An Application-Based Analysis of M-PHATE

Matthew Waismann
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093
mwaismann@ucsd.edu

Chiye Ma
Department of Computer Science
University of California San Diego
La Jolla, CA 92093
cma@ucsd.edu

Abstract

M-PHATE is a visualization method designed to capture the evolution of the hidden units of a neural network over the course of its training. This paper explains how M-PHATE works and how it is useful. From there, an application-based analysis is used to better understand M-PHATE's behavior and interpretation. Such analysis is done by applying M-PHATE to different network types and architectures. Additionally, limitations, such as scalability, are addressed.

1 Summary of the paper and its contributions

M-PHATE is a novel visualization algorithm which combines a multislice kernel and PHATE embeddings to capture the evolution of an evolving graph structure - in our case, neural networks. The goal of the multislice kernel is to capture affinities between hidden activations within and across slices of time (epochs) all the while preserving the temporal structure of the network's evolution through training. The M-PHATE paper provides evidence that suggests only with the combination of the multislice kernel and PHATE do we get these useful temporally-structured embeddings. The goal of these embeddings is to provide useful, interpretable information on the network's generalizablity, convergence, and overall effectiveness. Two separate kernels make up the multislice kernel:

$$K^{(\tau)}_{interslice}(i,j) = \exp(-\frac{||T(\tau,i) - T(\tau,j)||_2^{\alpha}}{\sigma^{a}_{(\tau,i)}})$$

$$K^{(i)}_{intraslice}(\tau,\upsilon) = \exp(-\frac{||T(\tau,i) - T(\upsilon,i)||_2^{2}}{\epsilon^{2}})$$

The idea behind the interslice kernel is to construct affinities between standardized hidden unit activations T for fixed units across epochs τ and υ. The idea behind the intraslice kernel is to construct affinities between standardized hidden unit activations T for pairs of units i and j within epochs.

Combining the temporal structure of the interslice kernel with the intraslice kernel we obtain a multislice kernel which preserves the temporal structure of the evolving units all the while containing information on the units' affinities as well. This structure can be seen in Figure 1.

To visualize the multislice kernel, M-PHATE uses PHATE. PHATE is a dimensionality reduction algorithm that provides 2D and 3D embeddings by applying Multi-Dimensional Scaling to the "informational distance" between rows i and j of the diffusion kernel P^t

$$\phi_t(i,j) = ||\log P^t(i) - \log P^t(j)||_2$$

where P is just a normalized and symmetrized multislice kernel. The hope is that these PHATE visualizations of the multislice kernel will provide insight into the performance, characteristics, and

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

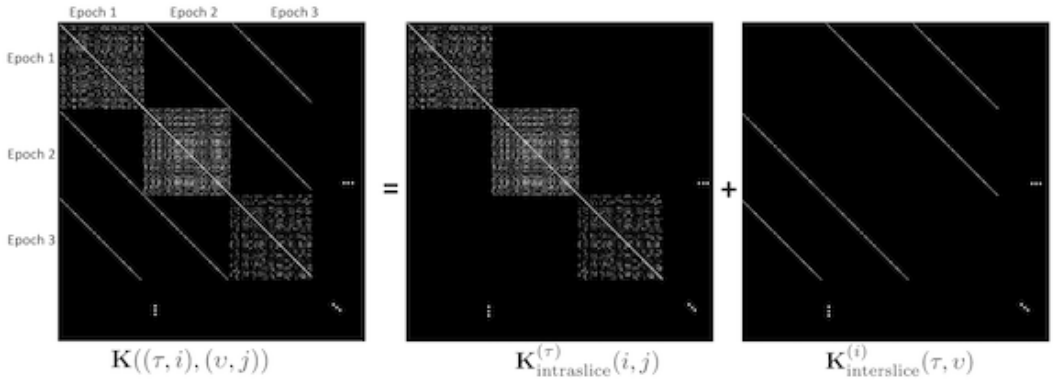

Figure 1: Multislice Kernel

behavior of a neural network. Before M-PHATE, the evolution of a network's hidden layers was shrouded in mystery. However, now with a tool like M-PHATE, there's room for more to be learned about how the hidden units evolve and eventually converge to a model.

