notebook

April 28, 2023

0.0.1 Description

This project contains following sub-folders and scripts: - analysis, that contains functions for ANN analysis, - generated/models, where you should paste your downloaded states of the MLP and CNN networks for specific datasets, - generated/tsne, where t-sne arrays will be saved - network/constants.py, where you should specify paths to folders described above, - network/data_loader.py, which define helper loader methods, - network/network.py, which defines the simple networks architecture

Paper attached to lab materials is an overview of what we will do in this assignment.

[ ]:
```
from analysis.network_analysis import *

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

0.1 INTER-LAYER EVOLUTION

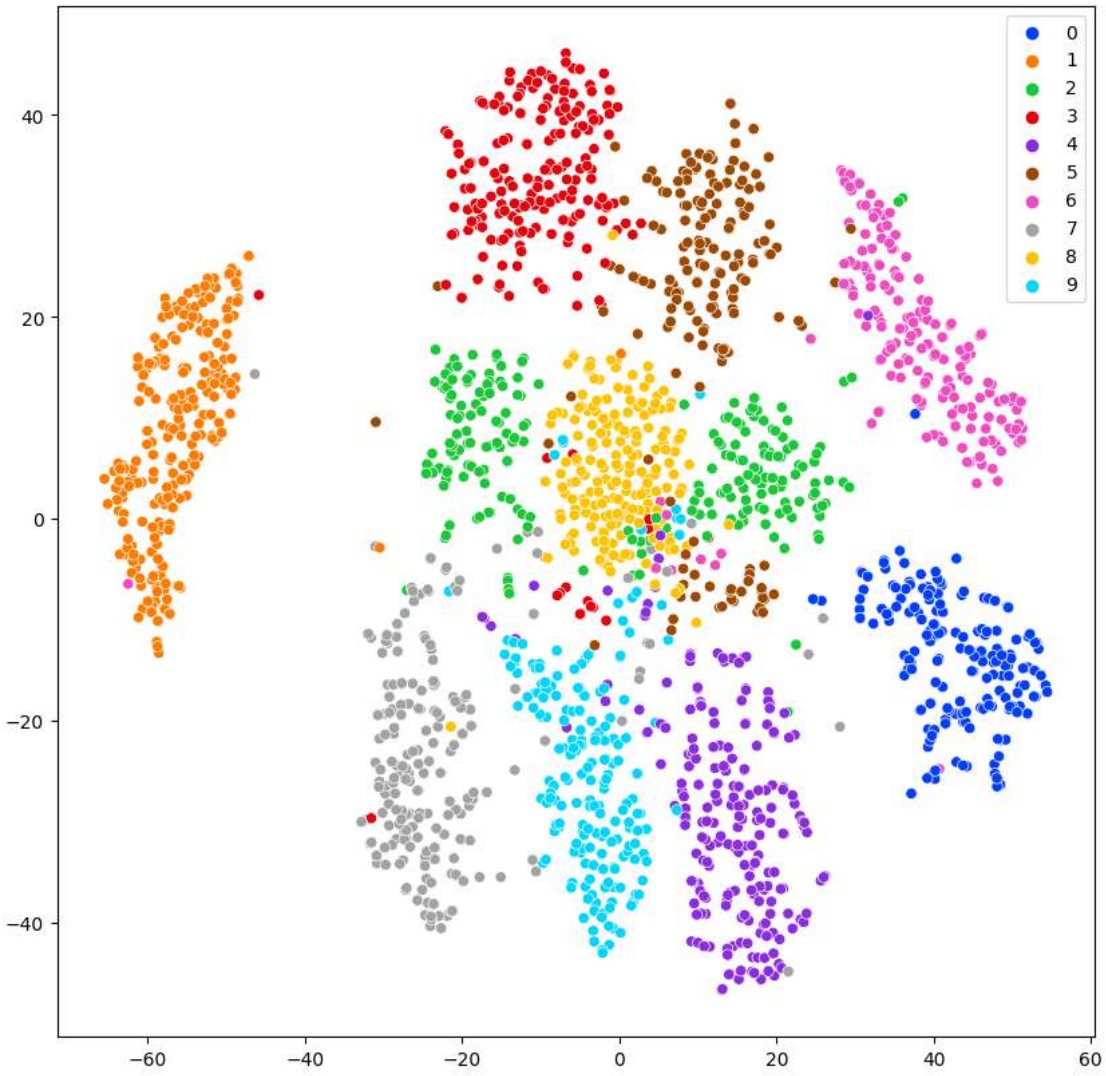
The bundled image summarizes a sequence of `N` projections, one per hidden layer, shown as thumbnails. For MLP `N=4`, and for CNN `N=2`, Trail hues encode classes, and edge brightness encodes layer number (depth). Thus, the brightness gradient shows how activation data "flow" through the four network layers. The same idea, but slightly modified, can be employed to visualize "inter-epoch" evolution.

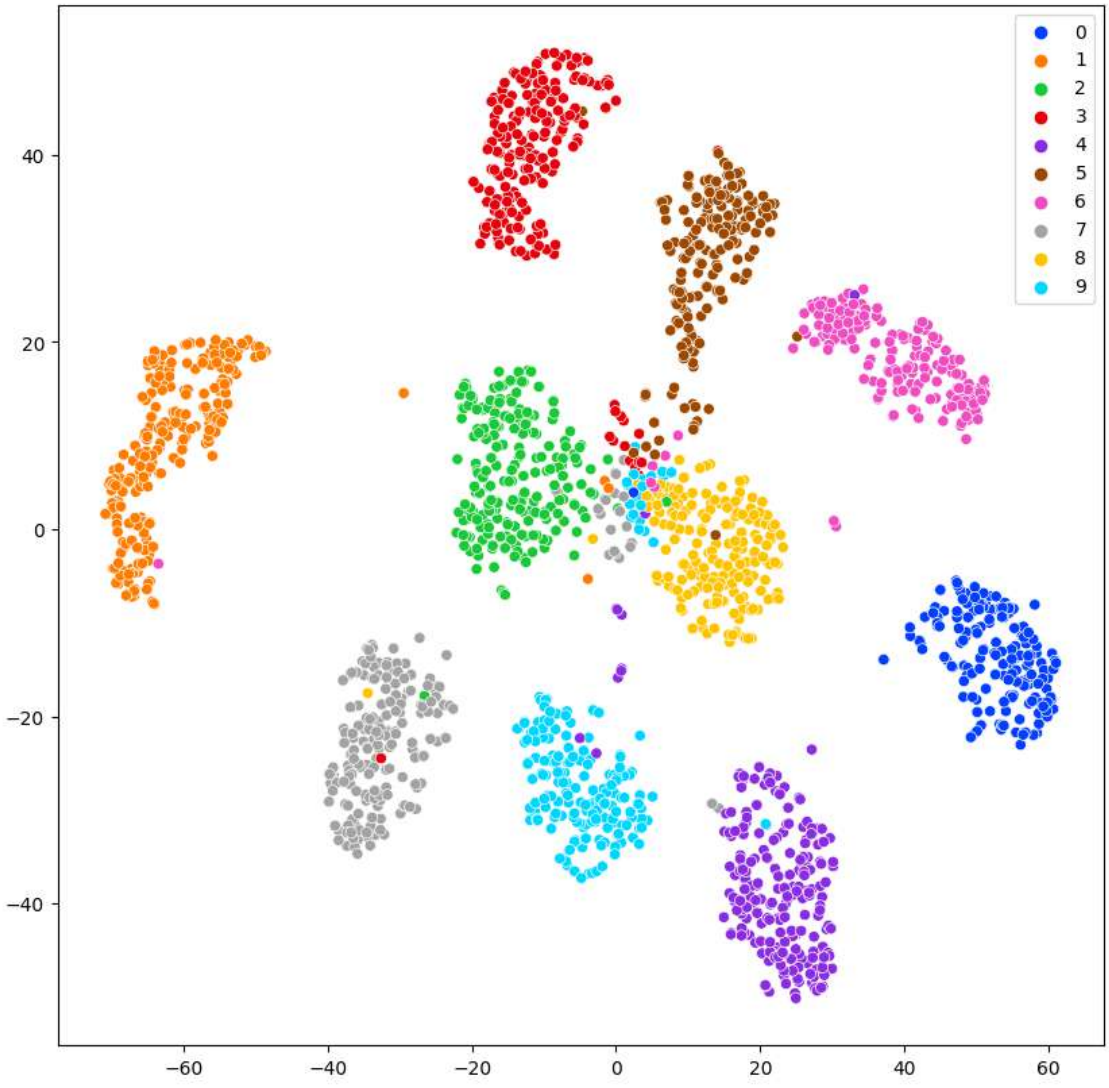
TODO: Use `show_seq_projections` function to calculate points and targets for `N` layers of mnist_mlp model. Use `inter_layer_evolution()` and `show_trace()` for plotting the actual evolution between layers.

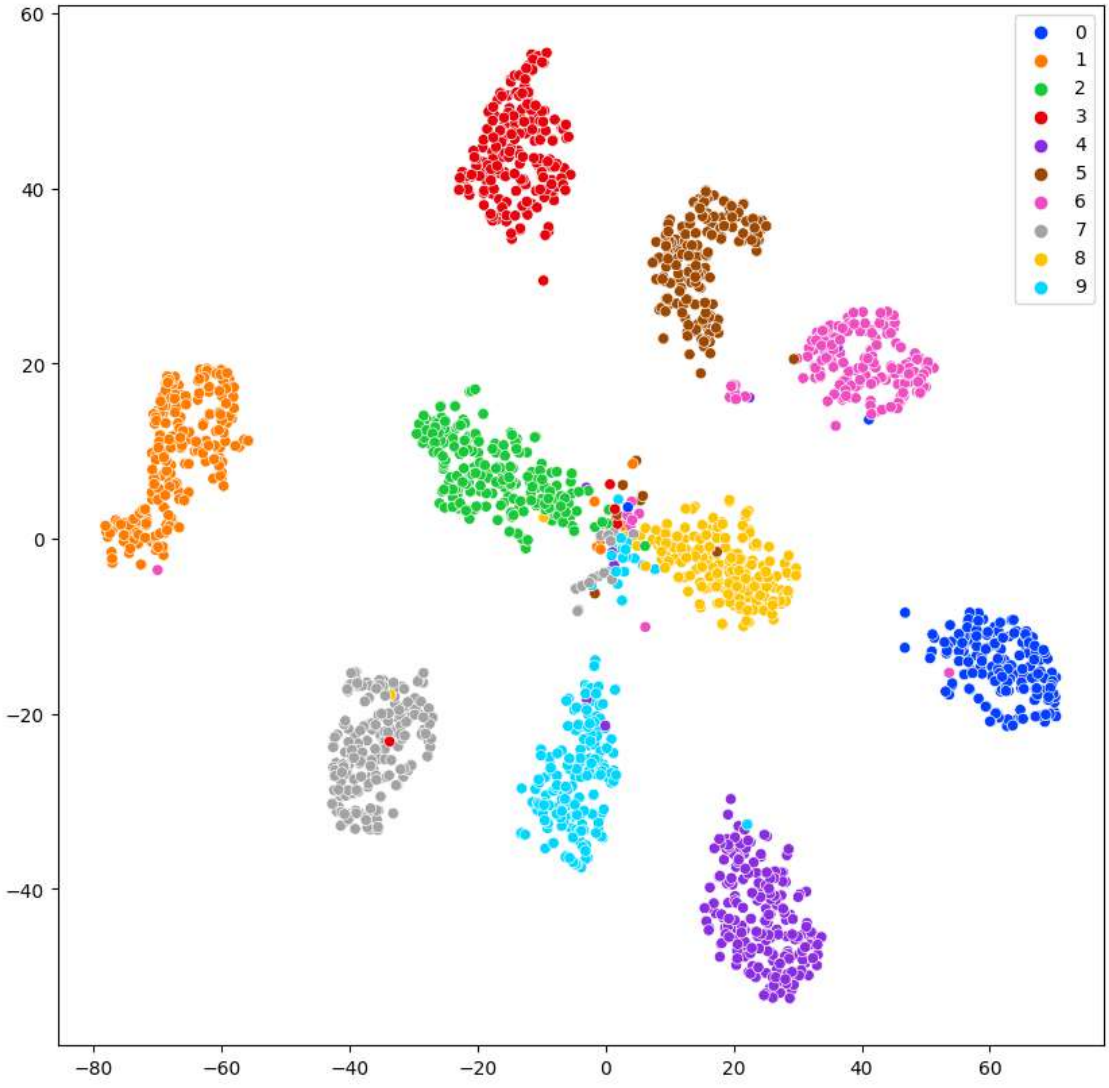
IMPORTANT: You need to implement minor detail in `show_seq_projections`

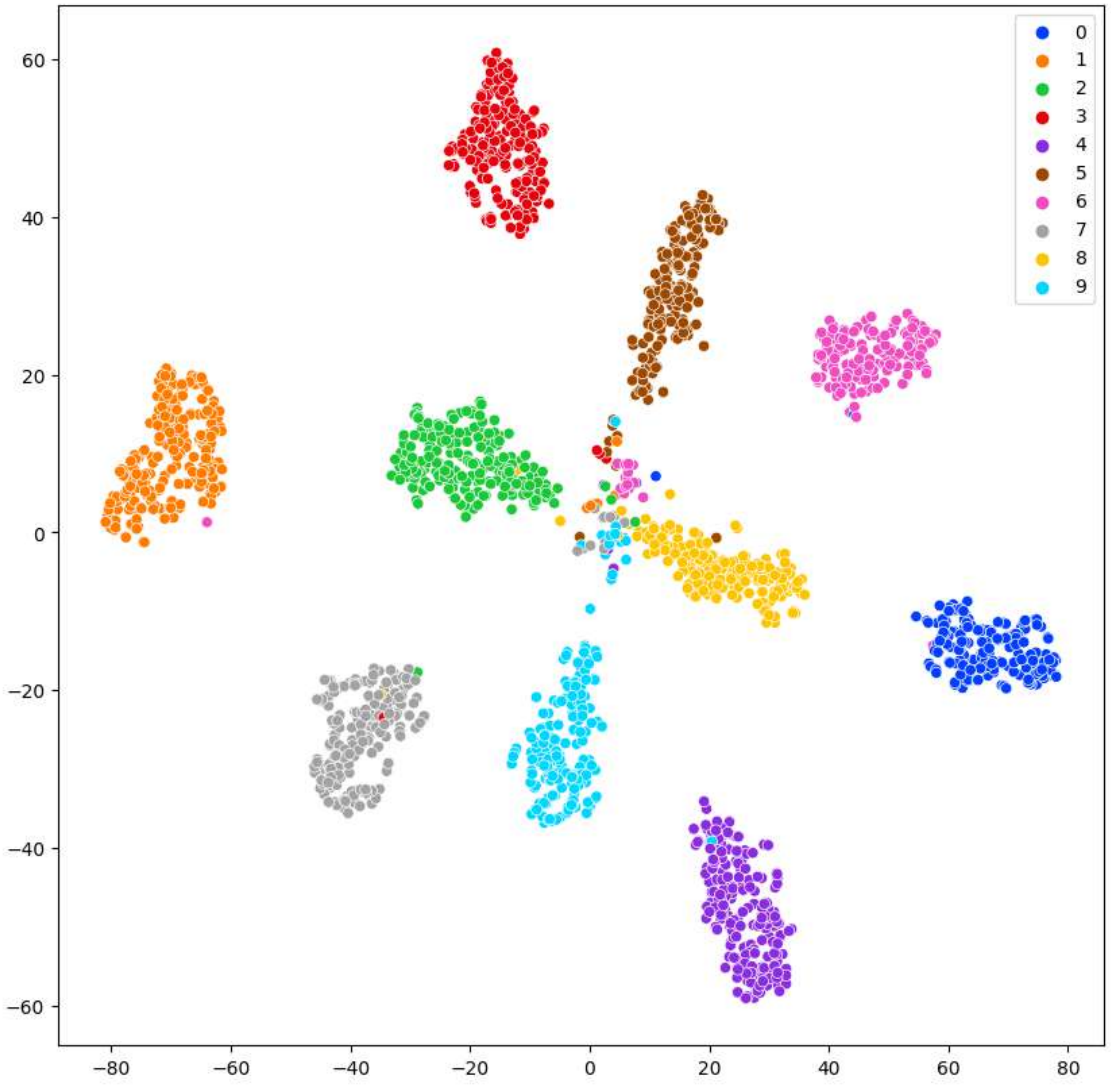
[ ]:
```
import analysis.network_analysis as na
from network.constants import DataType

points_bucket = []
for n_layer in range(0, 4):
    points, targets = na.show_seq_projections(DataType.MNIST, 'mnist_mlp',␣
 ↪n_layer=n_layer, epoch=100, size=2000)
    points_bucket.append(points)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434 [==============================] - 5s 0us/step
```



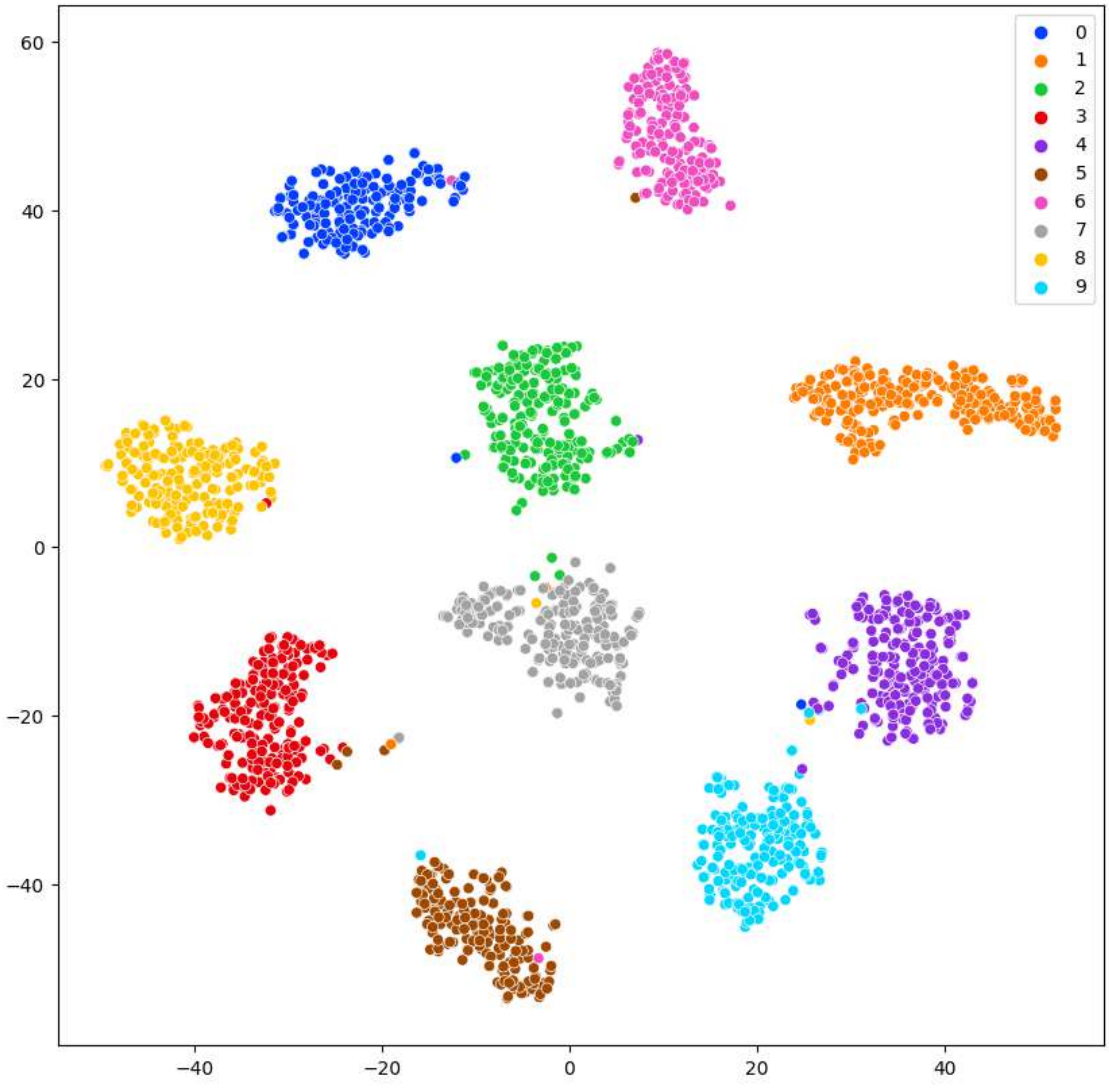

0
1
2
3
4
5
6
7
8
9
40
20
0
−20
−40
−60
−40
−20
0
20
40
60

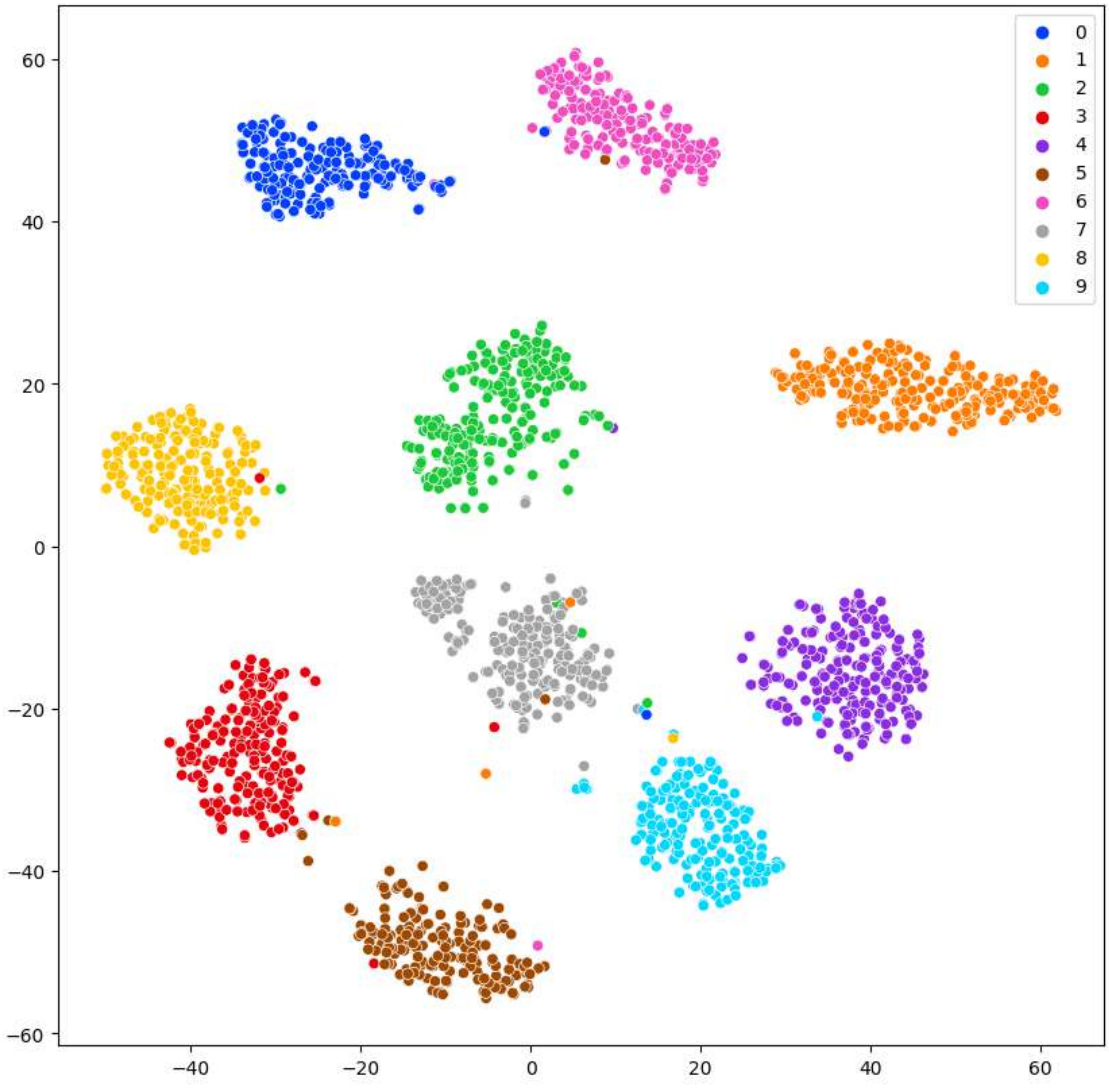

0
1
2
3
4
5
6
7
8
9
60
40
20
0
−20
−40
−80
−60
−40
−20
0
20
40
60

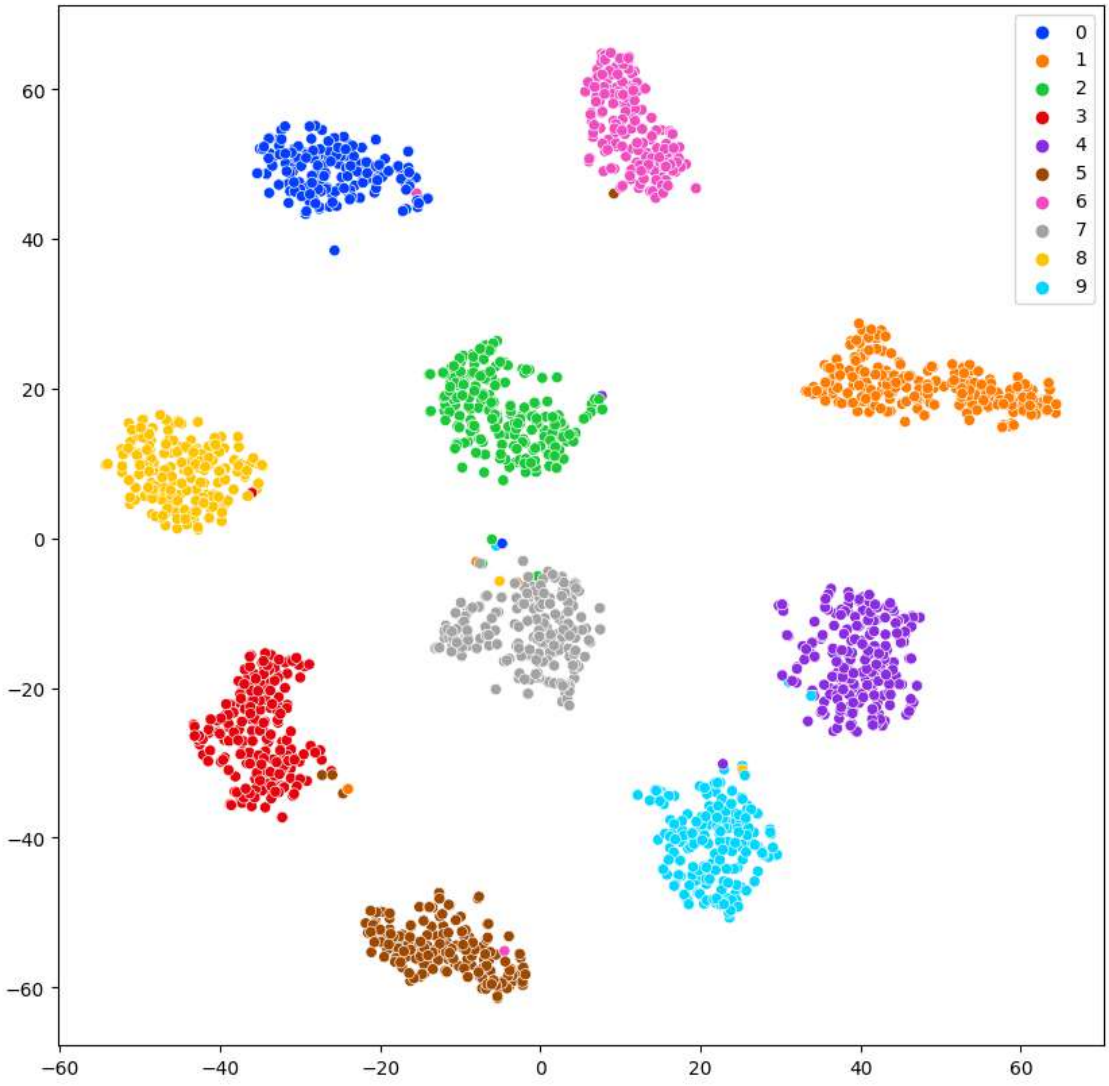


TODO: Please repeat the previous task for `mnist_cnn` and compare the two. Additionally, using the provided article, try to determine the meaning behind the results.

