05_ann_visualization

April 11, 2024

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
```

```
2024-04-10 20:40:55.698057: I external/local_tsl/tsl/cuda/cudart_stub.cc:32]
Could not find cuda drivers on your machine, GPU will not be used.
2024-04-10 20:40:55.828308: I external/local_tsl/tsl/cuda/cudart_stub.cc:32]
Could not find cuda drivers on your machine, GPU will not be used.
2024-04-10 20:40:56.353591: I tensorflow/core/platform/cpu_feature_guard.cc:210]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
2024-04-10 20:40:57.740798: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not
find TensorRT
```

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.

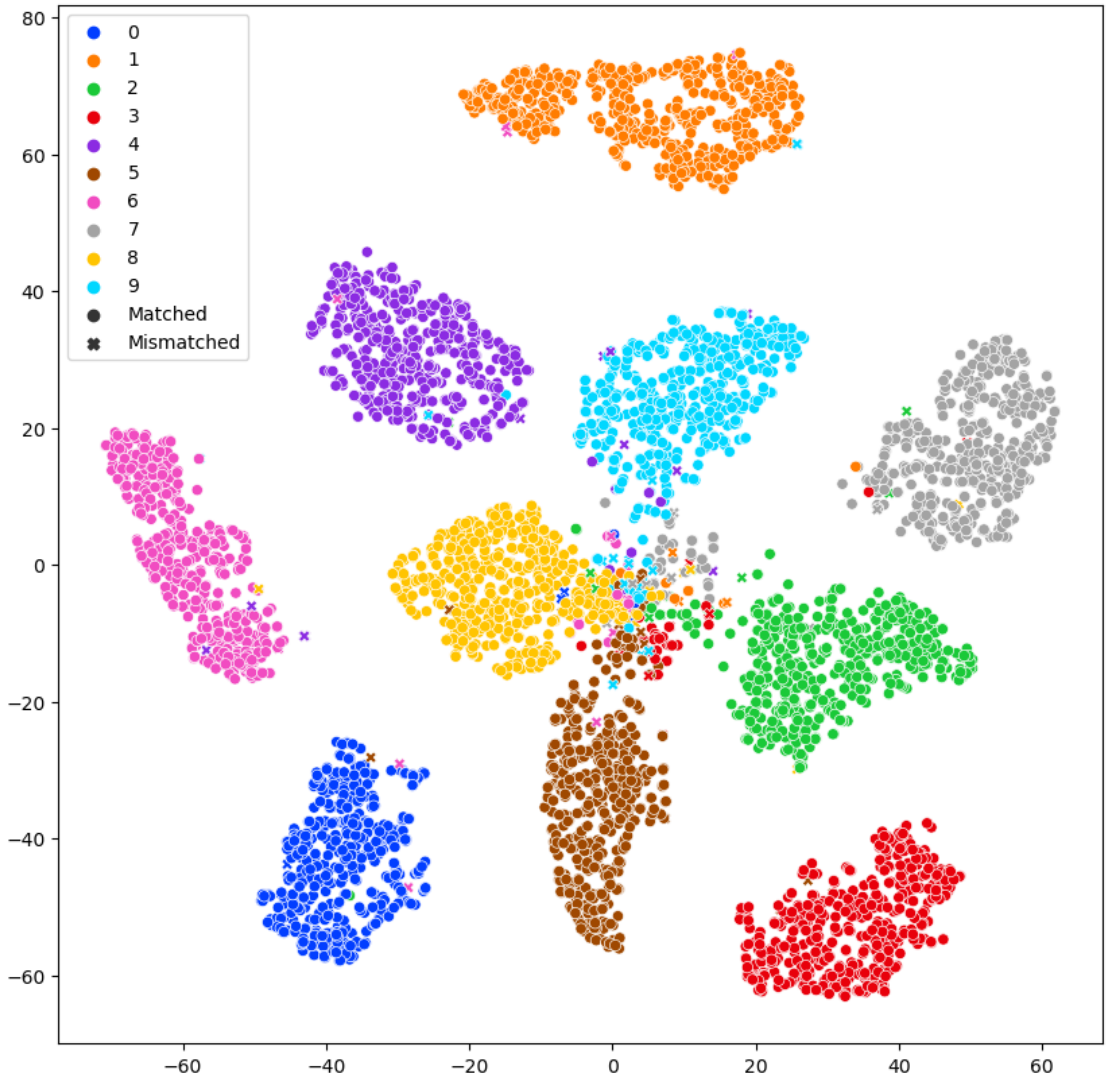
TODO: Use `show_seq_projections` function to calculate points and targets for `N` layers of

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`

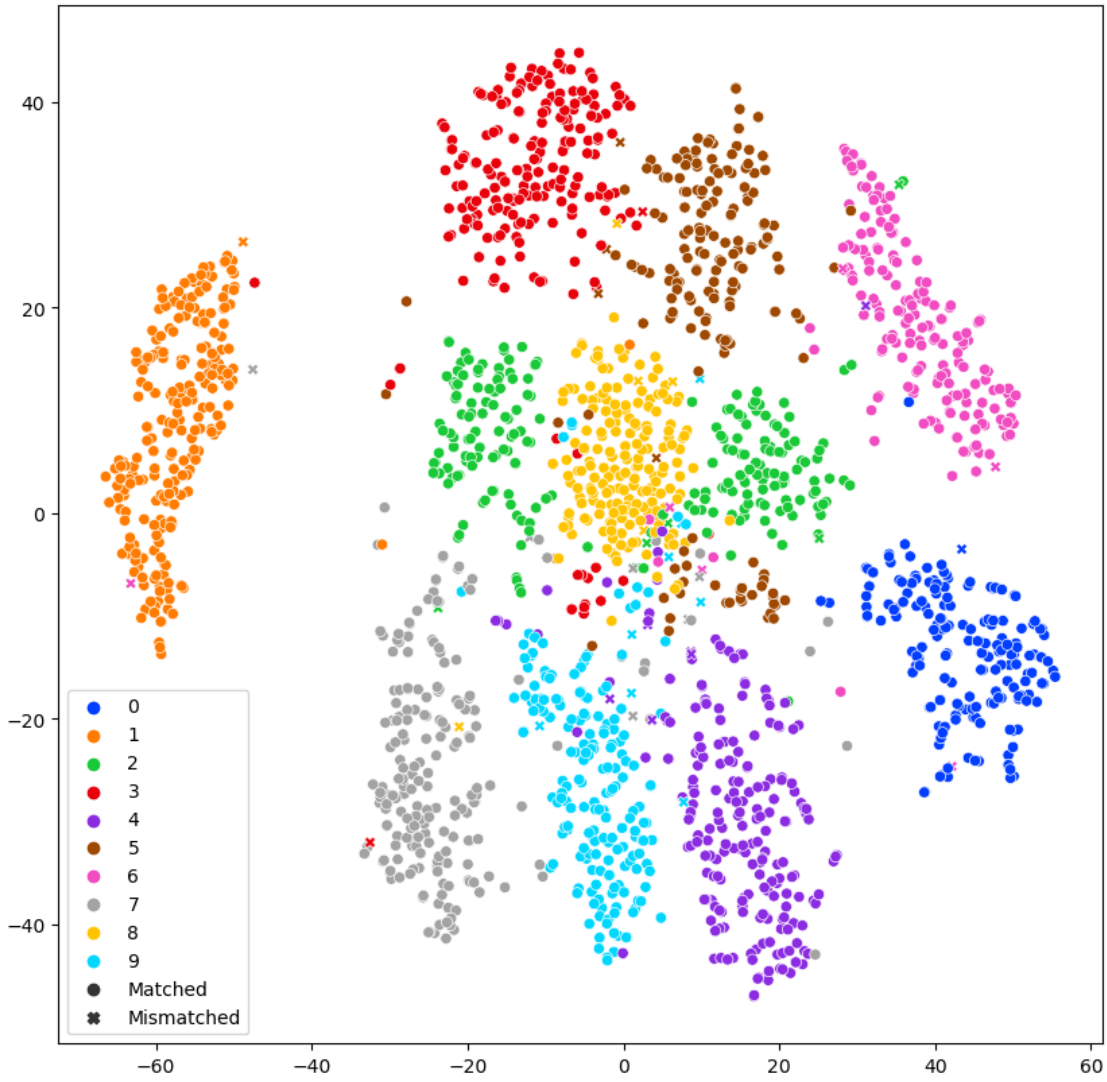
[ ]:
```
# test out the function
transformed_points, targets = show_seq_projections(DataType.MNIST, "mnist_mlp",␣
 ↪1, 100, 5000)
```

```
2024-04-10 20:40:58.903378: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:998] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2024-04-10 20:40:59.082752: W
tensorflow/core/common_runtime/gpu/gpu_device.cc:2251] Cannot dlopen some GPU
libraries. Please make sure the missing libraries mentioned above are installed
properly if you would like to use GPU. Follow the guide at
https://www.tensorflow.org/install/gpu for how to download and setup the
required libraries for your platform.
Skipping registering GPU devices…