2 Extended Research

2.1 Shallow and deep neural networks

A neural network consists of an input layer, an output layer, and hidden layers. A neural network can be categorized as shallow or deep neural networks in terms of how many hidden layers the network has. But how the composition of hidden layers will affect the performance of neural networks is still an ongoing study field in artificial intelligence. Based on the M-PHATE visualization techniques, we were trying to figure out an interpretation of the visualization to help us understand how the neural network performs.g

We were focusing on classification neural networks on MNIST dataset. To compare the visualization of neural networks with a different number of layers, we trained three neural networks with different construction and created multislice kernel for them. All of these three neural networks have an input layer of 192 units which is the pixel number of each digit graph from MNIST and an output layer of 10 units representing digit categories. However, each of these neural networks has different hidden layers. One of the neural networks has three hidden layers each of which has 64 units while the other two neural networks have 4 layers of 48 units and 6 layers of 32 units. Note that the three neural networks share the same number of hidden units in total which help us to reduce the effect of different amount of neurons.

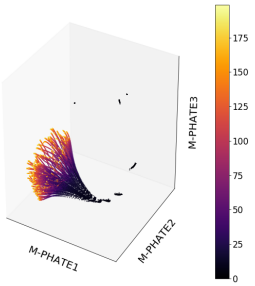

Figure 2: M-PHATE on the ANN with 3 layers

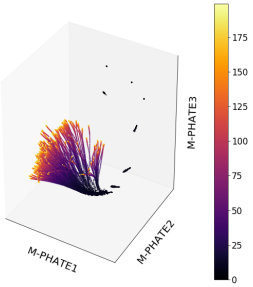

Figure 3: M-PHATE on the ANN with 4 layers

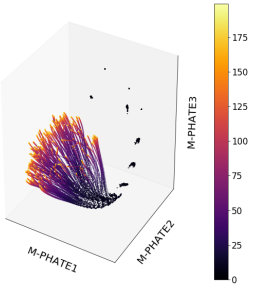

Figure 4: M-PHATE on the ANN with 6 layers

The set of Figure 2, Figure 3 and Figure 4 introduce how the visualizations look differently in shape among these three neural networks. As a whole, the major discrepancy among them is how much the

# of layers	# of neurons per layer	Train Accuracy	Validation Accuracy
3	64	0.9713167	0.96499997
4	48	0.9680167	0.95859998
6	32	0.95385	0.95029998

Table 1: Accuracy of neural networks after epochs of 200

# of layers	# of neurons per layer	Train Accuracy	Validation Accuracy
3	64	0.9713167	0.96499997
4	64	0.97178334	0.963
6	64	0.97941667	0.96429998

Table 2: Accuracy of neural networks after epochs of 200

neurons are separated in the midterm of training. With more layers in the neural networks, neurons seemingly separate earlier. It is believed that the performance of a neural network is positively related to the ability of the neural network to capture complex features from input data. Meanwhile, deep neural networks are inclined to extract more complicated features than shallow neural networks. However, this is not the case here. According to Table 1, with a fixed total number of neurons, an increase of the number of layers will lower the accuracy of classifying digits from MNIST. Thus, these figures should be perceived in another way. Note that these the separation of neurons in these figures shouldn't be compared quantitatively, because all of these visualizations are re-scaled to fit into the plot. In fact, the performance or efficiency of the neural networks can be reflected by how fast the separation of neurons increases over time of training. For instance, the variation of neurons in the figure for the 3-layer neural network increases more expeditiously than that in the figure for the 6-layer neural network, which implies that as the network was being trained, unique and complex features are singled out efficiently which will lead the network trained to an impressive performance.

For further proof of the correlation between thickness distribution and performance of neural networks, we also trained two new 4-layer and 6-layer neural networks while each layers in both neural networks has the same 64 neurons as it has in the 3-layer neural networks. The visualization of the new set of shallow to deep neural networks are shown in Figure 5, Figure 6 and Figure 7.

