CMPT 733 Assignment 4 PointNet

301474102 Sihui Wang

1. Environment Configurations

1.1 Training:

I uploaded '.py' files to google drive and trained the models by running python scripts in Colab.

The following is the link for my workspace on Google Drive:

https://drive.google.com/drive/folders/1U3ojSJp_XSnqqPz9O5wHA5CgcpO3cdG3?usp=sharing

I trained 6 models in total:

1) Classification Network with Feature Transformation:

The Notebook showing the training process:

https://colab.research.google.com/drive/1FjOhSOztuWnt8hHVKg5FechXbFR2pbsd?usp=sharing

Saved Models (from epoch 1 to epoch 250):

https://drive.google.com/drive/folders/1GKIoAYmli_POBlARSIys49SA3qiPfks0?usp=sharing

2) Classification Network without Feature Transformation:

The Notebook showing the training process:

https://colab.research.google.com/drive/1aHgOg6aq_M8HRdfYTWR_AjCBAZOfwjLe?usp=sharing

Saved Models (from epoch 1 to epoch 250):

https://drive.google.com/drive/folders/1WNMLuAuL9hNGsqa1h8jRjTAzbeLS3-8F?usp=sharing

3) Segmentation Network for Chairs with Feature Transformation:

The Notebook showing the training process:

https://colab.research.google.com/drive/1kwGVVqXNd2aXZcTrrDLoZu2sysOpRLCP?usp=sharing

Saved Models (from epoch 1 to epoch 25):

https://drive.google.com/drive/folders/1S7MfBKfZLhPdMibxQzaF00MGixSDWJDa?usp=sharing

4) Segmentation Network for Chairs without Feature Transformation:

The Notebook showing the training process:

https://colab.research.google.com/drive/1-YrVgRn7t8DQgLZMieGFTz5yc5l9Prrn?usp=sharing

Saved Models (from epoch 1 to epoch 25):

https://drive.google.com/drive/folders/19-tfmmth3lYbzPEPNkPMSTyfeLmhbYGF?usp=sharing

5) Segmentation Network for Airplanes with Feature Transformation:

The Notebook showing the training process:

https://colab.research.google.com/drive/1WpB4lwu8FhCUEVGqazXvTbZgG3FnN2Jn?usp=sharing

Saved Models (from epoch 1 to epoch 25):

https://drive.google.com/drive/folders/1zw0unPGDjqj8cdsNsk6MmqGy53_ZnBHF?usp=sharing

6) Segmentation Network for Airplanes without Feature Transformation:

The Notebook showing the training process:

https://colab.research.google.com/drive/1lqwNBY_ZfGeEQYNwWVRxiv-675k0L98C?usp=sharing

Saved Models (from epoch 5 to epoch 25, epoch 1~4 were somehow lost):
https://drive.google.com/drive/folders/1PYVKr9dpRT1wsPOnZ7vWcCVDUwEs76lf?usp=sharing

1.2 Testing:

I ran inference locally on my computer with the following configurations:

Operating System: Ubuntu 20.04 LTS

Python: 3.7.11

PyTorch: 1.8.1

CUDA: 9.0.176

2. Classification Networks

2.1 Classification Accuracy:

2.1.1 Classification with Feature Transformation:

Testing Commands:

On my own computer, the folders are organized as follows:

```
framework
  └─codes
  └─pretrained_networks
  └─misc
  └─shapenet_data
```

And the testing command is:

```
python show_cls.py --model ../pretrained_networks/cls_with_trans_epoch250.pt --feature_transform
```

For this submission, the folders are organized differently as required:

```
codes
  └─codes
  └─cls
  └─seg
  └─misc (not uploaded in this submission, needed from evaluator's side)
  └─shapenet_data (not uploaded in this submission, needed from evaluator's side)
```

So, the testing command *for this submission* is:

```
python show_cls.py --model ../cls/cls_with_trans_epoch250.pt --feature_transform
```

In the remaining part of this report, I will only mention the testing commands for this submission.

Classification Accuracy:

Please note that the data loader *randomly* loads 2500 points for each object:
`choice=np.random.choice(len(seg),self.npoints,replace=True)` (`dataset.py` line 102)

So, the test accuracy *can change a little bit each time*. Other results, such as mIOU, critical points, and segmentation, will also show a little variation for each run due to this randomness.

Classification Accuracy on Testing Dataset (with Feature Transformation)

Class Name	Correct/Total	Accuracy	Class Name	Correct/Total	Accuracy
Overall	**2796/2874**	**97.29%**	Airplane	337/341	98.83%
Bag	13/14	92.86%	Cap	7/11	63.64%
Car	154/158	97.47%	Chair	694/704	98.58%

Earphone	11/14	78.57%	Guitar	157/159	98.74%
Knife	79/80	98.75%	Lamp	263/286	91.96%
Laptop	82/83	98.80%	Motorbike	49/51	96.08%
Mug	37/38	97.37%	Pistol	42/44	95.45%
Rocket	11/12	91.67%	Skateboard	30/31	96.77%
Table	830/848	97.88%			

Classification Accuracy on Training Dataset (with Feature Transformation)

Class Name	Correct/Total	Accuracy	Class Name	Correct/Total	Accuracy
Overall	12061/12137	99.37%	Airplane	1956/1958	99.90%
Bag	54/54	100.00%	Cap	39/39	100.00%
Car	656/659	99.54%	Chair	2650/2658	99.70%
Earphone	47/49	95.92%	Guitar	542/550	98.55%
Knife	262/277	94.58%	Lamp	1106/1118	98.93%
Laptop	324/324	100.00%	Motorbike	124/125	99.20%
Mug	130/130	100.00%	Pistol	207/209	99.04%
Rocket	46/46	100.00%	Skateboard	106/106	100.00%
Table	3812/3835	99.40%			

Classification Accuracy on Validation Dataset (with Feature Transformation)

Class Name	Correct/Total	Accuracy	Class Name	Correct/Total	Accuracy
Overall	1833/1870	98.02%	Airplane	391/391	100.00%
Bag	6/8	75.00%	Cap	3/5	60.00%
Car	81/81	100.00%	Chair	393/396	99.24%
Earphone	4/6	66.67%	Guitar	74/78	94.87%
Knife	34/35	97.14%	Lamp	129/143	90.21%
Laptop	44/44	100.00%	Motorbike	24/26	92.31%
Mug	16/16	100.00%	Pistol	30/30	100.00%
Rocket	5/8	62.50%	Skateboard	15/15	100.00%
Table	584/588	99.32%			

