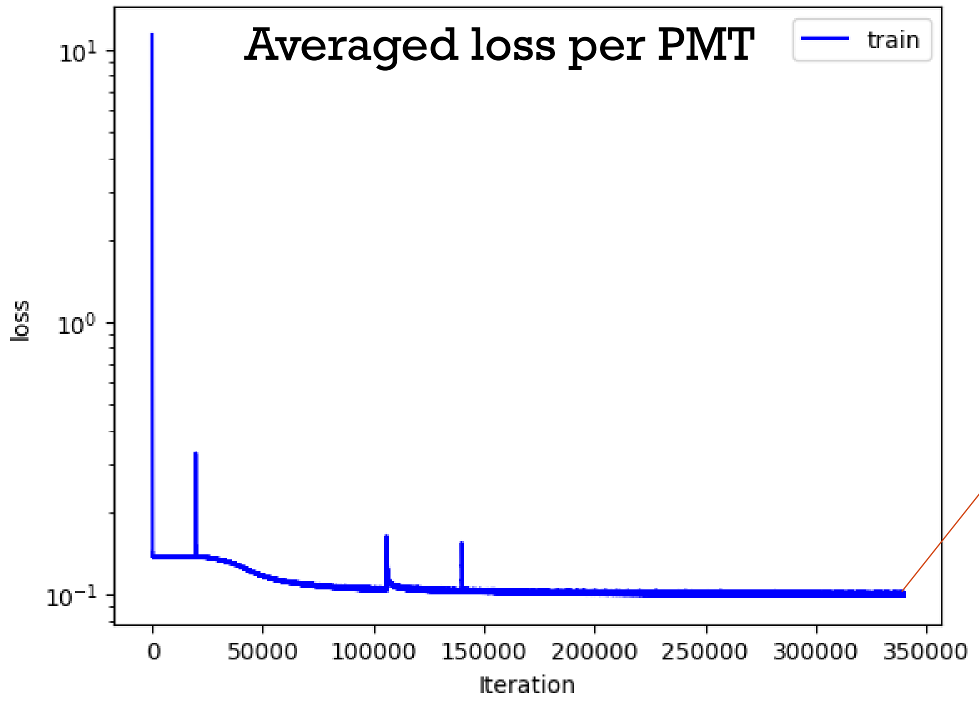
Training setup

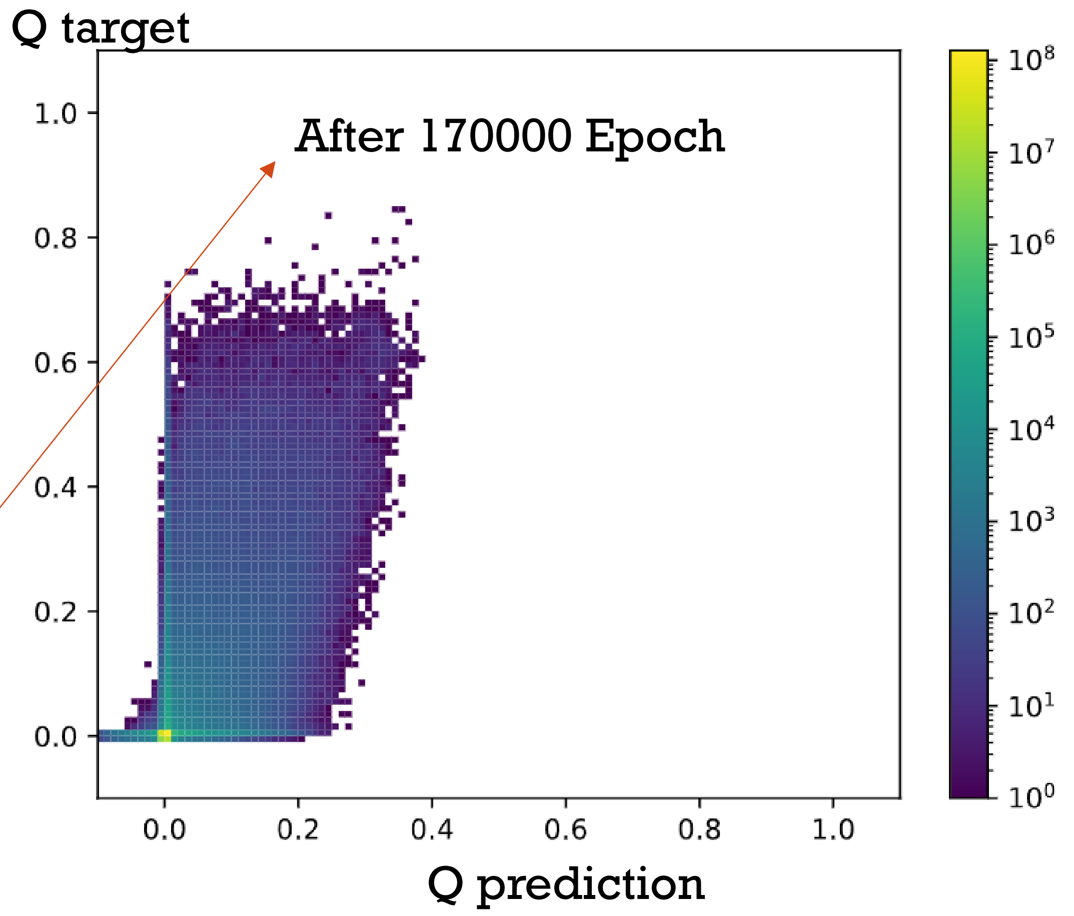
- Q linear normalization: [0,100000] → [0,1]
- Model architecture
  - `hidden_features: 64`
  - `hidden_layers: 5`
  - `in_features: 5`
  - `out_features: 1843` (T is omitted for now)
- `optimizer:`
  - `name: Adam`
  - `parameters:`
    - `lr: 5.0e-05`
- To speed up training
  - Load all data (Q) into GPU
  - 1 epoch = 8 batches