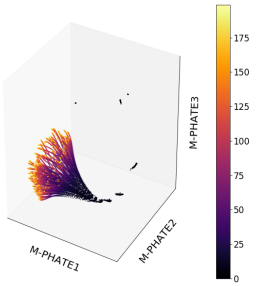

Figure 5: M-PHATE on ANN with 3 layers

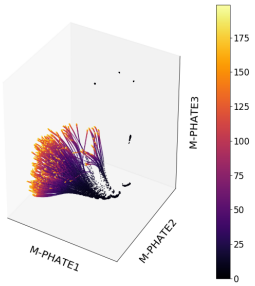

Figure 6: M-PHATE on ANN with 4 layers

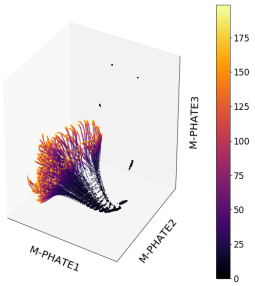

Figure 7: M-PHATE on ANN with 6 layers

The visualization of the two new neural networks exhibits a different shape to those in the first set. The new shapes are having a slimmer necks which reflects as we know the neural networks are competent in figuring out the features from the input are they are supposed to have a better result. The observation can be certified by the accuracy data from Table 2. The two new neural networks are producing as good results as the 3-layer one which is much better than the two networks in the previous set. By the way, in this specific case, more layers don't guarantee a better performance which contradicts the commonsense.

Besides the shape of neurons in the plot, how these neurons are affecting the classifying results could be another aspect to look at. Take the 3-layer neural networks as an example, Figure 8 provides a rough idea of how neurons are distributed on the three layers. The color of the neurons represents

the digit on which the neuron has the highest values on average when classifying. Though they are displaying some kind of clustering situation, it can only be analyzed case by case. That's because, for different neural networks or those same neural networks retrained, the features captured during training might vary a lot and those neurons embodying them will be distributed differently every time. Thus, it is hard to conclude a general rule for all cases.

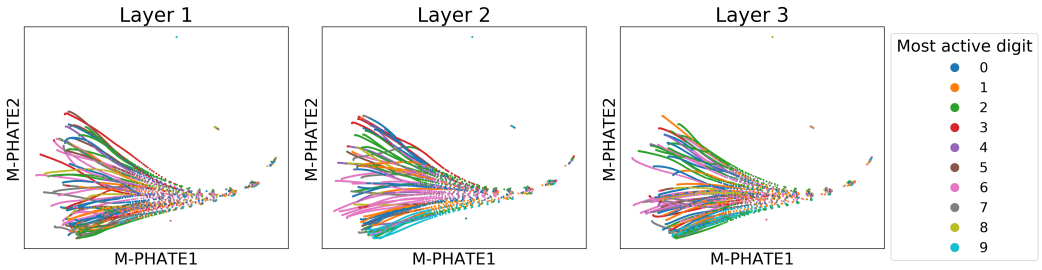

Figure 8: Most active digit on the 3-layer ANN on each layer

2.2 Different types of neural networks

2.2.1 Autoencoder

An autoencoder neural network is an unsupervised Machine learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. An autoencoder is trained to attempt to copy its input to its output. Internally, it has a hidden layer that describes a code used to represent the input. We firstly trained an autoencoder with three hidden layers, in which the middle layer is known as the code as the size of 24 while the other two layers contain 84 neurons respectively. The M-PHATE visualization for the neural network is on Figure 9.