313/313             1s 4ms/step
```

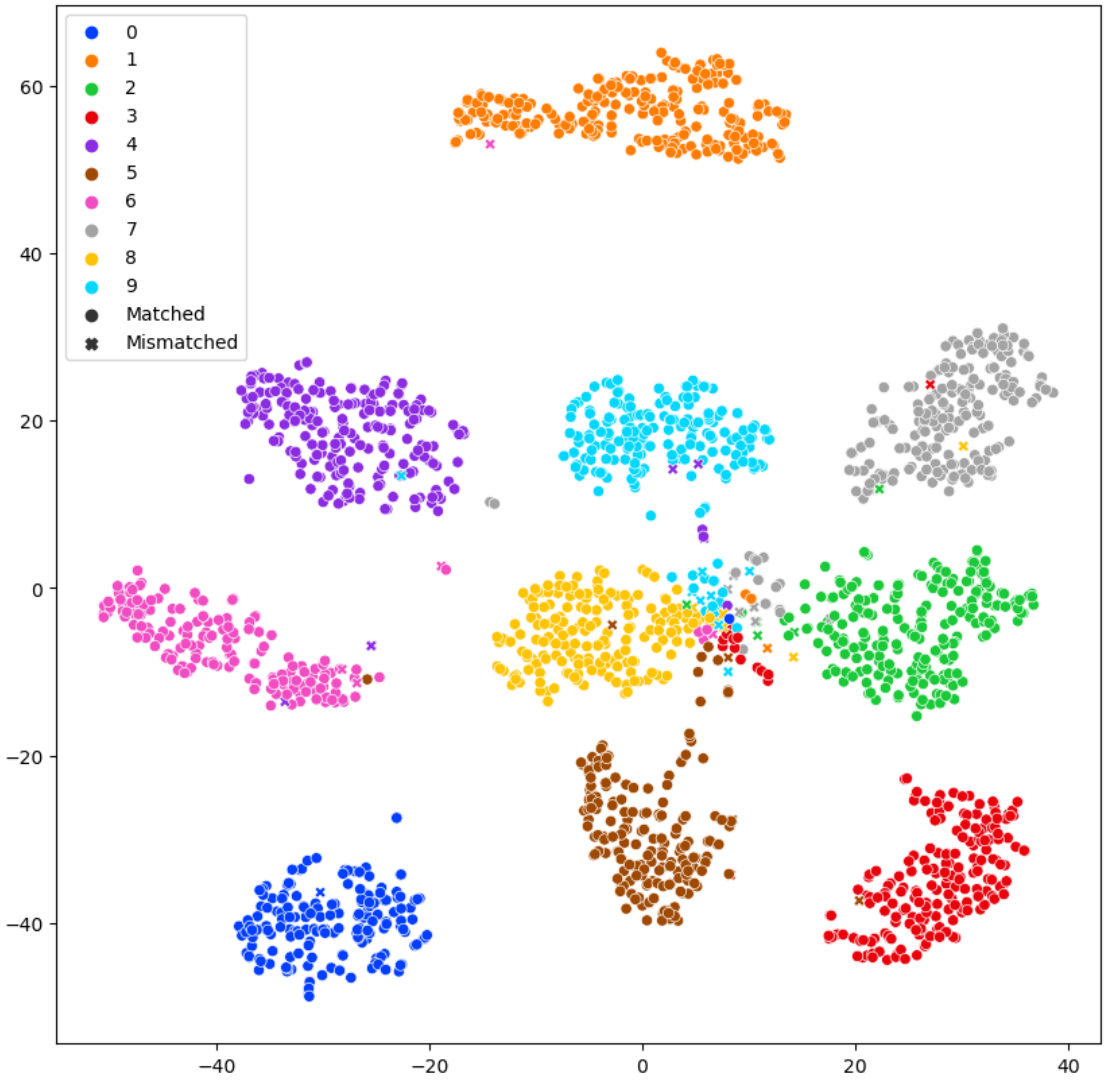


[ ]:
```
points_by_layers = []
N = 4 # for mlp
for n in range(N):
    transformed_points, targets = show_seq_projections(DataType.MNIST,␣
 ↪"mnist_mlp", n, 100, 2000)
    points_by_layers.append(transformed_points)
```

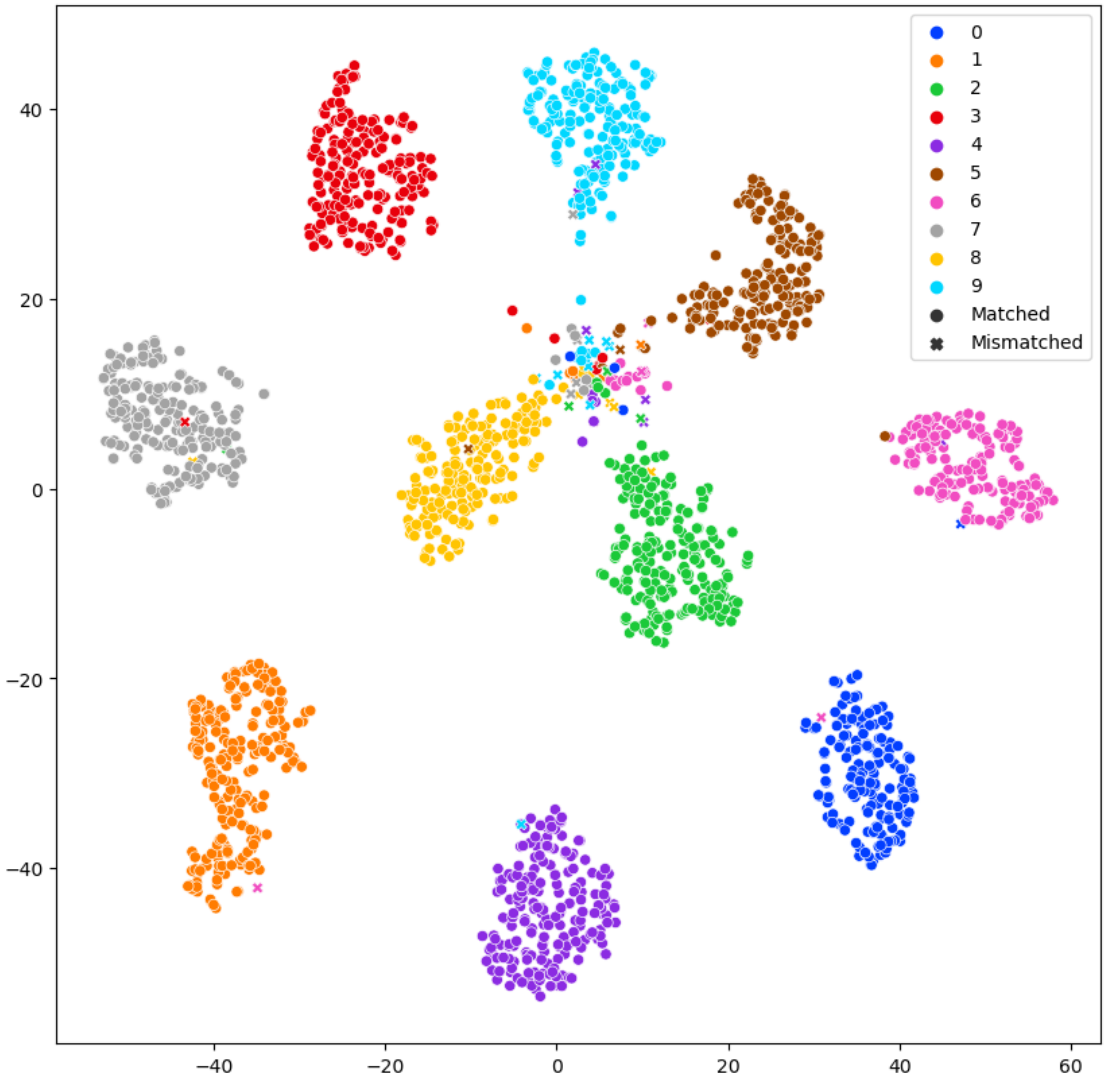
```
313/313             1s 4ms/step
```



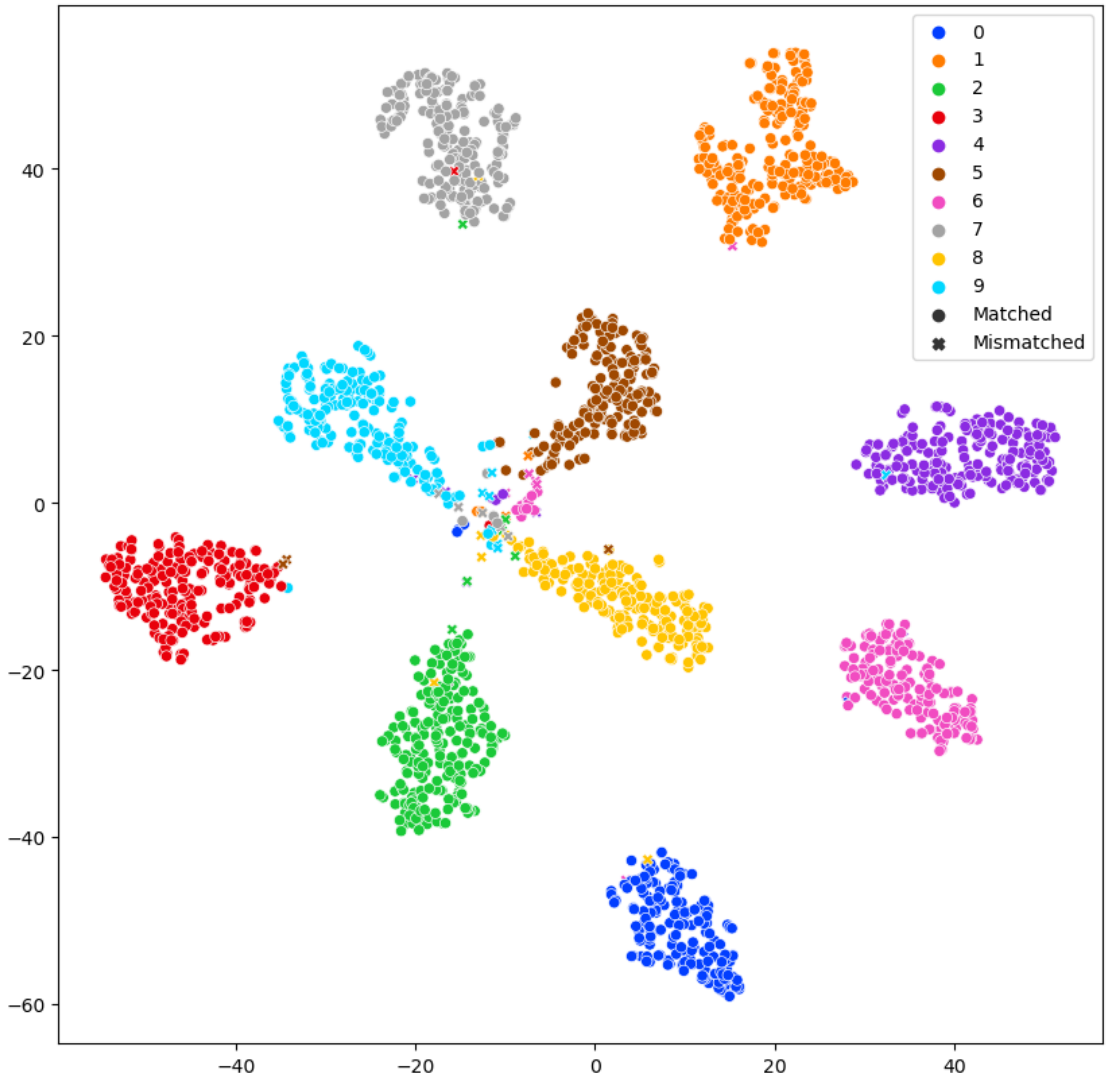

```
313/313                1s 4ms/step
```



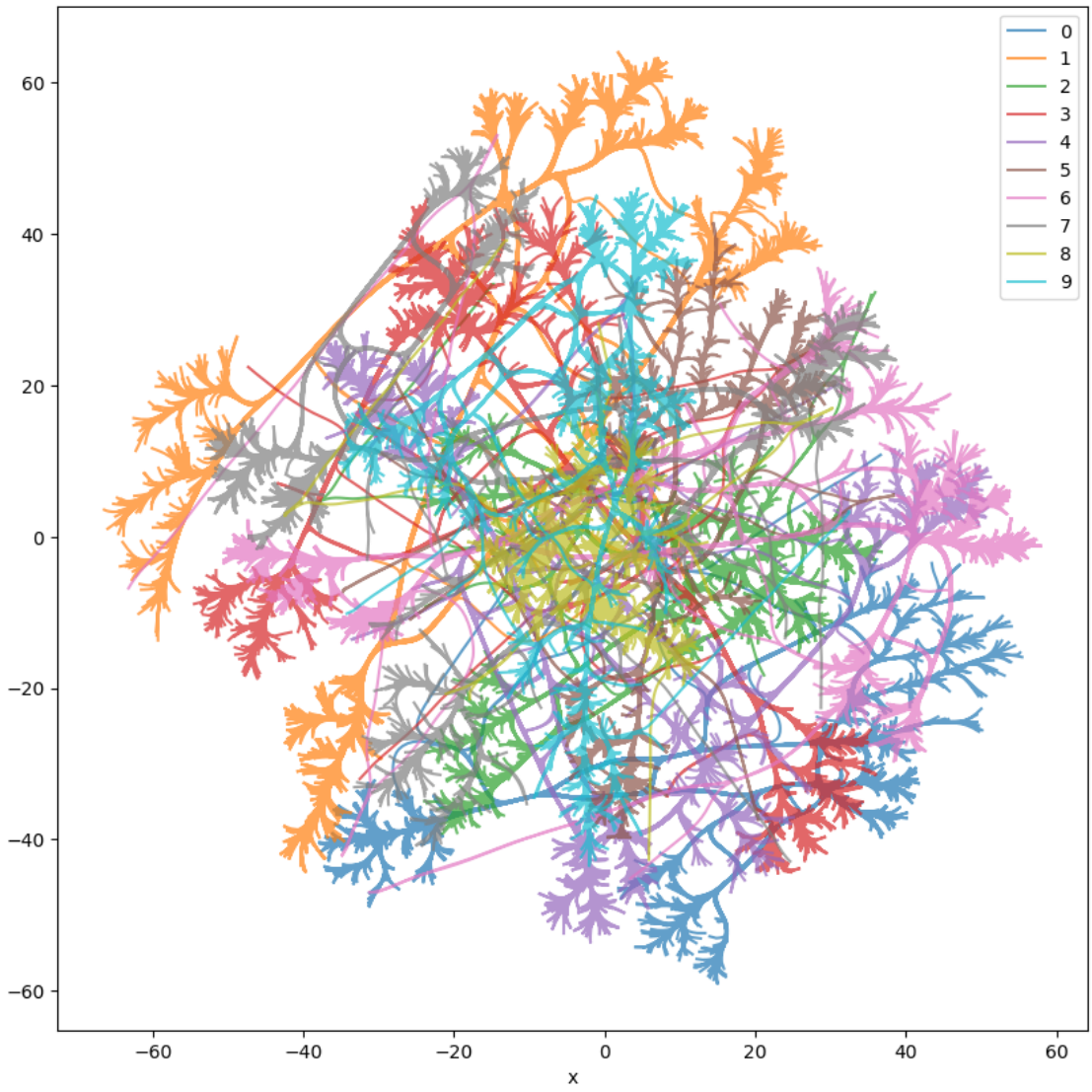

```
313/313                1s 4ms/step
```



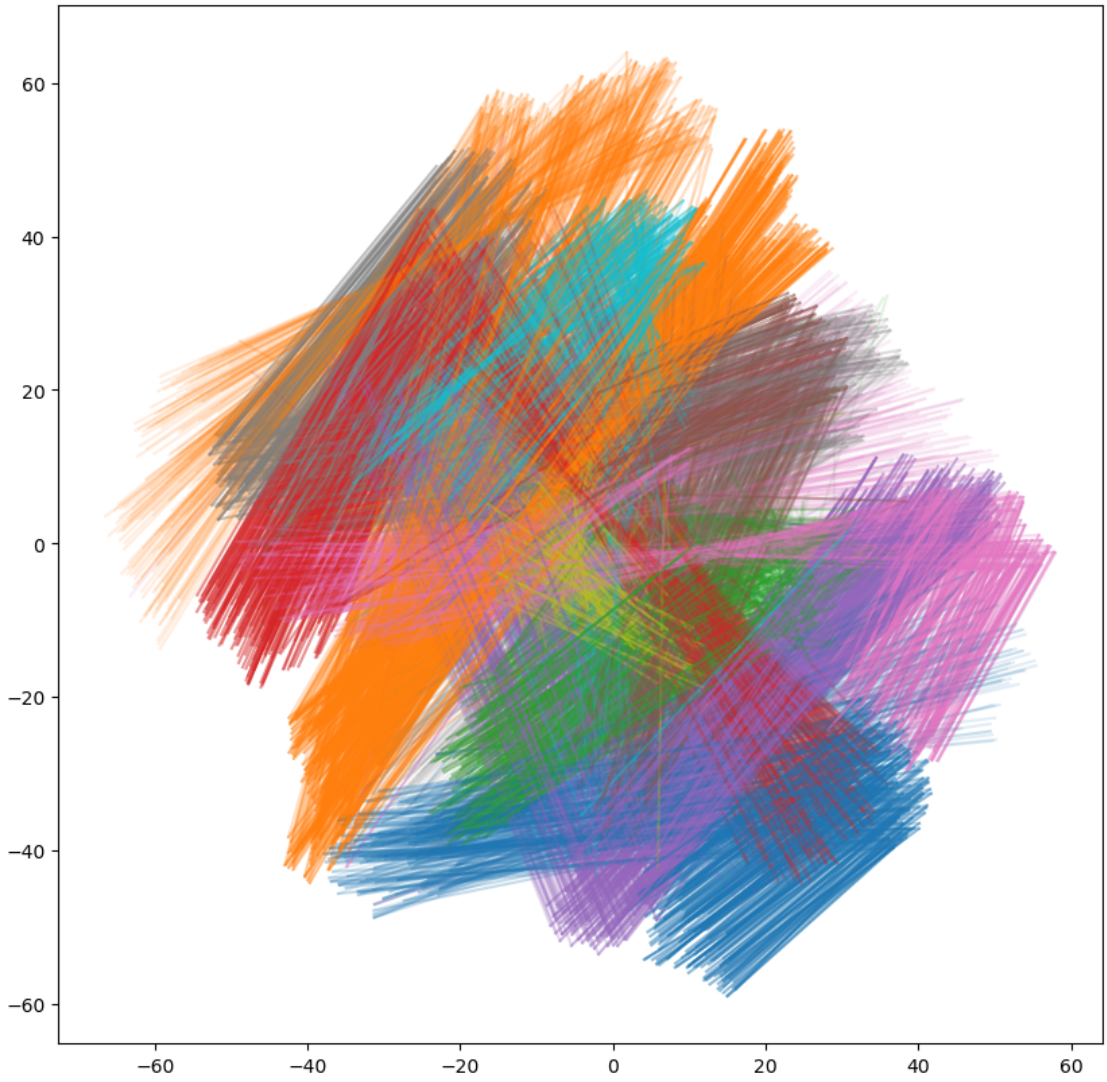