[ ]:
```
points_bucket_cnn = []
for n_layer in range(3):
    points, targets = na.show_seq_projections(DataType.MNIST, 'mnist_cnn',␣
 ↪n_layer=n_layer, epoch=100, size=2000)
    points_bucket_cnn.append(points)
```


0
1
2
3
4
5
6
7
8
9
60
40
20
0
−20
−40
−40
−20
0
20
40

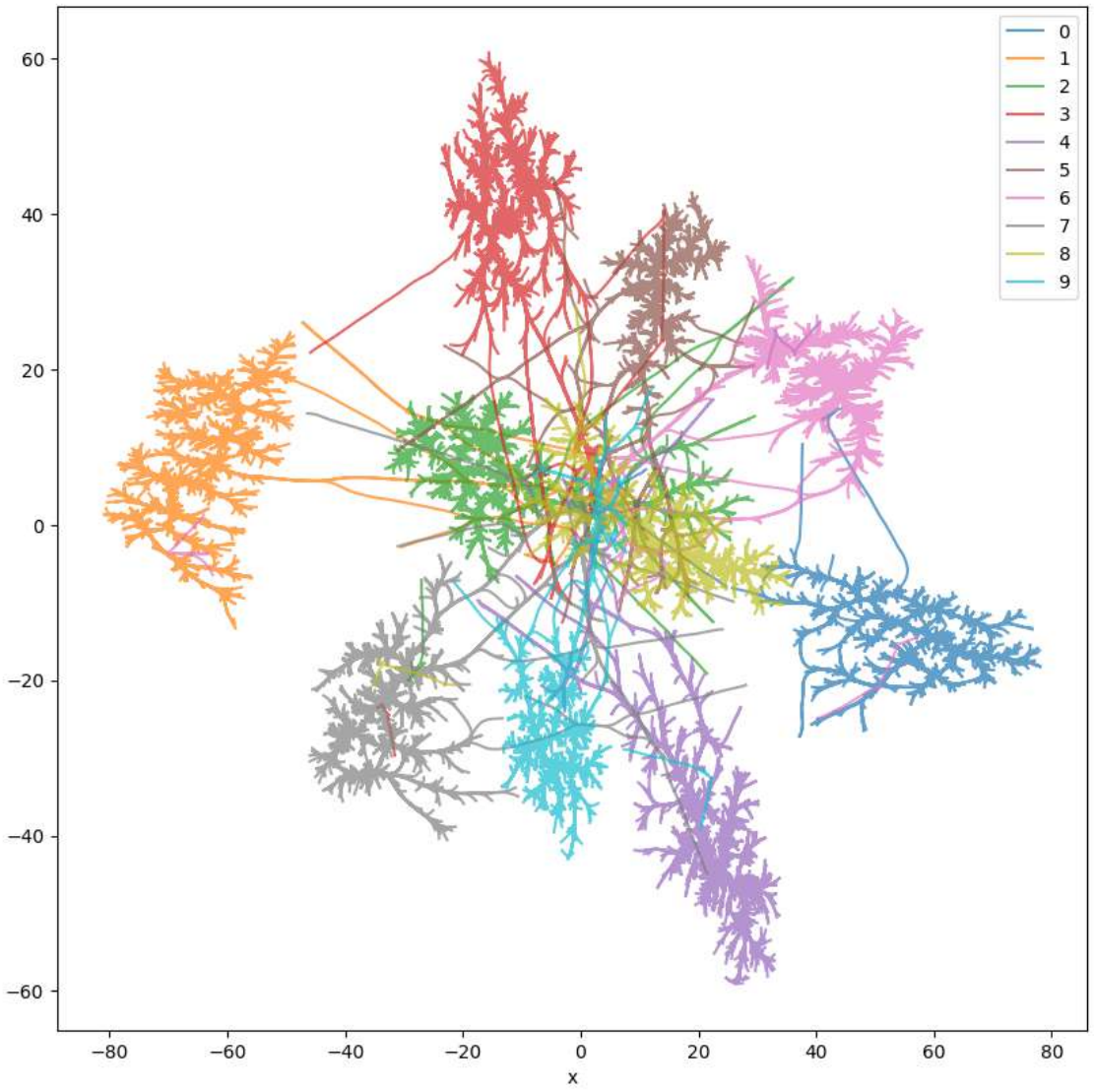

0
1
2
3
4
5
6
7
8
9
60
40
20
0
−20
−40
−60
−40
−20
0
20
40
60



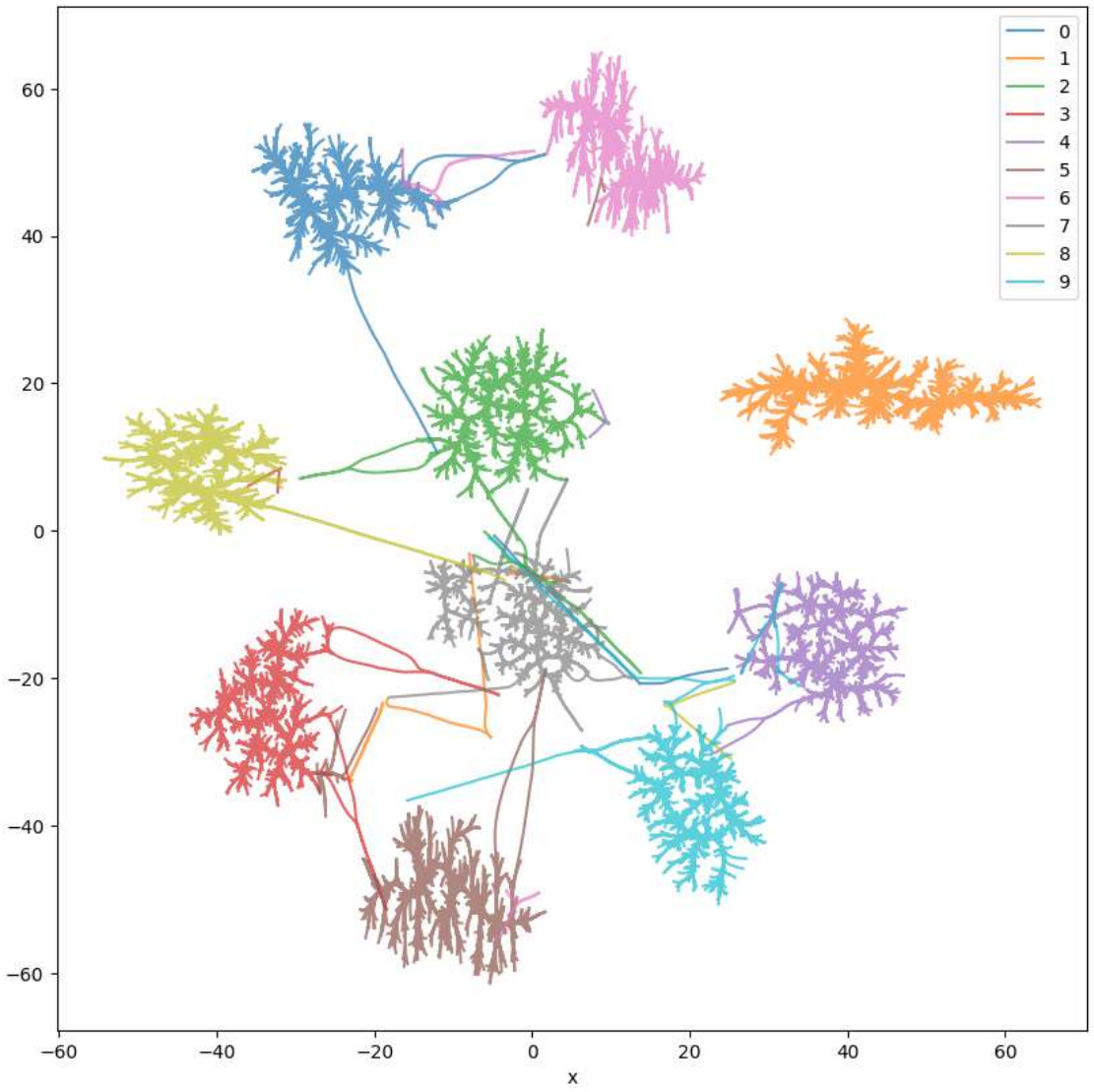
0.2 INTER-EPOCH EVOLUTION

TODO: To complete this task, start by obtaining points and targets after hidden layers using the `get_all_activations` and `process_activations` functions. Then, use analogous functions for generating the inter-epoch evolution plot and trace. Finally, perform this task for both the `mnist_mlp` and `mnist_cnn` networks.

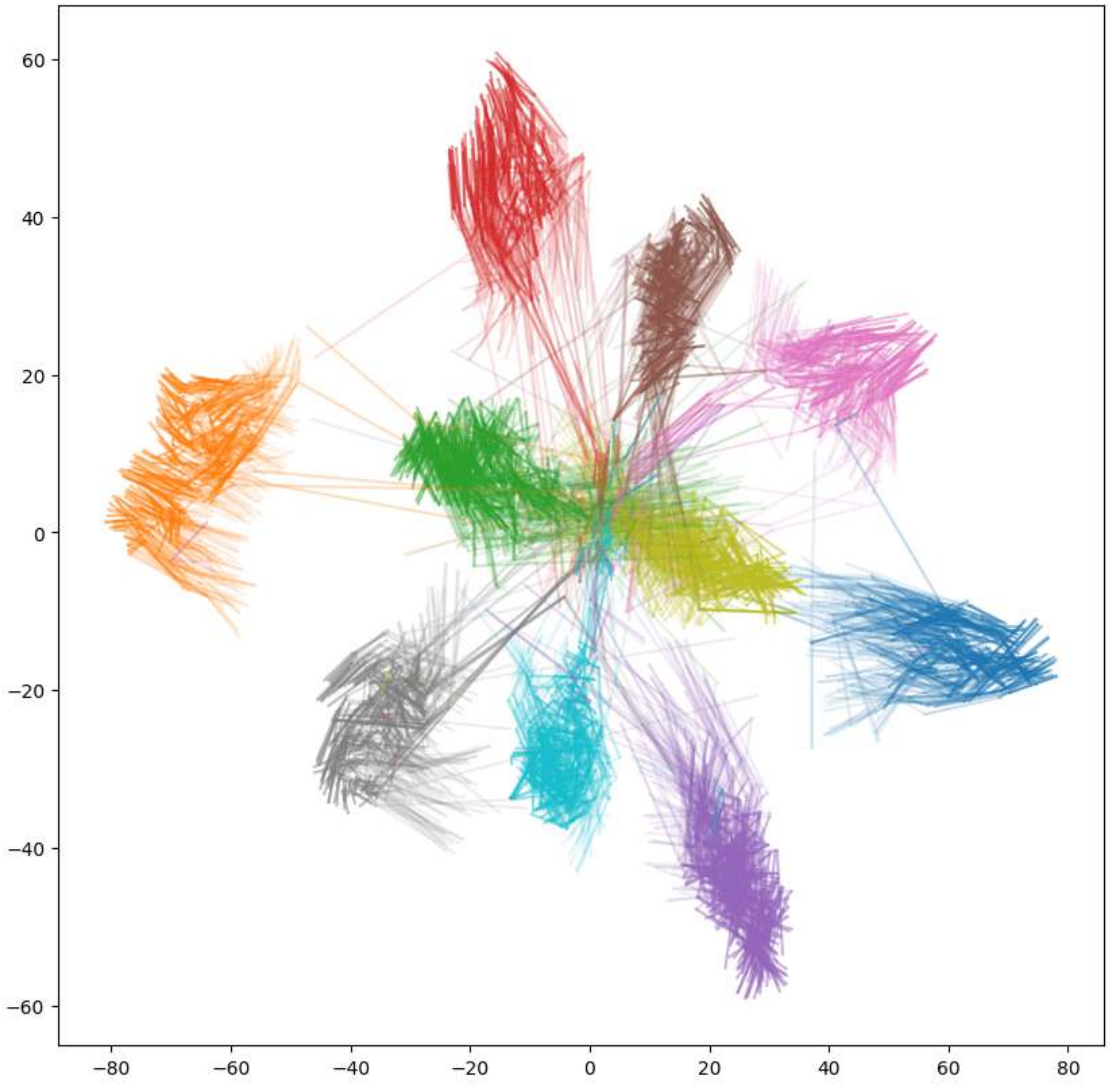
[ ]:
```
inter_layer_evolution(points_bucket, targets)
```



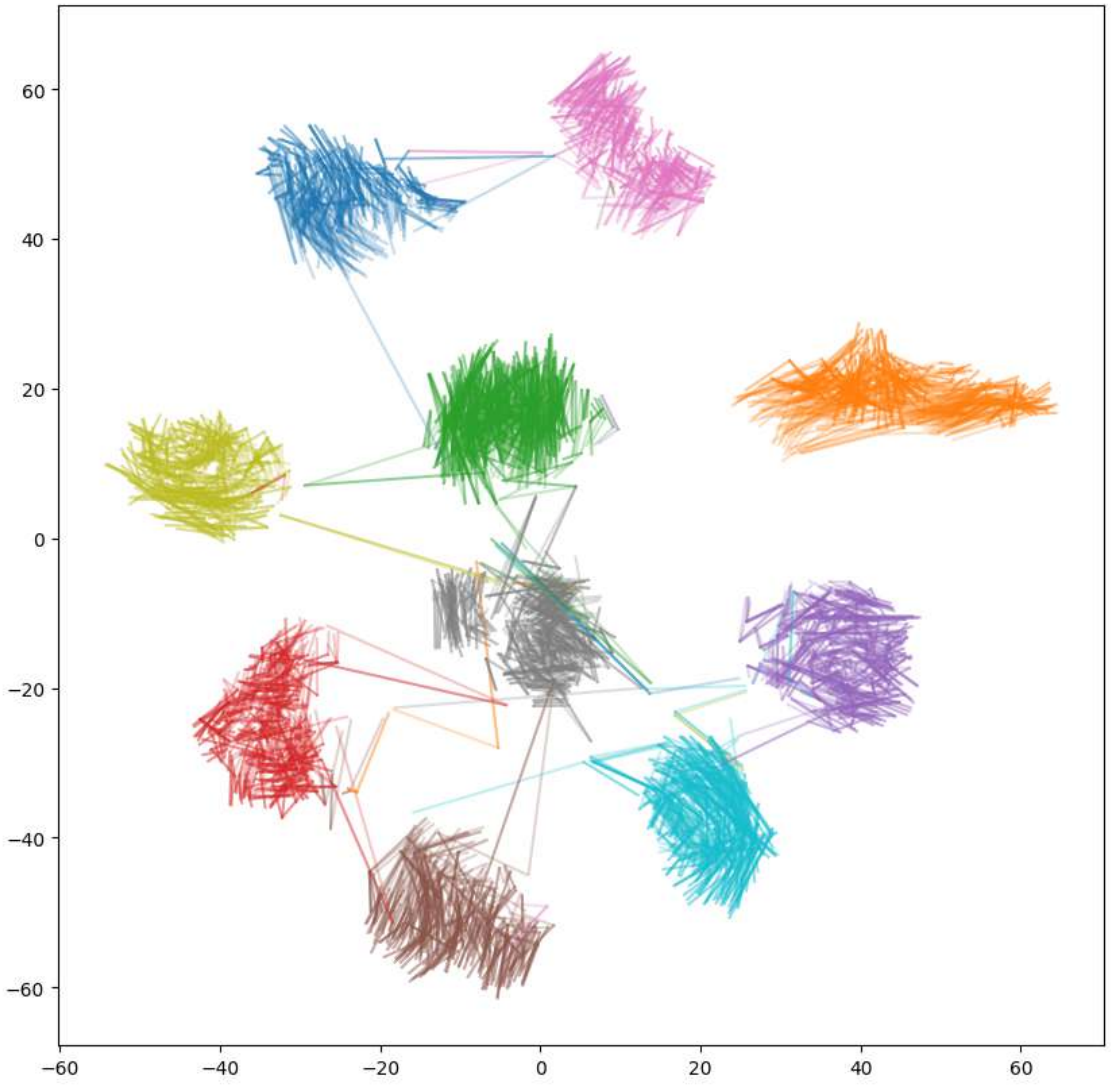
[ ]:
```
inter_layer_evolution(points_bucket_cnn, targets)
```



[ ]:
```
show_trace(points_bucket, targets)
```

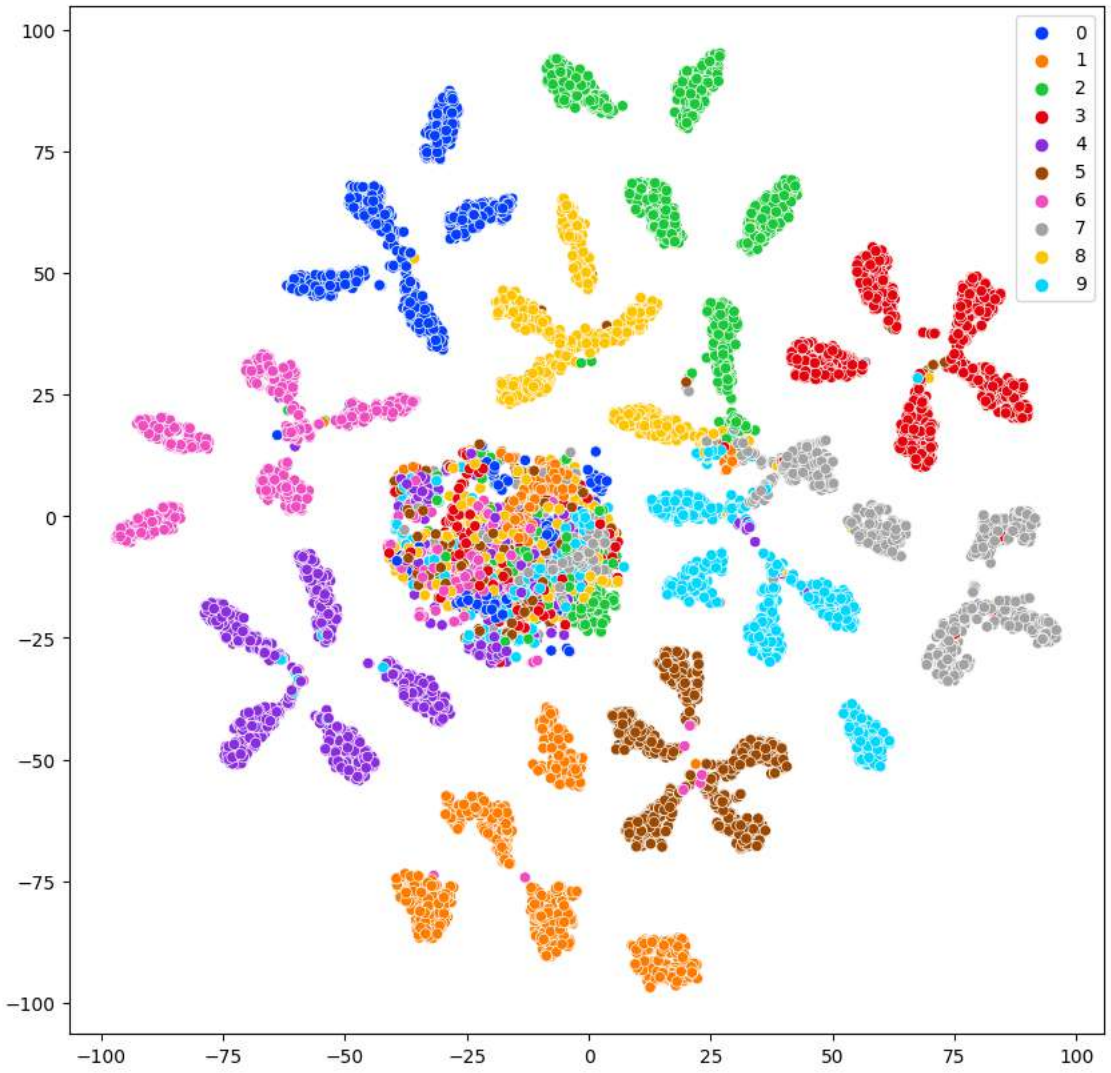


[ ]:
```
show_trace(points_bucket_cnn, targets)
```



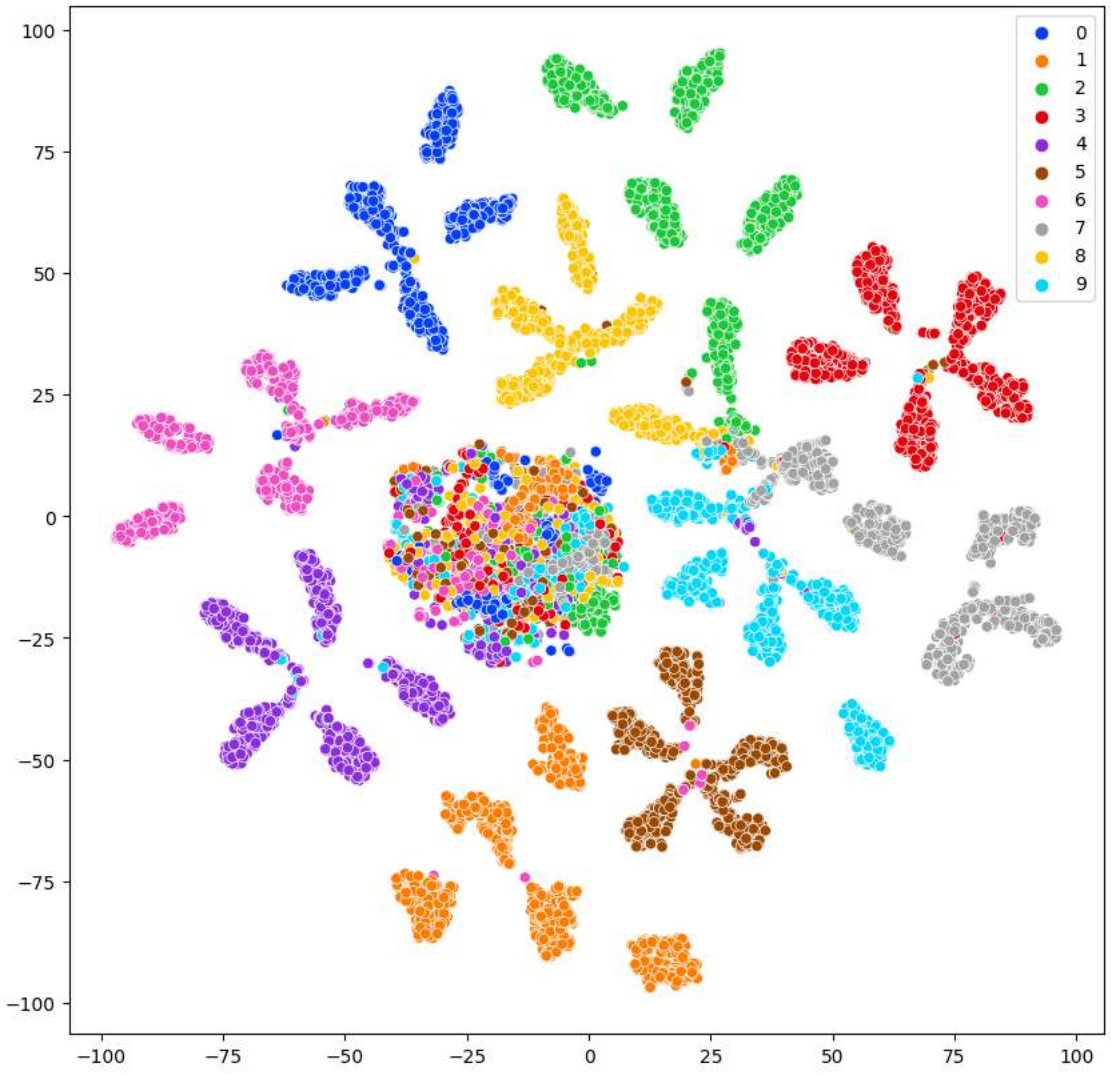
[ ]:
```
act, targets = get_all_activations(DataType.MNIST, 'mnist_mlp', n_layer=4,␣
↪size=2000)
```

[ ]:
```