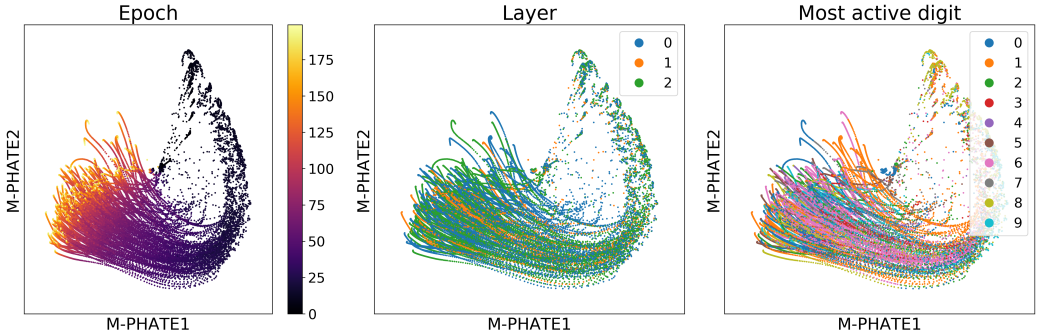

Figure 9: M-PHATE on autoencoder

The three figures successfully depict how the neurons are distributed at a specific timestamp and they are smoothly updated over time of training. We are able to evaluate the performance by looking at several digit graphs from MNIST after encoded and then decoded. The reconstructed digits from the 3-layer autoencoder are inserted as Figure 11. When comparing to the corresponding original input digit shown in Figure 10, we can get that the rough sketch of the original digits have been preserved and the reconstructed digits are still recognizable.

For comparing other autoencoders with different structures, we trained another two autoencoders that have the same size of code which means the middle layer of these autoencoders have remained as 24. The difference between these autoencoders is the number of layers they have. The previous autoencoder has two extra hidden layers each of that has 84 neurons. One of the new autoencoders has 4 extra hidden layers each of that has 42 neurons and the other has 6 extra hidden layers each of that

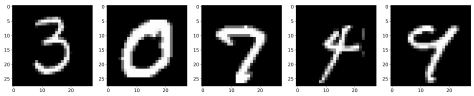

Figure 10: Sample input digit from MNIST

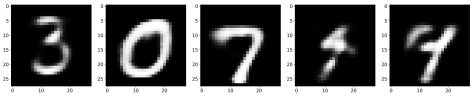

Figure 11: Sample output digit for the 3-layer autoencoder

has 28 neurons. Note that to control the influence of the amount of neurons, all three autoencoders have the same number of neurons in total.

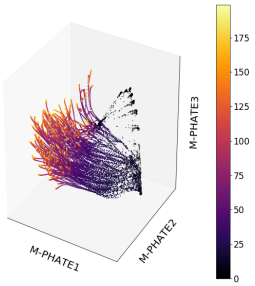

Figure 12: M-PHATE on autoencoder with 3 layers

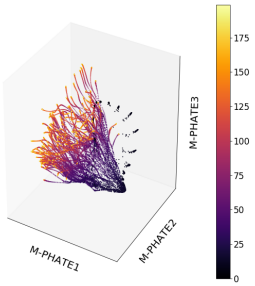

Figure 13: M-PHATE on autoencoder with 5 layers

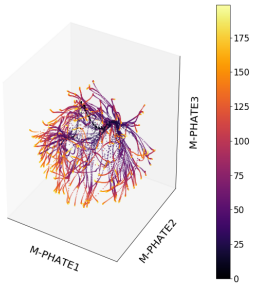

Figure 14: M-PHATE on autoencoder with 7 layers

The set of Figure 12, Figure 13 and Figure 14 are the M-PHATE of these three autoencoders with different structures. As we can see, the plot for the 7-layer autoencoder produces a more chaotic stretching pattern than the other two autoencoders with fewer layers, from which, we obtain the information that the last autoencoder is not trained with a steady descent toward the local optimum. If we are looking at the reconstructed results from the two new autoencoders from Figure 15 and Figure 16, the 7-layer autoencoders performs apparently worse than the other two.