2.1.2 Classification without Feature Transformation

Testing Commands:

```
python show_cls.py --model ../cls/cls_no_trans_epoch250.pt
```

Classification Accuracy:

Classification Accuracy on Testing Dataset (no Feature Transformation)

Class Name	Correct/Total	Accuracy	Class Name	Correct/Total	Accuracy
Overall	**2798/2874**	**97.36%**	Airplane	335/341	98.24%
Bag	13/14	92.86%	Cap	9/11	81.82%
Car	155/158	98.10%	Chair	690/704	98.01%
Earphone	14/14	100.00%	Guitar	157/159	98.74%
Knife	80/80	100.00%	Lamp	269/286	94.06%
Laptop	82/83	98.80%	Motorbike	49/51	96.08%
Mug	38/38	100.00%	Pistol	42/44	95.45%
Rocket	10/12	83.33%	Skateboard	27/31	87.10%
Table	828/848	97.64%			

Classification Accuracy on Training Dataset (no Feature Transformation)

Class Name	Correct/Total	Accuracy	Class Name	Correct/Total	Accuracy
Overall	12100/12137	99.70%	Airplane	1956/1958	99.90%
Bag	53/54	98.15%	Cap	39/39	100.00%
Car	39/39	100.00%	Chair	2649/2658	99.66%
Earphone	49/49	100.00%	Guitar	547/550	99.45%
Knife	275/277	99.28%	Lamp	1112/1118	99.46%
Laptop	324/324	100.00%	Motorbike	125/125	100.00%
Mug	130/130	100.00%	Pistol	209/209	100.00%
Rocket	45/46	97.83%	Skateboard	106/106	100.00%
Table	3822/3835	99.66%			

Classification Accuracy on Validation Dataset (no Feature Transformation)

Class Name	Correct/Total	Accuracy	Class Name	Correct/Total	Accuracy
Overall	1839/1870	98.34%	Airplane	391/391	100.00%
Bag	6/8	75.00%	Cap	3/5	60.00%
Car	81/81	100.00%	Chair	393/396	99.24%
Earphone	5/6	83.33%	Guitar	74/78	94.87%
Knife	35/35	100.00%	Lamp	129/143	90.21%
Laptop	44/44	100.00%	Motorbike	26/26	100.00%
Mug	16/16	100.00%	Pistol	30/30	100.00%
Rocket	7/8	87.50%	Skateboard	15/15	100.00%
Table	584/588	99.32%			

2.2 Visualization of Critical Points

2.2.1 Critical Points with Feature Transformation

Instance 1 (Pistol, with Feature Transformation):

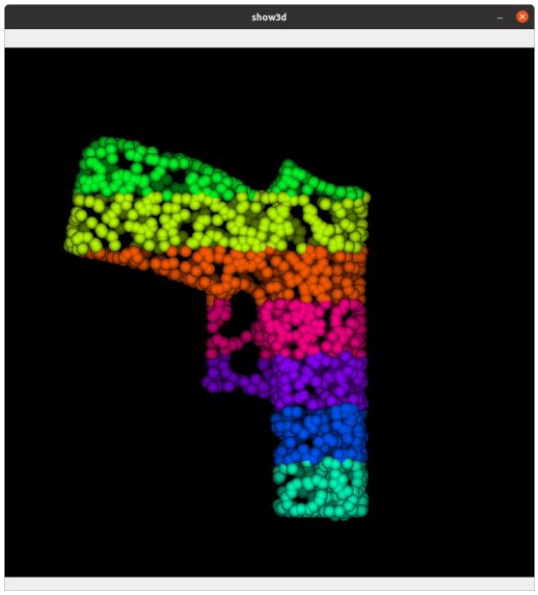
Testing Commands:

`python critical_points.py --model ../cls/cls_with_trans_epoch250.pt --idx 10 --class_choice Pistol --feature_transform` (show original points)

`python critical_points.py --model ../cls/cls_with_trans_epoch250.pt --idx 10 --class_choice Pistol --feature_transform --critical_point` (show critical points)

Visualization:

Original Points **Critical Points**



Instance 2 (Table, with Feature Transformation):

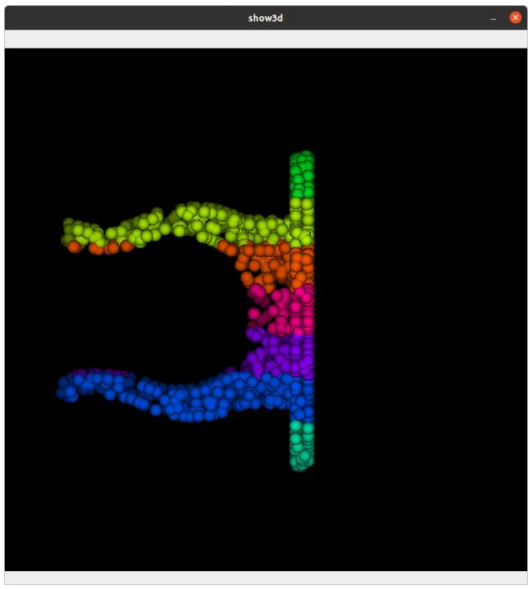
Testing Commands:

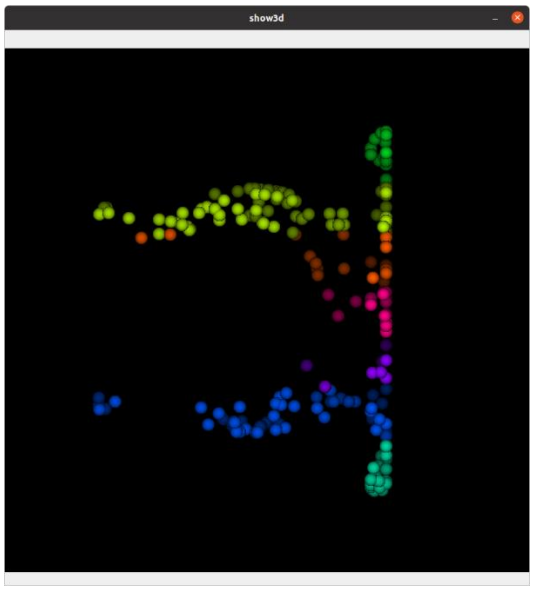
`python critical_points.py --model ../cls/cls_with_trans_epoch250.pt --idx 3 --class_choice Table --feature_transform` (show original points)