points, targets =  process_activations(act, targets, size=2000)
```

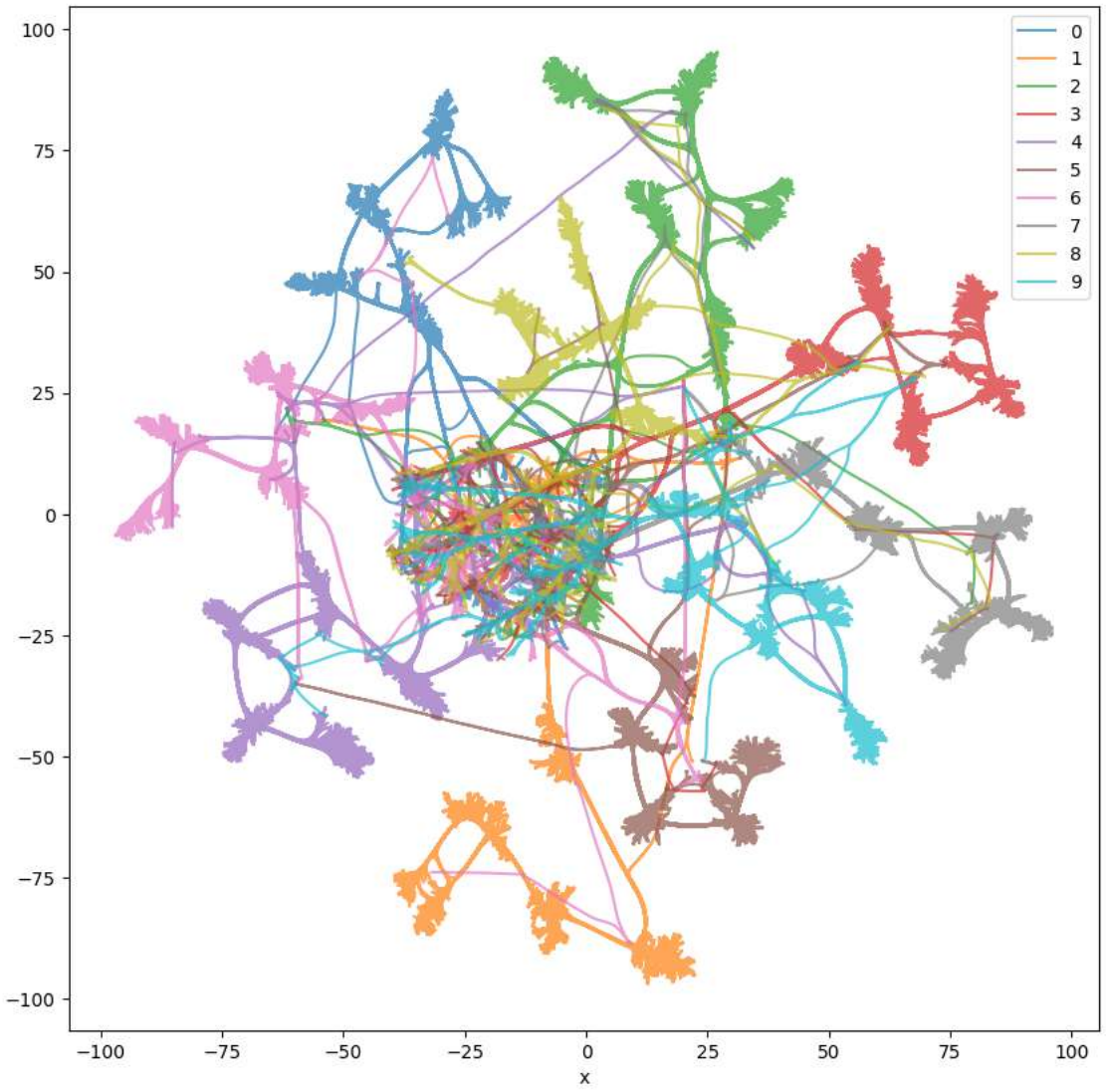


[ ]:
```
act_cnn, targets = get_all_activations(DataType.MNIST, 'mnist_cnn', n_layer=2,␣
↪size=2000)
```

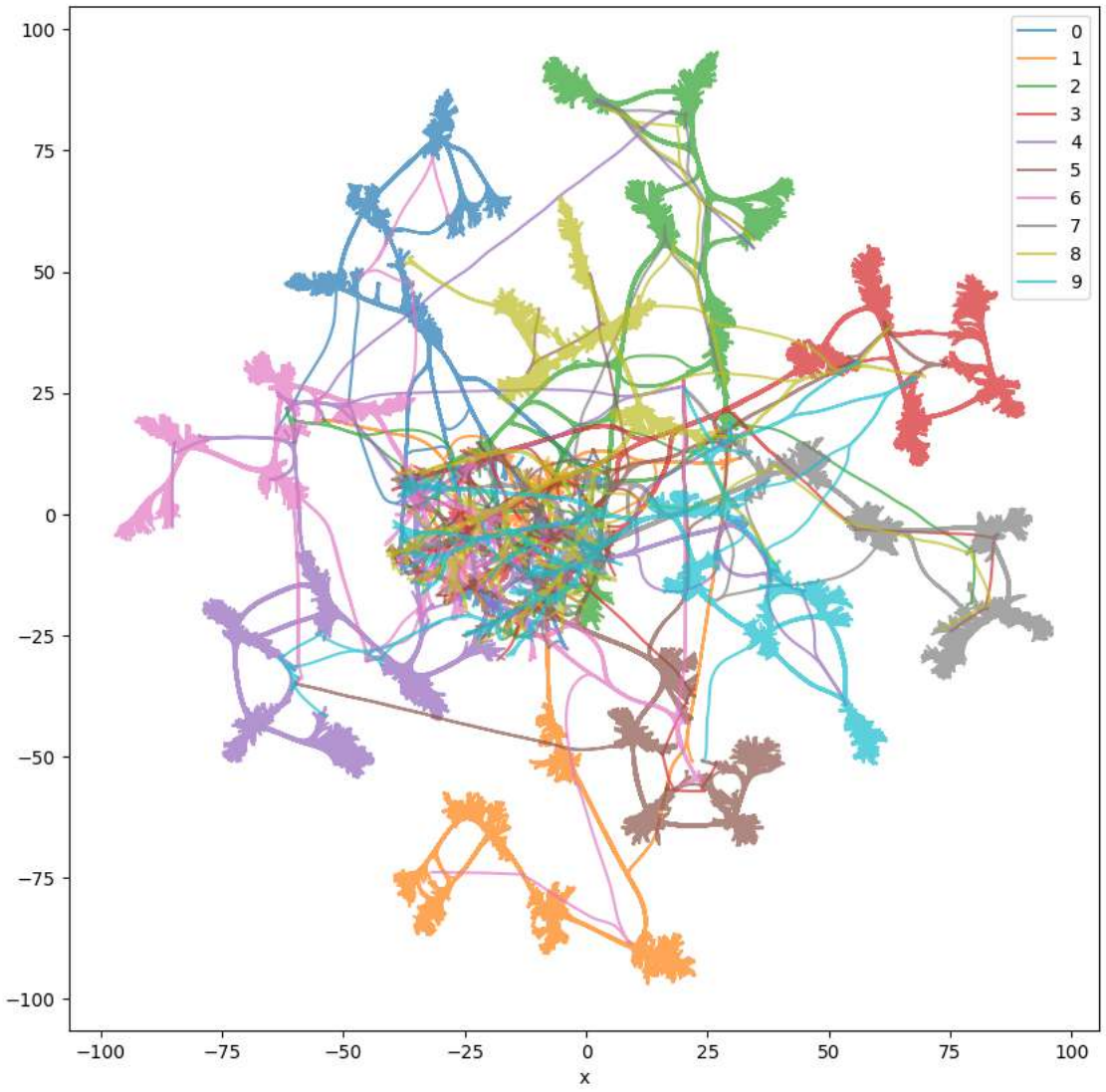
[ ]:
```
points_cnn, targets =  process_activations(act, targets, size=2000)
```



[ ]:
```
inter_epoch_evolution(points, targets)
```



[ ]:
```
inter_epoch_evolution(points_cnn, targets)
```



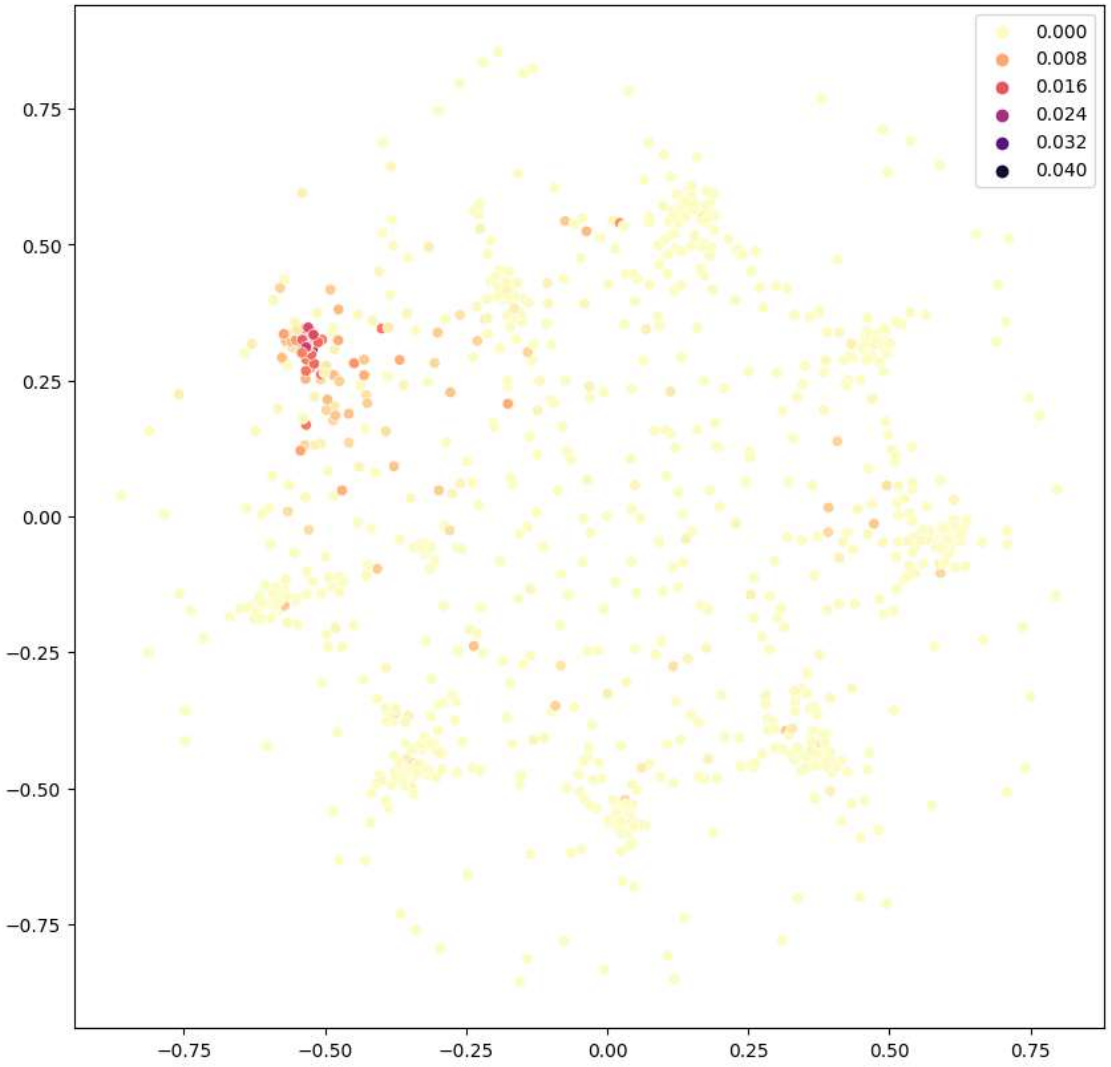
0.3 PROJECTION COMPARISION

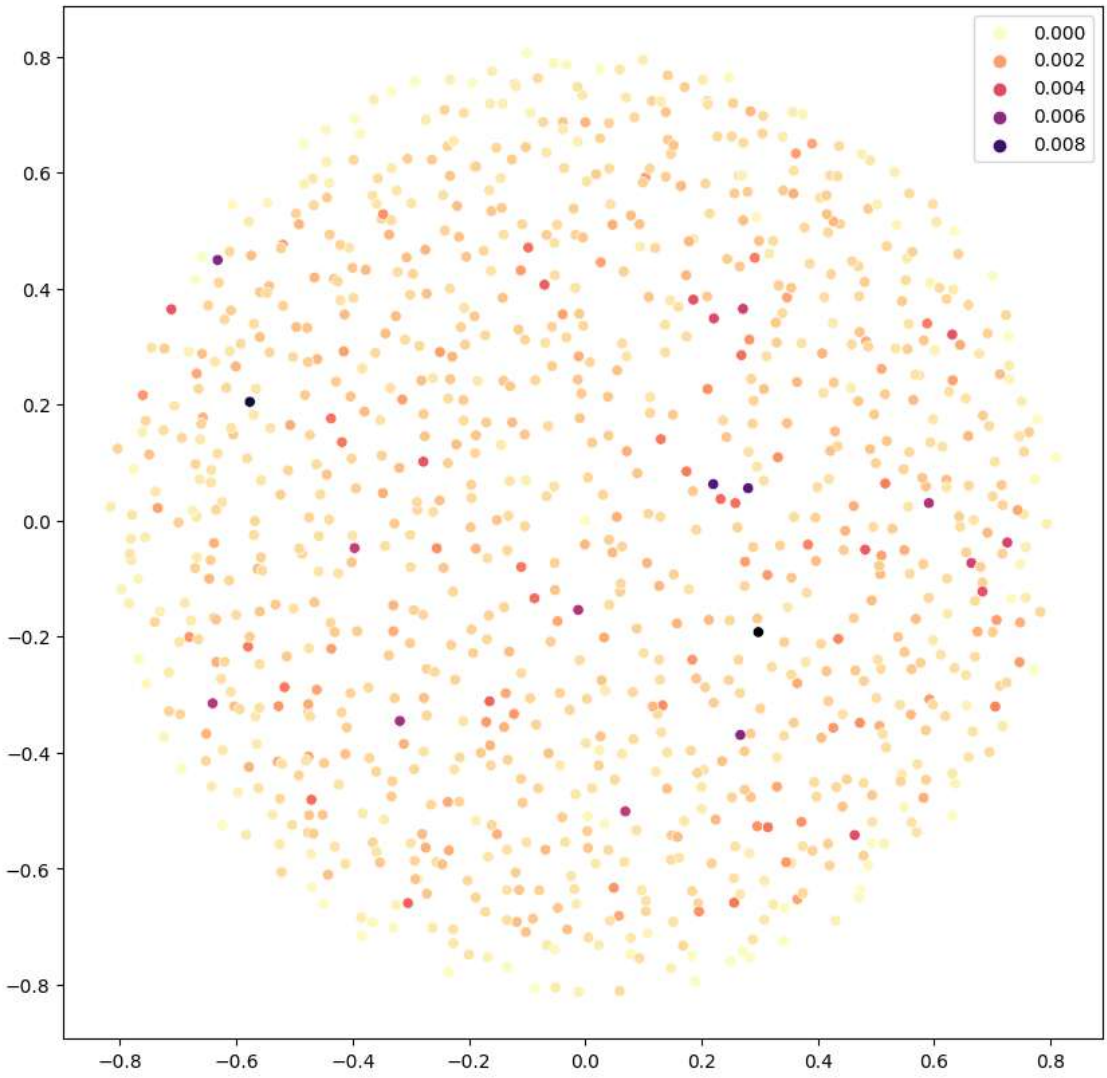
This exercise shows the neuron projection for the last CNN/MLP hidden layer activations, after training. Ignoring the colors for a moment, we see no clear pattern in the neuron projection, except for some ill-defined visual clusters. We next color each point (neuron) based on its ability to discriminate between specified class (here we define class 0) and all other classes, computed by a standard feature selection technique, based on extremely randomized trees.

TODO: Utilize the `compare_projections` method to generate a projection for both `mnist_mlp` and `mnist_cnn`, focusing on different labels (e.g., 0 and 5). IMPORTANT: Few things in `compare_projections` needs to be implemented.

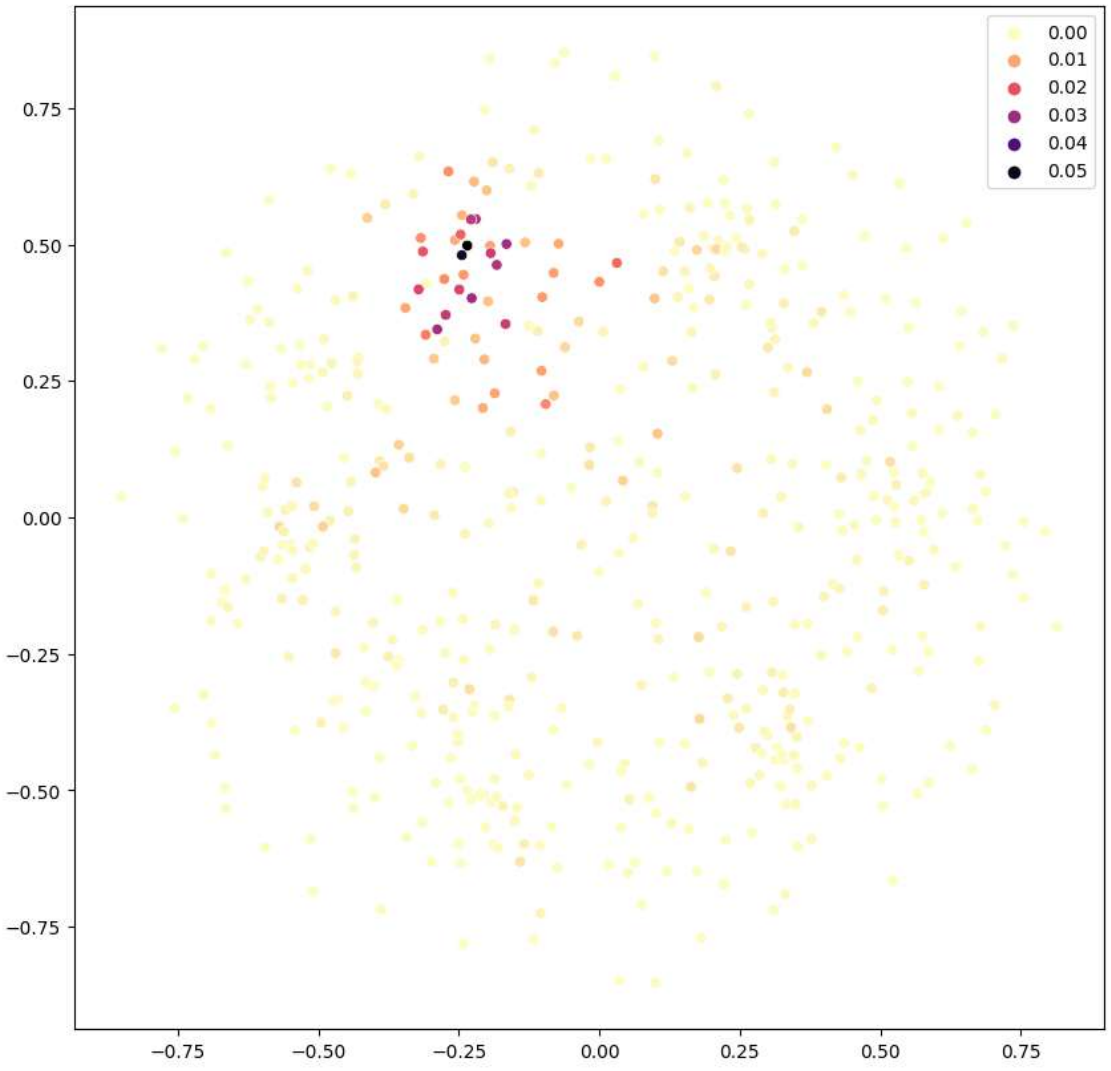
What observations can be made?