Training

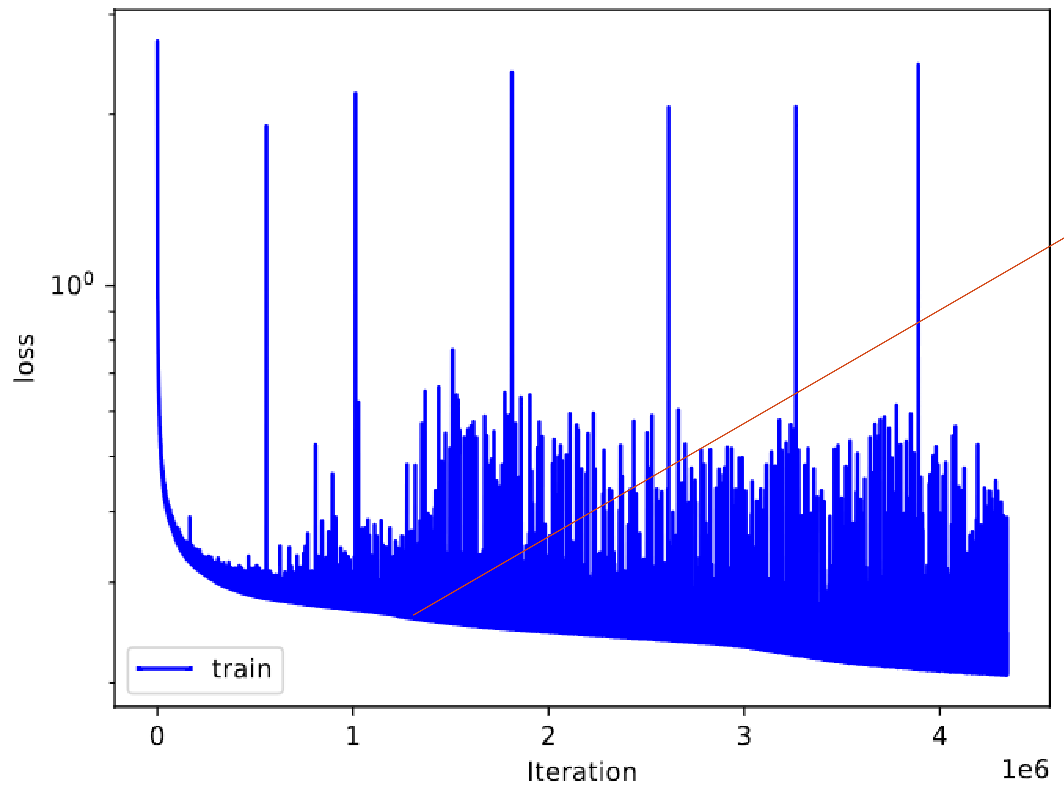
- Loss function: MeanSquared (no weight)
- Average speed ~ 4.43 epoch/s