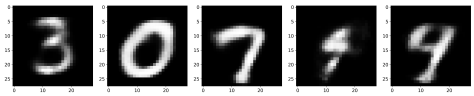

Figure 15: Sample output digit for the 5-layer autoencoder

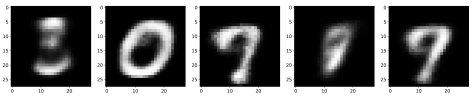

Figure 16: Sample output digit for the 7-layer autoencoder

2.2.2 Knowledge distillation

A fairly recent development in machine learning was the design of knowledge distillation networks. The main idea of knowledge distillation is to "distill" the knowledge learned from a large, cumbersome model (e.g. deep networks, ensembles, etc.) to a smaller, less computationally expensive network. This knowledge transfer can be achieved by taking temperature-modified softmax outputs in the cumbersome model and using those along with a traning set (with or without labels) to train the "student" network with a modified loss function. In our case, this loss function is a weighted sum of correct labels and temperature-modified softmaxes from the teacher model. These student networks are shown to outperform their traditional counterparts.

Here, we are interested in visualizing the PHATE of these student networks with the goal of understanding how these students learn, generalize, and perform. Additionally, we will compare the PHATE of these networks to similar network architectures trained in the conventional way.

Firstly, we will construct a simple teacher network consisting of 2 hidden layers with 1200 rectified linear units and Dropout as a regularizer. We then applied knowledge distillation, transferring the knowledge to a much smaller 2 hidden layer network with only 64 rectified linear units each. The performance of these models can be found in Table 3.

Model	# of layers	# of neurons per layer	Train Accuracy	Validation Accuracy
Teacher	2	1200	0.9975	0.9818
Student	2	64	0.9868	0.9762
Standard	2	64	0.9558	0.9618

Table 3: Comparison of Teacher and Student Networks with MLP architecture

We can see that the student model did outperform the standard model, with only 238 test errors compared to the standard model's 442. We can see the visualization of the PHATE of these two models in Figure 17 and Figure 18.

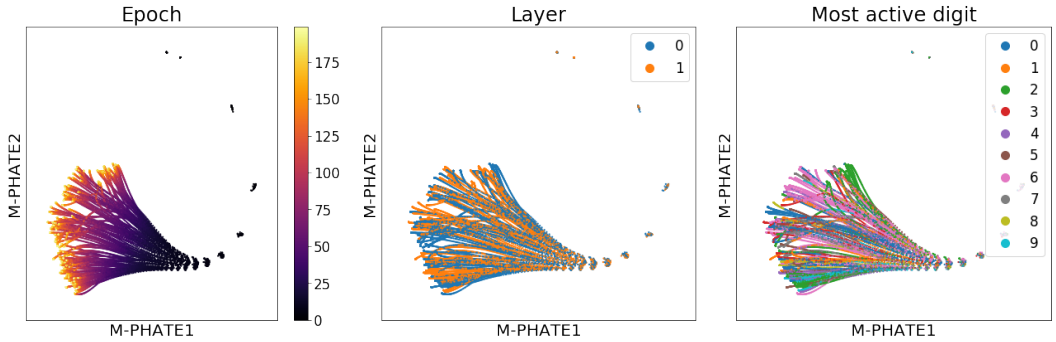

Figure 17: M-PHATE on standard network

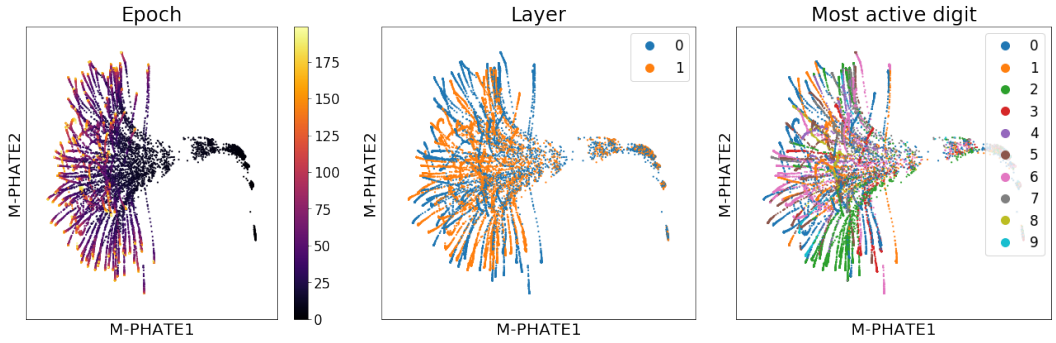

Figure 18: M-PHATE on student network

We see the stronger performing student network has more separation in later epochs than its standard counterpart. This supports the claim made by the authors of M-PHATE that this increased separation is indicator of increased generalization ability in the network. This more abstract structure speaks to the student network's inheritance of additional knowledge from the teacher model.