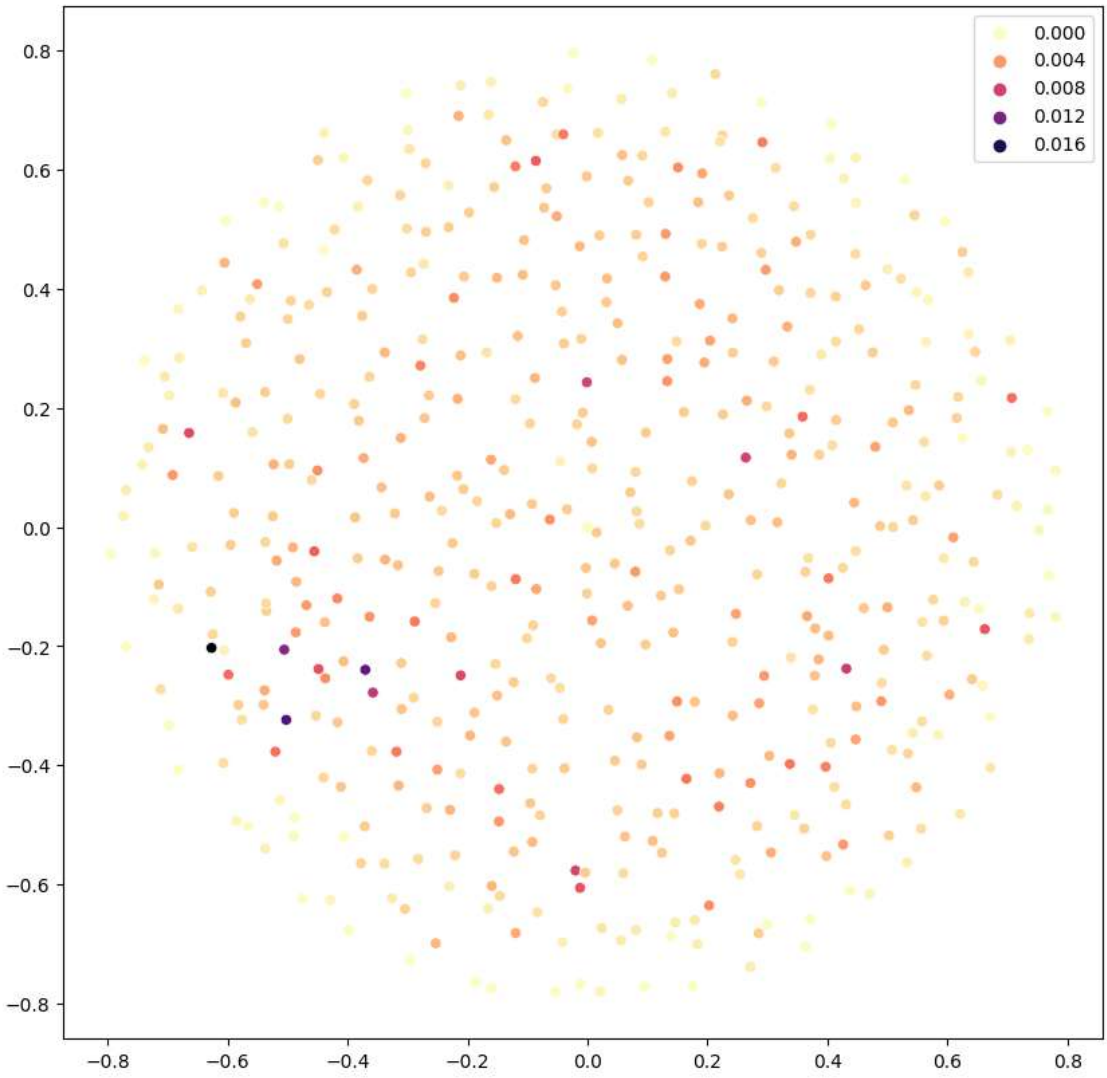
[ ]:
```
compare_projections(DataType.MNIST, 'mnist_mlp', n_layer=4, label=5)
```



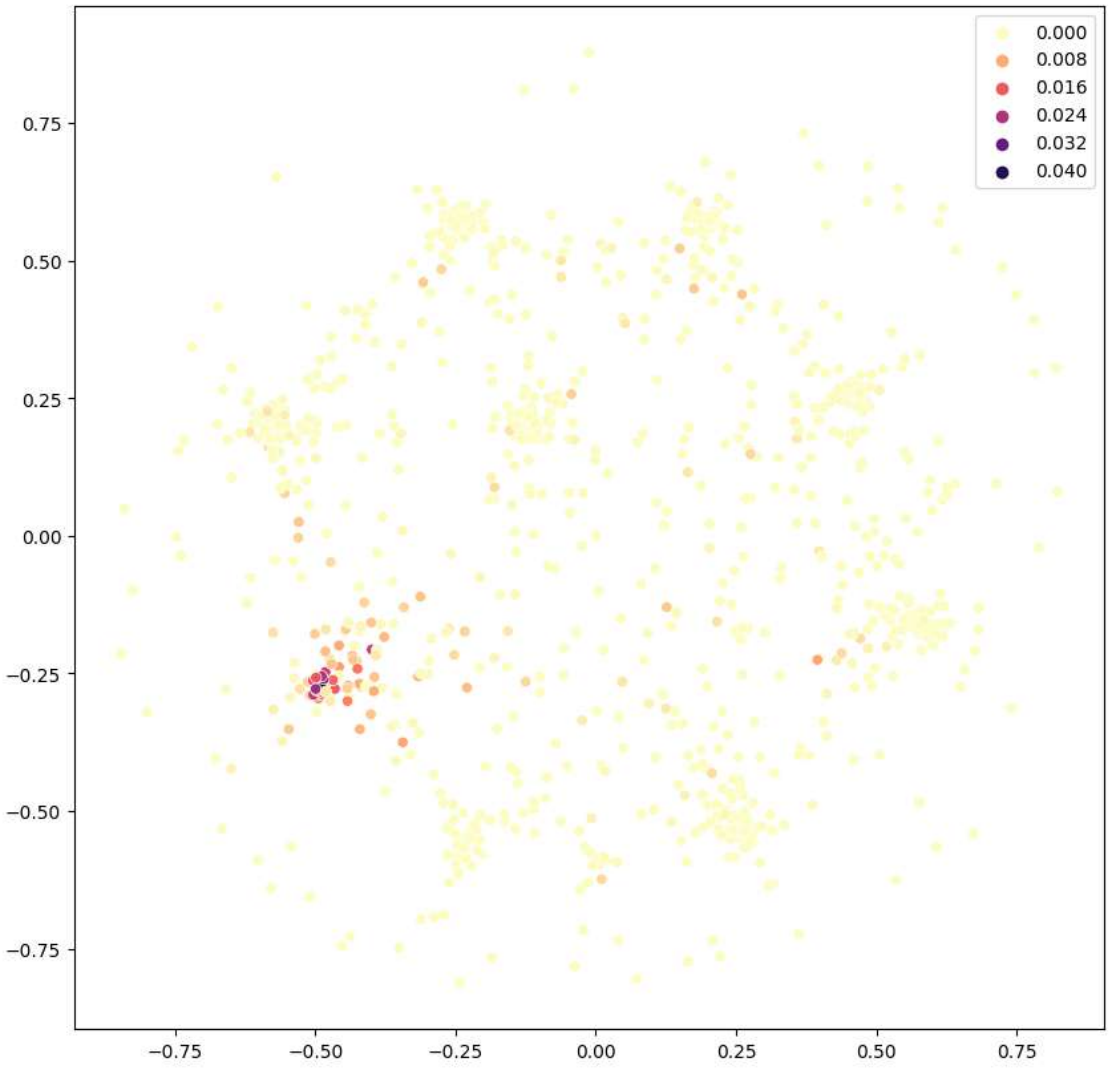


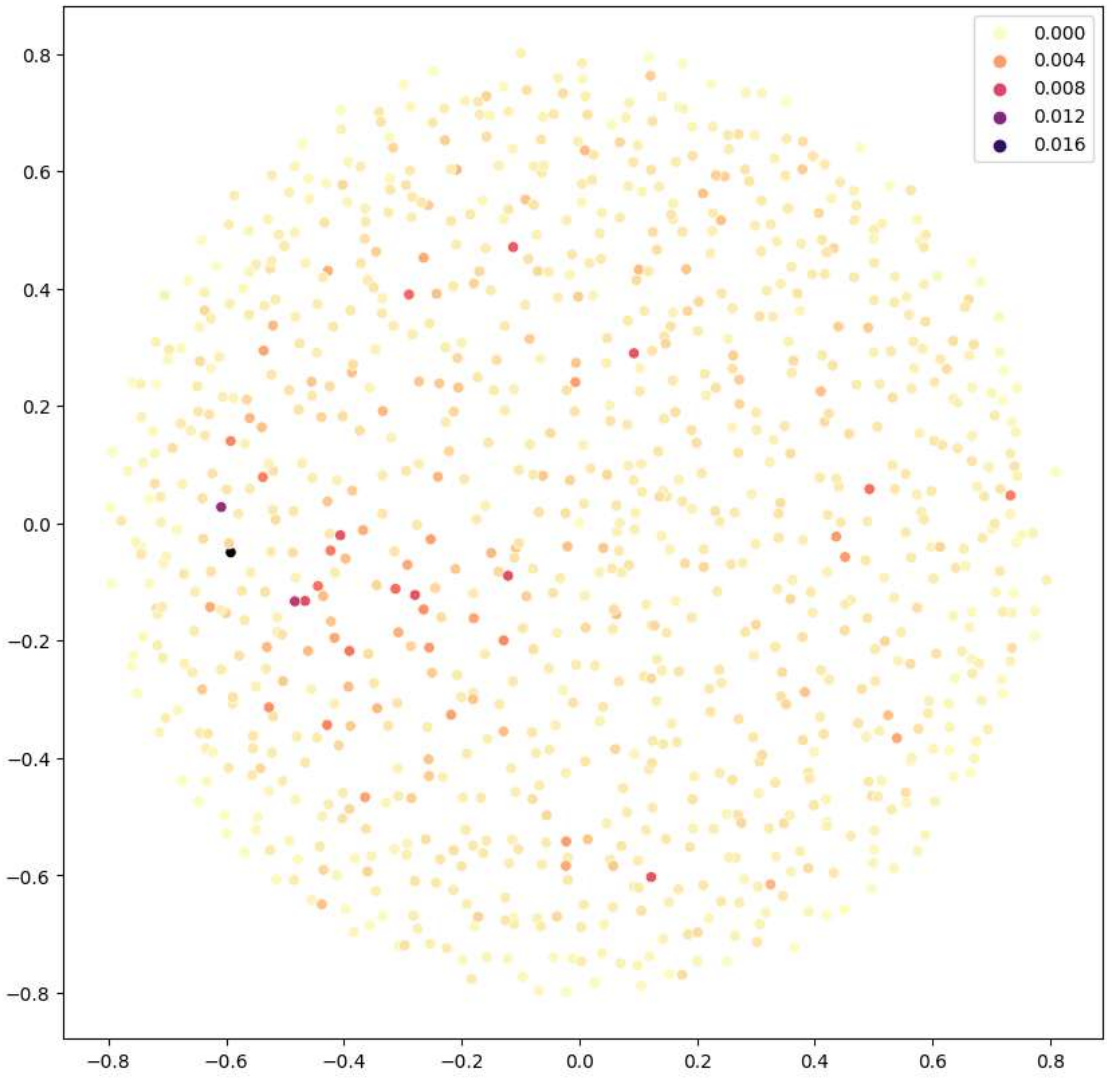
[ ]:
```
compare_projections(DataType.MNIST, 'mnist_cnn', n_layer=2, label=5)
```


0.00
0.01
0.02
0.03
0.04
0.05
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75

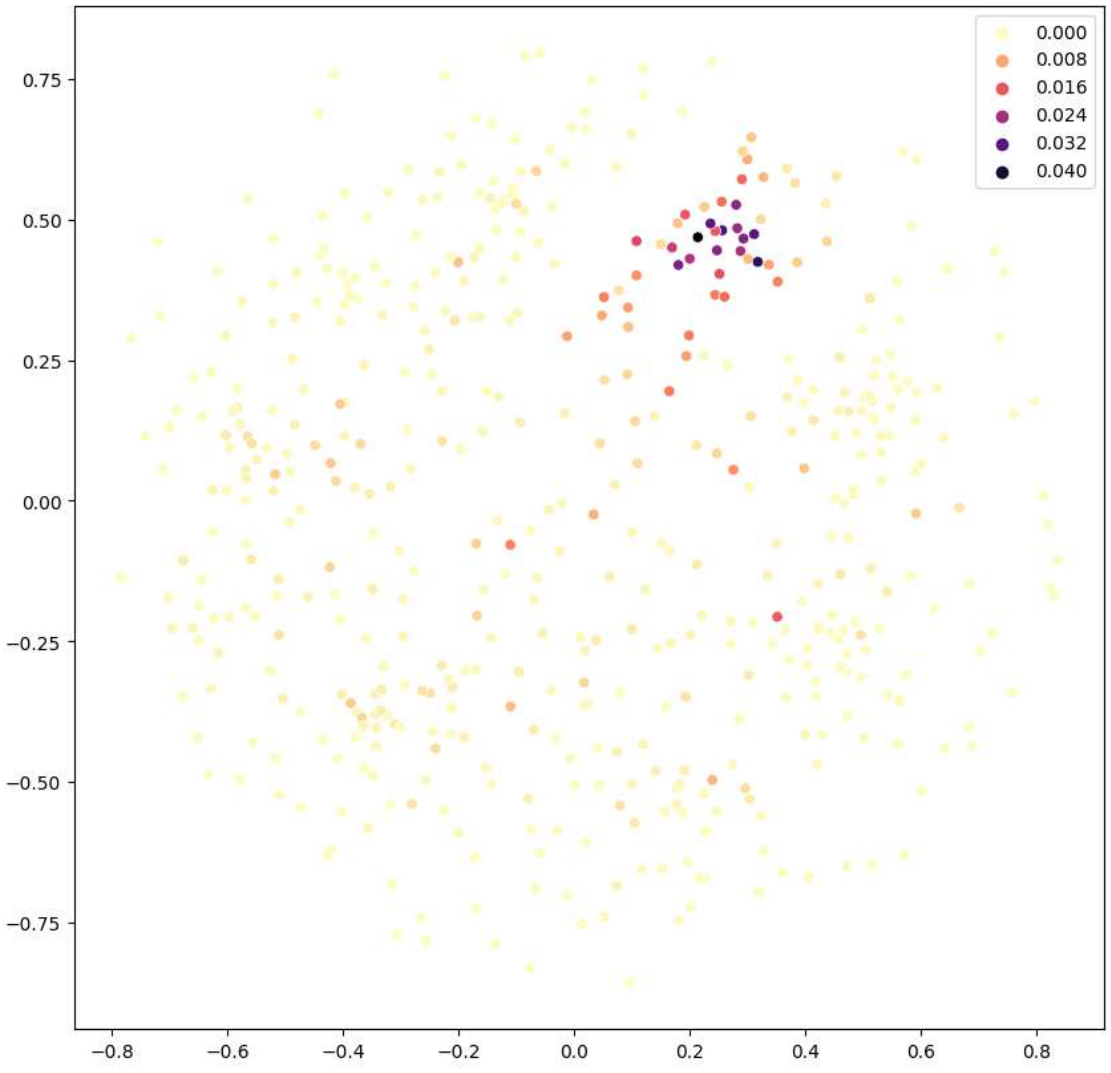


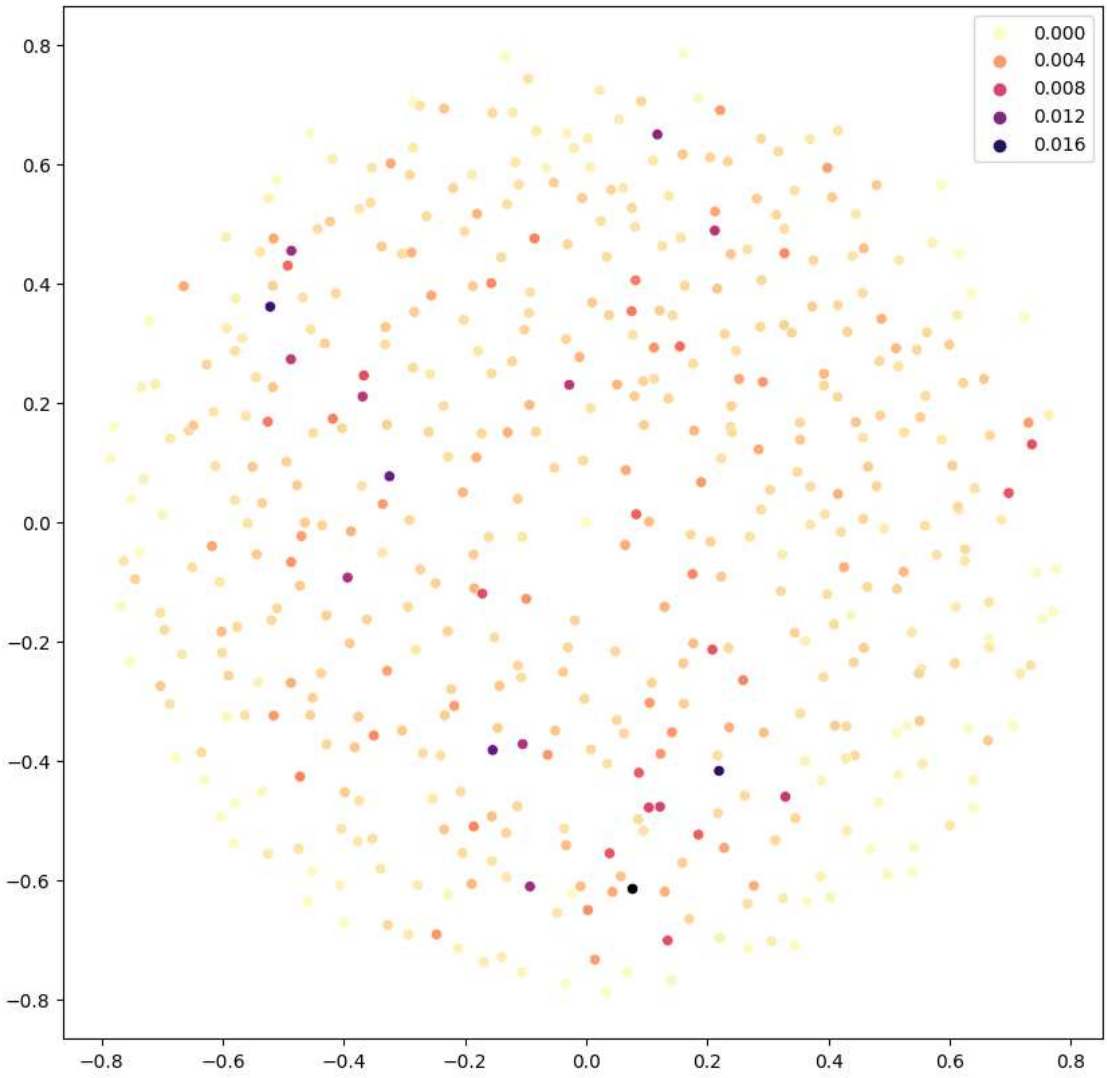
[ ]:
```
compare_projections(DataType.MNIST, 'mnist_mlp', n_layer=4, label=3)
```


0.000
0.008
0.016
0.024
0.032
0.040
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75



[ ]:
```
compare_projections(DataType.MNIST, 'mnist_cnn', n_layer=2, label=3)
```


0.000
0.008
0.016
0.024
0.032
0.040
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

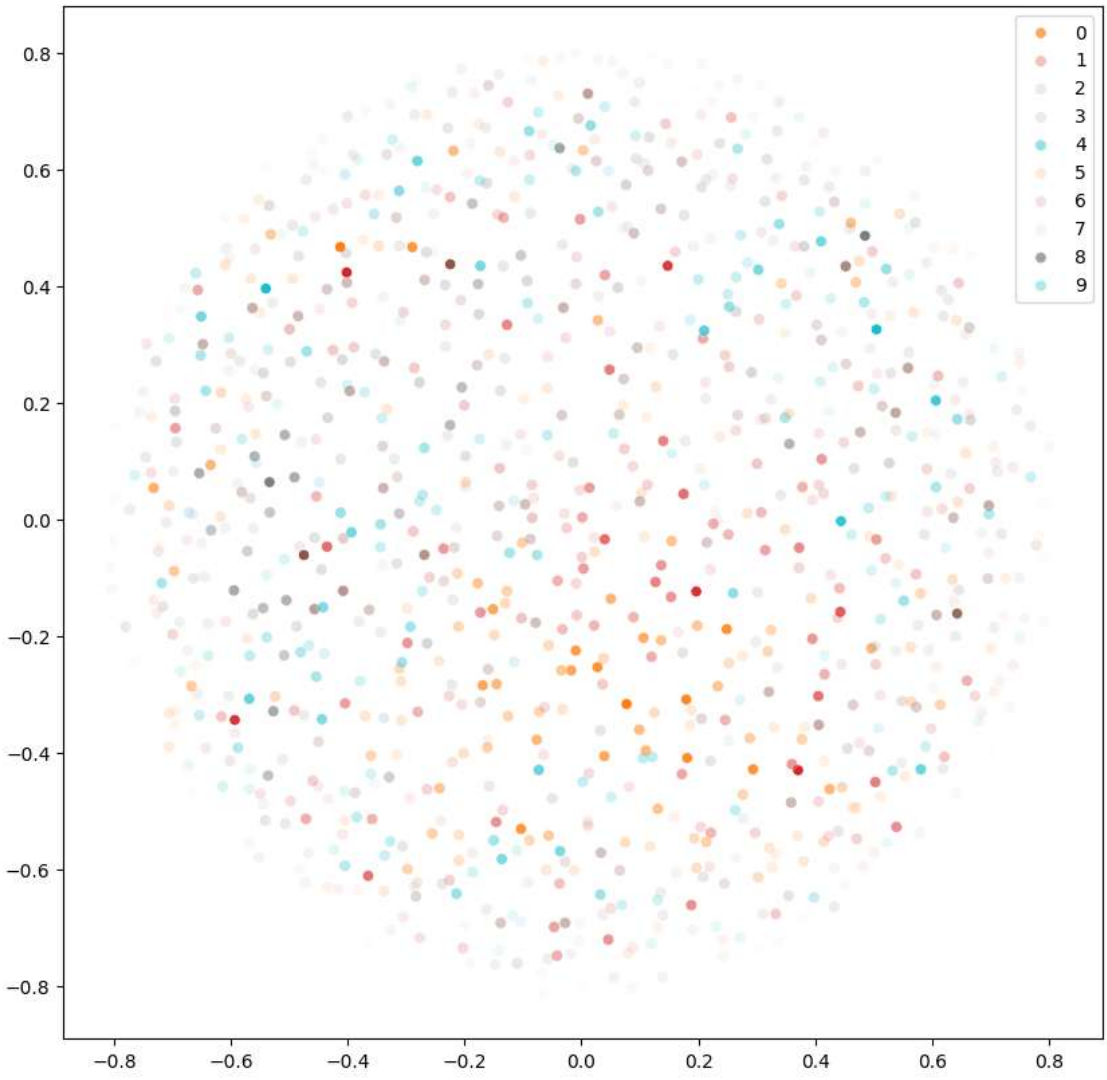


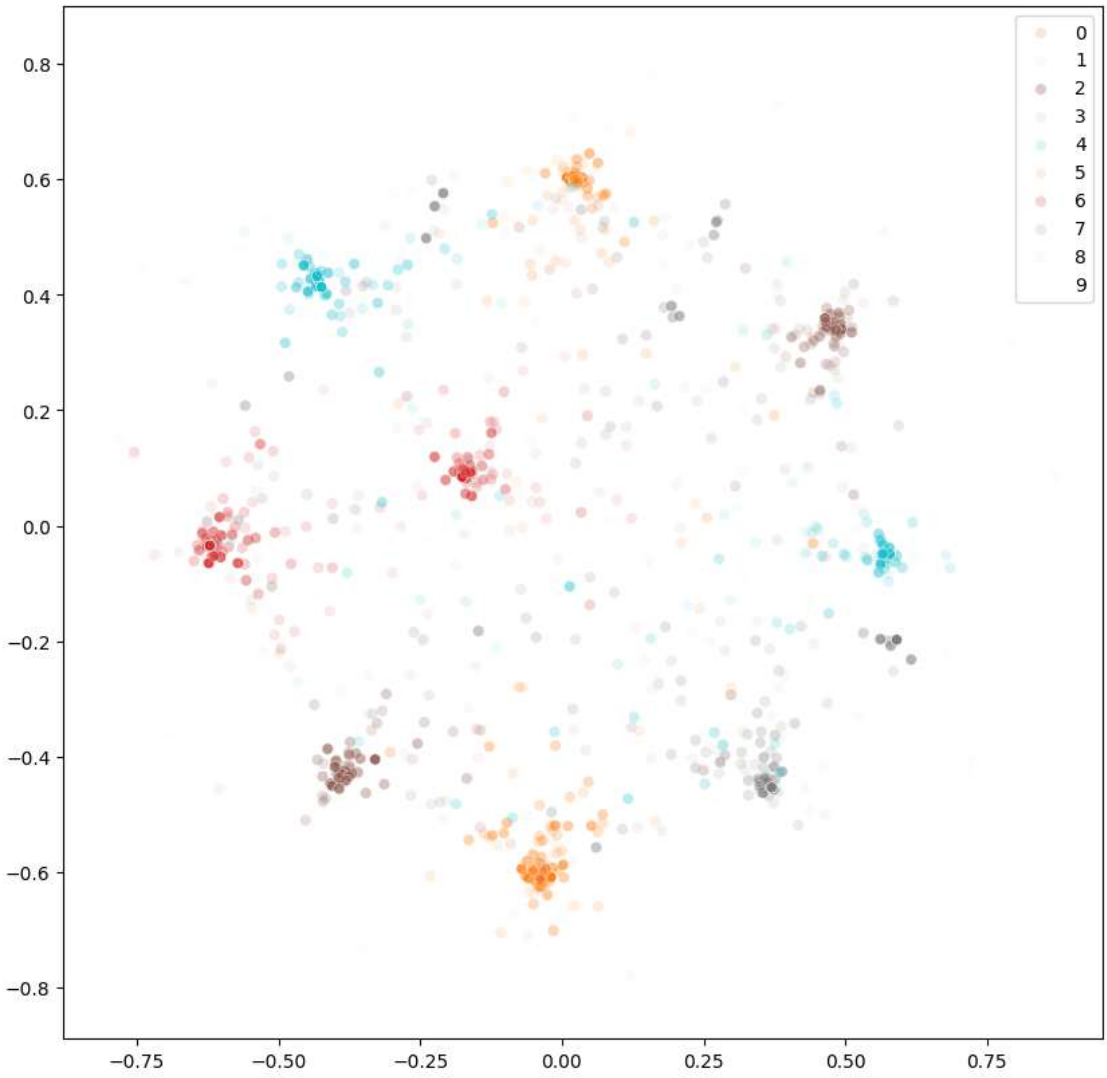
0.4 DISCRIMINATIVE NEURON MAP

For discriminative neuron map for the test subsets we use last hidden layer activations, after training. The presence of compact visual clusters shows how the entire set of neurons can be (almost) partitioned into groups with related discriminative roles (specializations), even though the neuron projection is created without any class information. The activation and neuron projections can be combined to elucidate the role of particular neurons.

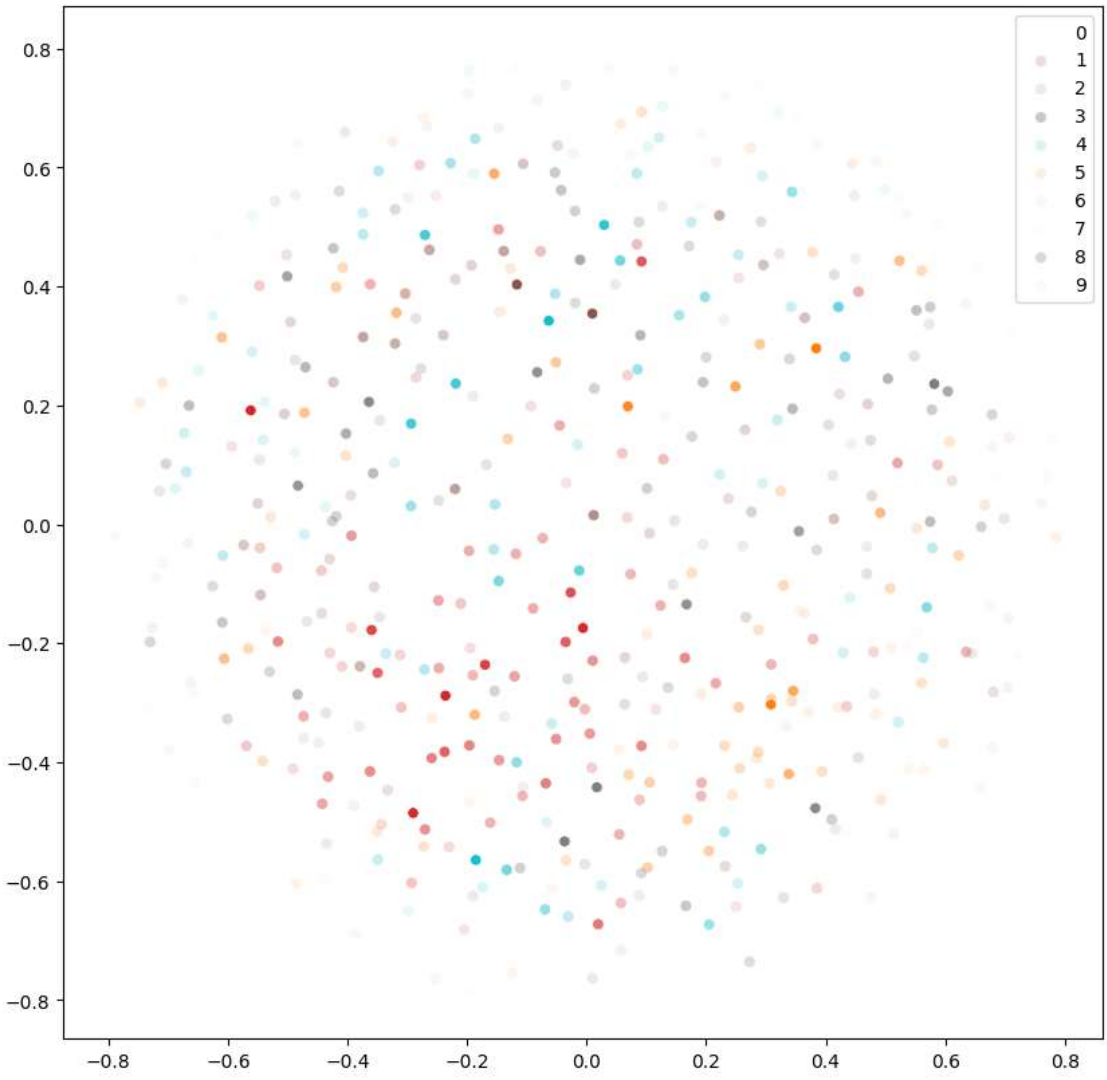
TODO: Utilize the `compare_discriminative_map` method to generate a discriminative neuron maps for both `mnist_mlp` and `mnist_cnn`. What observations can be made?