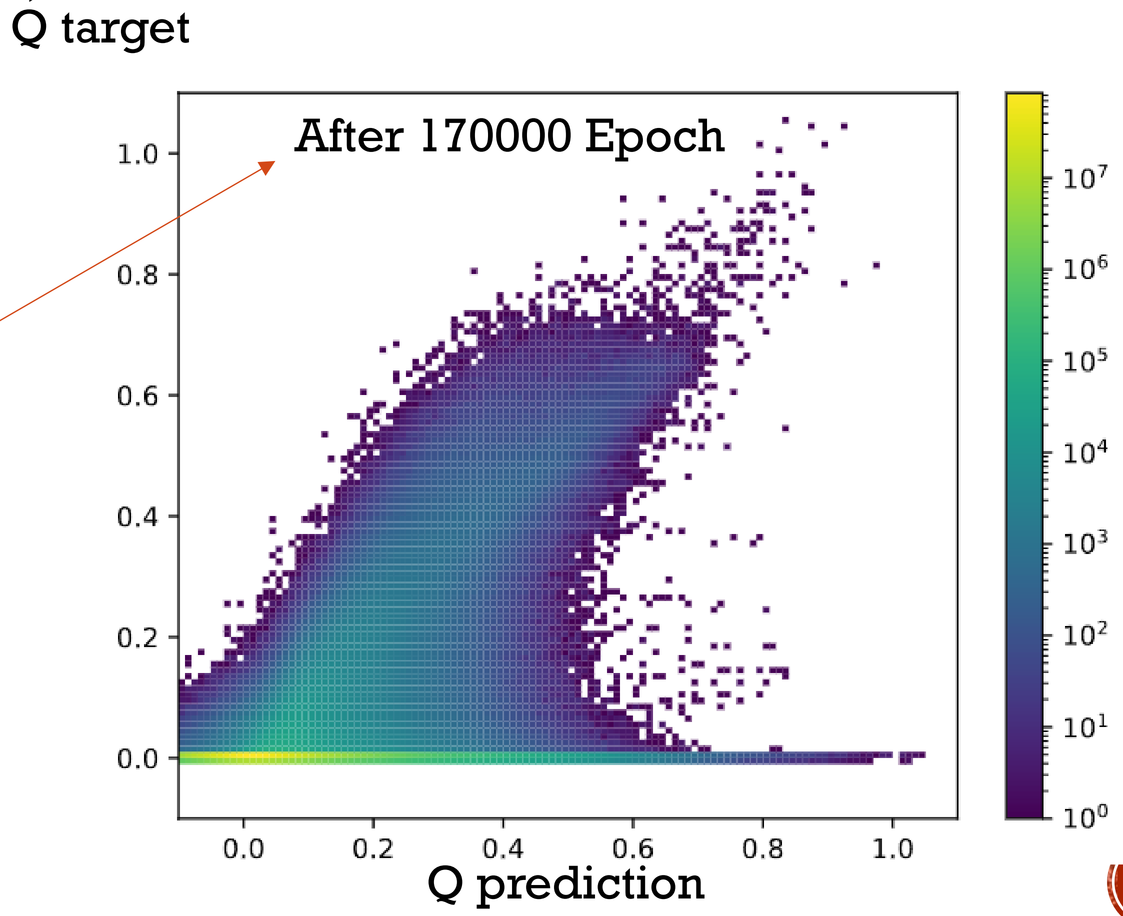




Training

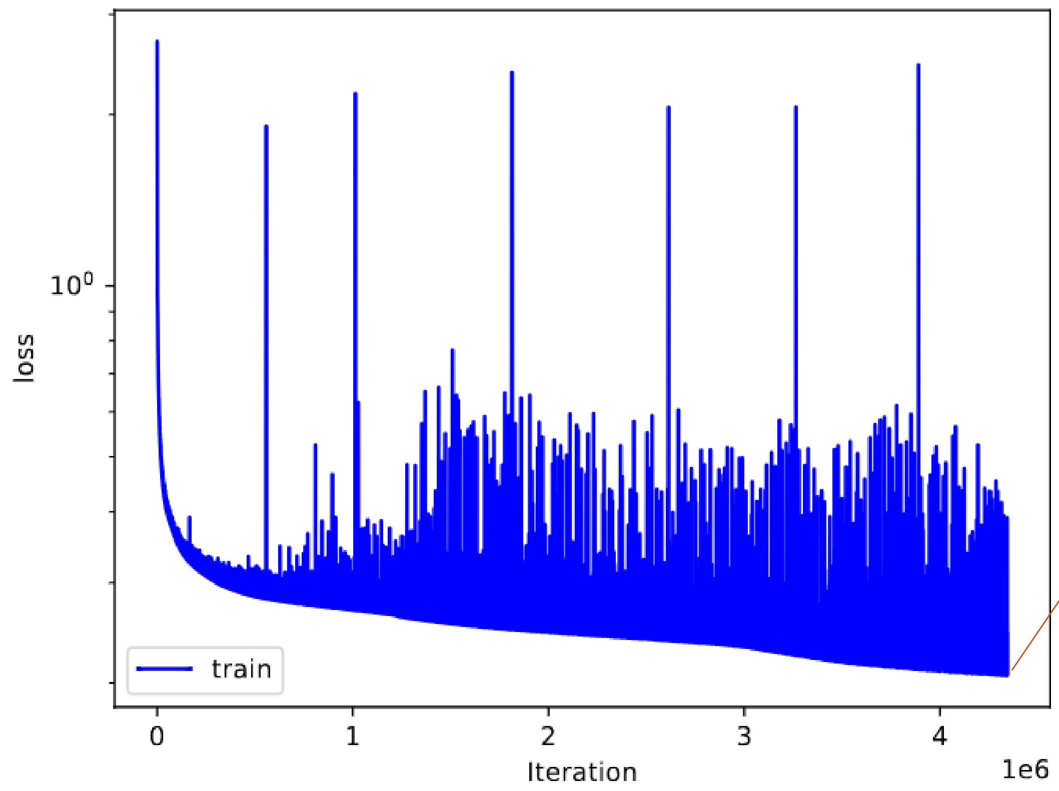
- Loss function: MeanSquared (weight = Q target)
- Average speed ~ 3 epoch/s