`python critical_points.py --model ../cls/cls_with_trans_epoch250.pt --idx 3 --class_choice Table --feature_transform --critical_point` (show critical points)

Visualization:

Original Points **Critical Points**





Instance 3 (Lamp, with Feature Transformation):

Testing Commands:

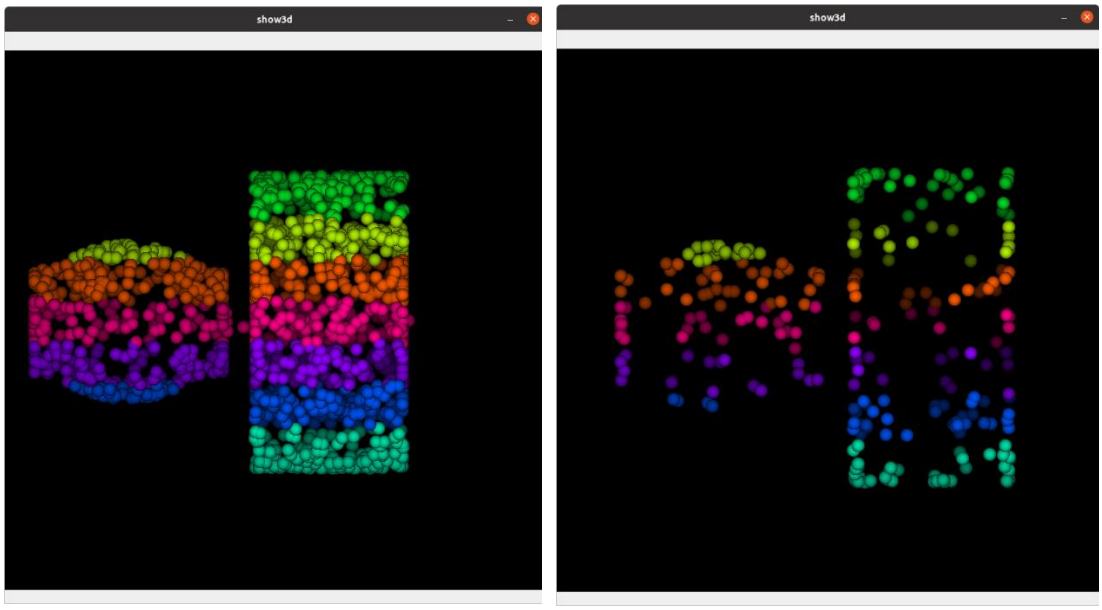
`python critical_points.py --model ../cls/cls_with_trans_epoch250.pt --idx 8 --class_choice Lamp --feature_transform` (show original points)

`python critical_points.py --model ../cls/cls_with_trans_epoch250.pt --idx 8 --`

`class_choice Lamp --feature_transform --critical_point` (show critical points)

Visualization:

Original Points **Critical Points**



Instance 4 (Motorbike, with Feature Transformation):

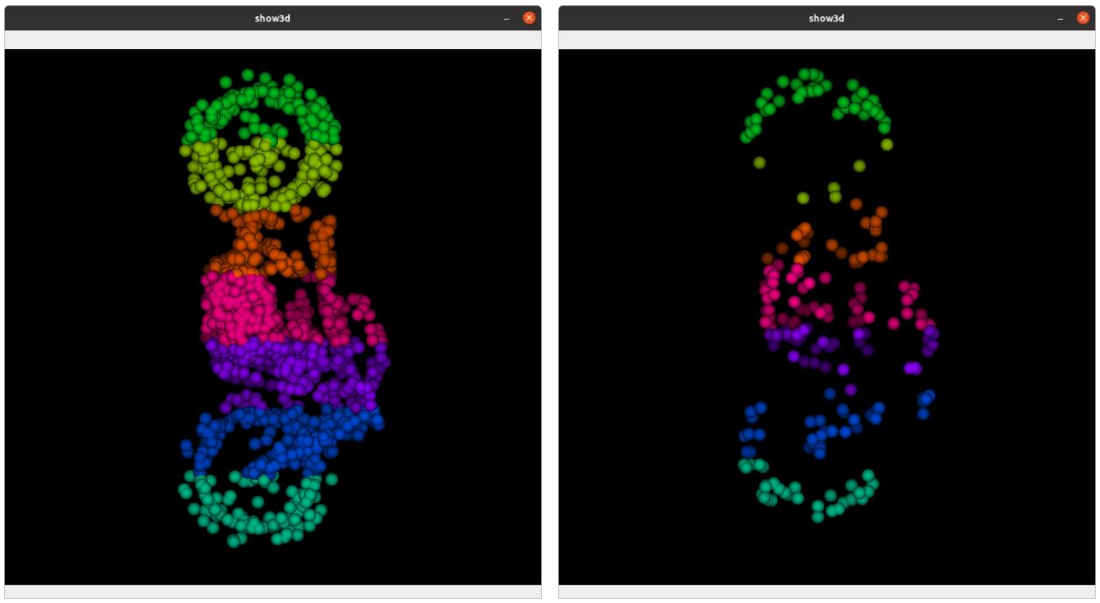
Testing Commands:

`python critical_points.py --model ../cls/cls_with_trans_epoch250.pt --idx 5 --class_choice Motorbike --feature_transform` (show original points)

`python critical_points.py --model ../cls/cls_with_trans_epoch250.pt --idx 5 --class_choice Motorbike --feature_transform --critical_point` (show critical points)

Visualization:

Original Points **Critical Points**



2.2.2 Critical Points without Feature Transformation

Instance 1 (Pistol, no Feature Transformation):