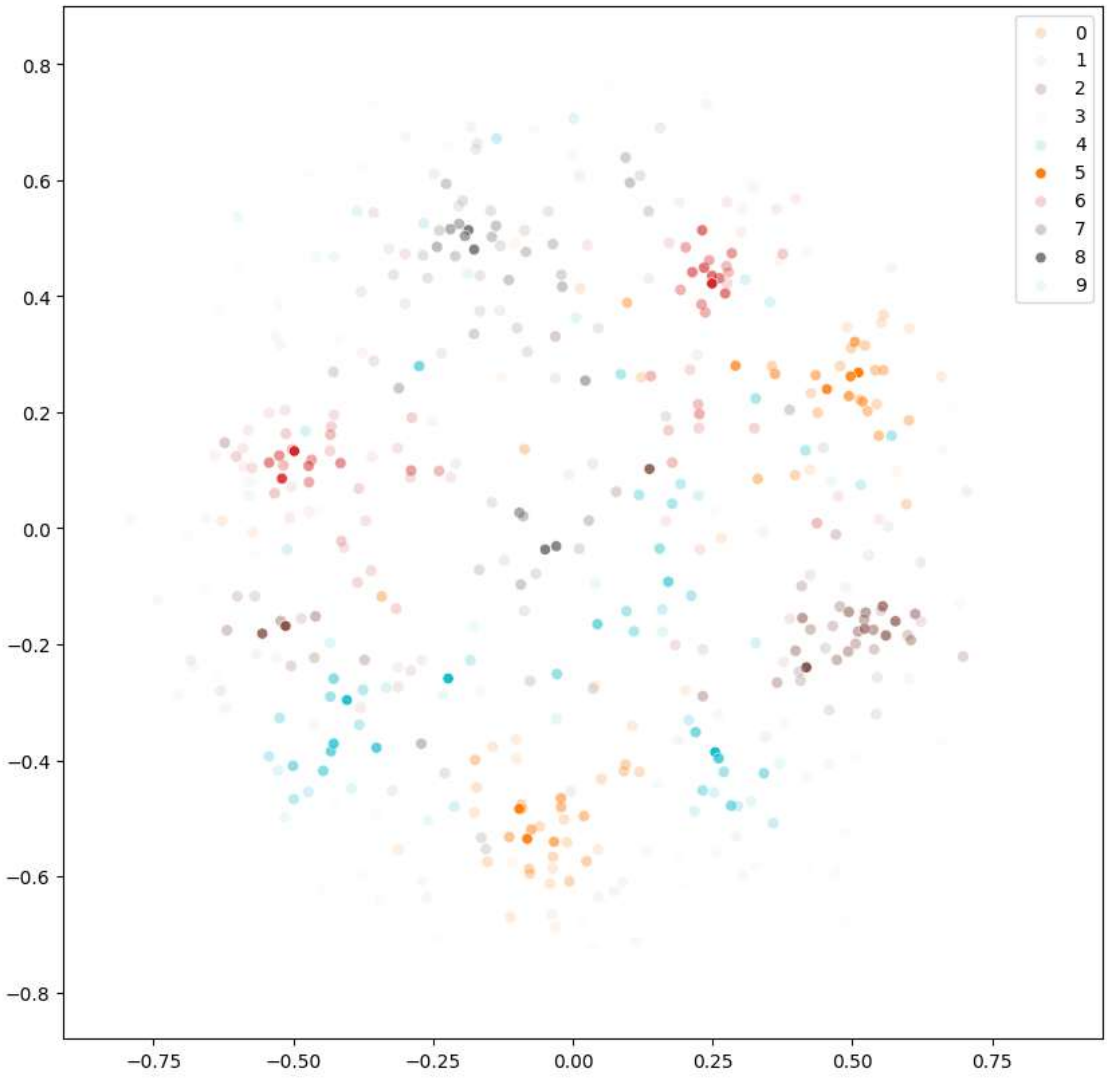
[ ]:
```
compare_discriminative_map(DataType.MNIST, 'mnist_mlp', n_layer=4, size=2000)
```


0
1
2
3
4
5
6
7
8
9
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8



[ ]:
```
compare_discriminative_map(DataType.MNIST, 'mnist_cnn', n_layer=2, size=2000)
```


0
1
2
3
4
5
6
7
8
9
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

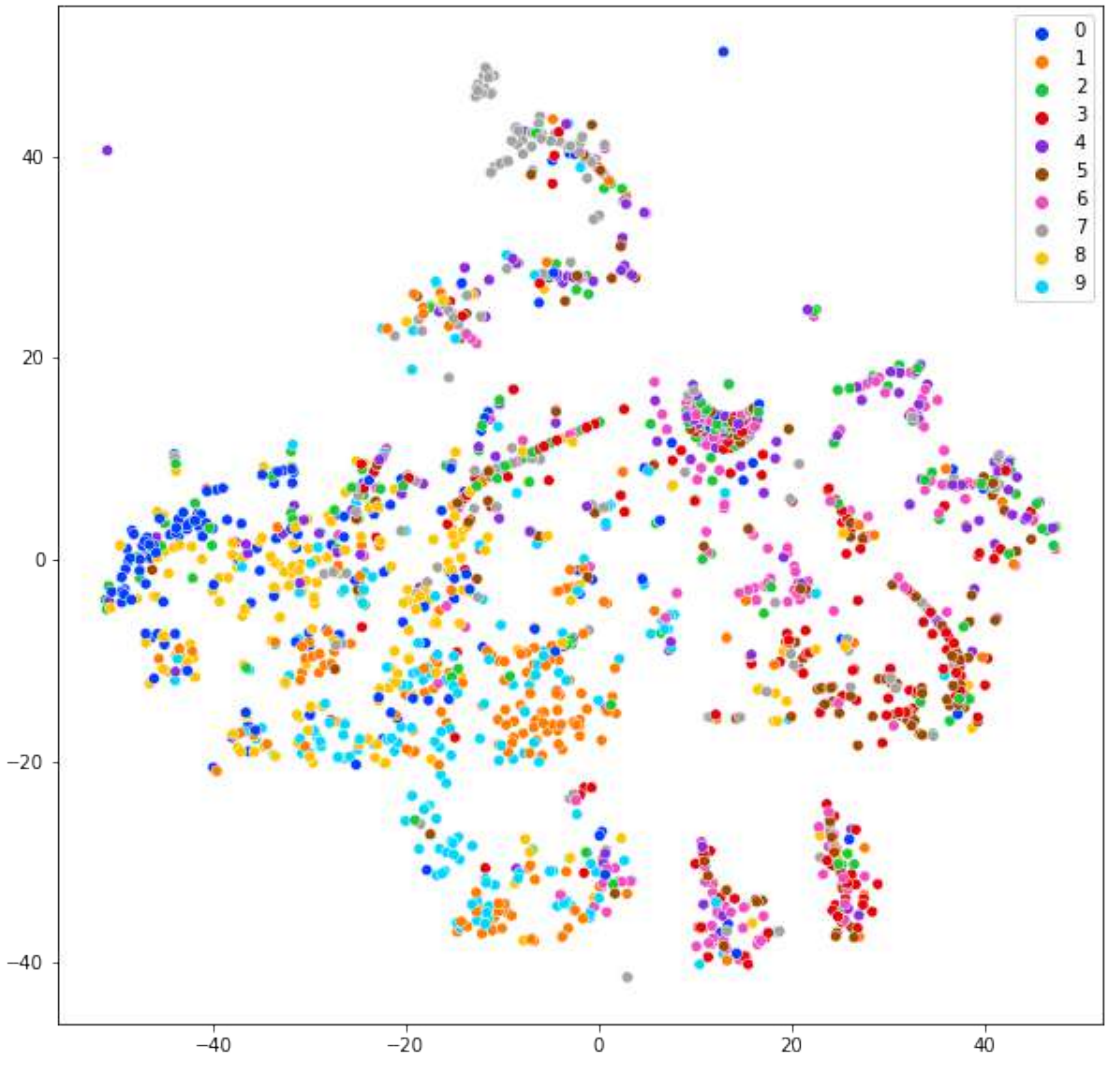


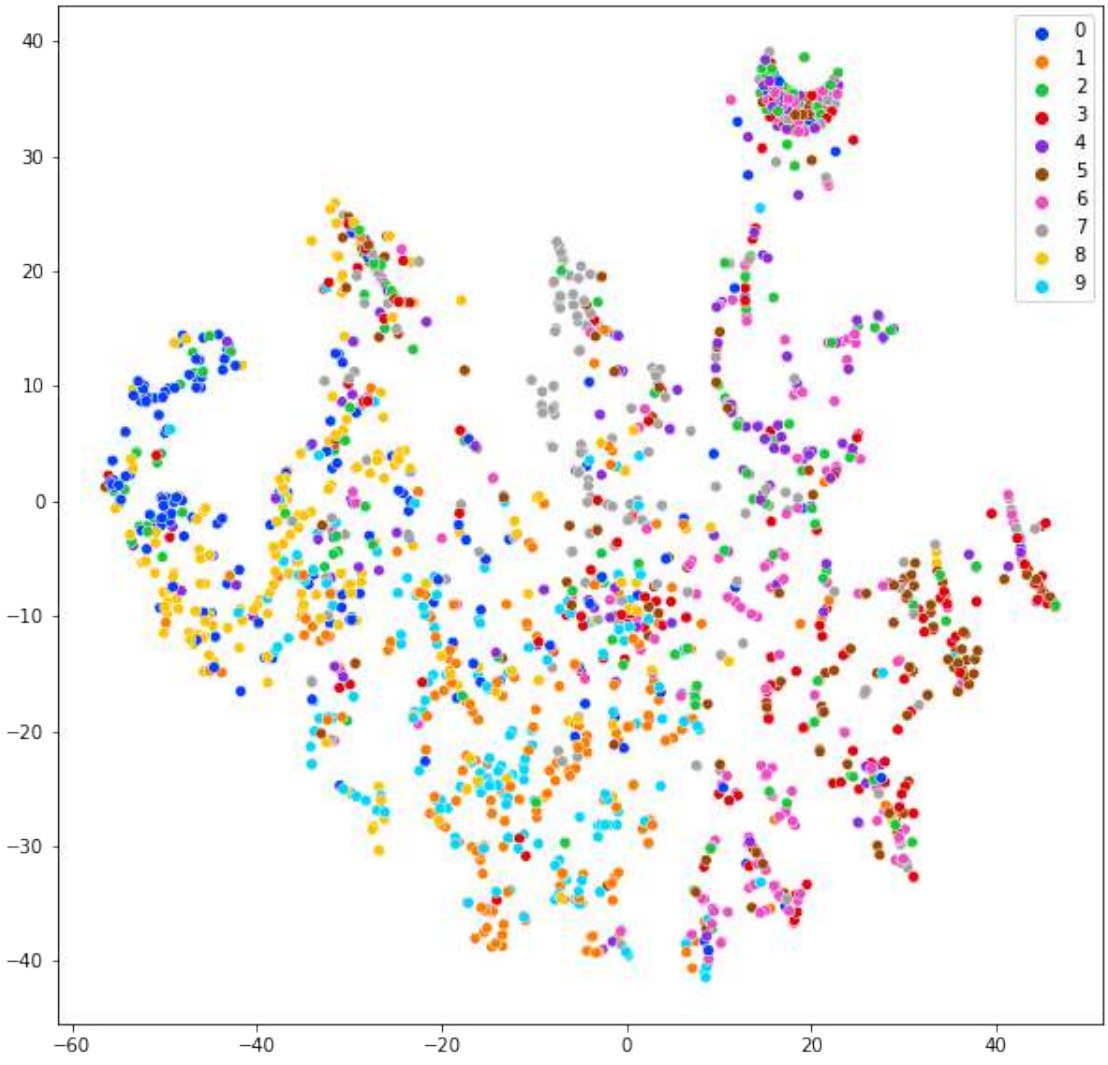
0.5 Homework

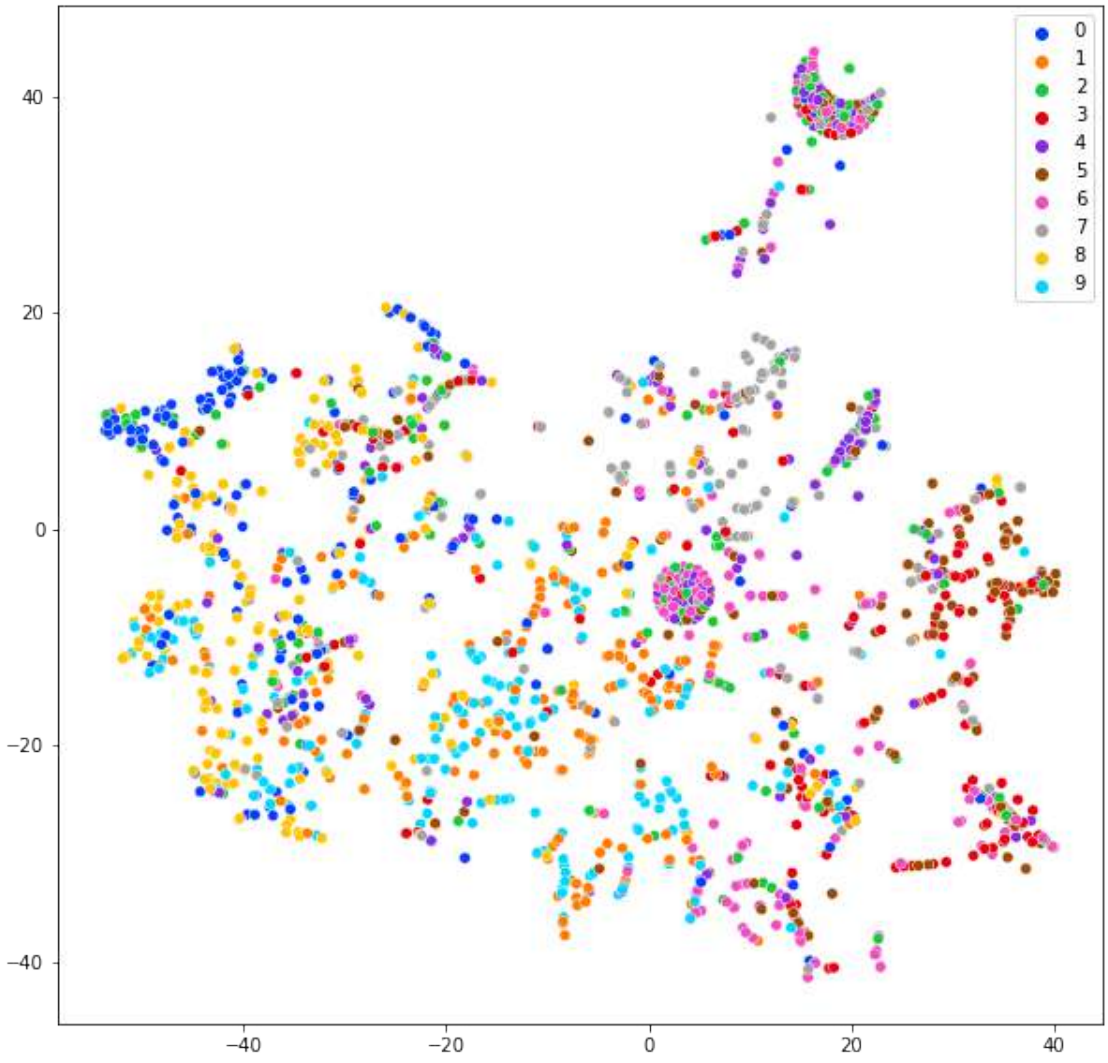
Repeat the analysis for the network parameters of `cifar_mlp` and `cifar_cnn`.

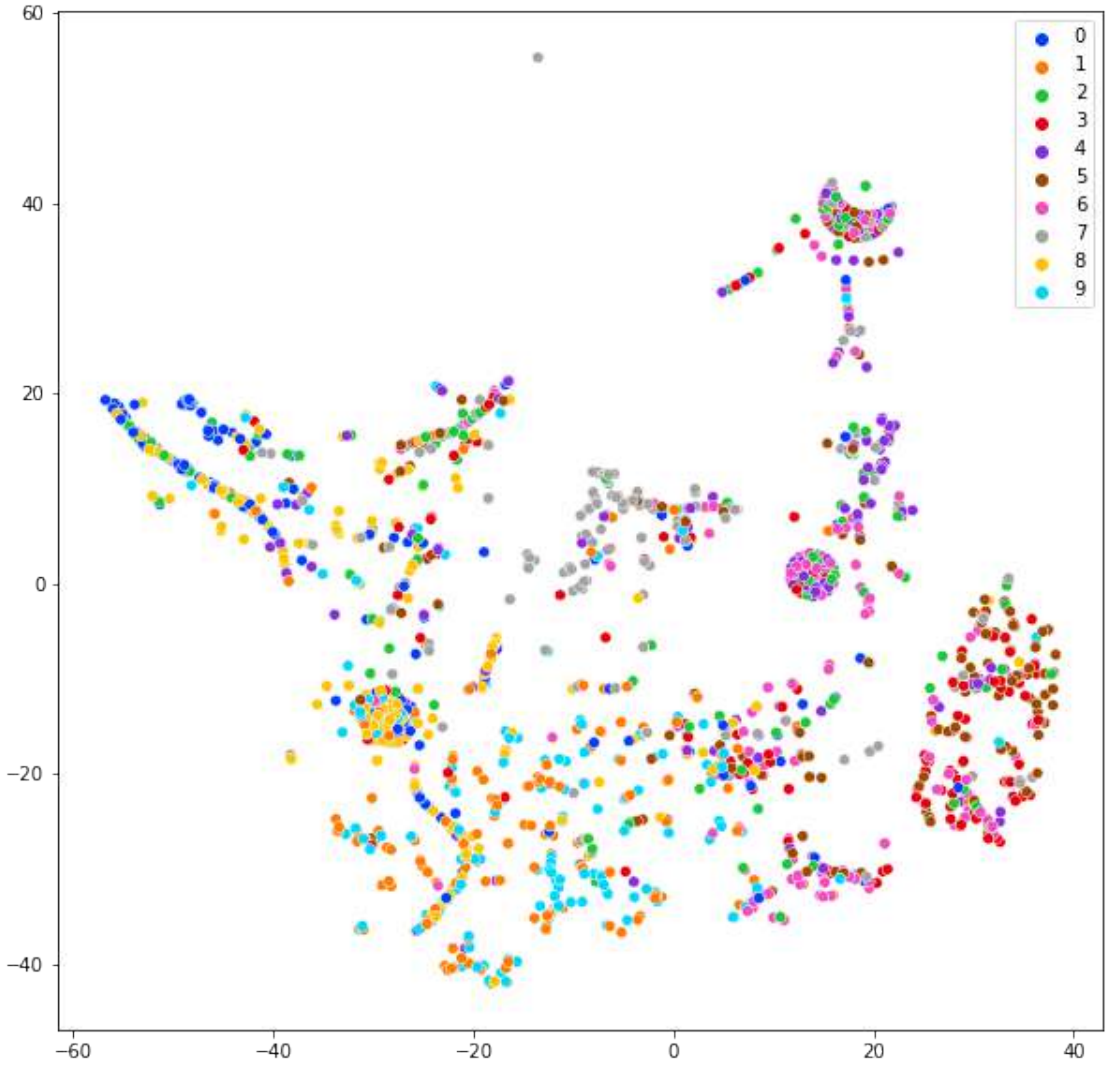
[ ]:
```
import analysis.network_analysis as na
from network.constants import DataType

points_bucket = []
for n_layer in range(0, 4):
    points, targets = na.show_seq_projections(DataType.CIFAR, 'cifar_mlp',␣
 ↪n_layer=n_layer, epoch=100, size=2000)
    points_bucket.append(points)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 202s 1us/step
```

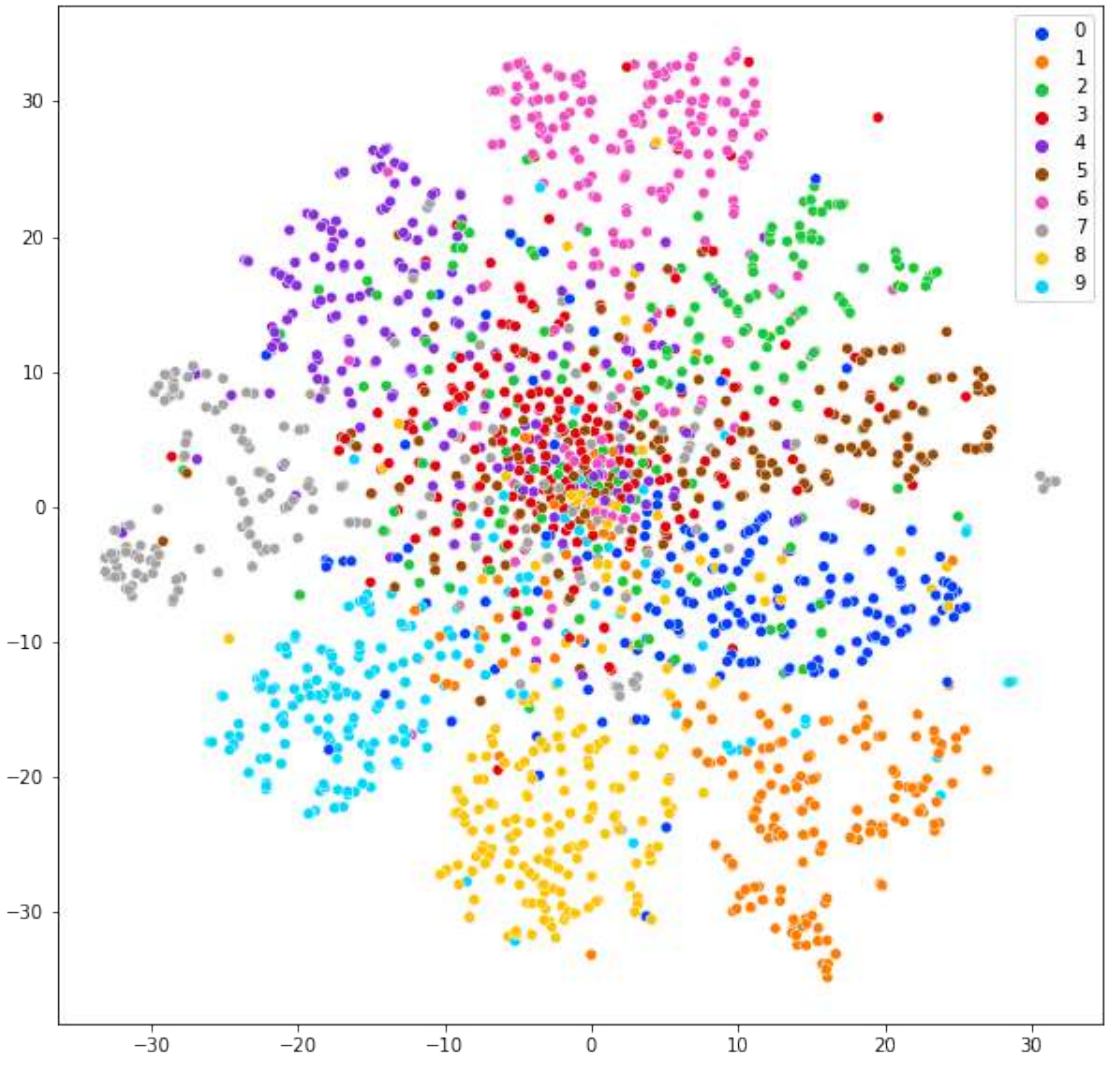

0
1
2
3
4
5
6
7
8
9
40
20
0
−20