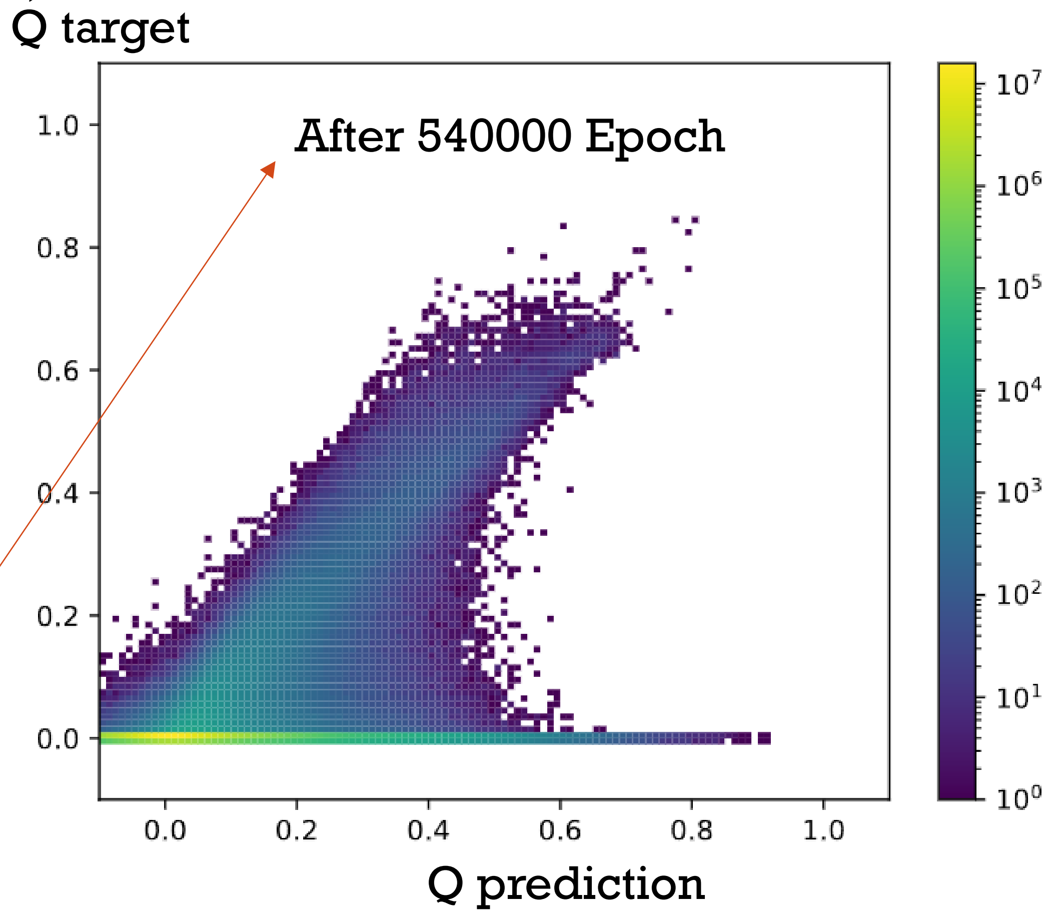




Training

- Loss function: MeanSquared (weight = Q target)
- Average speed ~ 3 epoch/s

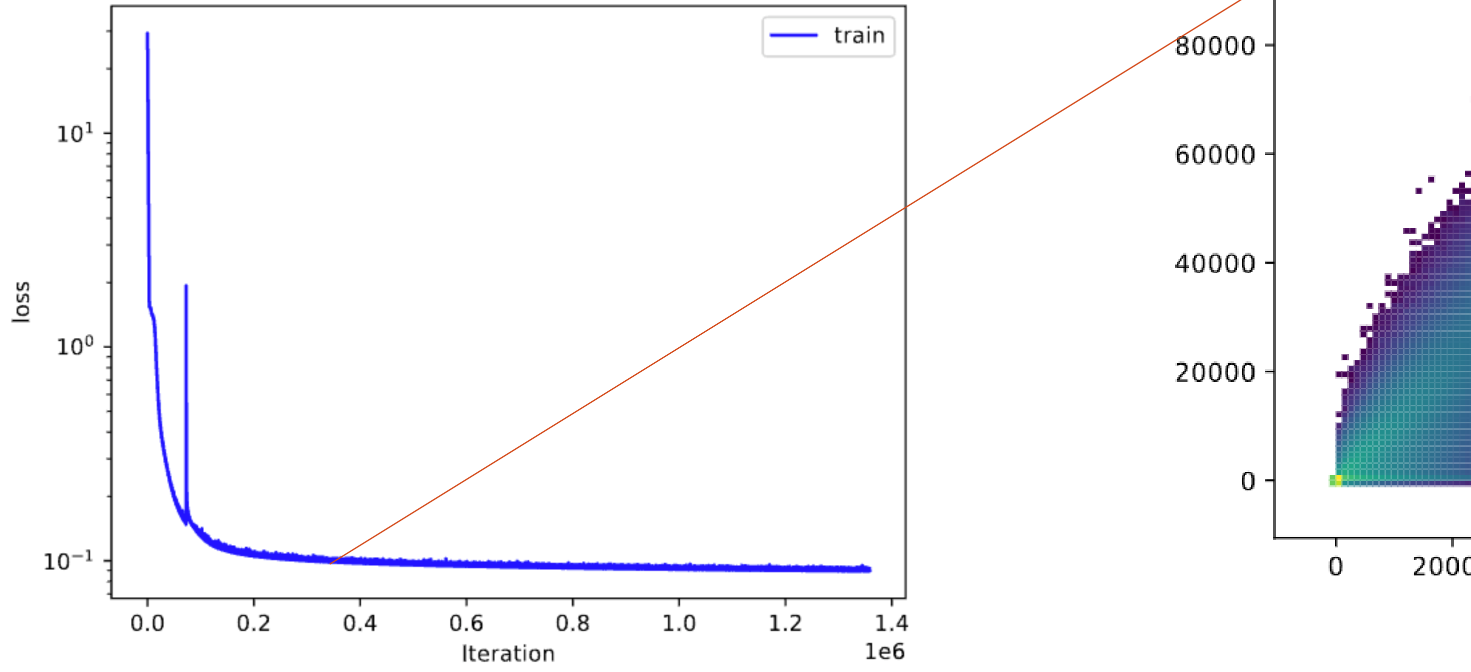




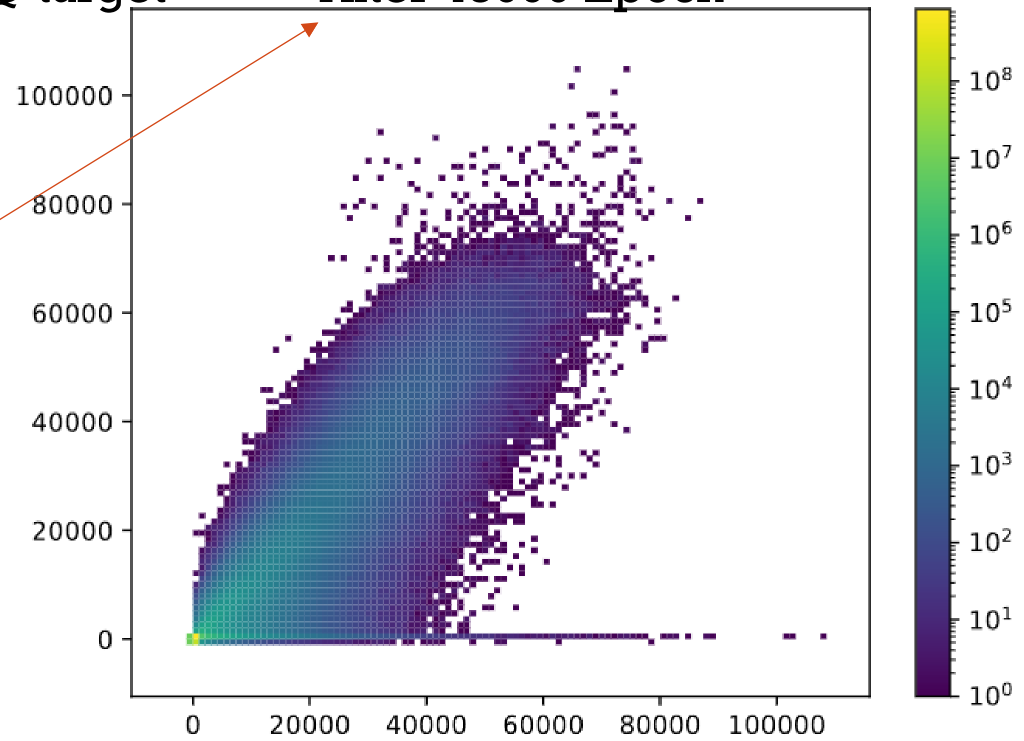
Training revised

```
if self.Q_norm == 'logNorm’:
    eps = 0.1
    y0 = np.log10(eps)
    y1 = np.log10(self.Qmax+eps)
    digi_nph = (torch.log10(digi_nph+eps)-y0)/(y1-y0)
```

- Q logNorm transformation
- Proper direction normalization
  - θ: [0,180]→[0,1]
  - φ: [0,360]→[0,1]