```
313/313              2s 5ms/step
```



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



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



0.1.1 COMMENT

Widzimy, że najłatwiej rozdzielalne klasy (0 i 1) praktycznie od pierwszej wartwy są łatwo separowalne od reszty. Dla trudniejszych klas (3, 8, 9) nawet w ostateniej warstwie ich reprezentacje leżą blisko siebie. Nie jestem pewien co miała pokazać ewolucja między warstwowa. Wizualizujemy tutaj arbitralne 2-wymiarowe reprezentacje aktywacji warstw ukrytych. Ich wzajemne położenie zdaje nie nieść jakiejś znaczącej informacji (której nie niosą scatter ploty samych 2D embeddingów - obrazkina samej górze)

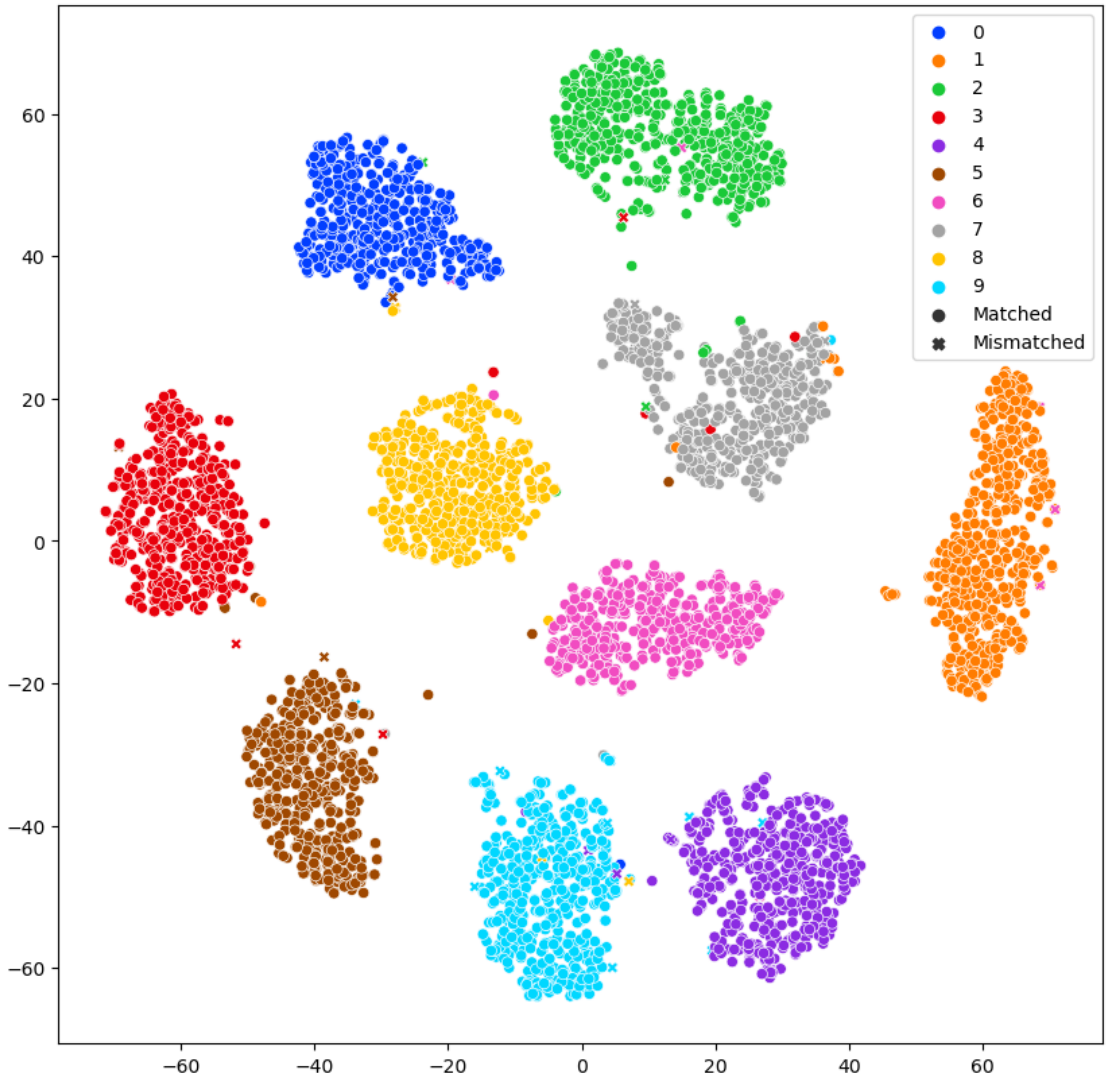
Dodatkowo, dla większej liczby punktów te wizualizaje są zupełnie nieczytelne (dla 5k była to wielka kolorowa plama - w raporcie pozostawiłem wyniki dla 2k)

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.

[ ]:
```
transformed_points, targets = show_seq_projections(DataType.MNIST, "mnist_cnn",␣
↪1, 100, 5000)
```

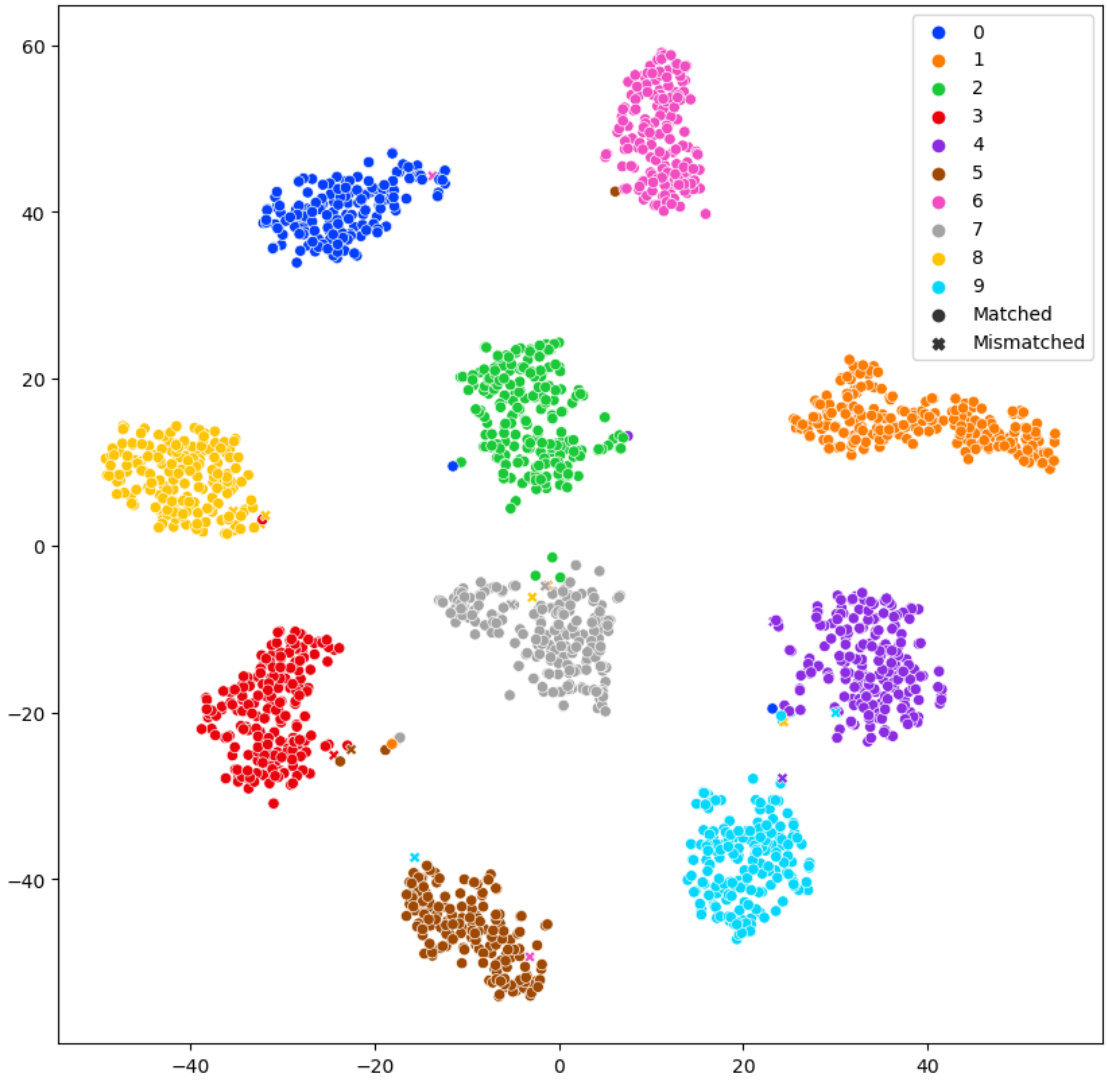
```
2024-04-11 00:20:23.844797: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
432640000 exceeds 10% of free system memory.
2024-04-11 00:20:24.032444: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
432640000 exceeds 10% of free system memory.
2024-04-11 00:20:24.261366: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
432640000 exceeds 10% of free system memory.
2024-04-11 00:20:24.398703: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
432640000 exceeds 10% of free system memory.
2024-04-11 00:20:24.535927: W
external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of
432640000 exceeds 10% of free system memory.
```