−40
−40
−20
0
20
40

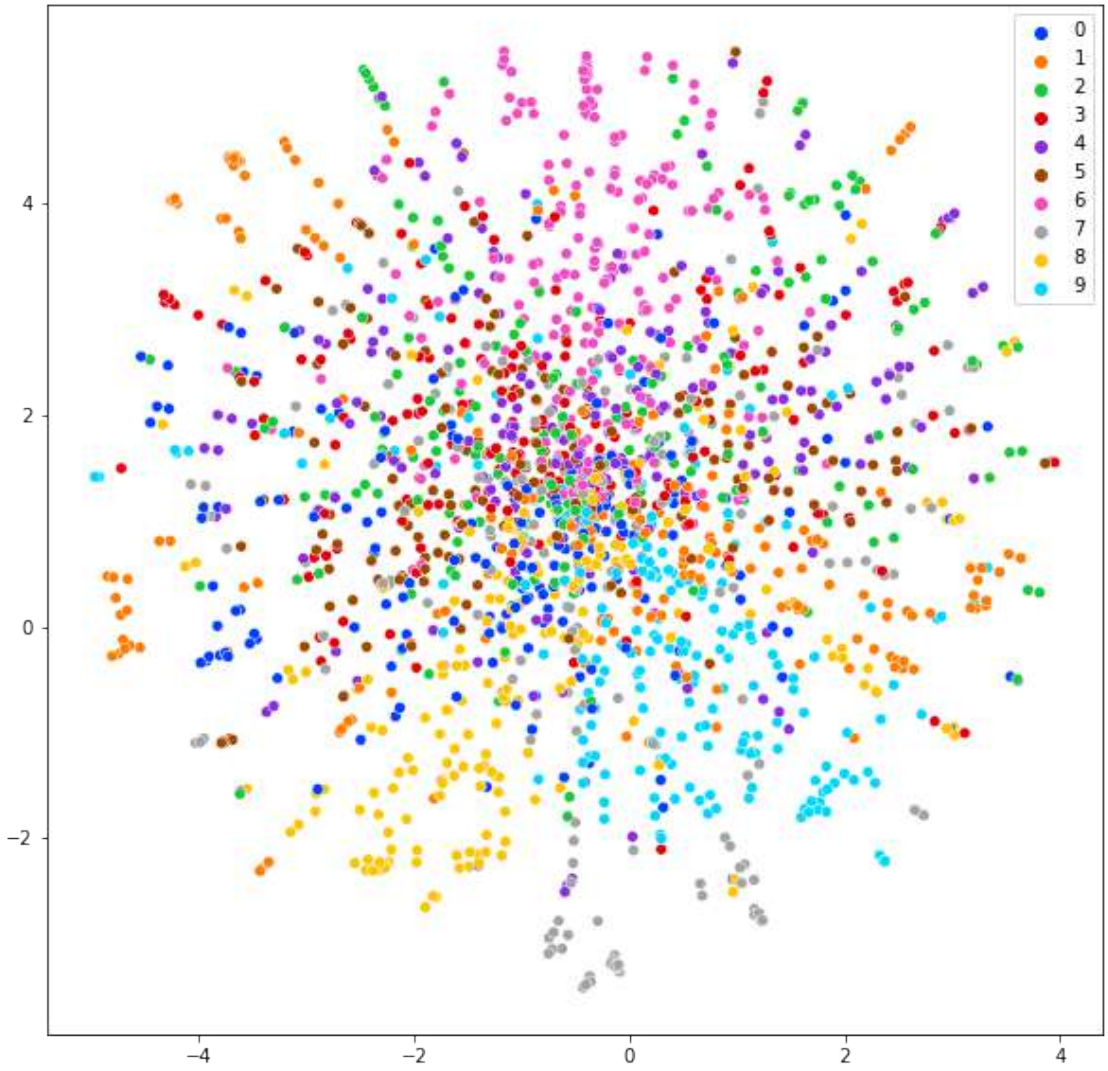

0
1
2
3
4
5
6
7
8
9
40
30
20
10
0
−10
−20
−30
−40
−60
−40
−20
0
20
40

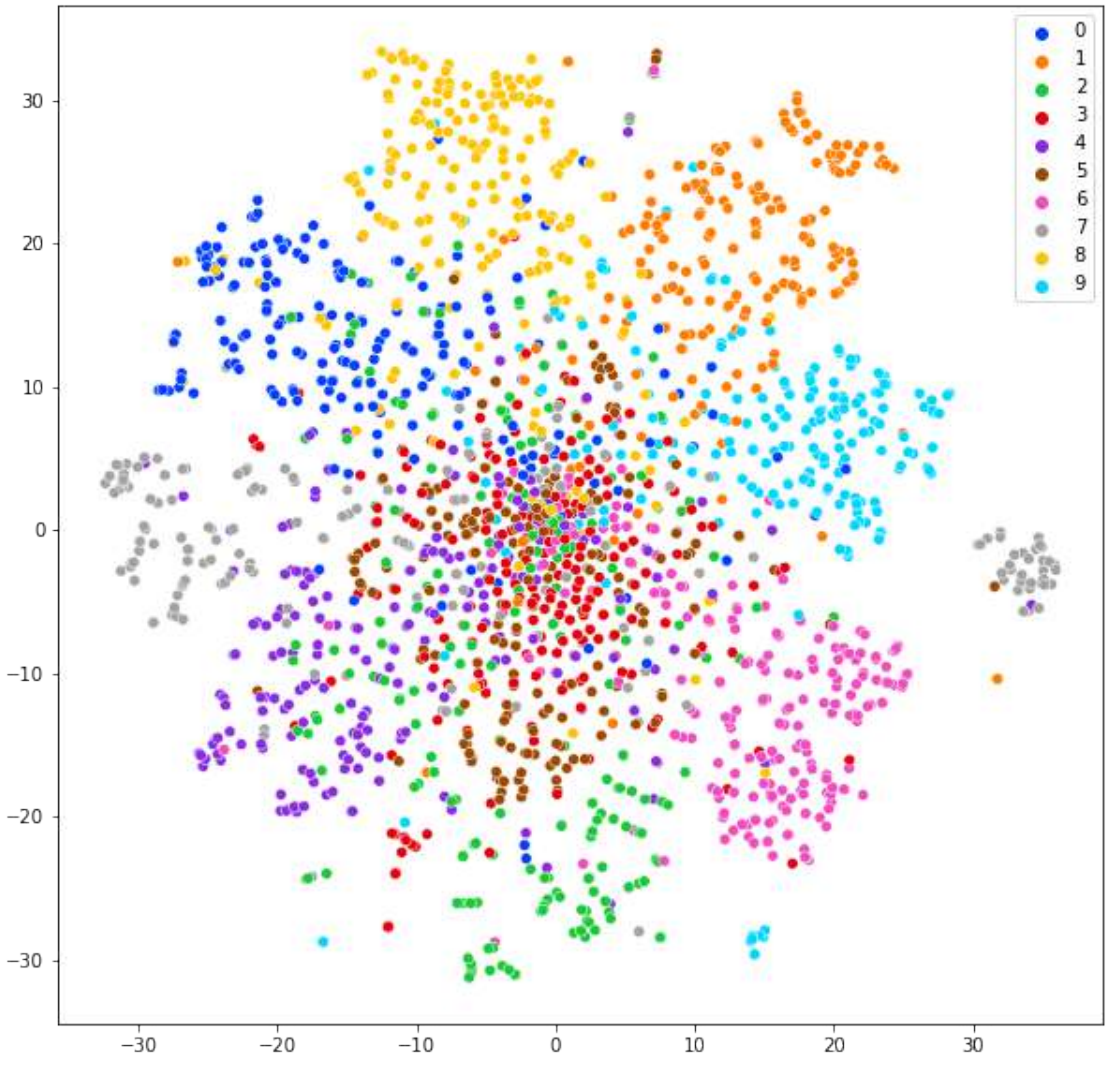

0
1
2
3
4
5
6
7
8
9
40
20
0
−20
−40
−40
−20
0
20
40



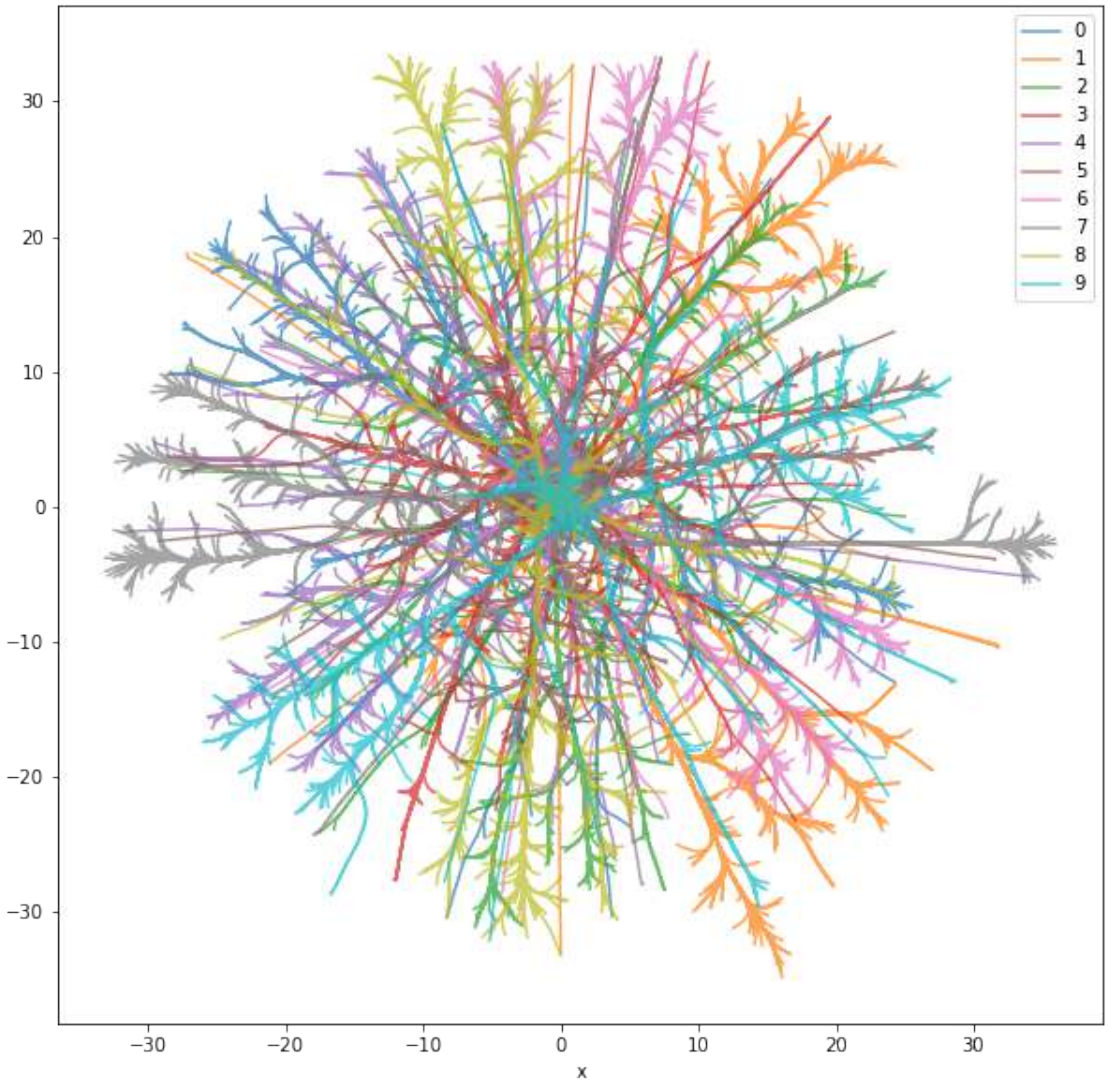
[ ]:
```
points_bucket_cnn = []
for n_layer in range(3):
    points, targets = na.show_seq_projections(DataType.CIFAR, 'cifar_cnn',␣
 ↪n_layer=n_layer, epoch=100, size=2000)
    points_bucket_cnn.append(points)
```


0
1
2
3
4
5
6
7
8
9
30
20
10
0
−10
−20
−30
−30
−20
−10
0
10
20
30

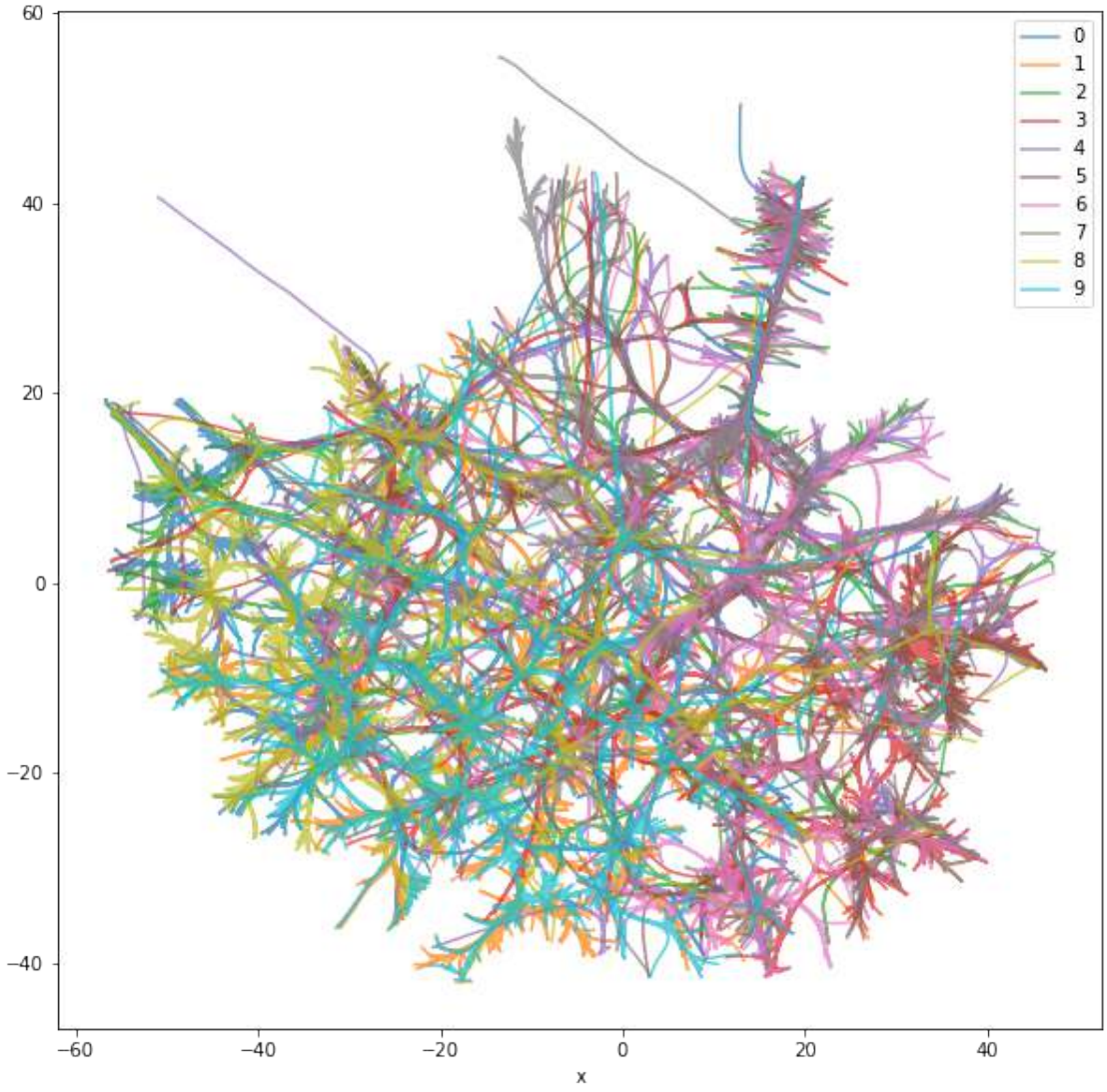

0
1
2
3
4
5
6
7
8
9
4
2
0
−2
−4
−2
0
2
4



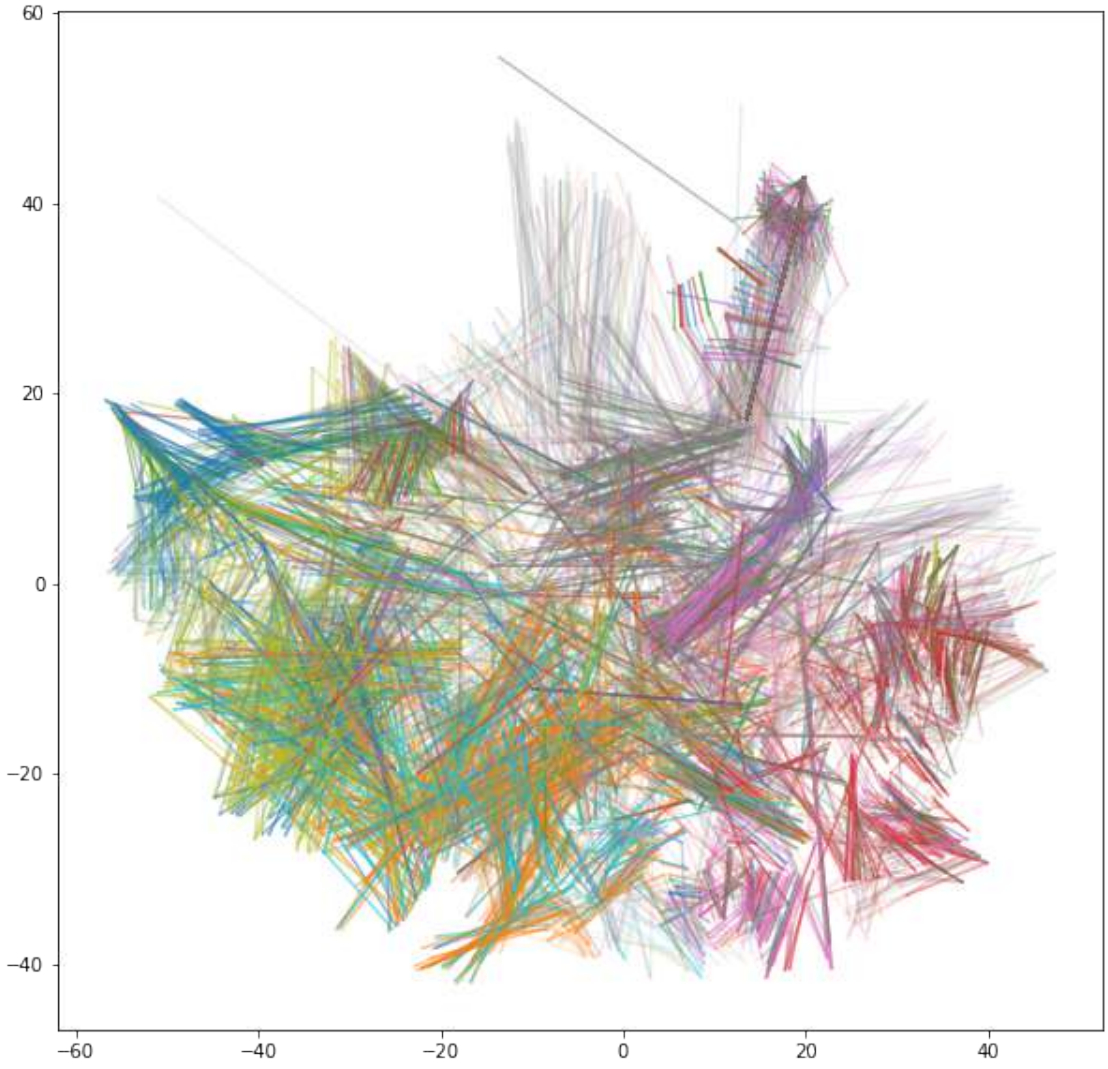
[ ]:
```
inter_layer_evolution(points_bucket_cnn, targets)
```



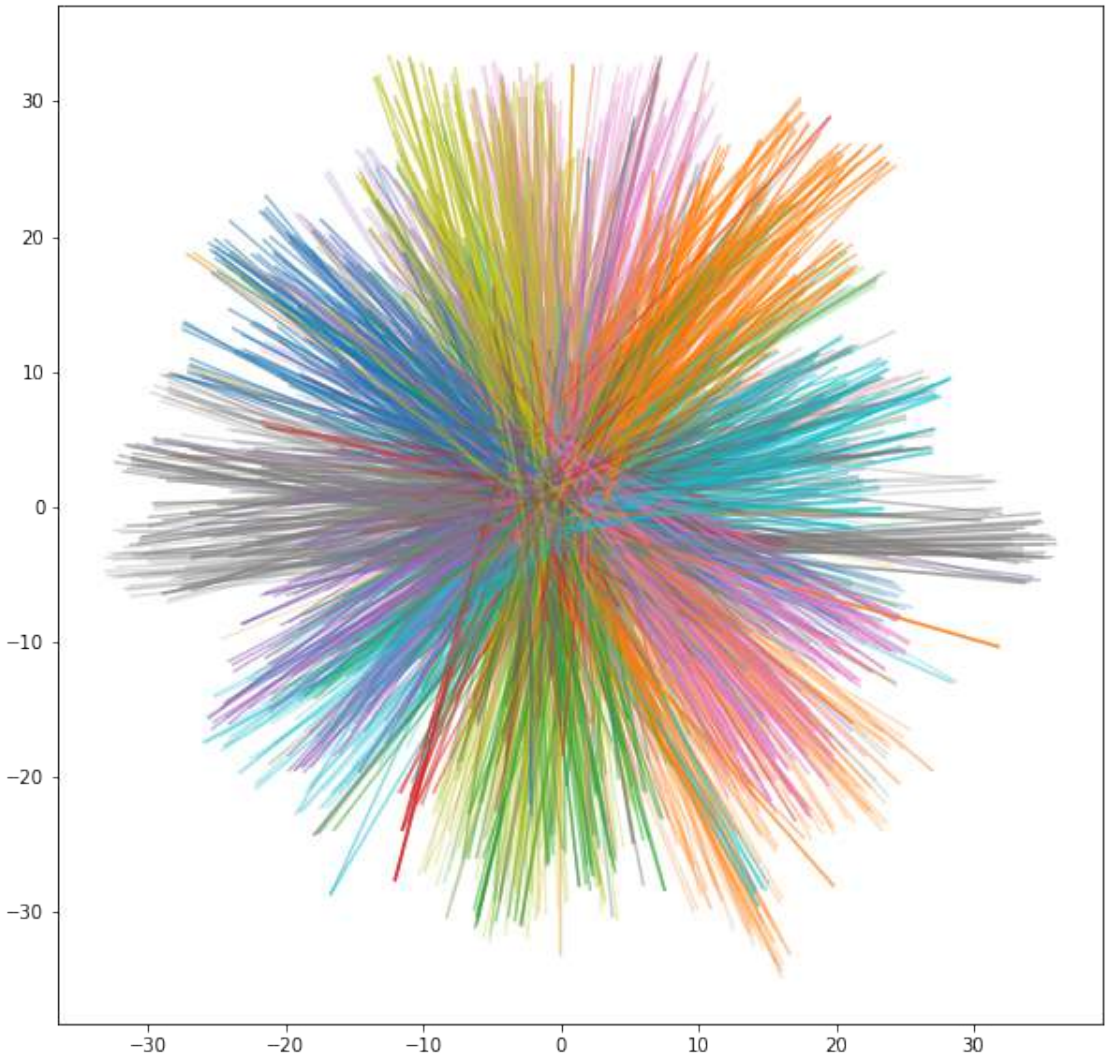
[ ]:
```
inter_layer_evolution(points_bucket, targets)
```



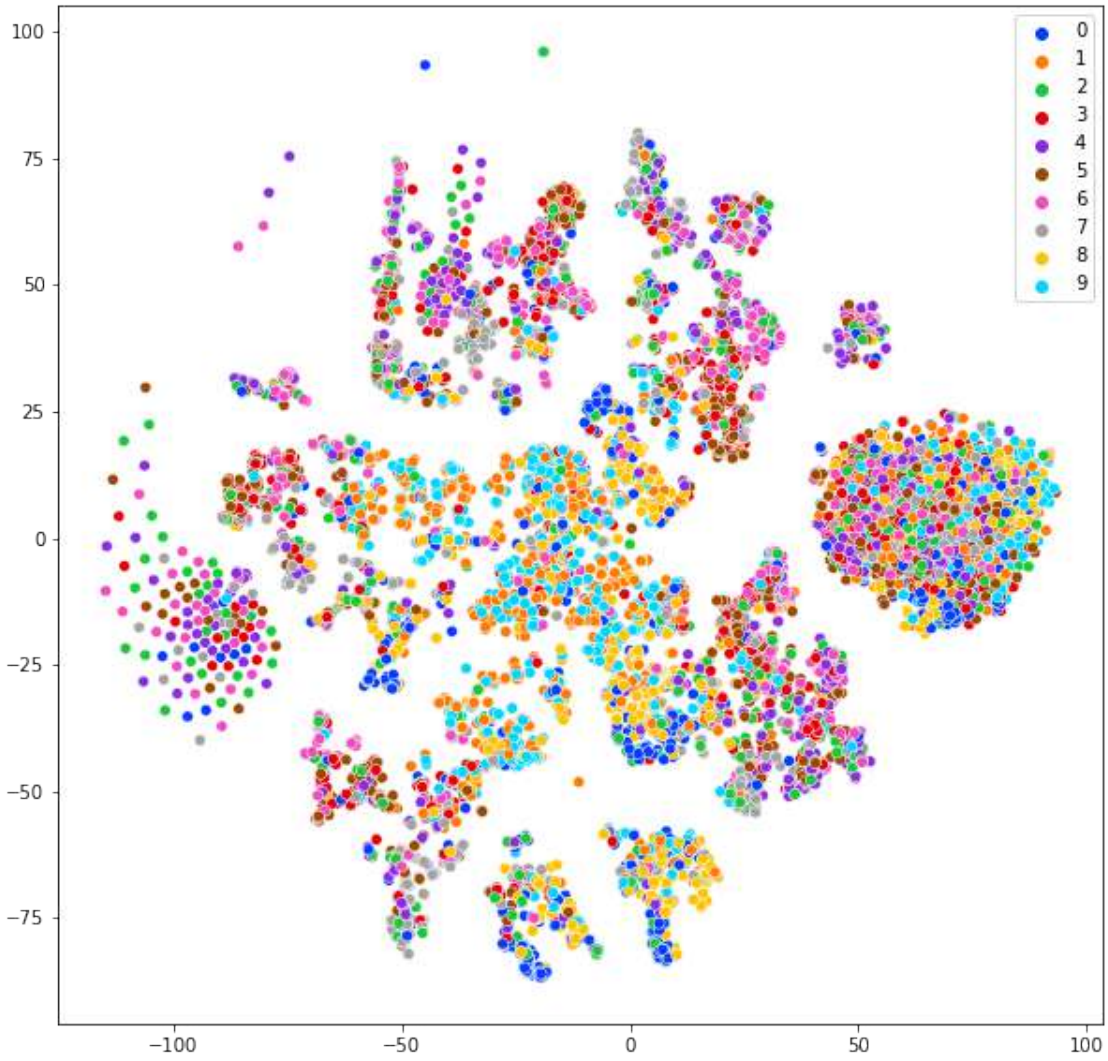
[ ]:
```
show_trace(points_bucket, targets)
```



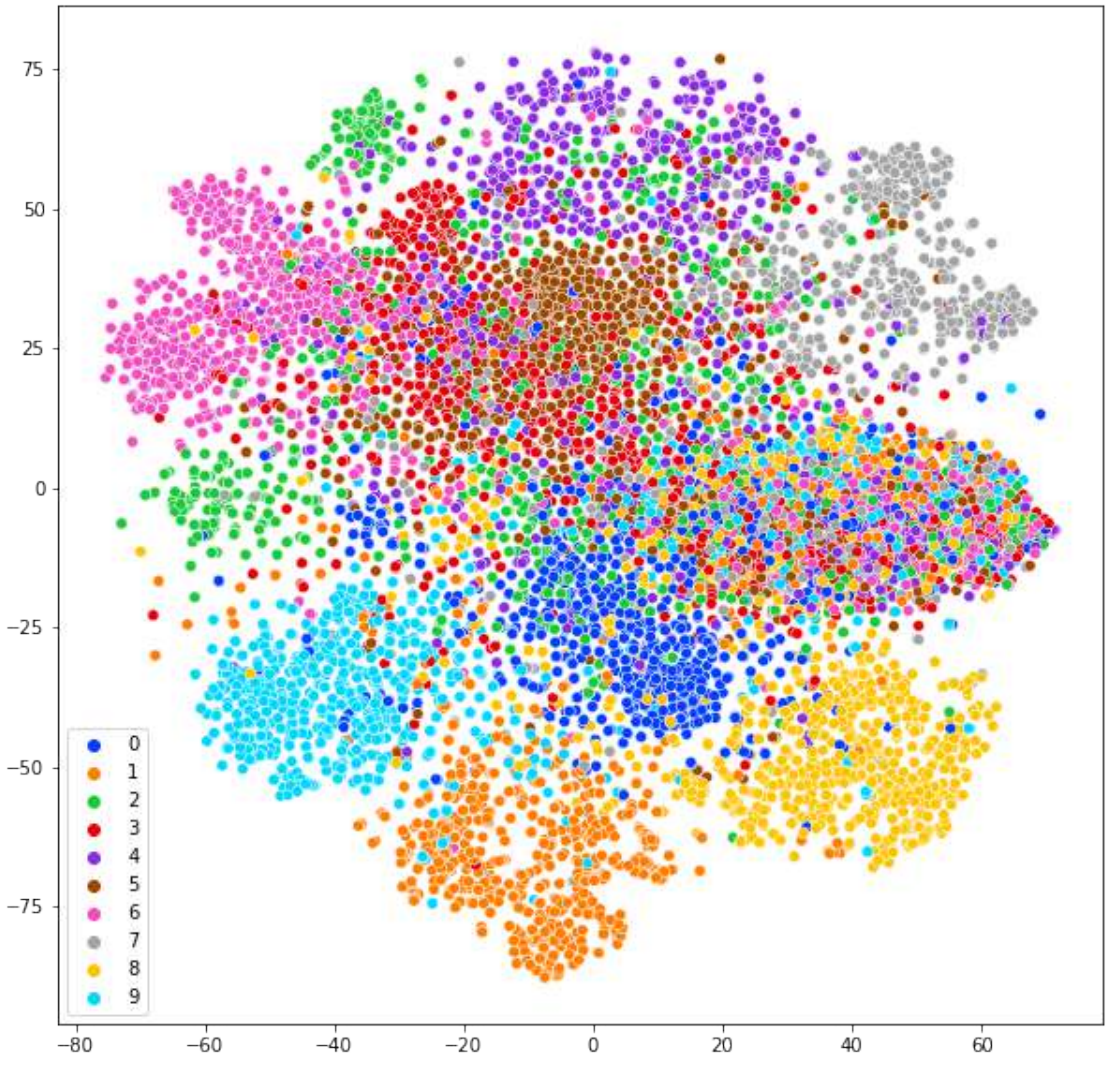
[ ]:
```
show_trace(points_bucket_cnn, targets)
```



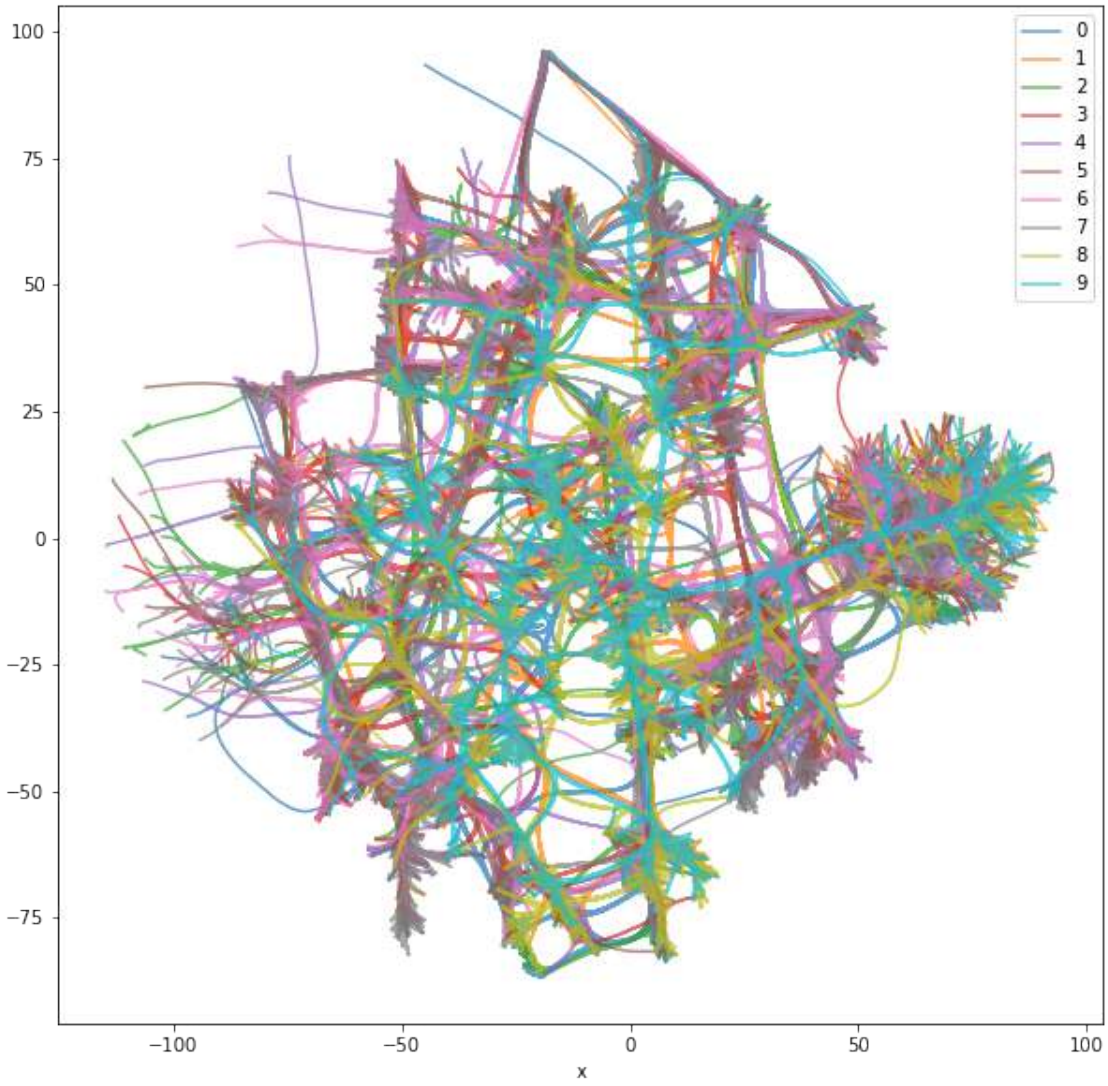