Another point of interest pertained to how PHATE would visualize student networks trained from different teacher networks. To better understand this, we created a new teacher network while keeping every other hyperparameter fixed, including the data. The teacher model was a convolutional neural network with a total of 1,199,882 parameters. A student network with the same architecture as before was trained with knowledge distilled from this teacher and labelled training data. The performance of these models can be found in Table 4.

Interestingly, while the student still performed better than the standard network it failed to outperform the other student network despite being distilled with knowledge from the most accurate teacher.

Model	Train Accuracy	Validation Accuracy
Teacher	0.9951	0.9935
Student	0.9892	0.9728
Standard	0.9558	0.9618

Table 4: Comparison of CNN Teacher and MLP Student Network on MNIST

We see in Figure 19 that the PHATE of this student network is drastically different from the traditional student network and the MLP student network. This result proves to be quite surprising since the student network had identical architecture to the other student network. This seems to suggest that the student inherited more complex behaviors from the teacher CNN. A 3D embedding of this student network can be found in Figure 20.

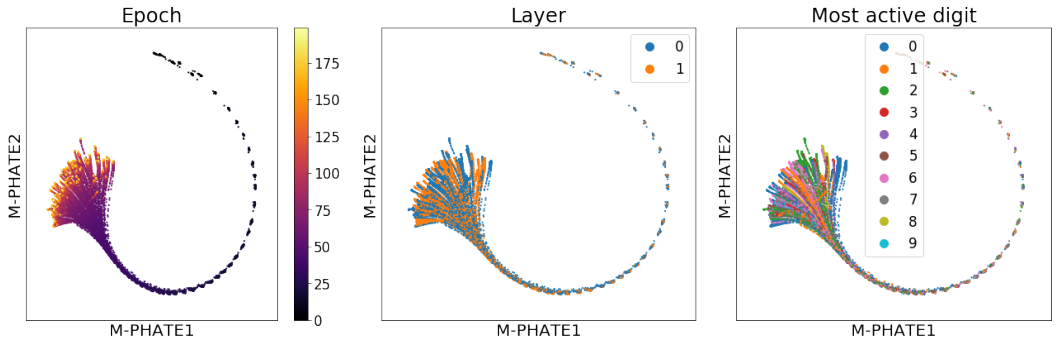

Figure 19: M-PHATE on student network

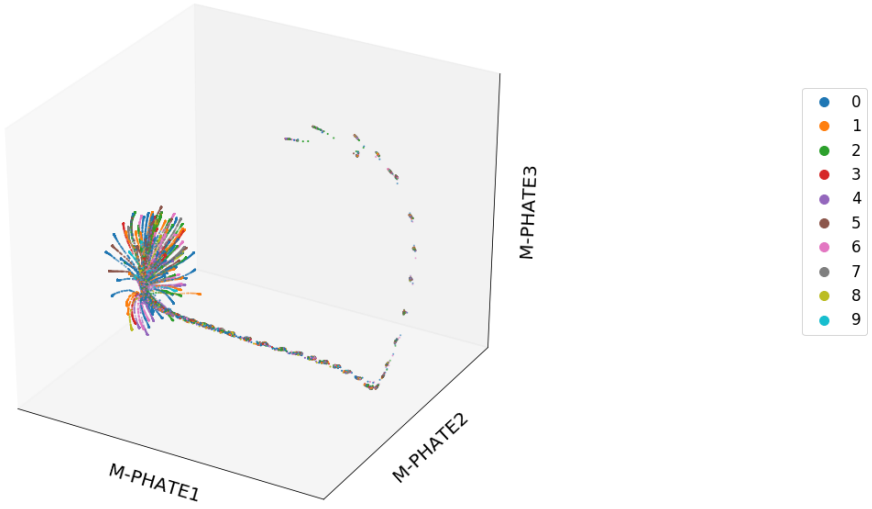

Figure 20: M-PHATE on student network

2.3 Limitations of M-PHATE

2.3.1 Scalability

The neural networks become more complex as one of its goals is to emulate human's brain which has 80 billion neurons in total. Thus, whether M-PHATE is able to convey meaningful information for a larger and more complicated neural network is crucial for the algorithm itself.

For evaluate the M-PHATE visualization for a comparatively large neural network, we trained a two-layer neural network each of whose layer includes 1000 neurons. The neural network is for purpose of classification on MNIST and the three major visualization figures are shown on Figure 21.

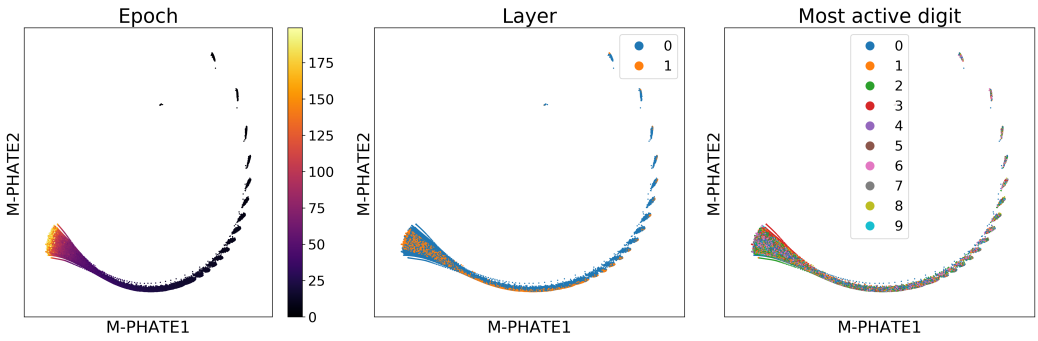

Figure 21: M-PHATE on huge neural networks

From the first figure with the color scheme for epochs, the shape of visualization turns out to be a long tail which apparently reduces the variation of neural networks plotted in the figure. However, the color transition reveals that in the first 50 epochs of training, neurons were rapidly altering when compared to the rest of epochs. The pattern of training will cause the end of neurons in the visualization unable to demonstrate various neurons geometrically. Take the right two figures as examples, neurons in these two figures are colored by which layer the neuron belongs to and for which digit the neuron has highest values in average for. Since the shape suppress the information beneath the distribution of neurons, we barely figure out any attributes of the neural network by looking at the figures. One possible solution for the issue is to visualize without the first part of epoch to center the tail part of visualization in the plot. Considering that the huge amount of neurons might interleave with other, randomly filtering a portion of neurons possibly convert the figure into a more easily interpretable one.

2.3.2 M-PHATE struggles to generalize to larger networks

While M-PHATE produces great visualizations of smaller, simpler neural networks, M-PHATE would not be a viable option for deeper and more complex architectures. As these networks become more complex with more and more parameters to be learned M-PHATE becomes less and less computationally viable. Furthermore, for larger networks trained on fewer epochs, there isn't much of a time trace to follow. It is worth noting, however, that we did show that simpler "knowledge distilled" student networks may contain valuable information about the underlying behavior of the teacher network. This is certainly an area that could use further exploration and hopefully provide some answers as to how the hidden units in these complex networks behave and evolve.

References

[1] Scott Gigante, Adam S. Charles, Smita Krishnaswamy,Gal Mishne. Visualizing the PHATE of Neural Networks. arXiv,1908.02831, 2019.

[2] Kevin R Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel Burkhardt, William Chen, Antonia van den Elzen, Matthew J Hirn, Ronald R Coifman, Natalia B Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing transitions and structure for high dimensional data exploration. bioRxiv, page 120378, 2017.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, “Distilling the knowledge in a neural network,” arXiv preprint arXiv:1503.02531, 2015.