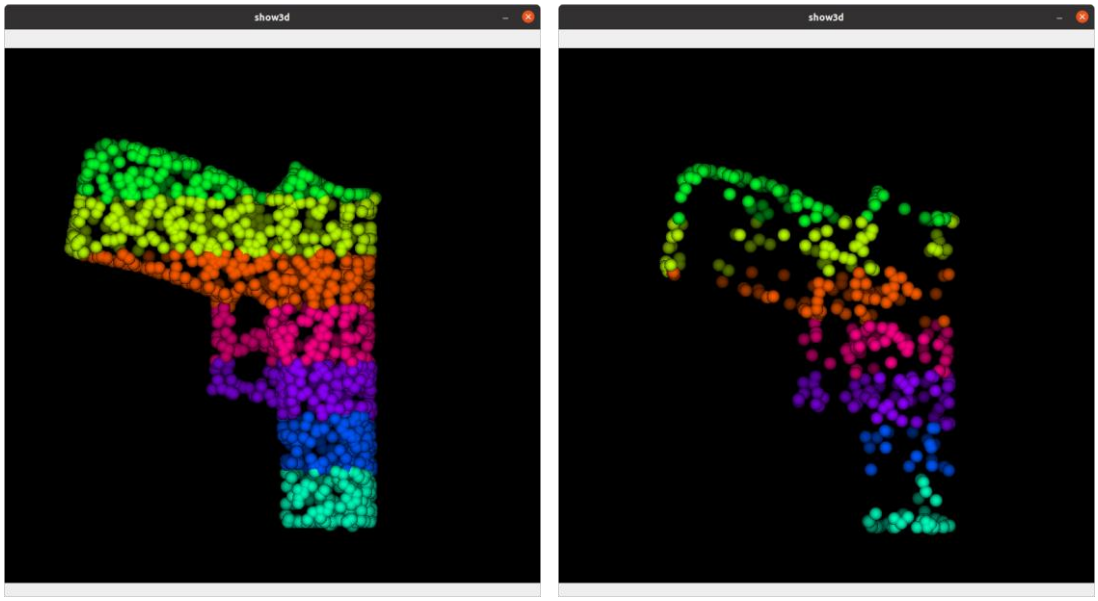
Testing Commands:

`python critical_points.py --model ../cls/cls_no_trans_epoch250.pt --idx 10 --class_choice Pistol` (show original points)

`python critical_points.py --model ../cls/cls_no_trans_epoch250.pt --idx 10 --class_choice Pistol --critical_point` (show critical points)

Visualization:

Original Points **Critical Points**



Instance 2 (Table, no Feature Transformation):

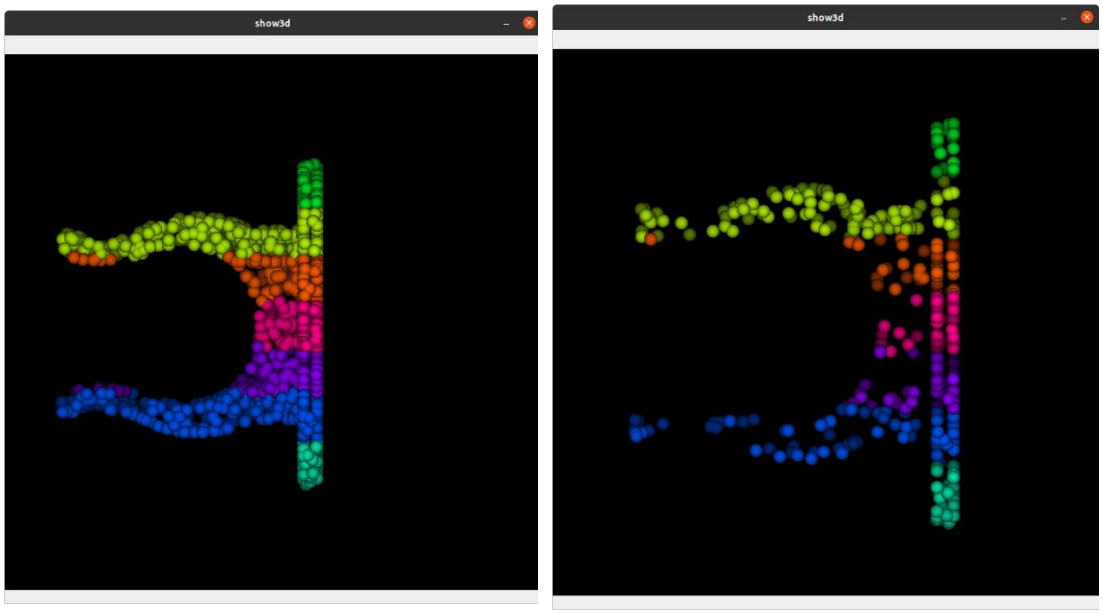
Testing Commands:

`python critical_points.py --model ../cls/cls_no_trans_epoch250.pt --idx 3 --class_choice Table` (show original points)

`python critical_points.py --model ../cls/cls_no_trans_epoch250.pt --idx 3 --class_choice Table --critical_point` (show critical points)

Visualization:

Original Points **Critical Points**



Instance 3 (Lamp, no Feature Transformation):

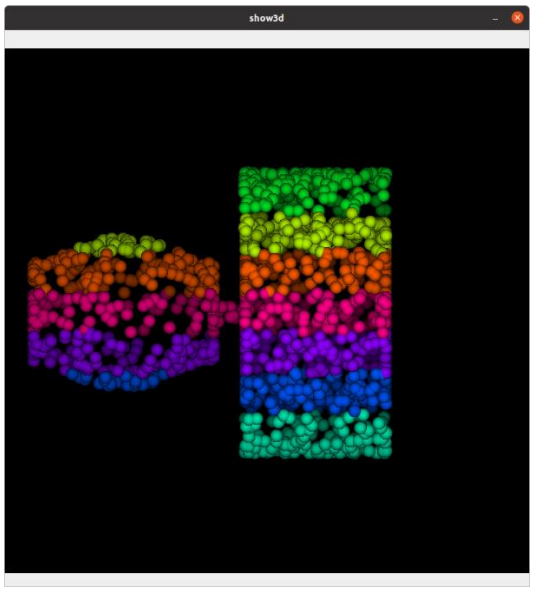
Testing Commands:

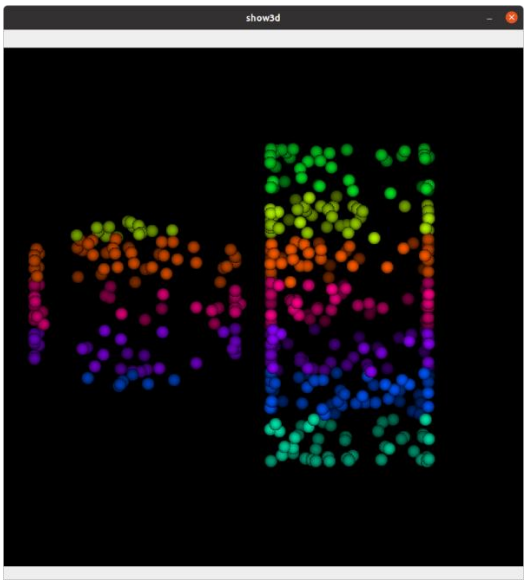
`python critical_points.py --model ../cls/cls_no_trans_epoch250.pt --idx 8 --class_choice Lamp` (show original points)

`python critical_points.py --model ../cls/cls_no_trans_epoch250.pt --idx 8 --class_choice Lamp --critical_point` (show critical points)

Visualization:

Original Points **Critical Points**





Instance 4 (Motorbike, no Feature Transformation):

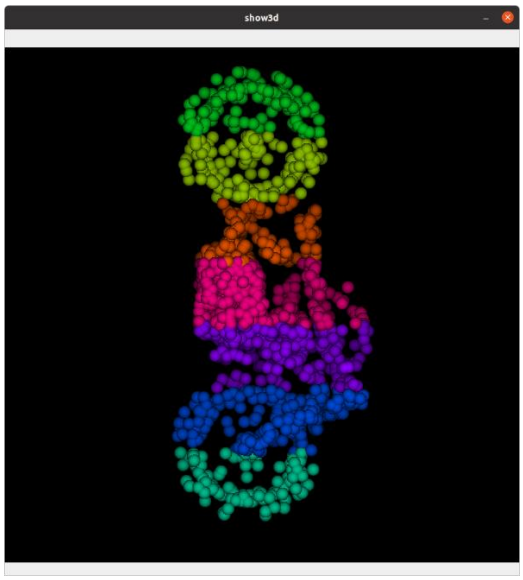
Testing Commands:

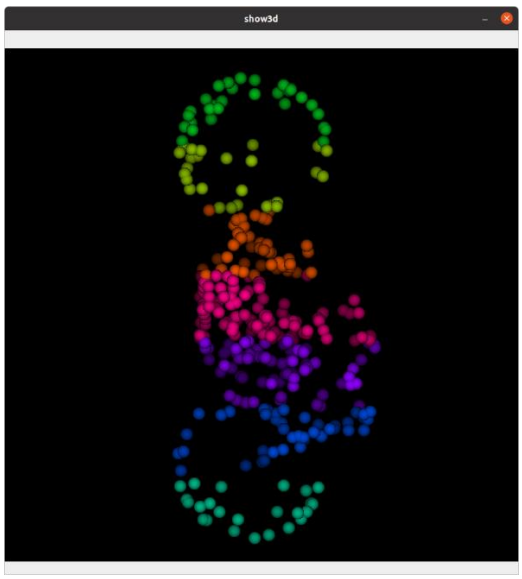
`python critical_points.py --model ../cls/cls_no_trans_epoch250.pt --idx 5 --class_choice Motorbike` (show original points)

`python critical_points.py --model ../cls/cls_no_trans_epoch250.pt --idx 5 --class_choice Motorbike --critical_point` (show critical points)

Visualization:

Original Points **Critical Points**





3. Segmentation Networks

3.1 mIOU:

In my implementation of `show_seg.py`, the code calculates mIOU first, and then shows the segmentation results. So, please run the testing commands for Section 3.2, 3.3, 3.4, and 3.5, to get the mIOU results.

Models	mIOU	Models	mIOU
Chair, with Transformation	0.8397	Chair, no Transformation	0.8640
Airplane, with Transformation	0.7318	Airplane, no Transformation	0.6772

3.2 Segmentation for Chairs with Feature Transformation

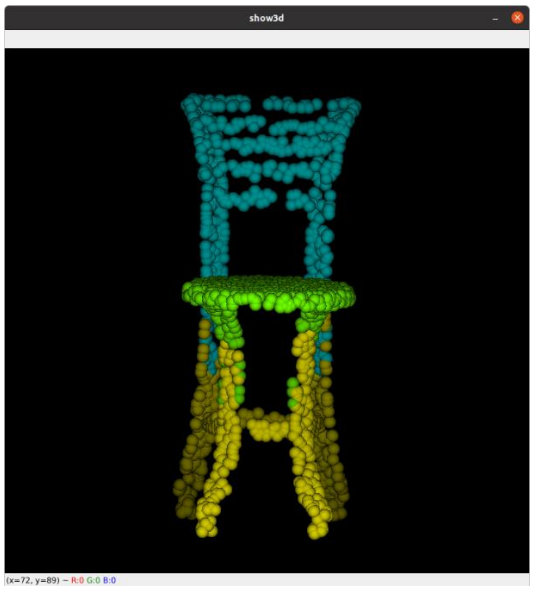
Chair Instance 1:

Testing Commands:

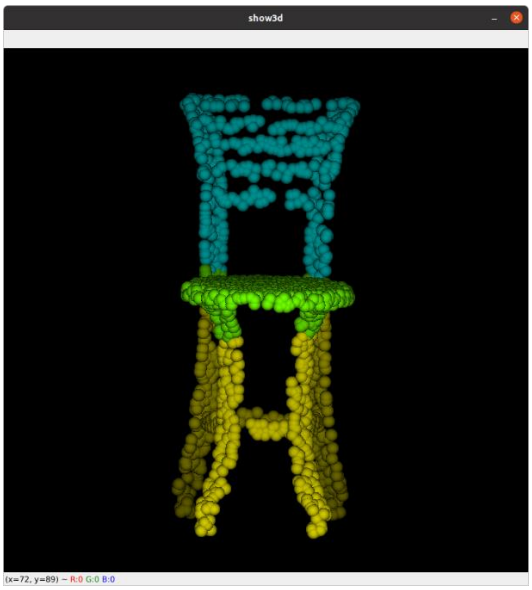
```
python show_seg.py --model ../seg/seg_with_trans_Chair_epoch25.pt --class_choice Chair --idx 5 --feature_transform
```

Segmentation:

Ground Truth



Prediction



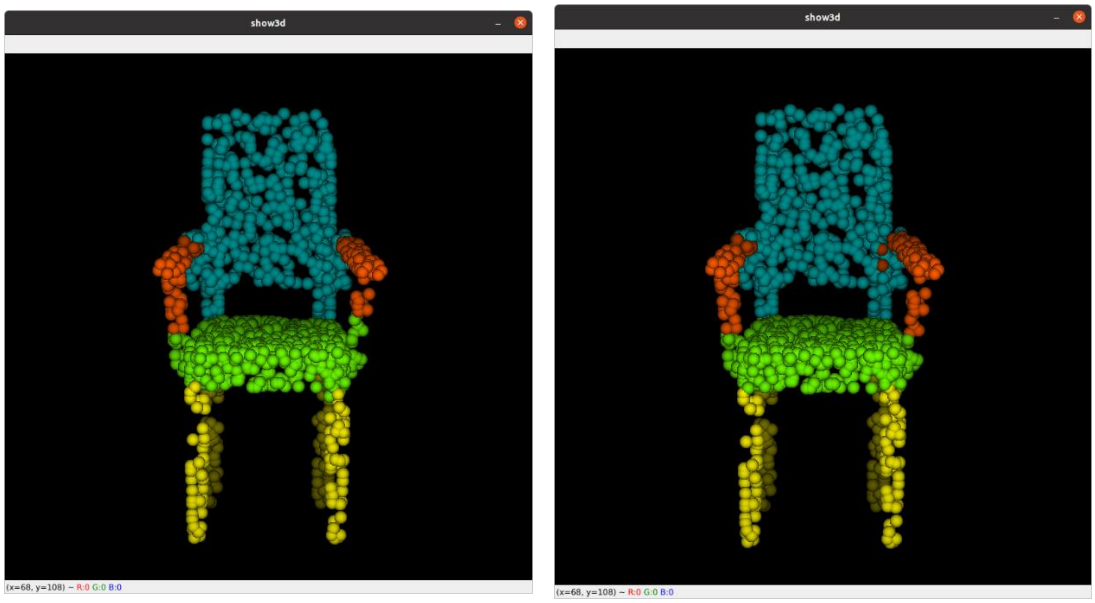
Chair Instance 2:

Testing Commands:

```
python show_seg.py --model ../seg/seg_with_trans_Chair_epoch25.pt --class_choice Chair --idx 25 --feature_transform
```

Segmentation:

Ground Truth **Prediction**



3.3 Segmentation for Chairs without Feature Transformation

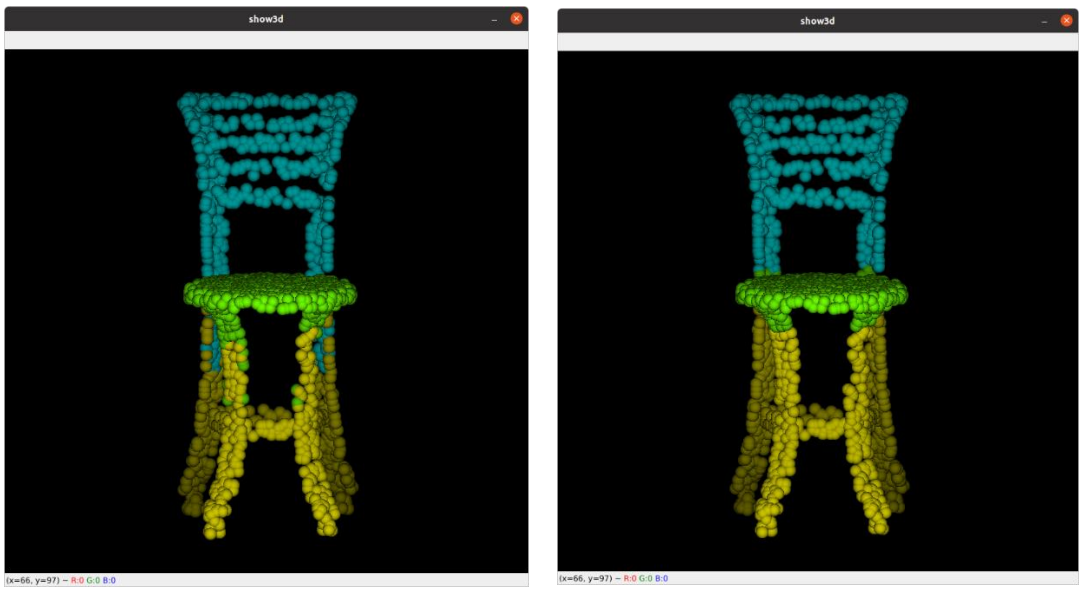
Chair Instance 1:

Testing Commands:

```
python show_seg.py --model ../seg/seg_no_trans_Chair_epoch25.pt --class_choice Chair --idx 5
```

Segmentation:

Ground Truth **Prediction**



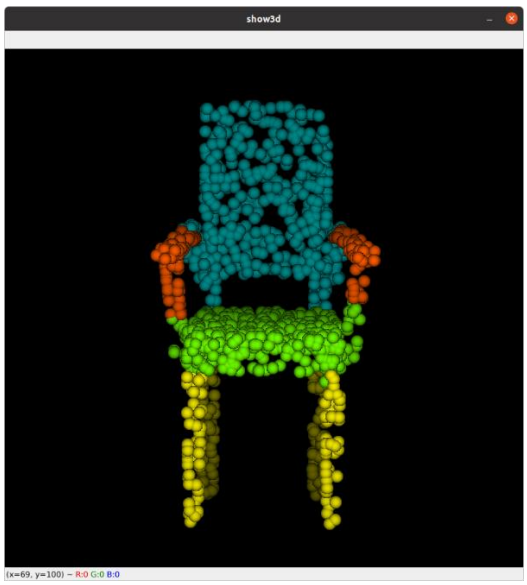
Chair Instance 2:

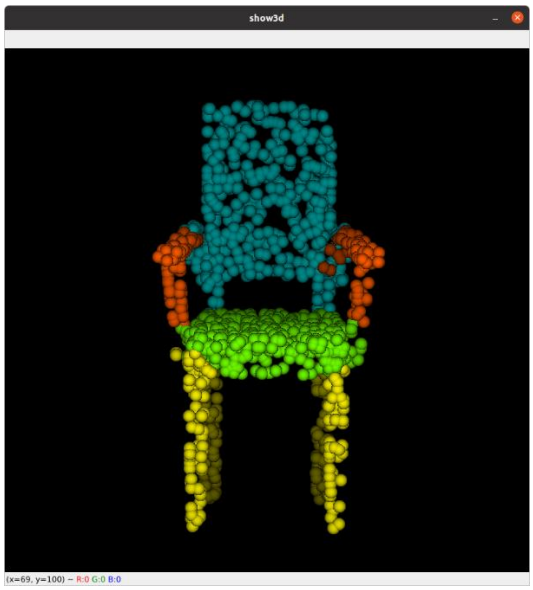
Testing Commands:

```
python show_seg.py --model ../seg/seg_no_trans_Chair_epoch25.pt --class_choice Chair --idx 25
```

Segmentation:

Ground Truth **Prediction**





3.4 Segmentation for Airplanes with Feature Transformation

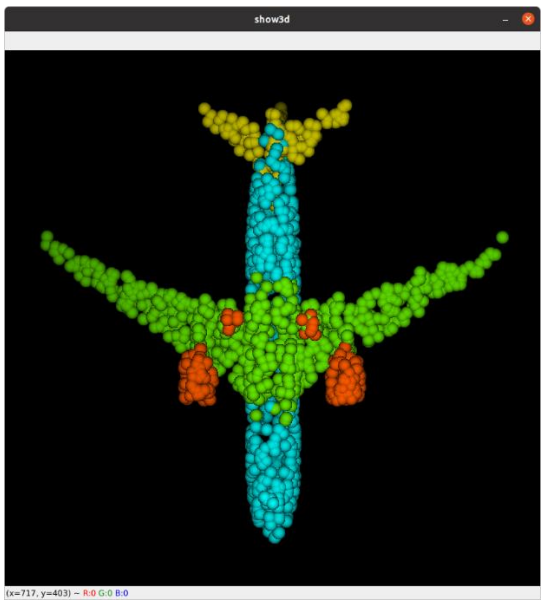
Airplane Instance 1:

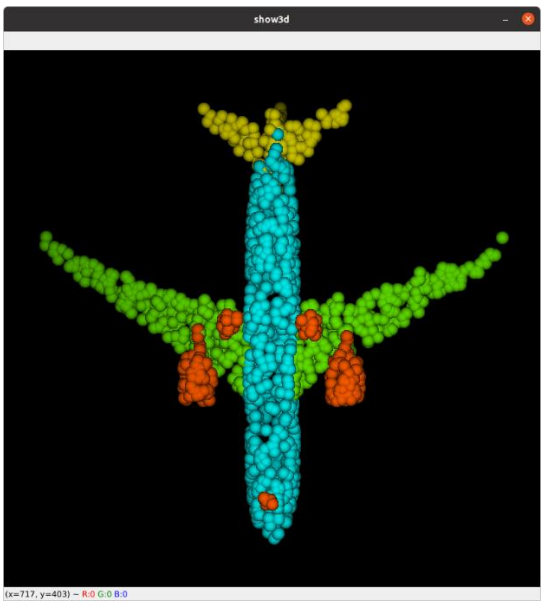
Testing Command:

```
python show_seg.py --model ../seg/seg_with_trans_Airplane_epoch25.pt --class_choice Airplane --idx 0 --feature_transform
```

Segmentation:

Ground Truth **Prediction**





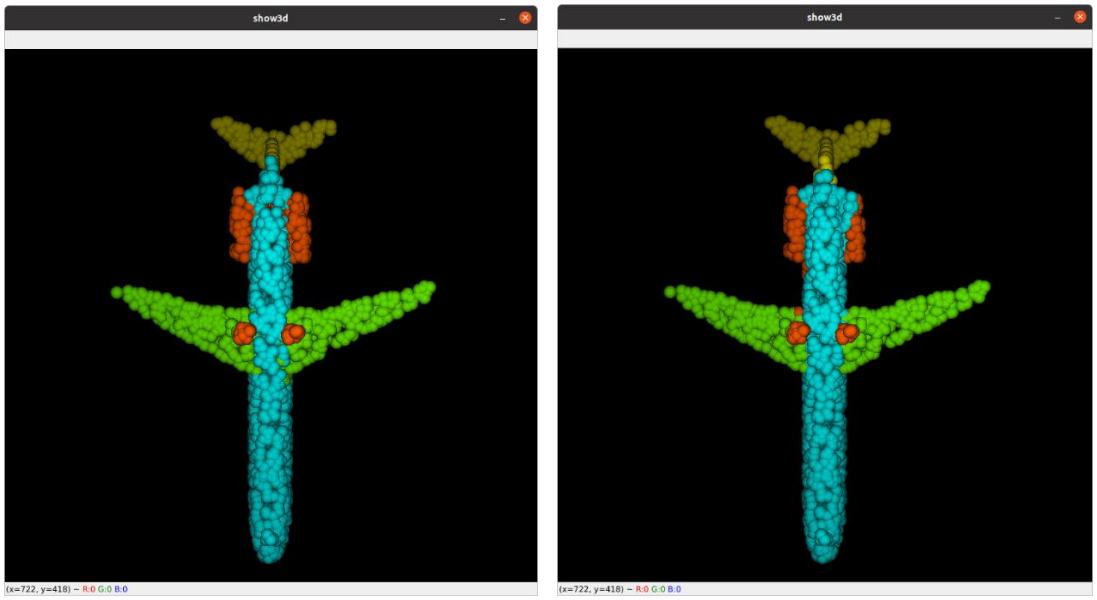
Airplane Instance 2:

Testing Command:

```
python show_seg.py --model ../seg/seg_with_trans_Airplane_epoch25.pt --class_choice Airplane --idx 9 --feature_transform
```

Segmentation:

Ground Truth **Prediction**



3.5 Segmentation for Airplanes without Feature Transformation

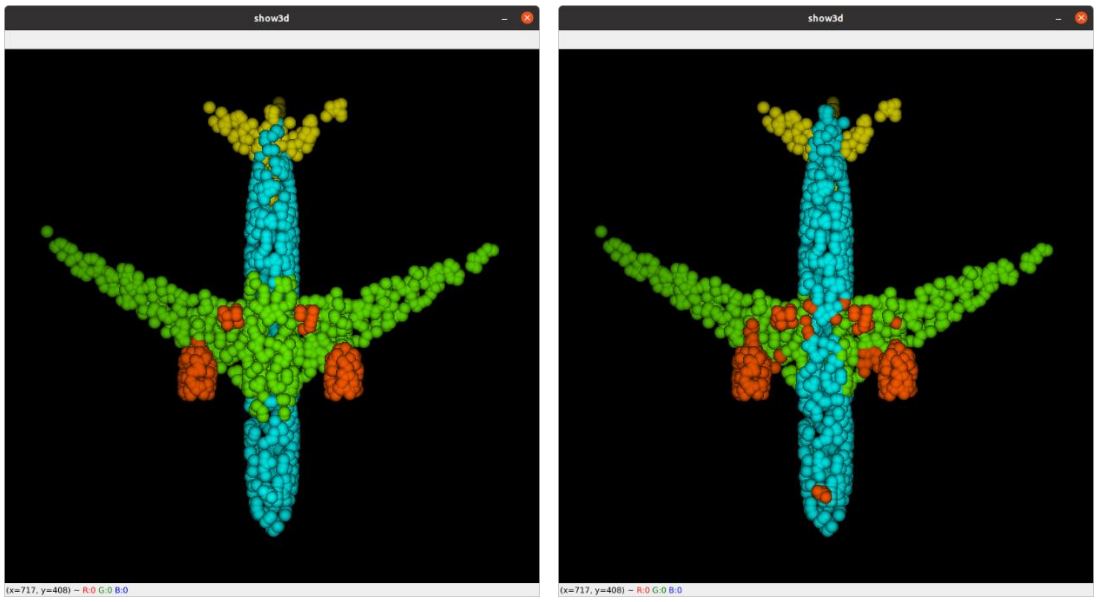
Airplane Instance 1:

Testing Command:

```
python show_seg.py --model ../seg/seg_no_trans_Airplane_epoch25.pt --class_choice Airplane --idx 0
```

Segmentation:

Ground Truth **Prediction**



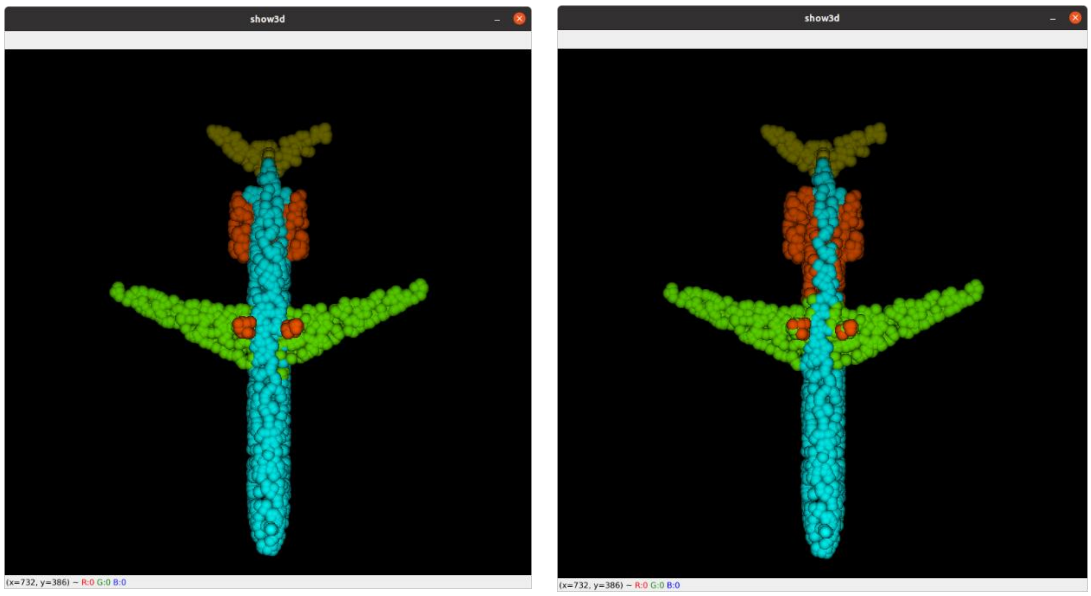
Airplane Instance 2:

Testing Command:

```
python show_seg.py --model ../seg/seg_no_trans_Airplane_epoch25.pt --class_choice Airplane --idx 9
```

Segmentation:

Ground Truth **Prediction**



4. Hyperparameters

I used default hyperparameters for training.

4.1 Training Classification Networks with Feature Transformation:

Batch Size:	32	**Number of epochs:**	250
Dataloader:			
Number of points:	2500	**Number of workers:**	4
Optimizer: Adam			
Learning Rate:	0.001	**Betas:**	(0.9,0.999)
feature_transform:	True		
Weight for regularization loss:		0.001	

4.2 Training Classification Networks without Feature Transformation:

Batch Size:	32	**Number of epochs:**	250
Dataloader:			
Number of points:	2500	**Number of workers:**	4
Optimizer: Adam			
Learning Rate:	0.001	**Betas:**	(0.9,0.999)
feature_transform:	False		
Weight for regularization loss:		0	

4.3 Training Segmentation Networks with Feature Transformation

Batch Size:	8	**Number of epochs:**	25
Dataloader:			
Number of points:	2500	**Number of workers:**	4
Optimizer: Adam			
Learning Rate:	0.001	**Betas:**	(0.9,0.999)
feature_transform:	True		
Weight for regularization loss:		0.001	

4.4 Training Segmentation Networks without Feature Transformation:

Batch Size:	8	**Number of epochs:**	25
Dataloader:			
Number of points:	2500	**Number of workers:**	4
Optimizer: Adam			
Learning Rate:	0.001	**Betas:**	(0.9,0.999)
feature_transform:	False		
Weight for regularization loss:		0	

4.5 Testing Classification Networks (Classification Accuracy, Critical Points):

For all dataloaders, `classification = True`, `data_augmentation = False`.

4.6 Testing Segmentation Networks (mIOU, Segmentation):

For all dataloaders, `classification = False`, `data_augmentation = False`.

5. Changes

Implementation of `feature_transform_regularizer`:

In my implementation, the regularization loss is the Frobenius-norm:

$$\|AA^T - I\|_F$$

not the squared Frobenius-norm defined in the original paper:

$$\|AA^T - I\|_F^2$$

Saved Models' dictionaries:

If in the model's dictionary, `'epoch'=n`, then its filename shows that epoch is `n+1`. I think I might have to clarify this to avoid misunderstandings.