Q target After 43000 Epoch

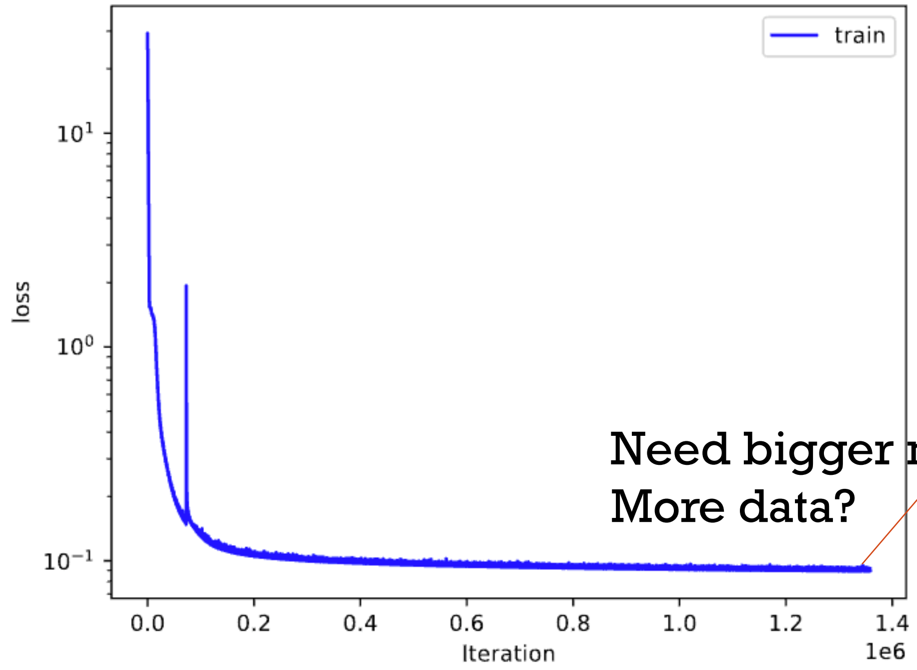


Q prediction

Training revised

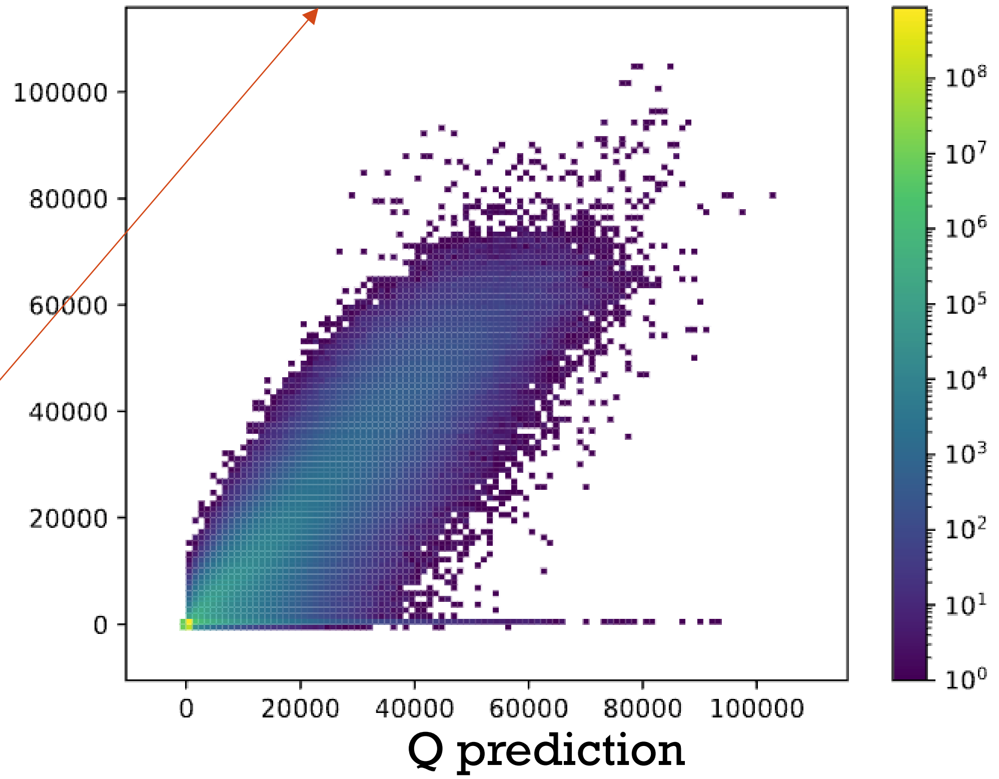
```
if self.Q_norm == 'logNorm':
        eps = 0.1
        y0 = np.log10(eps)
        y1 = np.log10(self.Qmax+eps)
        digi_nph = (torch.log10(digi_nph+eps)-y0)/(y1-y0)
```

- Q logNorm transformation
- Proper direction normalization
  - θ: [0,180]→[0,1]
  - φ: [0,360]→[0,1]



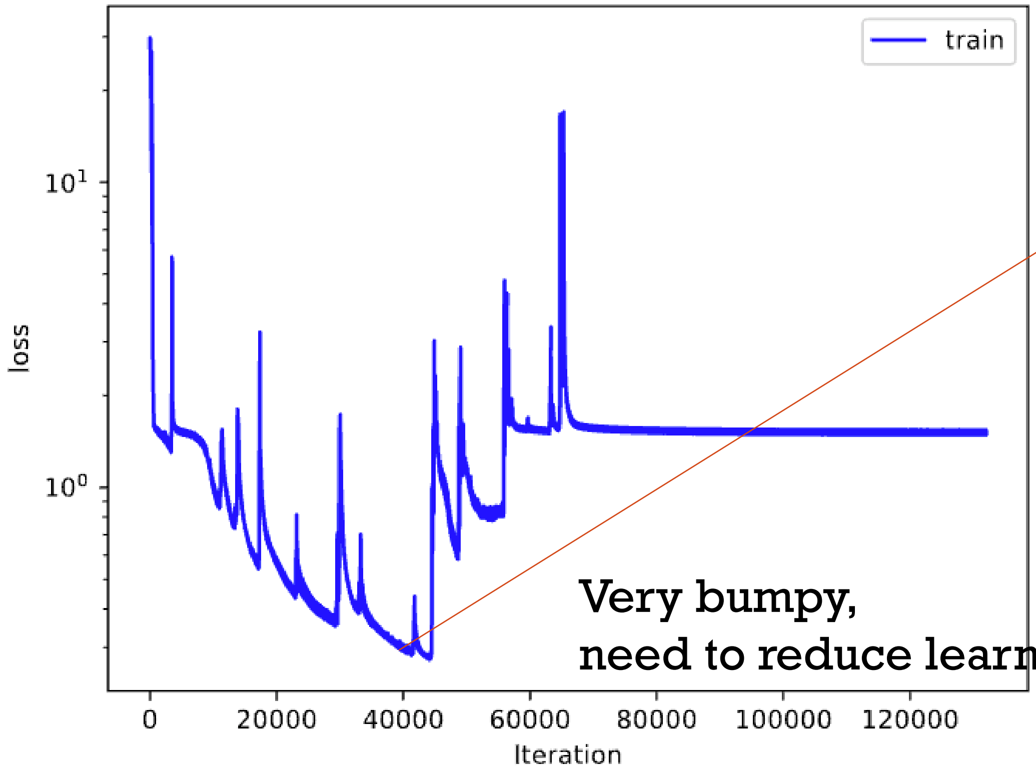
Need bigger network?
More data?

Q target After 169000 Epoch



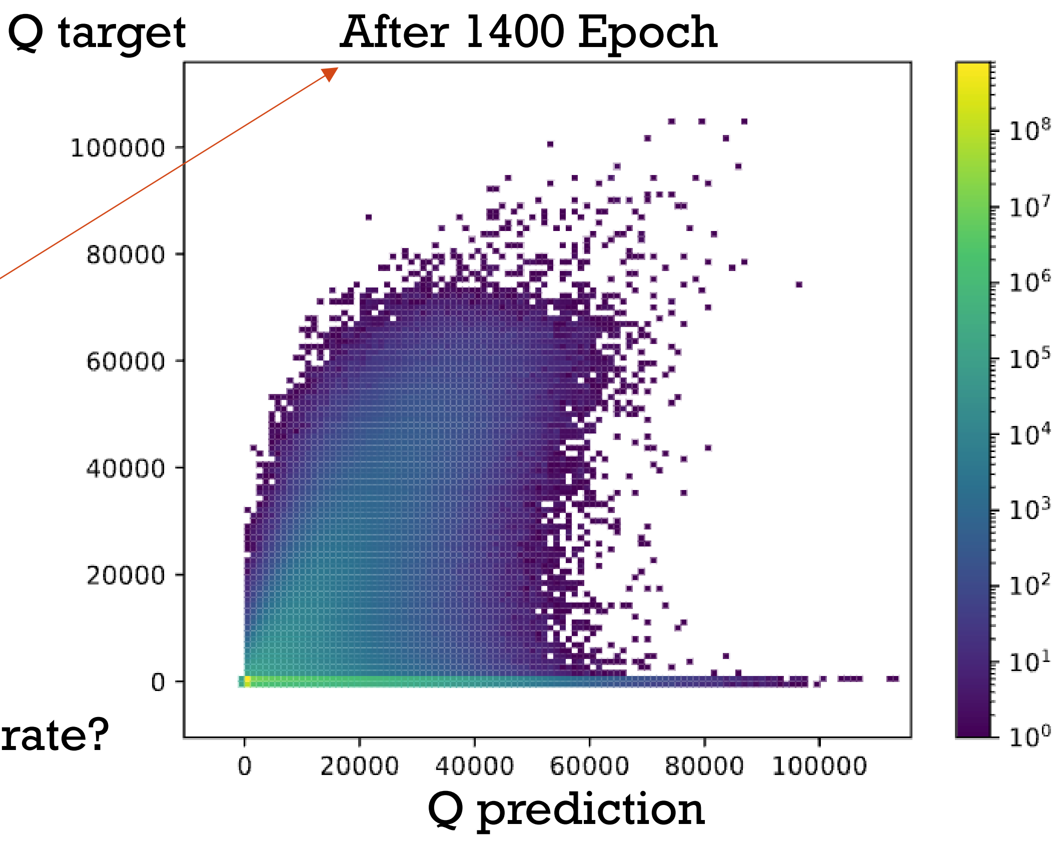
Extreme test

- Hidden features * layers: 64 *5 → 2048 * 15
- Much longer time: 3 epoch/s → 0.14 epoch/s



Very bumpy,
need to reduce learning rate?

After 1400 Epoch



Any advice

- Large network architure?
- Learning rate scheduler?
- More data?