[ ]:
```
act, targets = get_all_activations(DataType.CIFAR, 'cifar_mlp', n_layer=4,␣
↪size=2000)
points, targets =  process_activations(act, targets, size=2000)
```



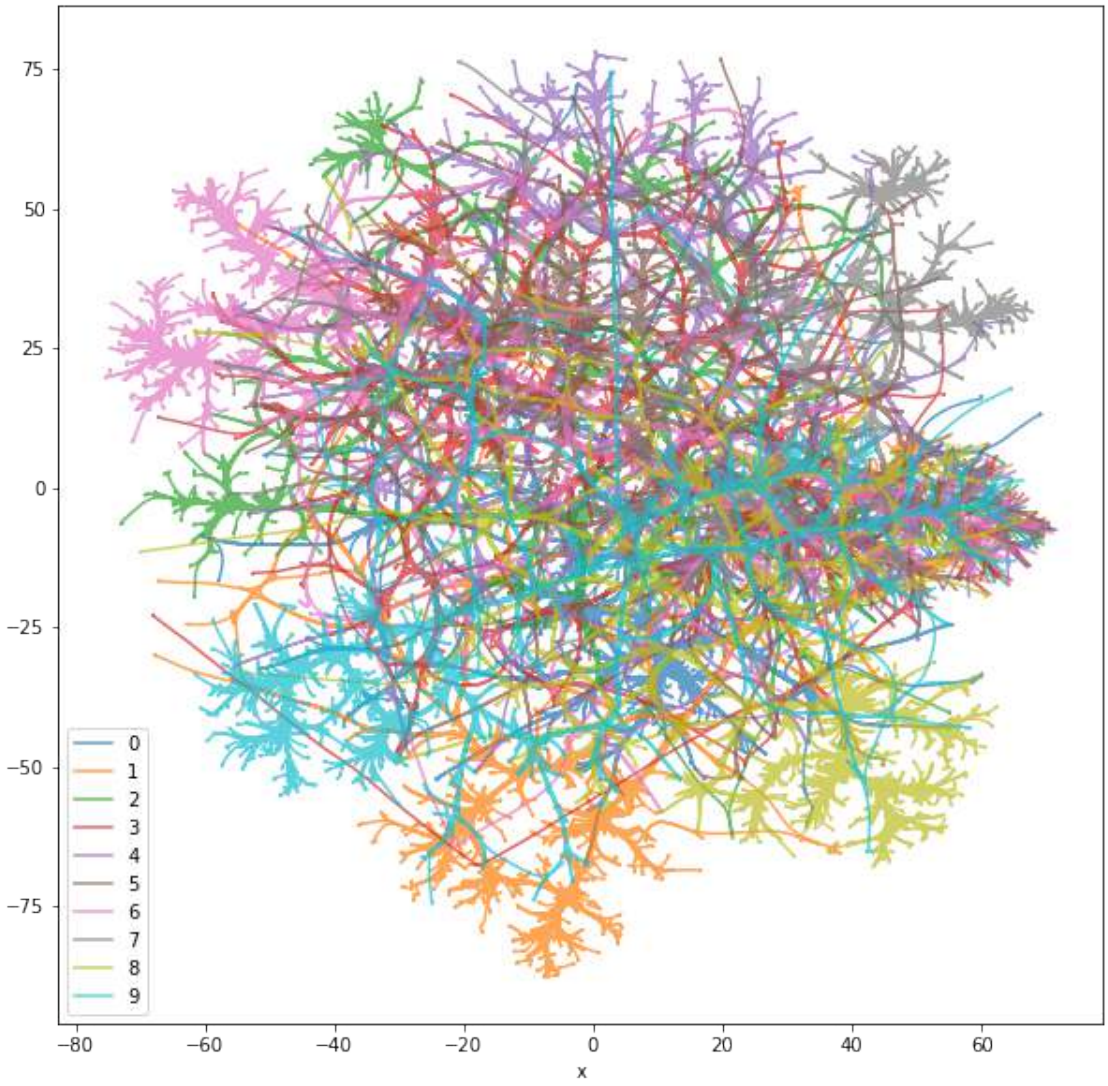
[ ]:
```
act_cnn, targets_cnn = get_all_activations(DataType.CIFAR, 'cifar_cnn',␣
↪n_layer=2, size=2000)
points_cnn, targets_cnn =  process_activations(act_cnn, targets_cnn, size=2000)
```



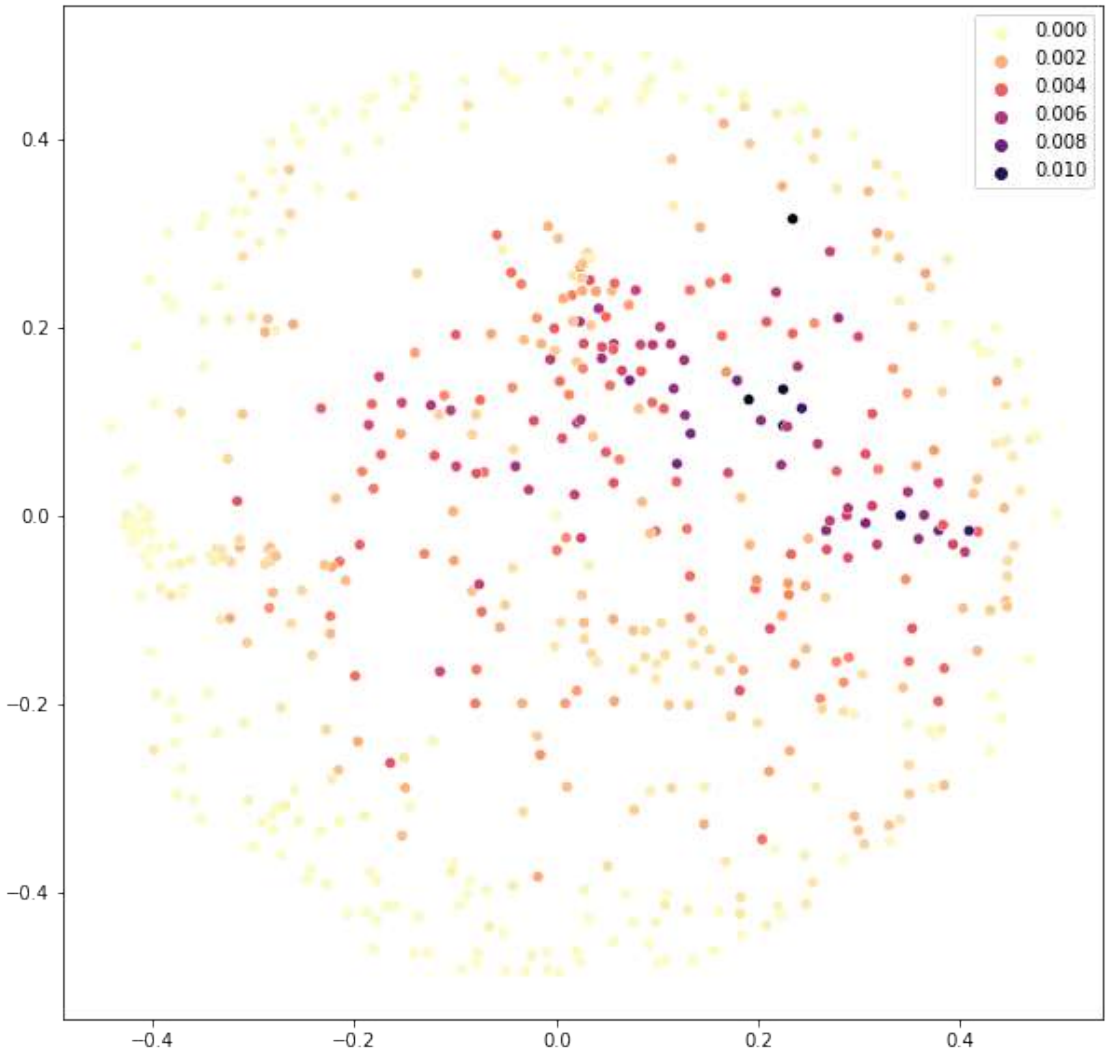
[ ]:
```
inter_epoch_evolution(points, targets)
```

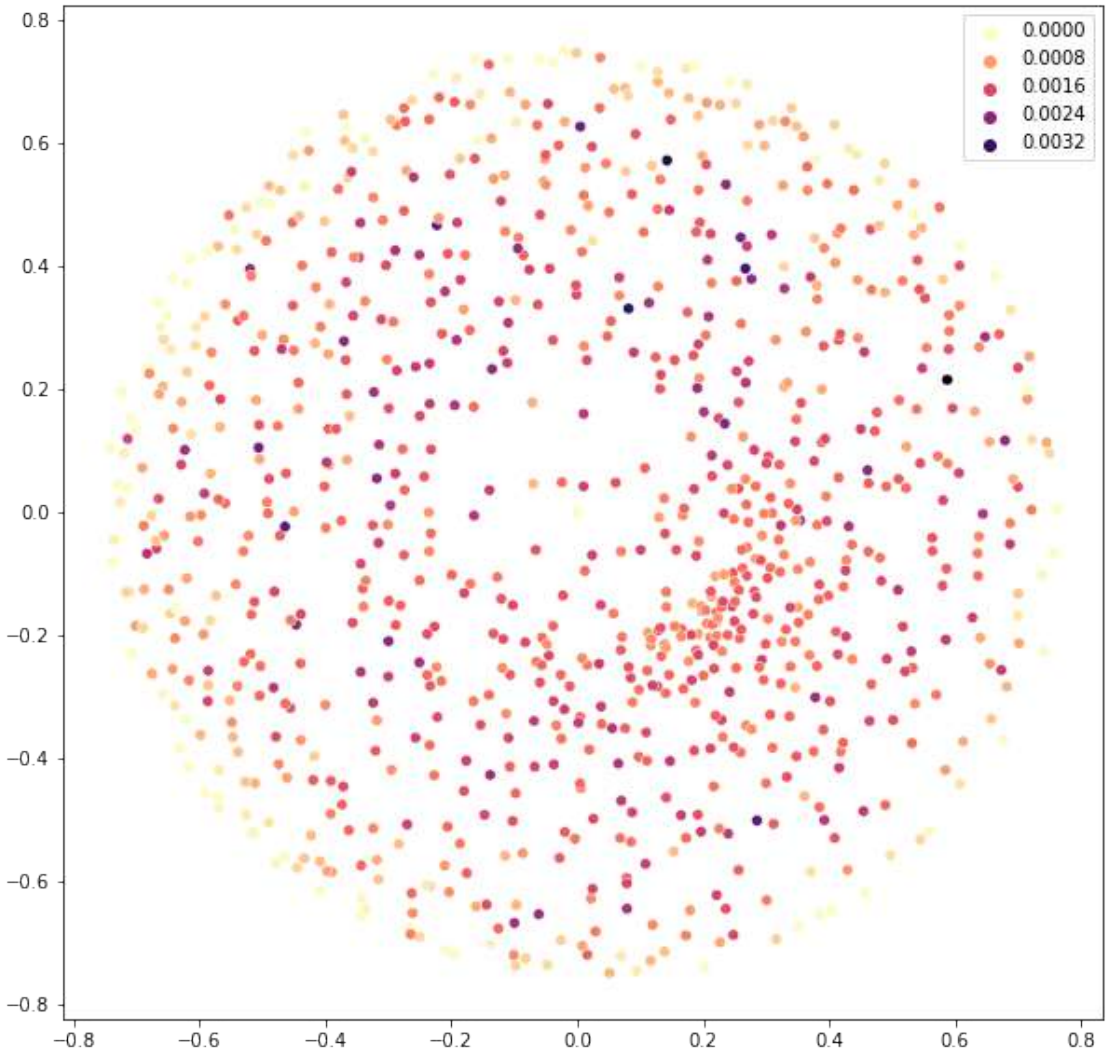


[ ]:
```
inter_epoch_evolution(points_cnn, targets_cnn)
```

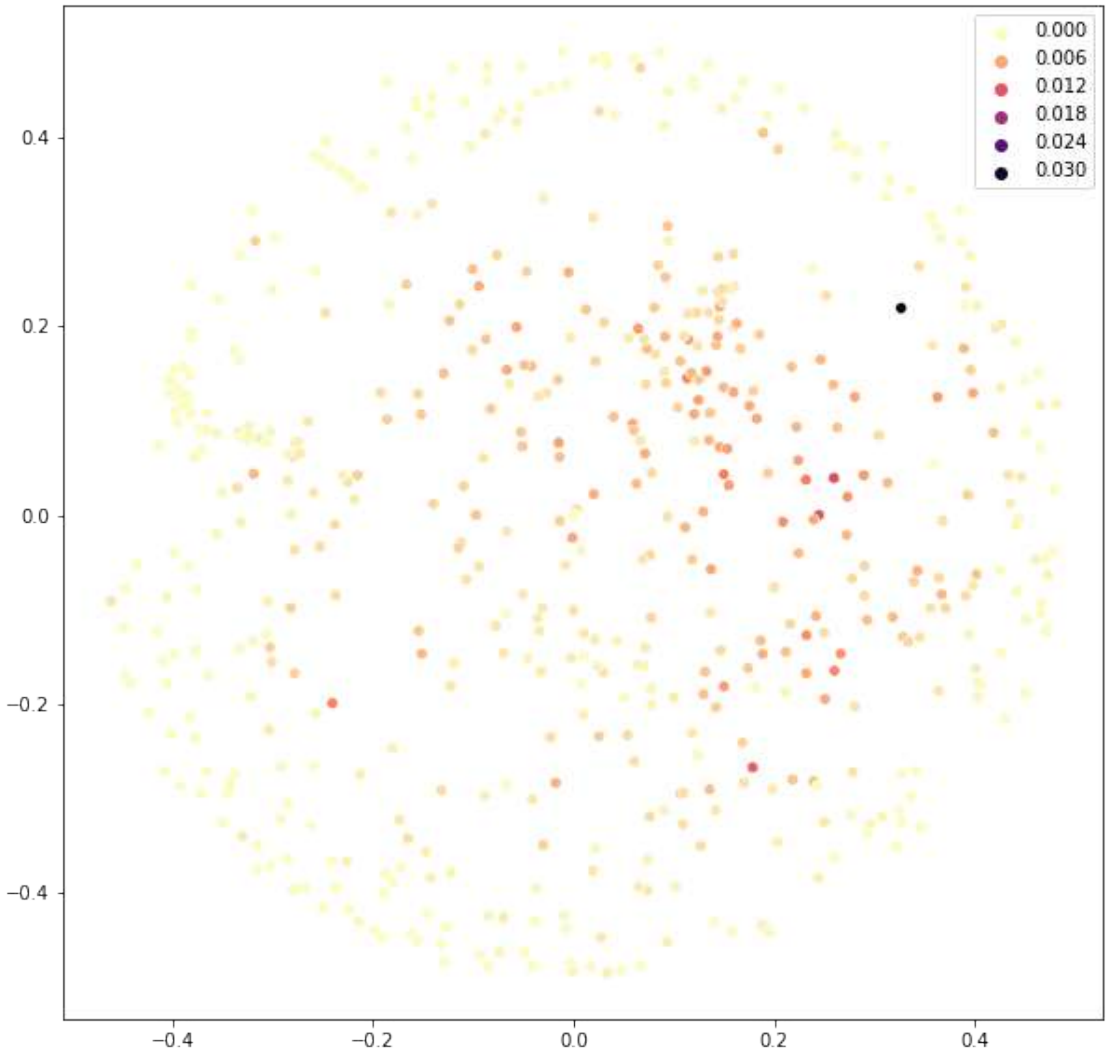


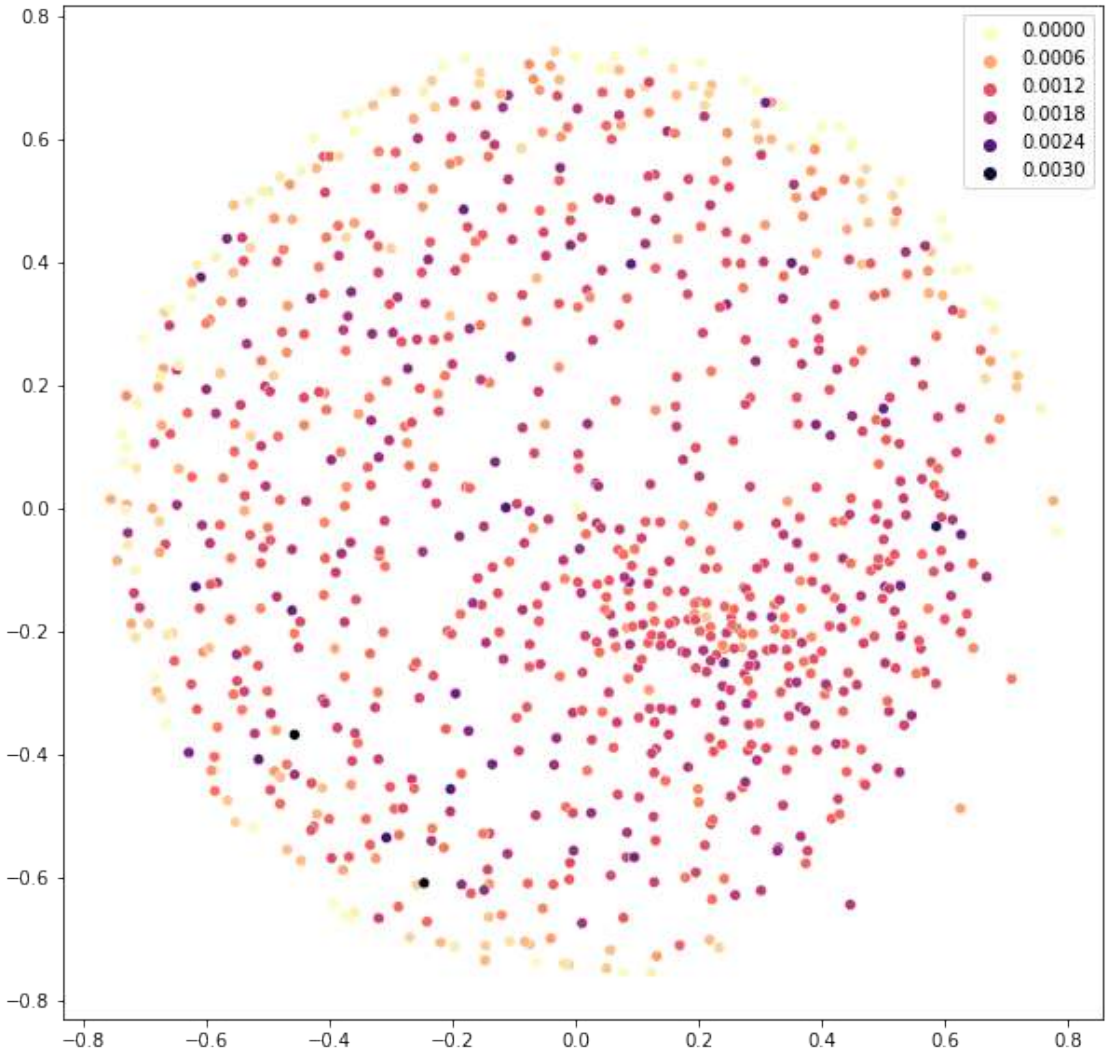
[ ]:
```
compare_projections(DataType.CIFAR, 'cifar_mlp', n_layer=4, label=5)
```


0.000
0.002
0.004
0.006
0.008
0.010
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4

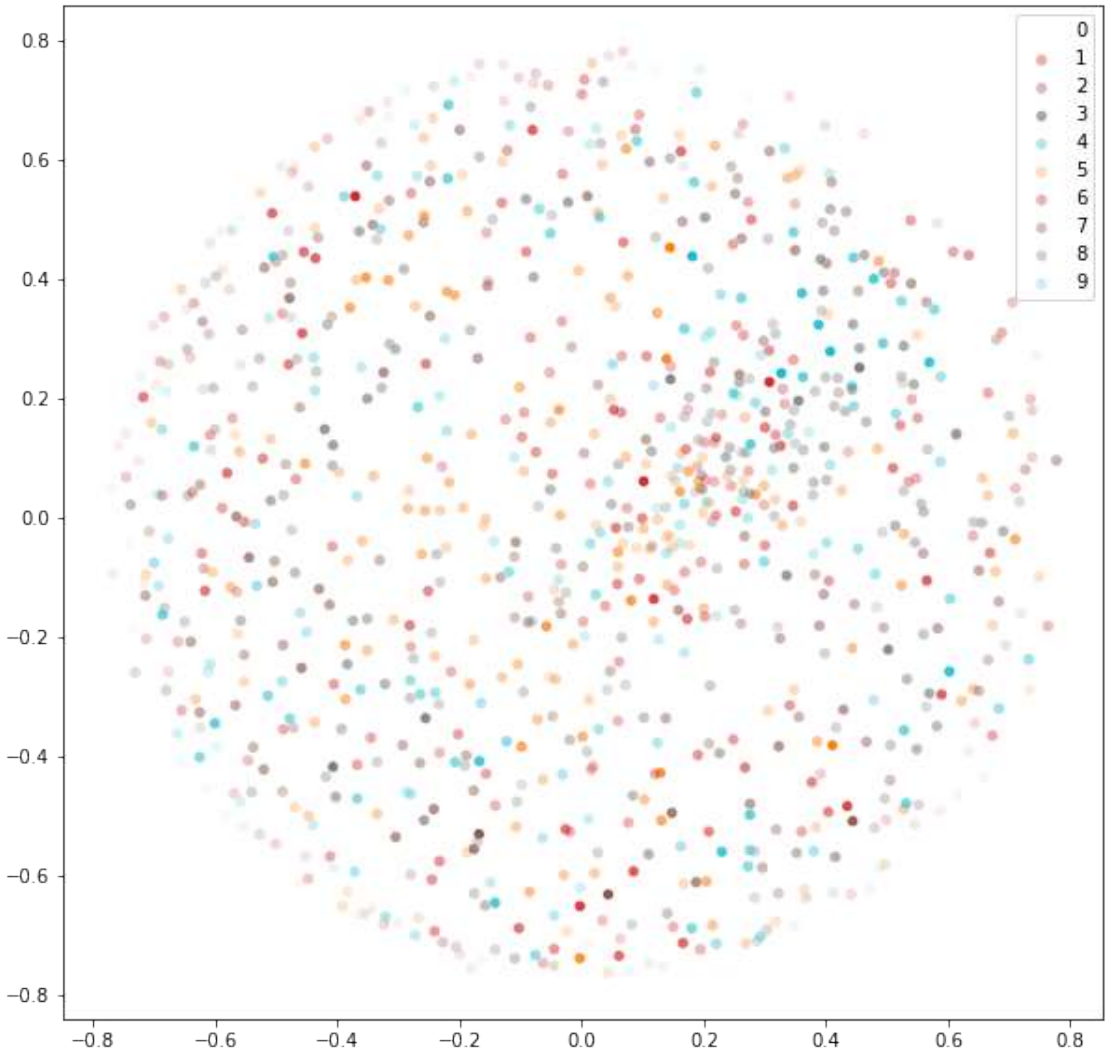


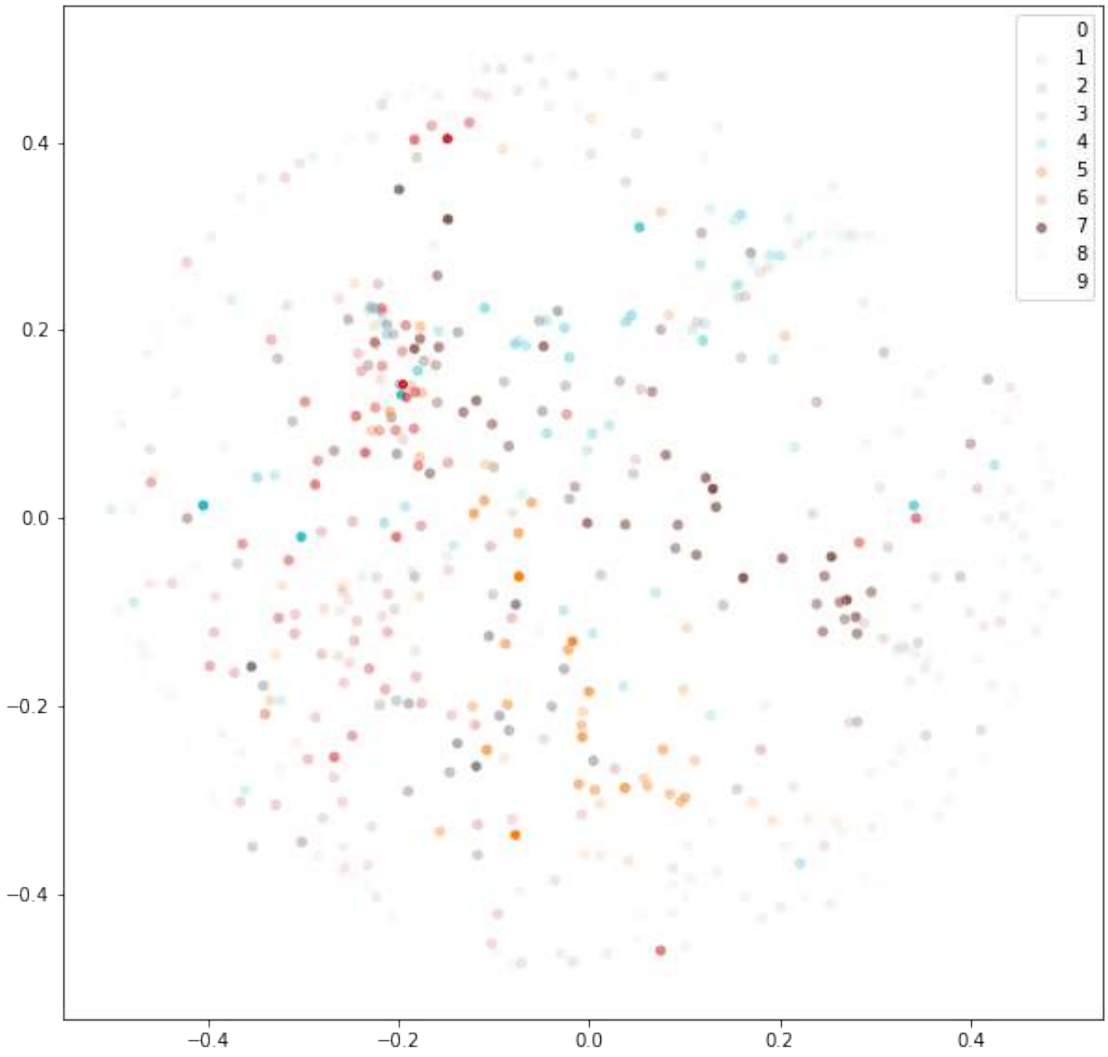
[ ]:
```
compare_projections(DataType.CIFAR, 'cifar_mlp', n_layer=4, label=3)
```


0.000
0.006
0.012
0.018
0.024
0.030