```
313/313             7s 22ms/step
```

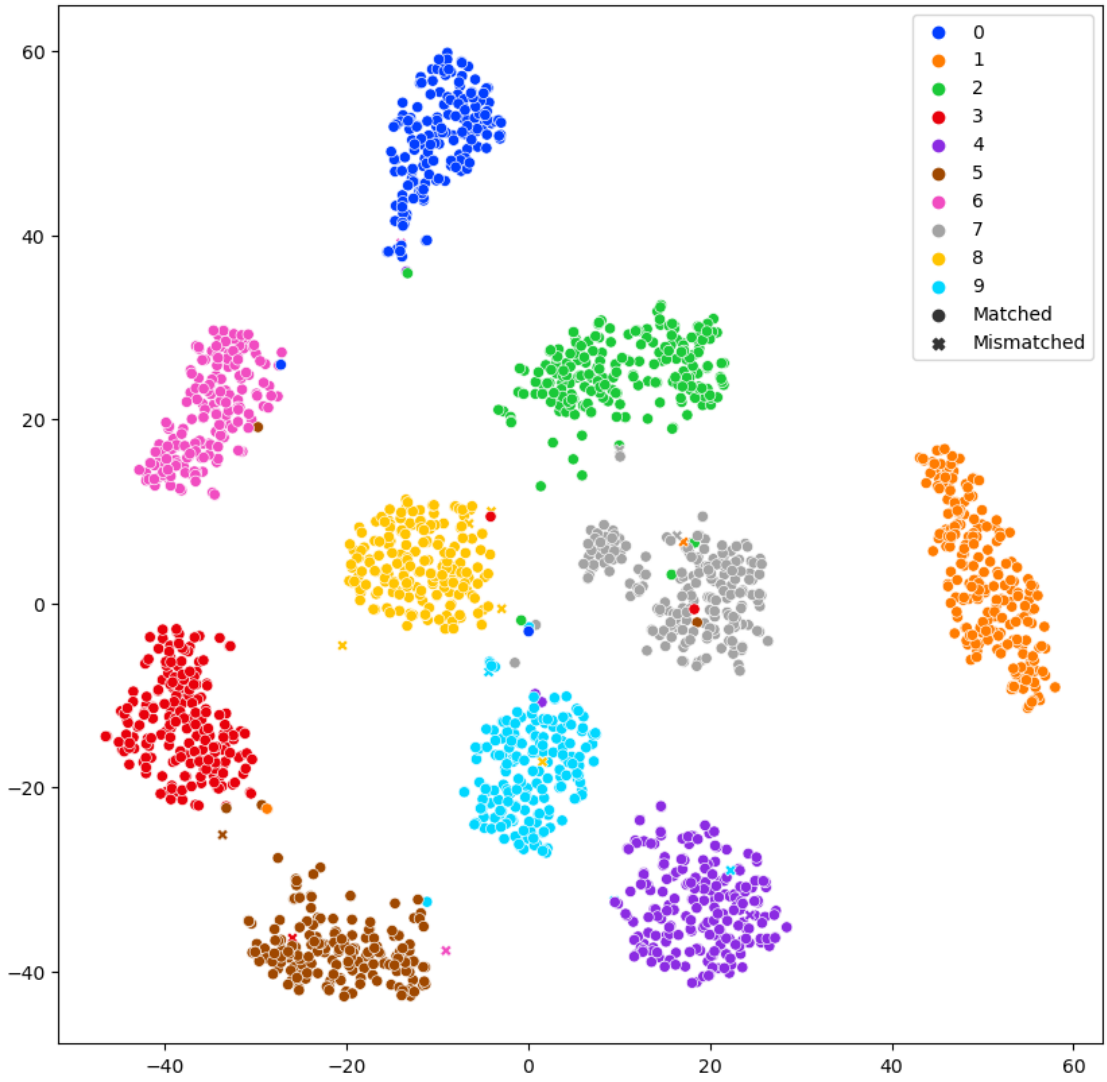


[ ]:
```
points_by_layers = []
N = 2 # 2 for cnn
for n in range(N):
    transformed_points, targets = show_seq_projections(DataType.MNIST,␣
 ↪"mnist_cnn", n, 100, 2000)
    points_by_layers.append(transformed_points)
```

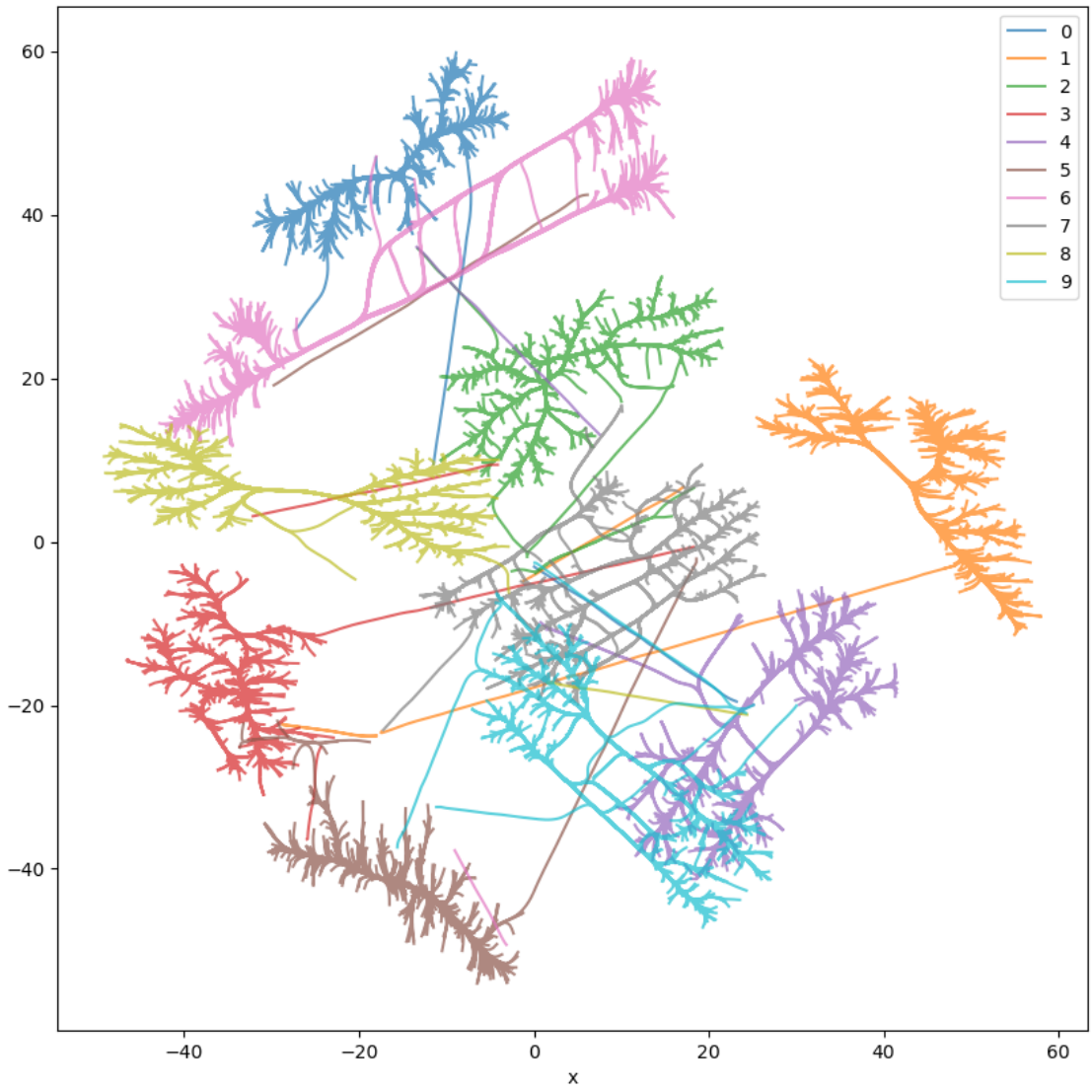
```
313/313             8s 26ms/step
```



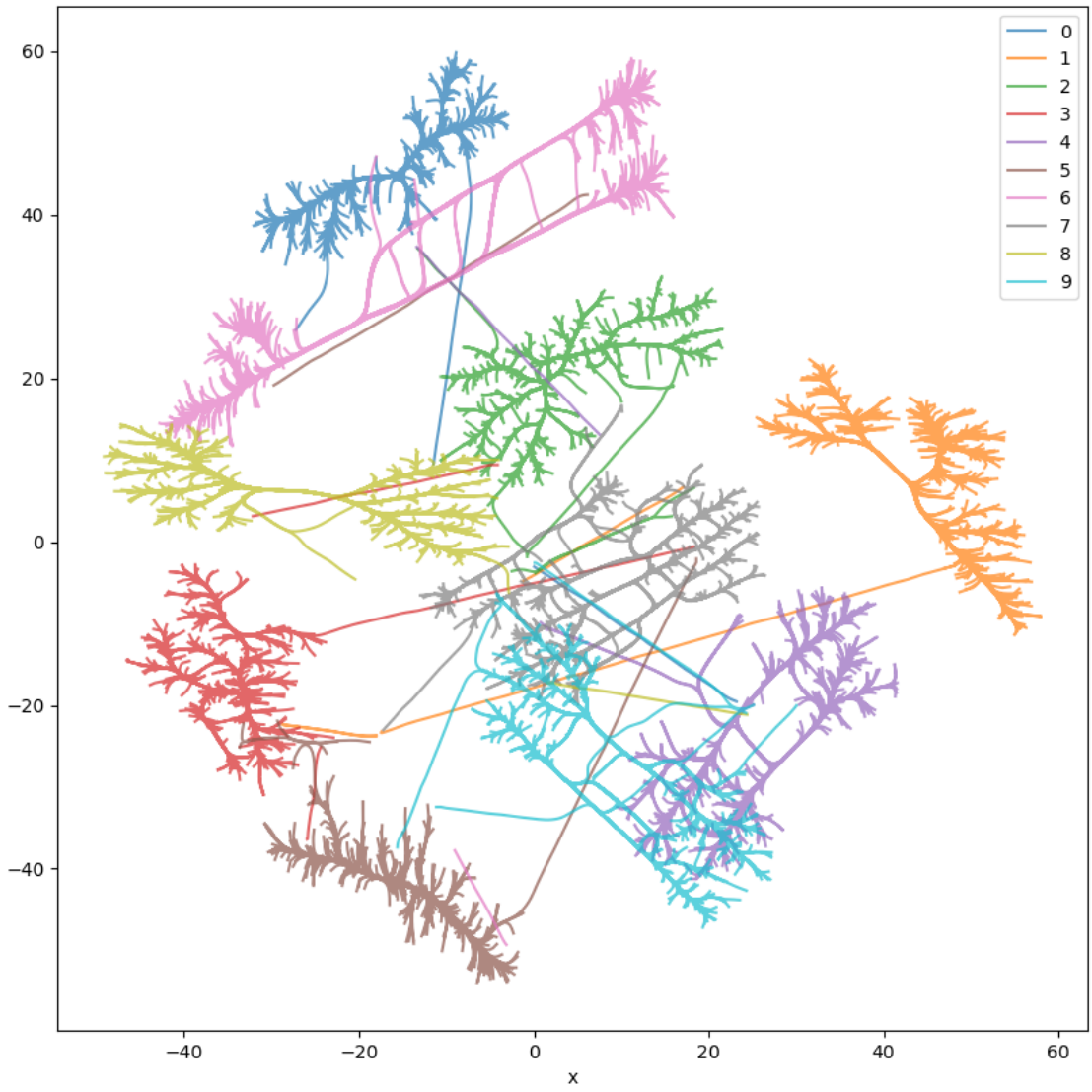

```
313/313              10s 30ms/step
```



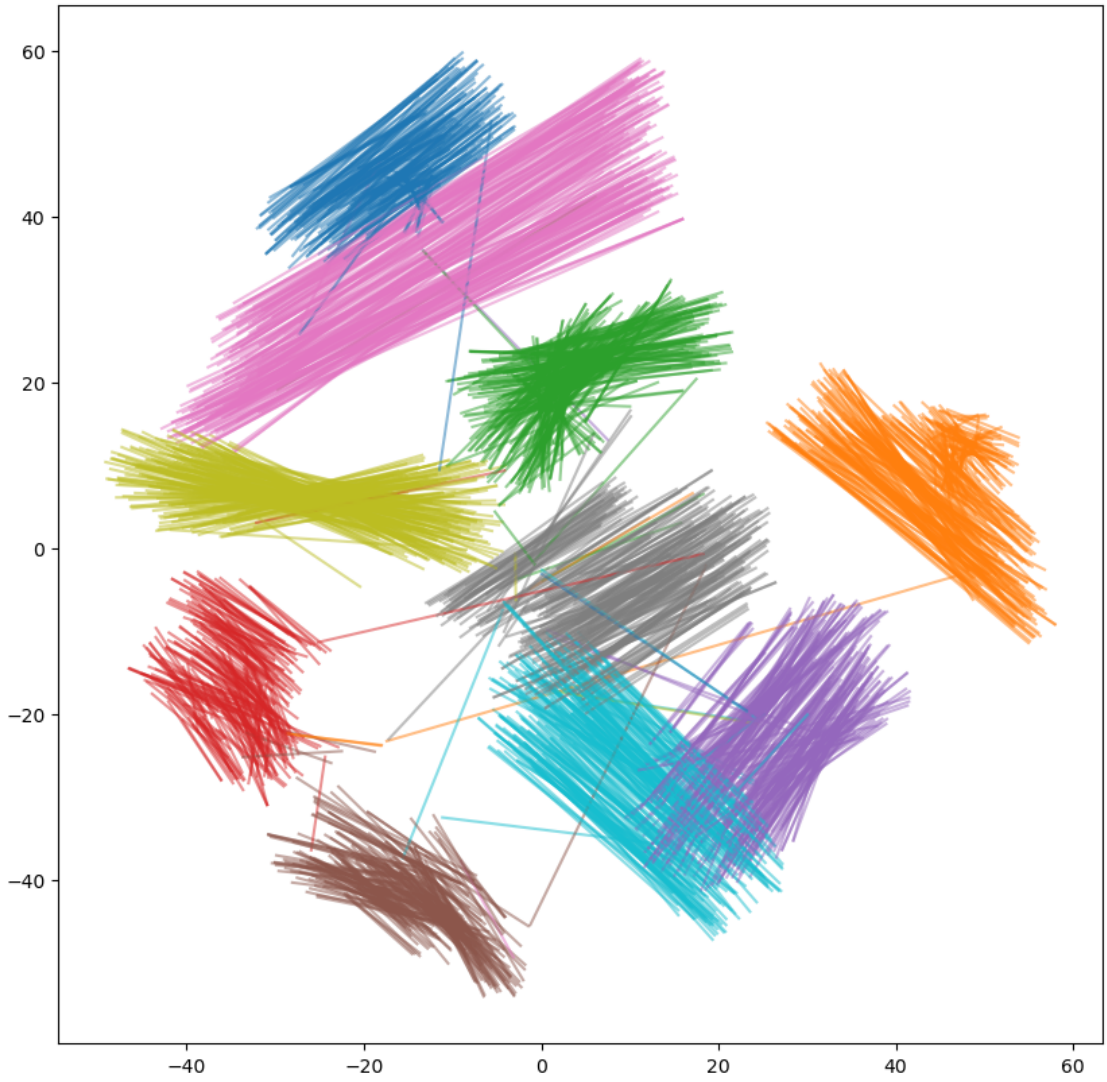
[ ]:
```
inter_layer_evolution(points_by_layers, targets)
```



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



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



0.1.2 COMMENT

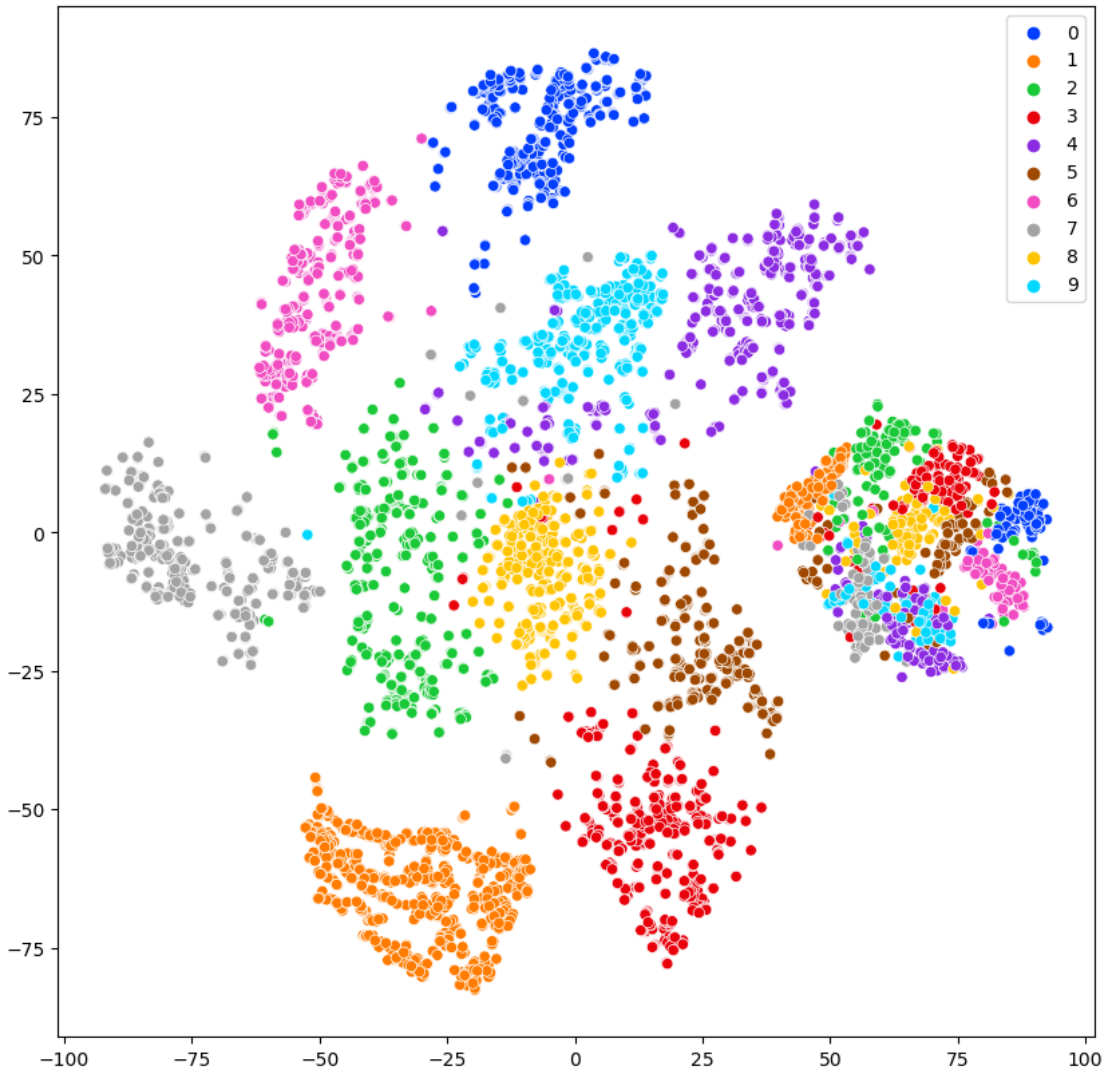
W porównaniu do MLP widzimy duże lepsze rozdzielenie klas i to już po pierwszej warstwie. Nie dziwi to, w końcu CNNy służą do przetwarzania obrazu. Widzimy, że MLP nie było w stanie dobrze zaklasyfikować części cyfr i ta niemoc jest zobrazowana przemieszanymi embeddingami. Tymczasem dla CNN widzimy wyraźne klastry, bez żadnego przenikania się i pojedyncze źle zaklasyfikowane data pointy. Widzimy, że już od początku embeddingi były rodzielone, a w wartwie drugiej takie pozostały i nie zmieniły aż tak swojego położenia jak w MLP

0.2 INTER-EPOCH EVOLUTION

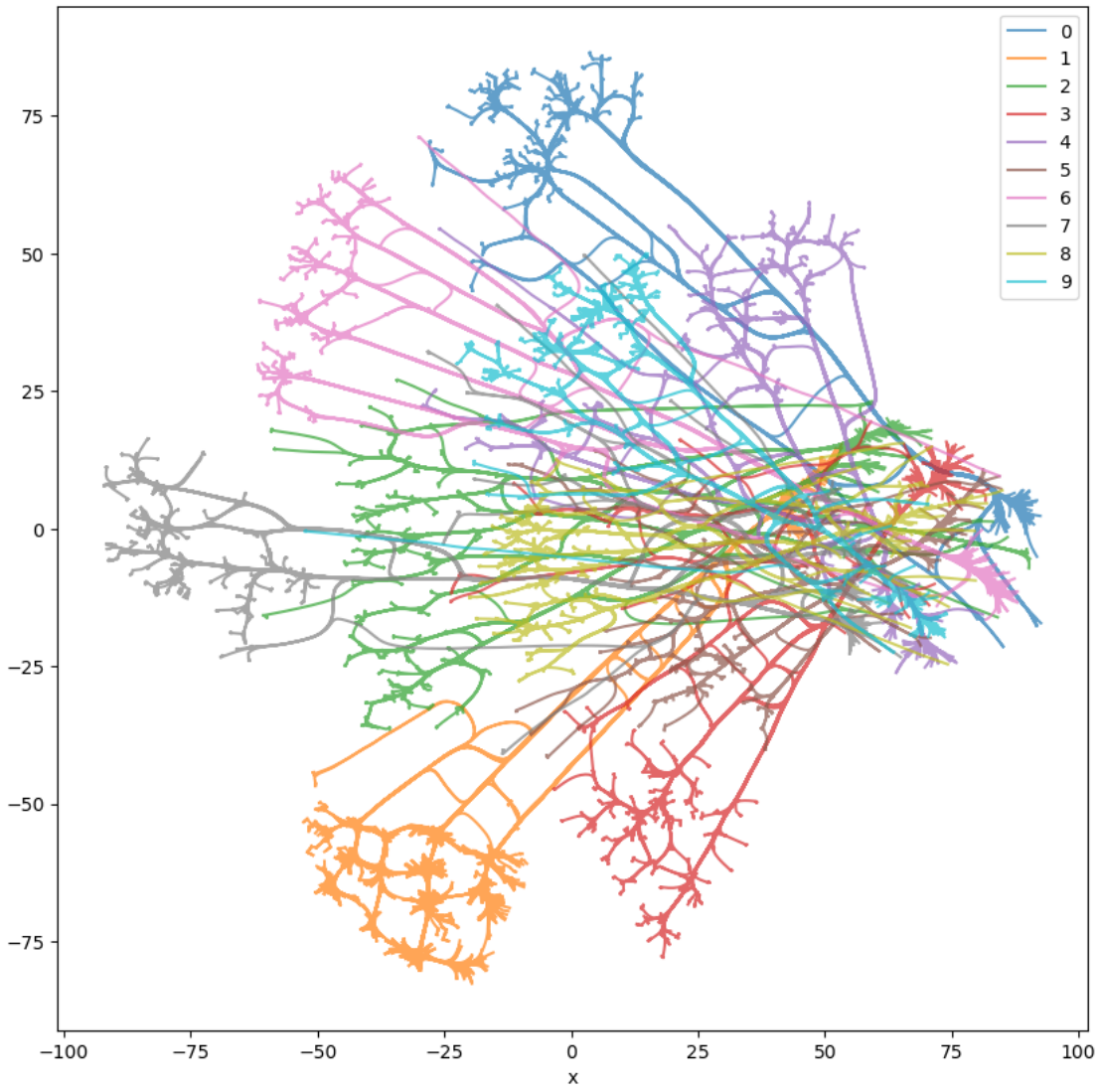
TODO: To complete this task, start by obtaining points and targets after hidden layers using the `get_all_activations` and `process_activations` functions. Then, use analogous functions for generating the inter-epoch evolution plot and trace. Finally, perform this task for both the

`mnist_mlp` and `mnist_cnn` networks.

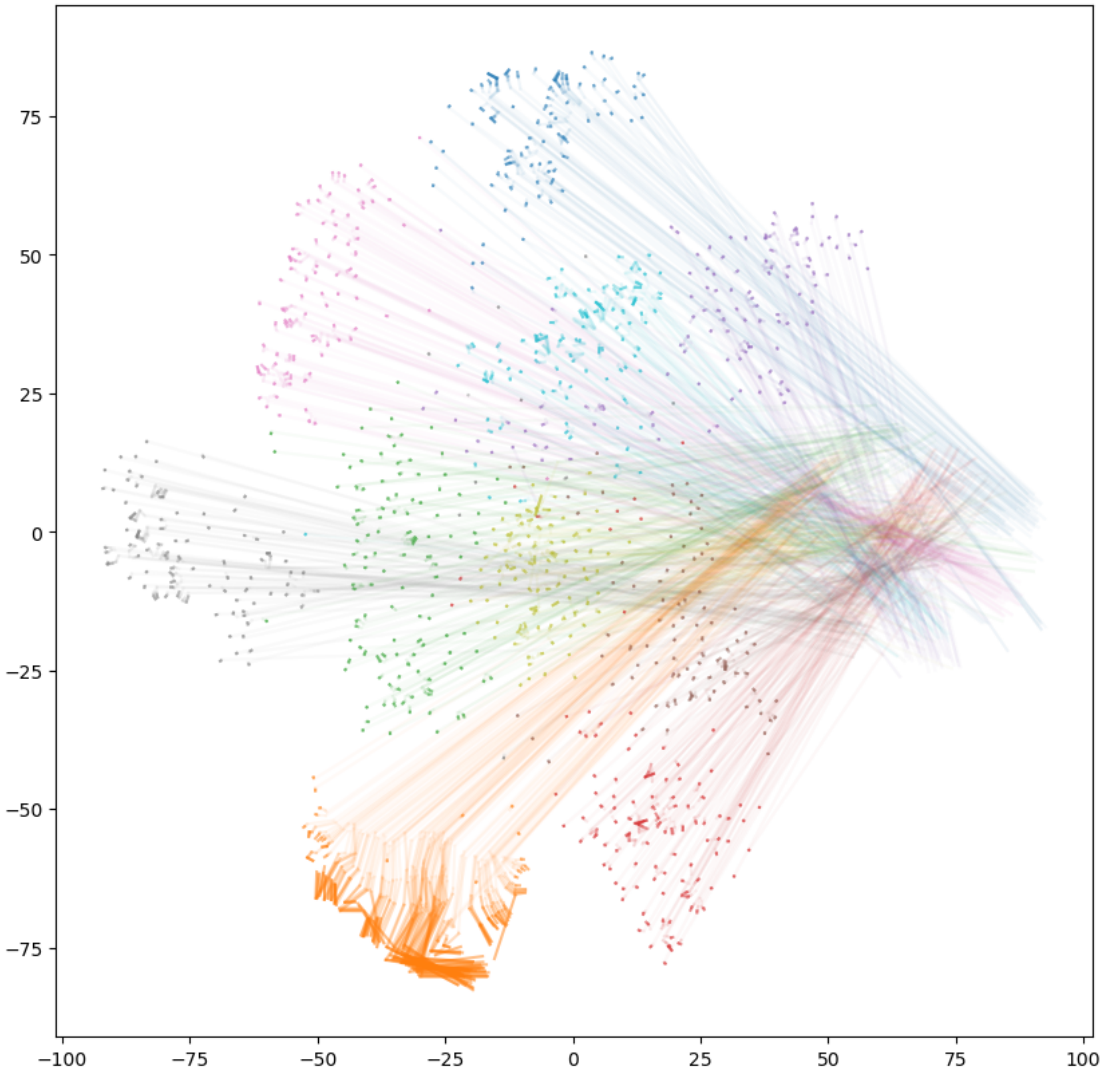
[ ]:
```
activations, targets = get_all_activations(DataType.MNIST, "mnist_mlp", 1, 1000)

points_lst, targets = process_activations(activations, targets, 1000)
```



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



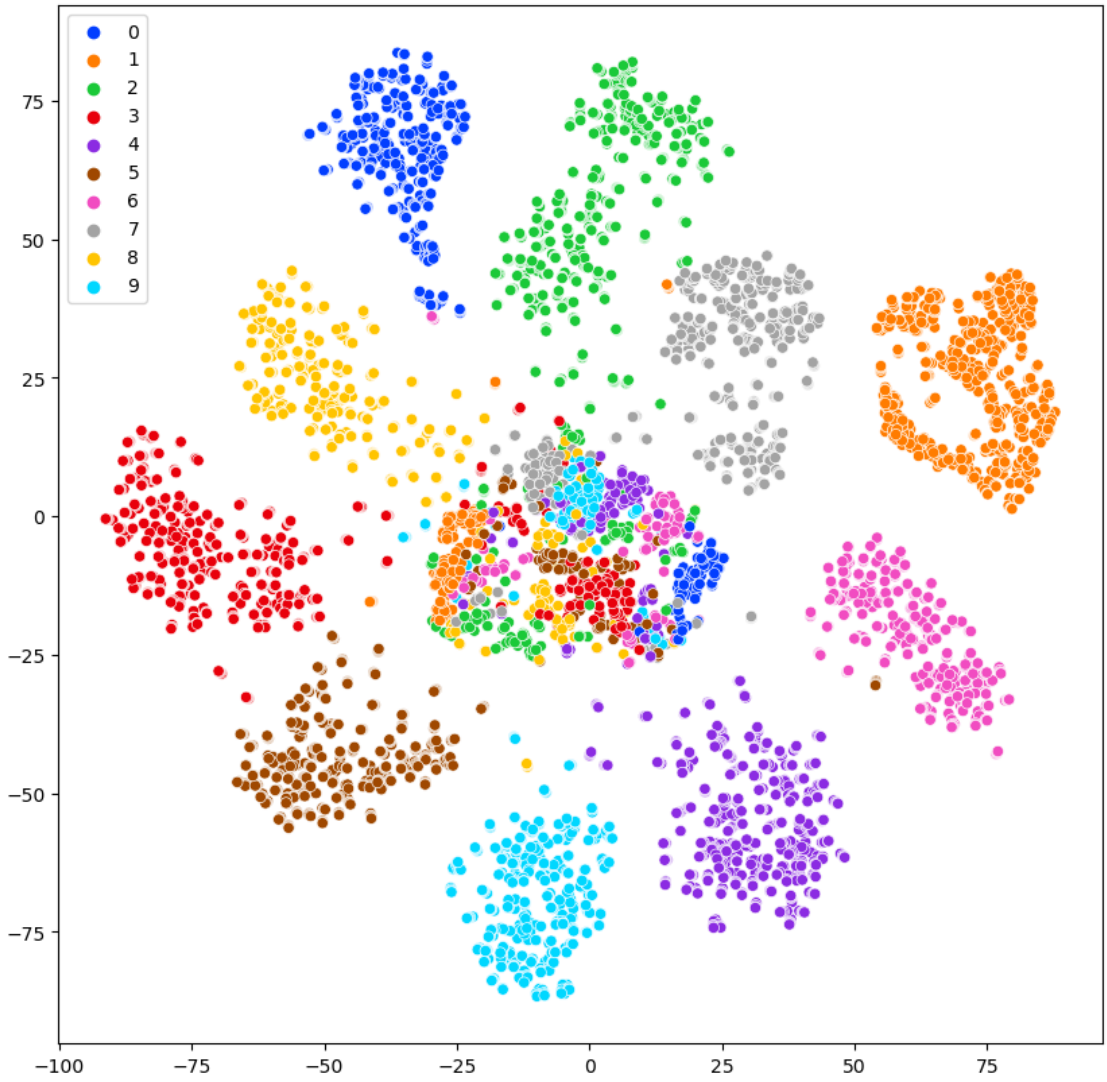
[ ]:
```
show_trace(points_lst, targets)
```



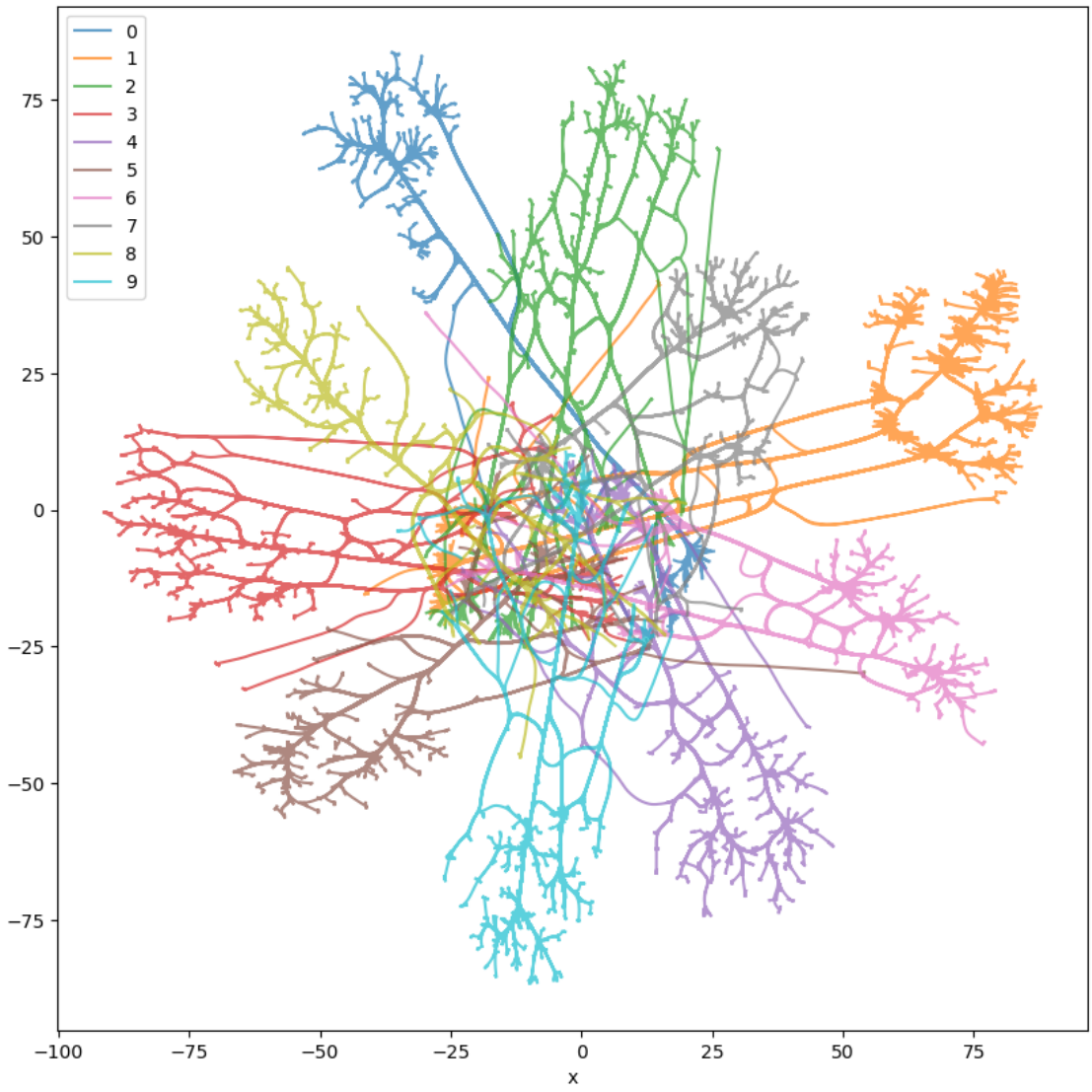
0.2.1 COMMENT

Tutaj mamy dużo ciekawsze wykresy niż w przypadku interlayer evolution. Widzimy jak mzieniają się reprezentacje aktywacji pierwszej warstwy ukrytej wraz z procesem uczenia. Uzykusjemy pożądany efekt, czyli reprezentacje robią się coraz lepiej separowalne.

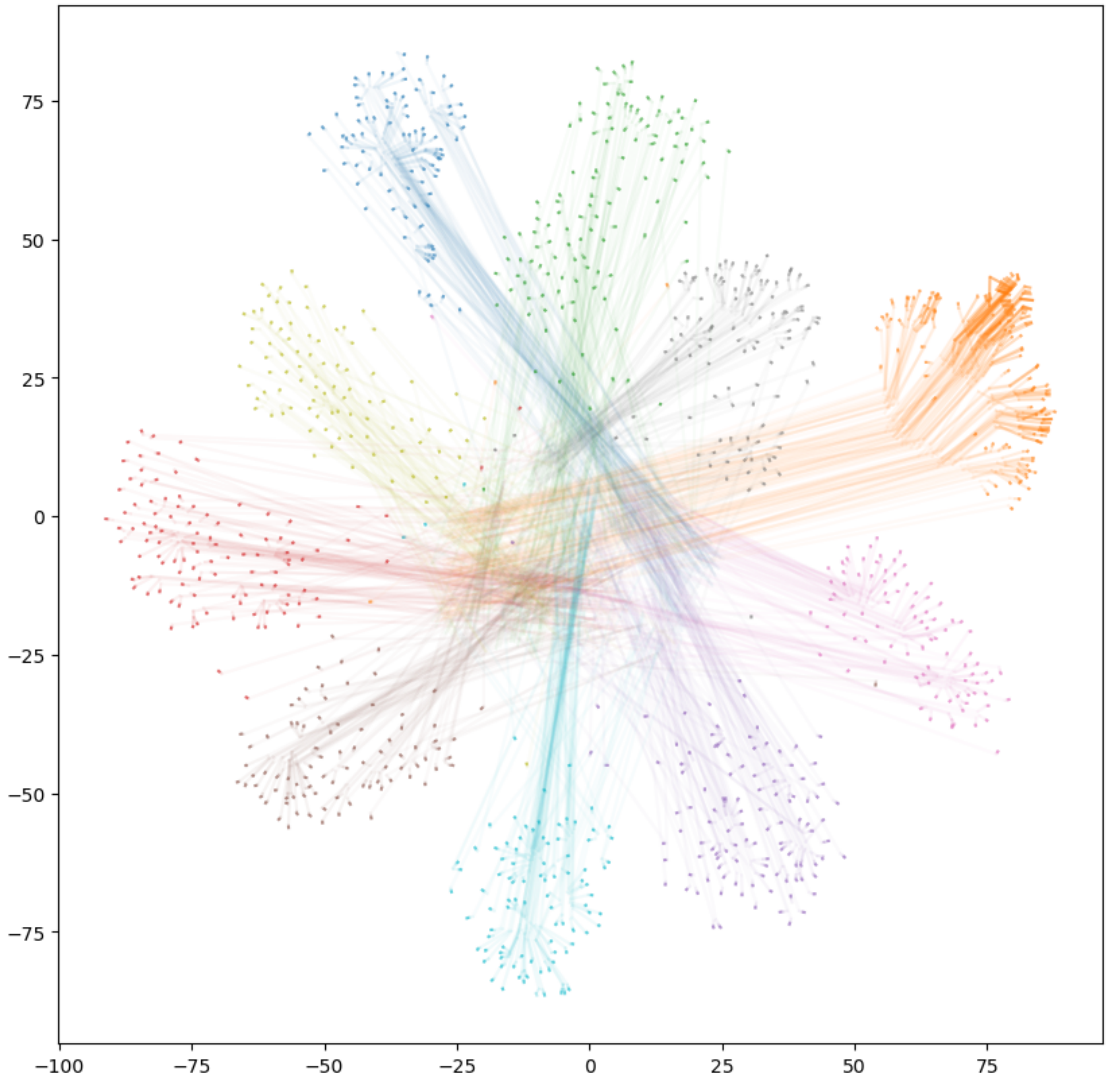
[ ]:
```
activations, targets = get_all_activations(DataType.MNIST, "mnist_cnn", 1, 1000)

points_lst, targets = process_activations(activations, targets, 1000)
```



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



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



0.2.2 COMMENT

Widzimy jak pięknie rozsuwają się reprezentacje warstwy, od nienauczonego szumu po wyraźne klastry.

0.3 PROJECTION COMPARISION

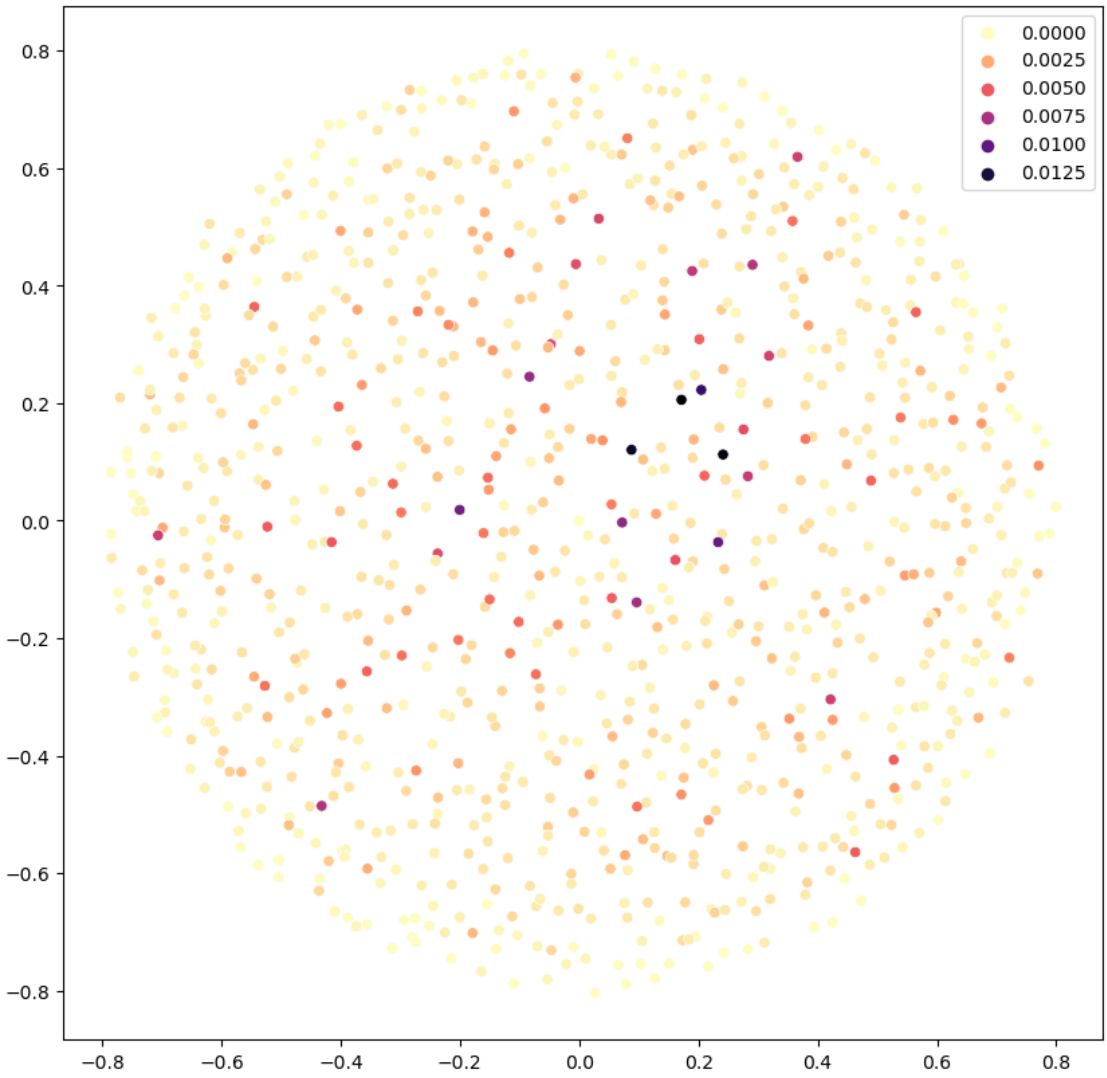
This exercise shows the neuron projection for the last CNN/MLP hidden layer activations, after training. Ignoring the colors for a moment, we see no clear pattern in the neuron projection, except for some ill-defined visual clusters. We next color each point (neuron) based on its ability to discriminate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

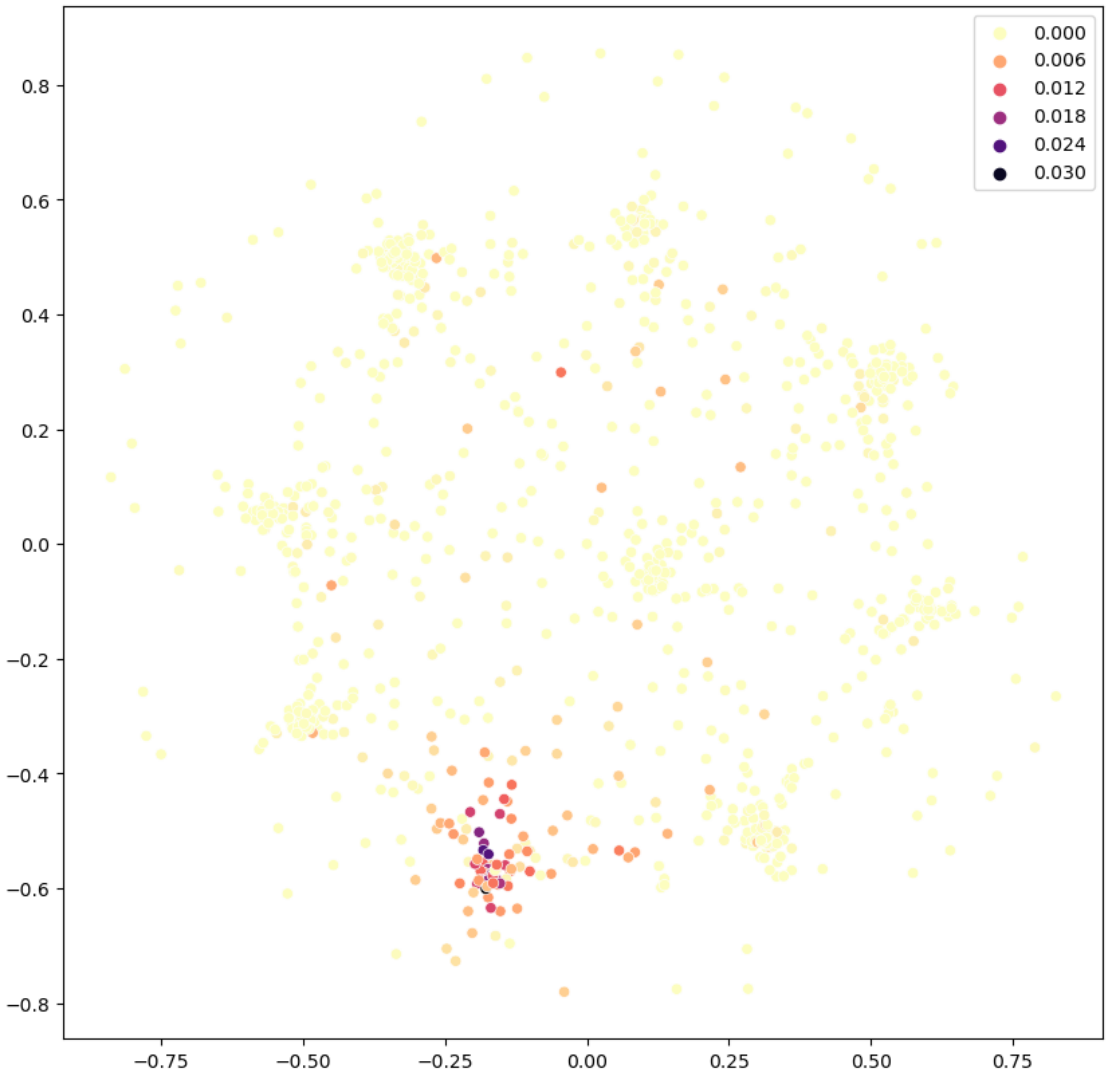
TODO: Utilize the `compare_projections` method to generate a projection for both `mnist_mlp` and `mnist_cnn`, focusing on different labels (e.g., 0 and 5). IMPORTANT: Few things in

`compare_projections` needs to be implemented.

What observations can be made?

[ ]:
```
compare_projections(DataType.MNIST, "mnist_mlp", n_layer=4, label=0, size=1000)
```

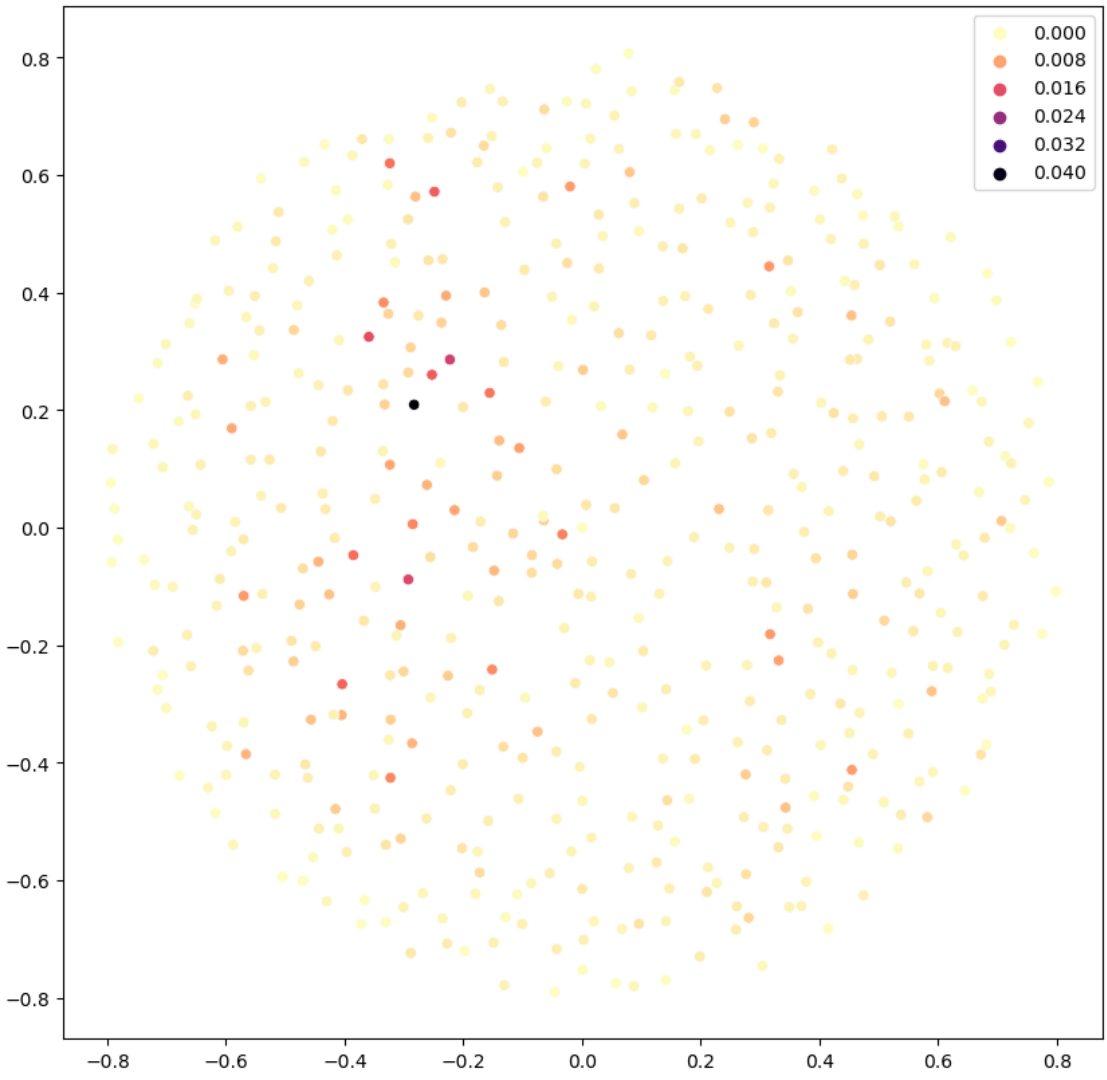


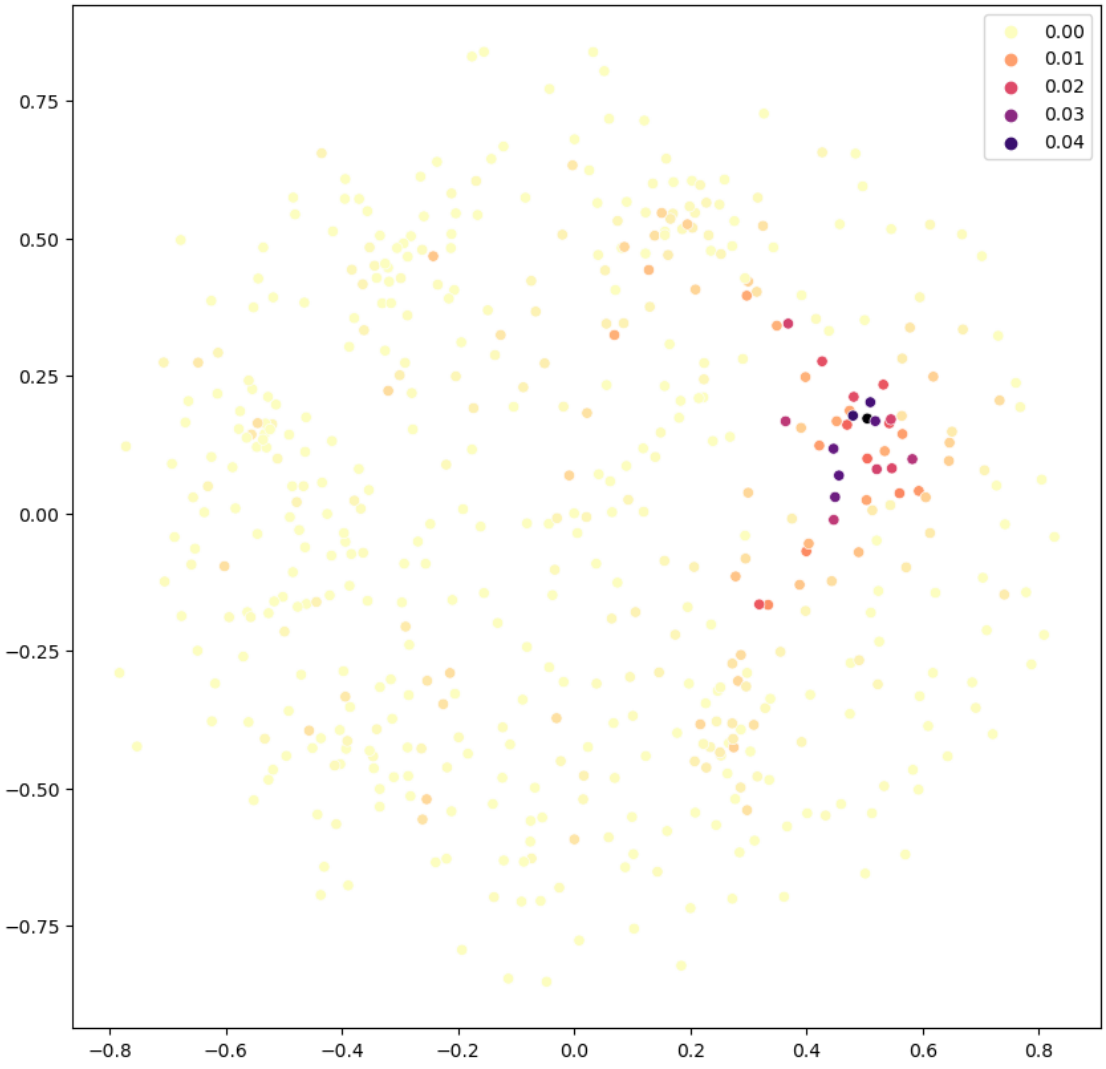


0.3.1 COMMENT

Widzimy wyraźnie, że neurony, które mają podobne embeddingi (wg. MDS - liczone na podstawie ich aktywacji na różnych przykładach) mają podobną siłę dyskryminatywną w stosunku do jednej z klas.

[ ]:
```
compare_projections(DataType.MNIST, "mnist_cnn", n_layer=2, label=0, size=1000)
```


0.000
0.008
0.016
0.024
0.032
0.040
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



0.3.2 COMMENT

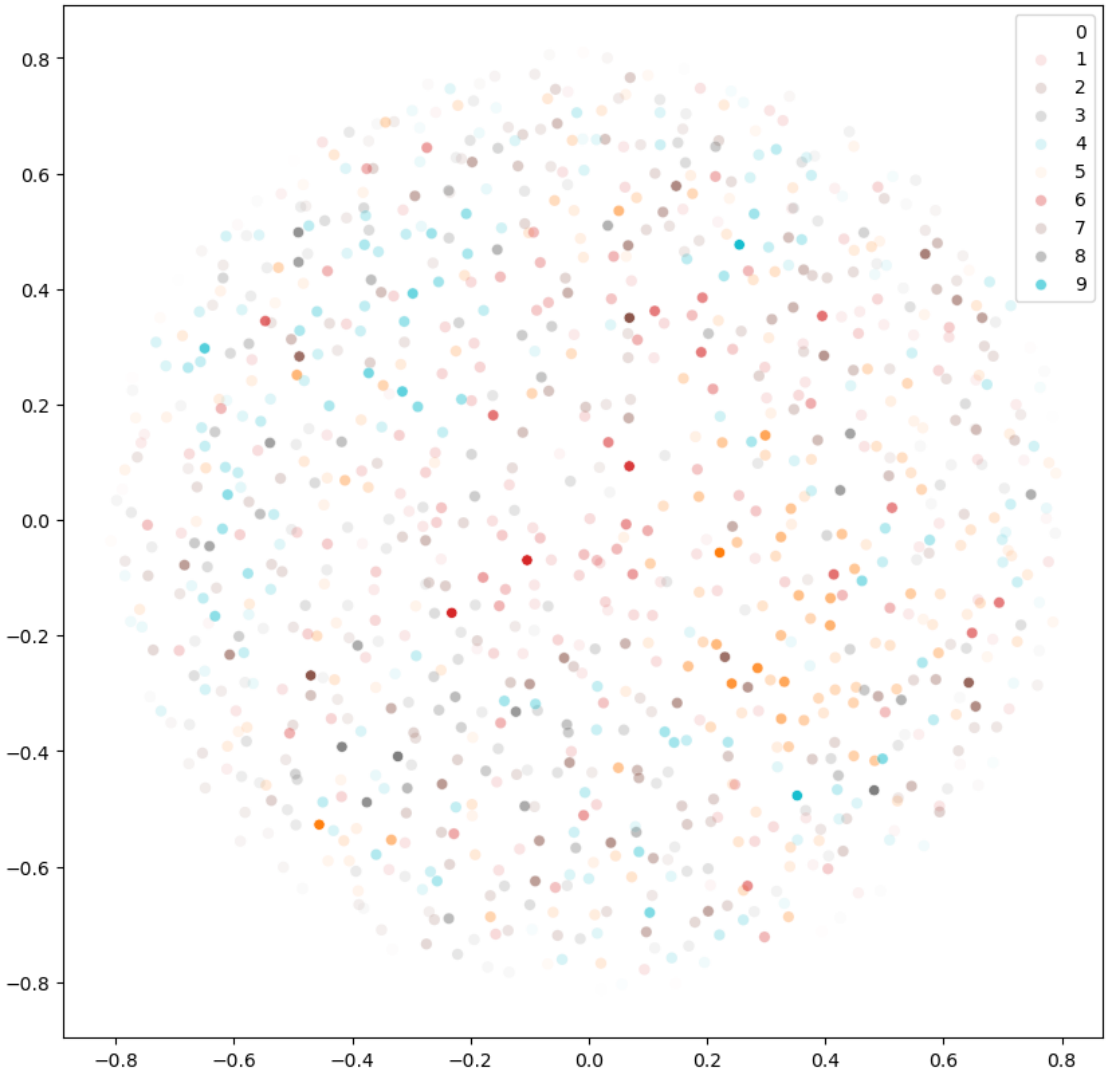
Podobnie jak w przypadku MLP widzimy, że większość neuronów odpowiadających za wykrywanie klasy "0" ma podobny embedding 2d.

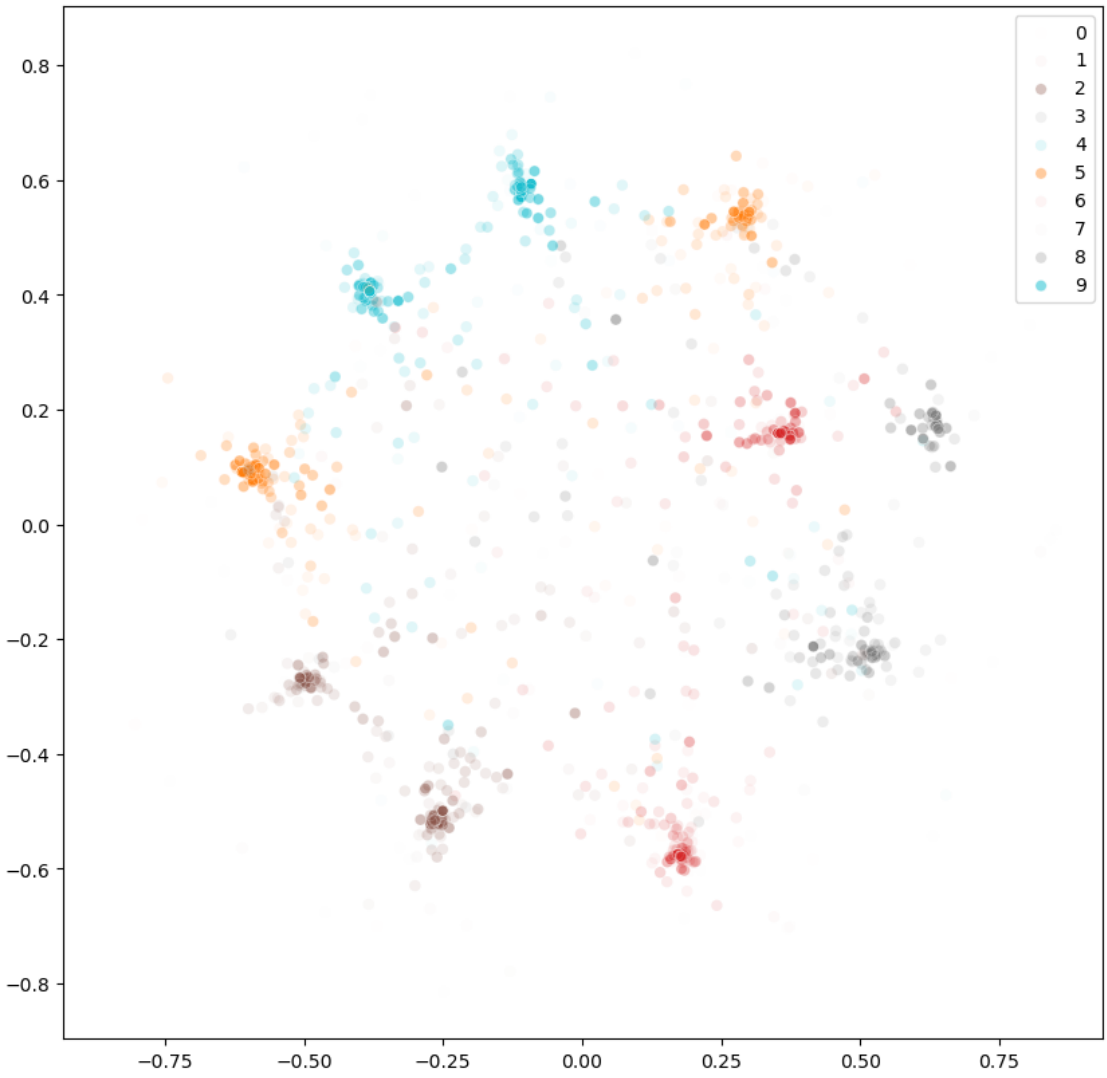
0.4 DISCRIMINATIVE NEURON MAP

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", 4, 2000)
```

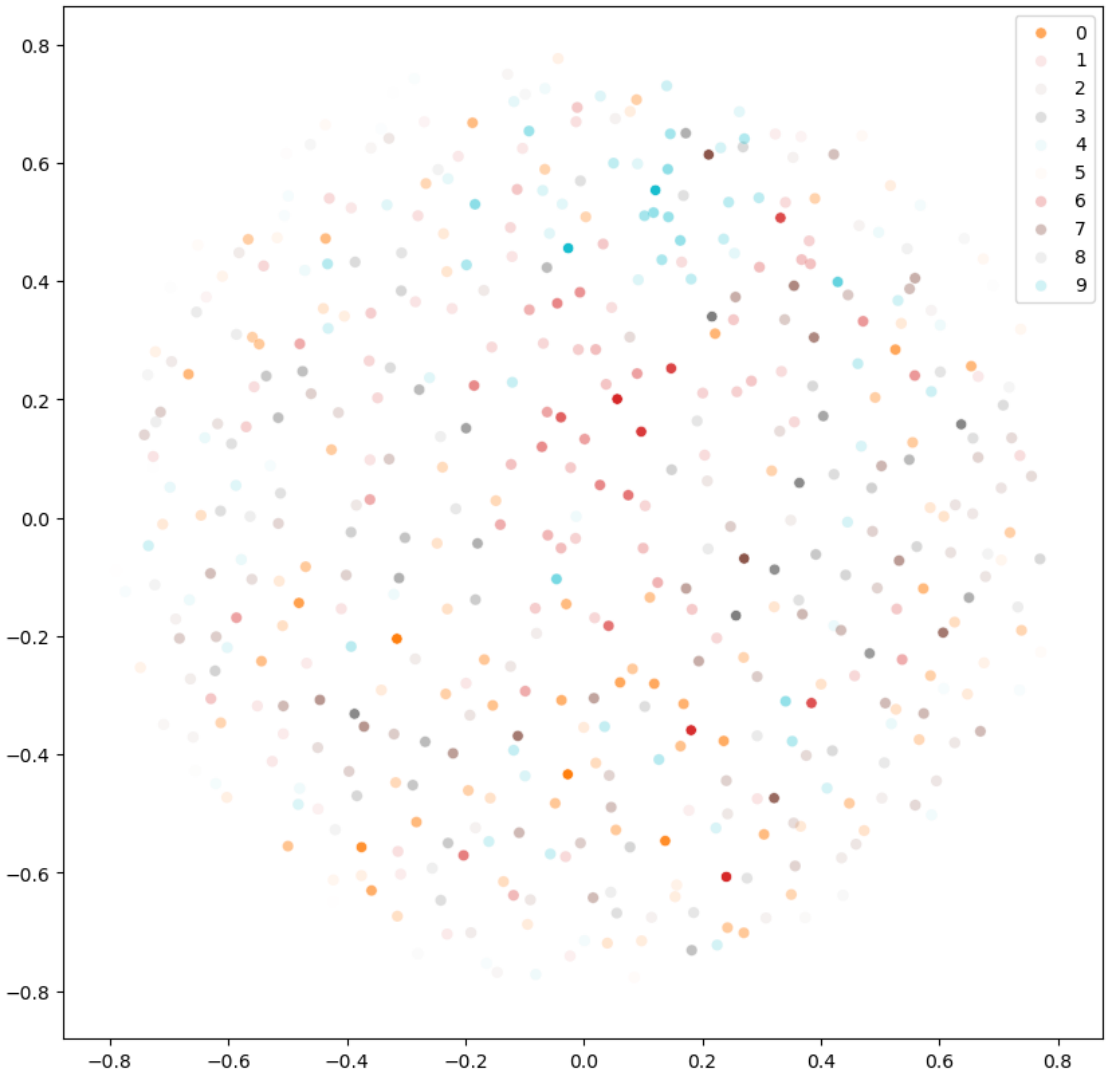


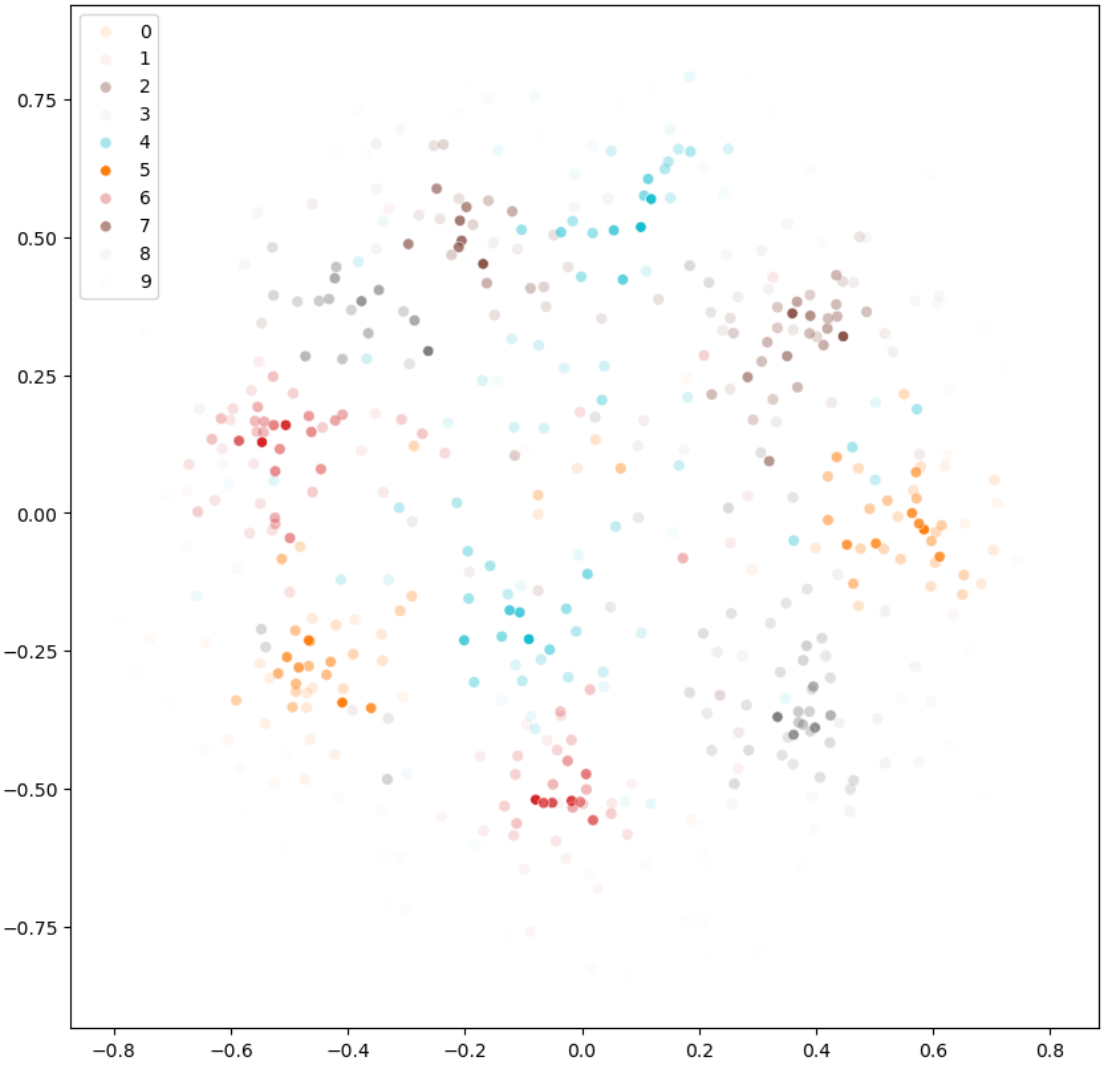


0.4.1 COMMENT

Wyraźnie widzimy, że po treningu embeddingi neuronów które mają dużą siłę dyskryminatywną w stosunku do danych klas leżą blisko siebie. Nie dziwi to, gdyż te neurony prawdopodobnie w większości aktywowały się właśnie na przykładach z tych klas.

[ ]:
```
compare_discriminative_map(DataType.MNIST, "mnist_cnn", 2, 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



0.4.2 COMMENT

Poobnie jak dla MLP widzimy klastry neuronów odpowiadających za wykrywanie danej klasy. Klastry te jednak są mniej spójne i wyraźne niż dla MLP. MOże to wynikać z tego, że filtry CNNów są bardziej uniwersalne i ich kombinacja może pomóc wykrywać konkretne klasy, nie musimy się uciekać tylko do zestawu kilku konkretnych neuronów