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4

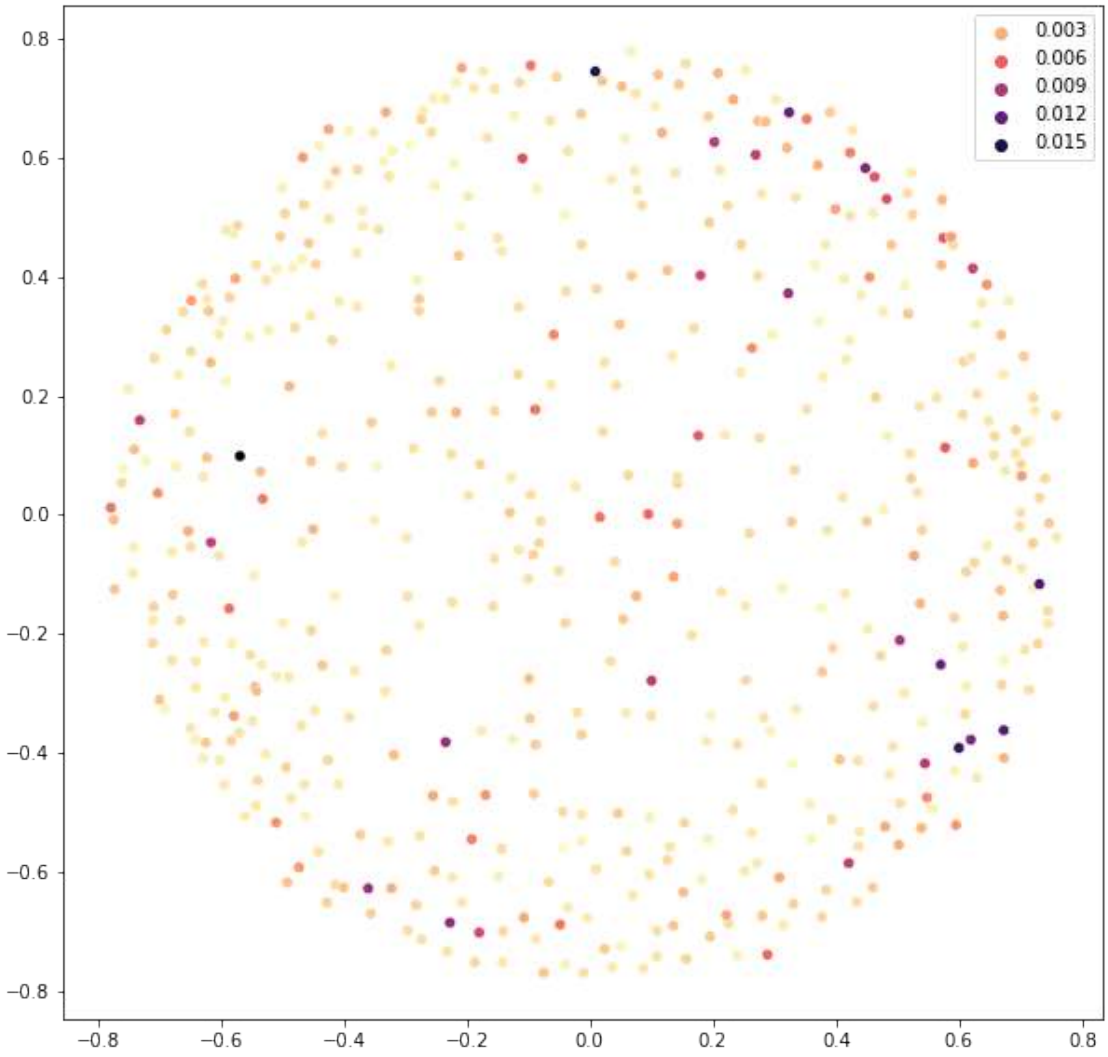


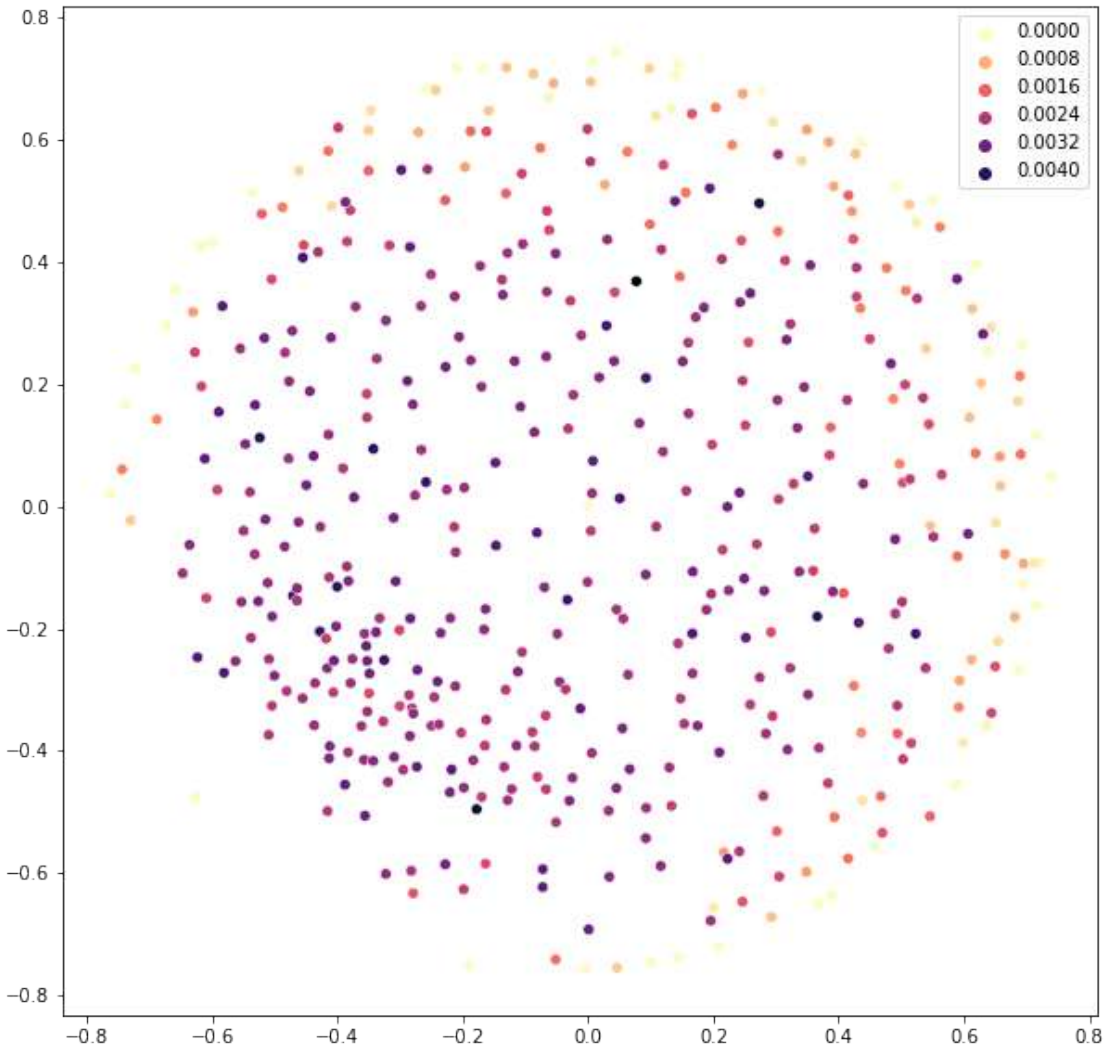
[ ]:
```
compare_discriminative_map(DataType.CIFAR, 'cifar_mlp', n_layer=4, size=2000)
```


0
1
2
3
4
5
6
7
8
9
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

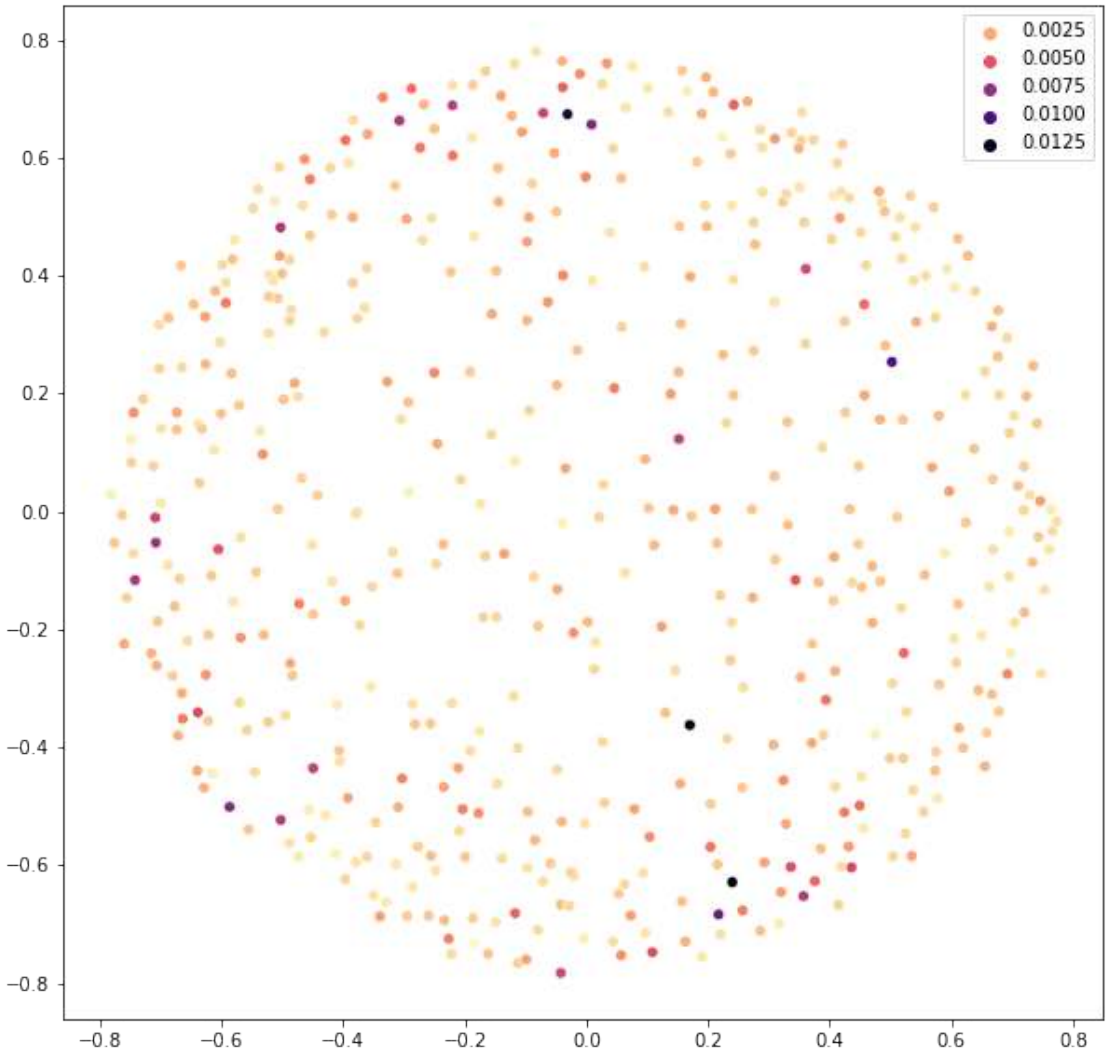


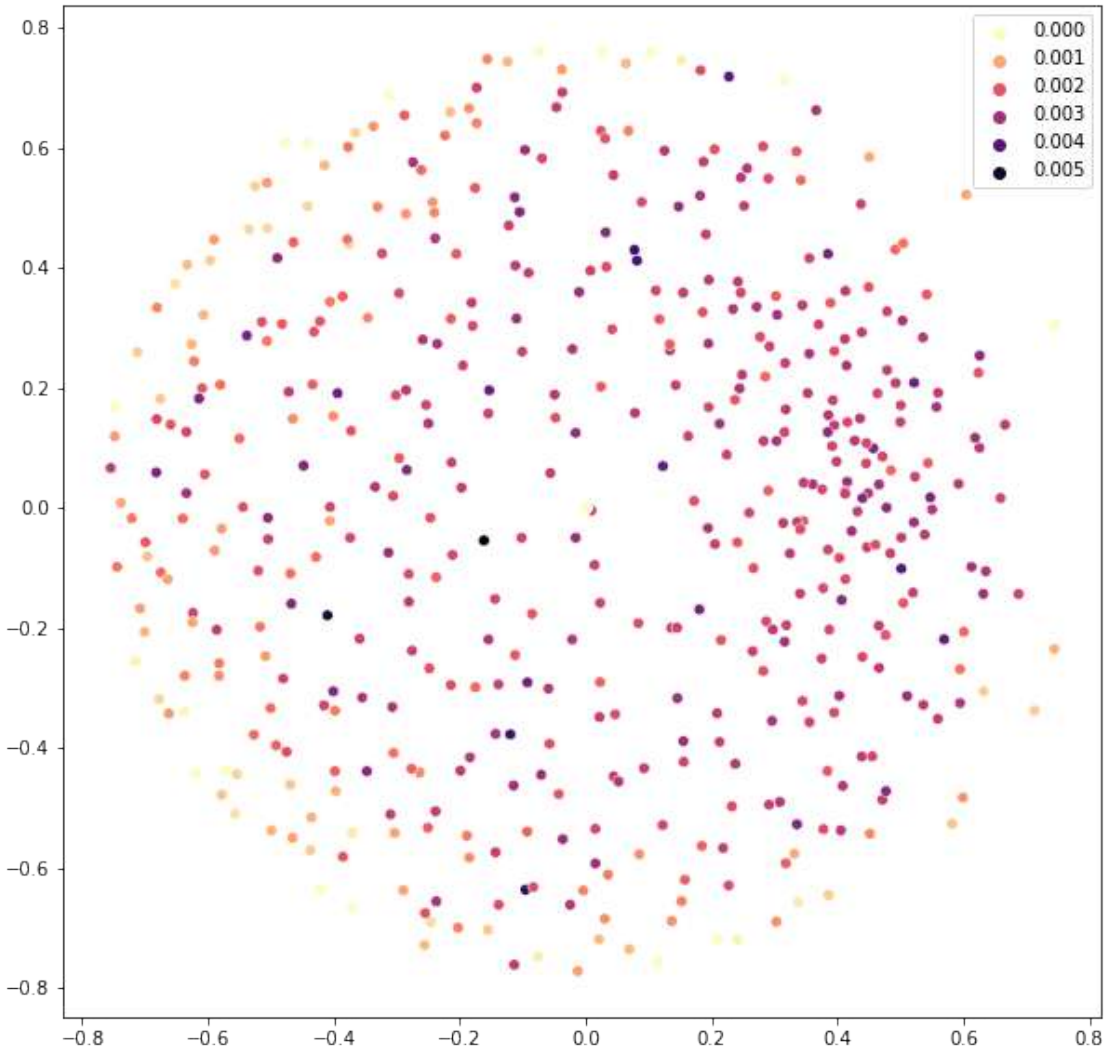
[ ]:
```
compare_projections(DataType.CIFAR, 'cifar_cnn', n_layer=2, label=5)
```


0.003
0.006
0.009
0.012
0.015
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

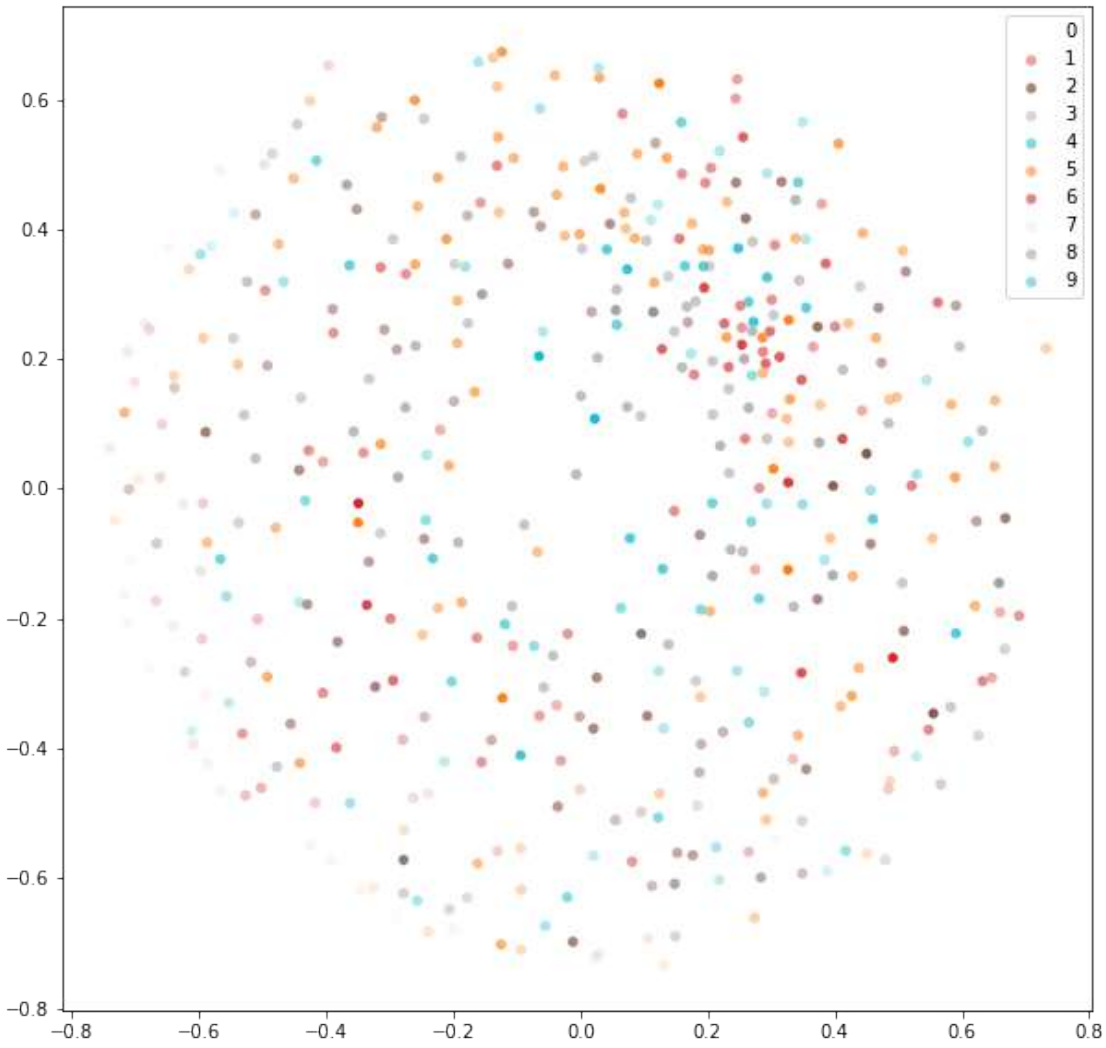


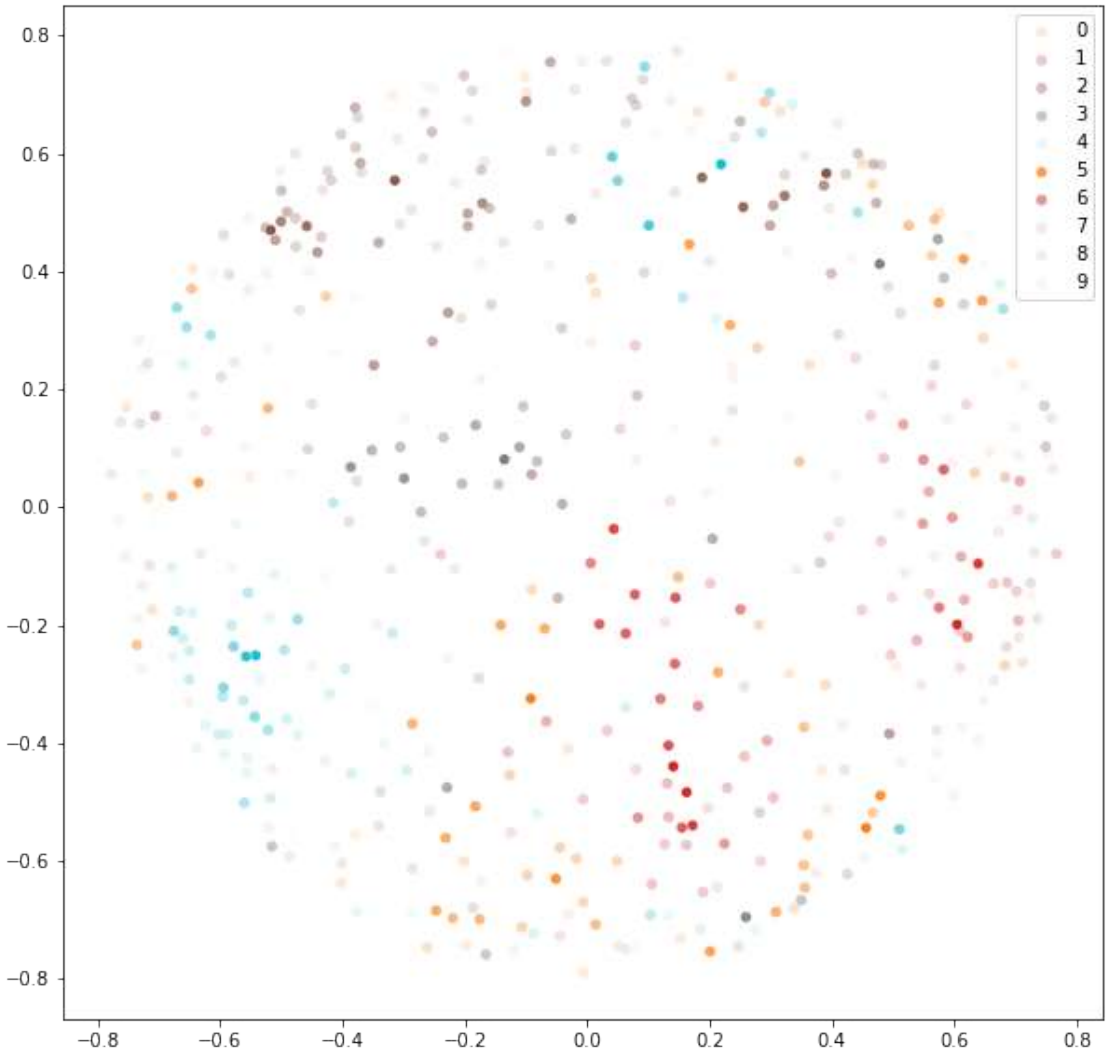
[ ]:
```
compare_projections(DataType.CIFAR, 'cifar_cnn', n_layer=2, label=3)
```


0.0025
0.0050
0.0075
0.0100
0.0125
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8



[ ]:
```
compare_discriminative_map(DataType.CIFAR, 'cifar_cnn', n_layer=2, size=2000)
```


0
1
2
3
4
5
6
7
8
9
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8


0
1
2
3
4
5
6
7
8
9
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

[ ]: