SNNVis: Visualizing Graph Embedding of Evolutionary Optimization for Spiking Neural Networks

Junghoon Chae
Oak Ridge National Laboratory
Oak Ridge, USA
chaej@ornl.gov

Seung-Hwan Lim
Oak Ridge National Laboratory
Oak Ridge, USA
lims1@ornl.gov

Shruti Kulkarni
Oak Ridge National Laboratory
Oak Ridge, USA
kulkarnisr@ornl.gov

Catherine Schuman
University of Tennessee
Knoxville, USA
cschuman@utk.edu

***Abstract*—While Spiking Neural Networks (SNNs) show a lot of promise, it is difficult to optimize them because applying traditional gradient-based optimization techniques is difficult. Even though evolutionary algorithms (EAs) have been shown to promise to optimize SNNs, understanding the relationship between evolving the characteristics of SNNs and their performance to improve the optimization algorithm is challenging because of the complex characteristics and huge population size. We propose visual analytics with novel graph embedding for evolutionary SNNs to address the challenges. While existing graph embedding techniques have limitations in preserving the specific features of the nodes and edges, our approach maintains them. Also, we develop visual analytics for understanding the relationship between the network performance and the features of nodes and edges and exploring and analyzing the evolving SNNs to build insights into improving the EA.**

***Index Terms*—Visual analytics, SNN, evolutionary algorithm**

I. Introduction

Researchers from the domains of computational neuroscience and neuromorphic engineering have designed neural network models called Spiking Neural Networks (SNNs) inspired by information processing in the biological human brain. A SNN is depicted as a directed graph where neurons are nodes and synaptic connections are edges with associated weights. Information propagation occurs through edge traversal, reflecting spike transmission between neurons. This graph-based representation facilitates analysis using graph algorithms, offering insights into SNN structures and dynamics.

While SNNs offer promising advantages such as biological plausibility, energy efficiency, and temporal processing capabilities, optimizing SNNs can be challenging due to the non-differentiable nature of spike-based computation and the discrete nature of spike timing. This can make it difficult to apply traditional gradient-based optimization techniques, requiring the development of specialized optimization algorithms. An evolutionary algorithm (EA) has been shown to promise to optimize SNNs [1]. EA starts by randomly creating networks within a specific criterion and then generates new child networks using high-performance parent networks rather than bad-performance ones based on principles of biological evolution: selection, crossover, and mutation. This evolutionary iteration continues until reaching a specific max generation (epoch) or gaining networks having good enough performance.

Understanding the relationship between SNN characteristics and performance is crucial to improving EA-based optimization methods because the methods aim to find characteristics of SNNs that can lead to better performance. However, it is difficult to find the important characteristics that are strongly related to performance from complex data in the form of graphs. Graph embedding techniques can be considered, converting the complex SNNs into a lower-dimensional space, represented by vectors. It aims to capture the important relationships between nodes in the original graph and makes the data more manageable and easier to process for machine learning algorithms. However, existing graph embedding algorithms have limitations in preserving the important characteristics of SNNs. Understanding embedding results and turning them into actionable insights can also be challenging.

In this paper, we introduce a visual analytics tool called *SNNVis* with novel graph embedding for evolutionary SNNs. While most existing graph embedding approaches mainly focus on the topological structure of a network, our approach supports not only the node connections but also the specific characteristics of SNNs. It unveils the relationship between the network performance and the features of nodes and edges, such as node threshold, edge delays, and edge weights. We use Evolutionary Optimization for Neuromorphic Systems (EONS) [2] as the SNN optimization algorithm for our tool. Also, *SNNVis* supports exploring and analyzing the evolving SNNs, visualizing how the characteristics evolve and converge to better performance SNNs as evolution progresses.

The manuscript is authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan. http://energy.gov/downloads/doe-public-access-plan.

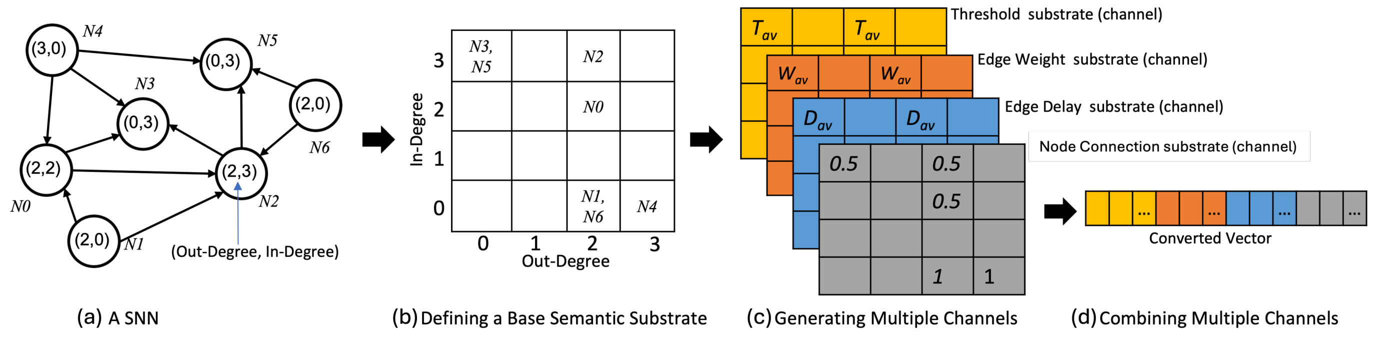
(3,0)
N4
(0,3)
N5
N3
(0,3)
(2,0)
N6
(2,2)
(2,3)
N0
N2
(2,0)
N1
(Out-Degree, In-Degree)
(a) A SNN
In-Degree
Out-Degree
N3, N5
N2
N0
N1, N6
N4
(b) Defining a Base Semantic Substrate
Threshold substrate (channel)
Edge Weight substrate (channel)
Edge Delay substrate (channel)
Node Connection substrate (channel)
T_{av}
W_{av}
D_{av}
0.5
1
(c) Generating Multiple Channels
Converted Vector
(d) Combining Multiple Channels

Fig. 1. Graph embedding process. Given a network, we compute In-Degree (i) and Out-Degree (o) for each node (First step). Then, we define a 2D array ($o_{\max} \times i_{\max}$) called as a Base Semantic Substrate and assign each node to a position of the substrate according to its i and o (Second step). We create four new substrates for the four network features: node connection, node threshold, edge delay, and edge weight respectively based on the base substrate. For each feature, we compute the feature values and map the values into the substrate (Third step). Finally, we generate a vector by combining the substrates (Fourth step).

II. Related Work

Many visualization methods have been developed to better understand the learning process and behavior of SNNs, but due to space limitations, we only present a few of the most relevant studies. Drouhard et al. [3] and SNN3DViewer [4] proposed a visual analytics method for SNNs of neuroscience-inspired architectures. Their visualization tool is to aid in the understanding of network learning activities and the decision-making process. Disney et al. [5] presented a software ecosystem for a neuromorphic hardware platform called Dynamic Adaptive Neural Network Arrays (DANNAs). The visualizer of the system allows users to visually explore the structure and properties of a running instance of DANNA, visualizing a snapshot of the DANNA network, how their states have changed between snapshots. The visualization is helpful in debugging the simulator and evaluating the hardware.

III. Graph Embedding for SNNs

Prior research on deep neural network models has predominantly centered on visualizing trained filters and attention maps to reveal learned features and areas of focus. However, evolving SNNs lack stored information and learned features, complicating our understanding of model behavior. Hence, our primary objectives revolve around extracting features from populated SNNs and translating them into actionable insights for model refinement. To achieve the goals, first we convert the complex SNNs into a lower-dimensional space, represented by vectors using graph embedding.

Each SNN is represented as a directed graph. Neurons and synapses are mapped as nodes and edges, respectively. The direction of each edge corresponds the direction of spike signals of each synapse. Therefore, it is necessary to consider the multiple features of a network as one, rather than computing each node separately. However, most existing graph embedding methods [6] focus on node embedding rather than whole-graph embedding and undirected graphs. Also, they have limitations in preserving the specific features of the nodes and edges: node threshold, edge delay, and edge weight that determine the operation of the SNNs. In this work, we consider not only the node connections but also the network features.

The overall embedding process is illustrated in Figure 1. First, given a network, we count the incoming edges (In-Degree) and outgoing edges (Out-Degree) for each node in Figure 1 (a) and then create a 2D array called Base Semantic Substrate using the In and Out degrees in Figure 1 (b). We assign each node to a position of the substrate array according to its In-Degree (i) and Out-Degree (o) where i and o are used as the row and column indexes respectively. The following computations are based on this base semantic substrate. First, we create a new *Node Connection* substrate. Then, we count the number of nodes (N) for each position of the substrate and normalize the numbers ($1 \geq N_{\mathrm{m}} \geq 0$). Finally, N_{m} is mapped into the node connection substrate (see the gray substrate at the third step in Figure 1 (c)). This substrate does not fully reflect the topological structure of the network, but it captures node connection changes.

We create a new substrate for the node threshold (T). For each node, its T is assigned to the substrate according to its i and o. For the nodes with the same i and o, we compute the average T (T_{av}). T_{av} are mapped into the *Threshold* substrate (yellow substrate in Figure 1 (c)). For edge delay and weight, we create two more substrates and map average delay (D_{av}) and weight (W_{av}) into the *Edge-Delay* (blue) and *Edge-Weight* (orange) substrates respectively in the same manner as above. As a result, the locations of mapping values inherit the node connections of a network and the values represent the characteristics of the nodes and edges. We note that the substrates of a network can be considered as different channels like the red, green, and blue channels of a color image. Then, we flatten the four substrates (2D arrays) and combine them into one vector as shown in Figure 1 (d). Finally, all populated networks are embedded into a set of vectors.

IV. Visual Analytics Dashboard

SNNVis aims to understand the graph embedding results and investigate the evolutionary process, evolving network characteristics through visualizations. As shown in Figure 2, *SNNVis*

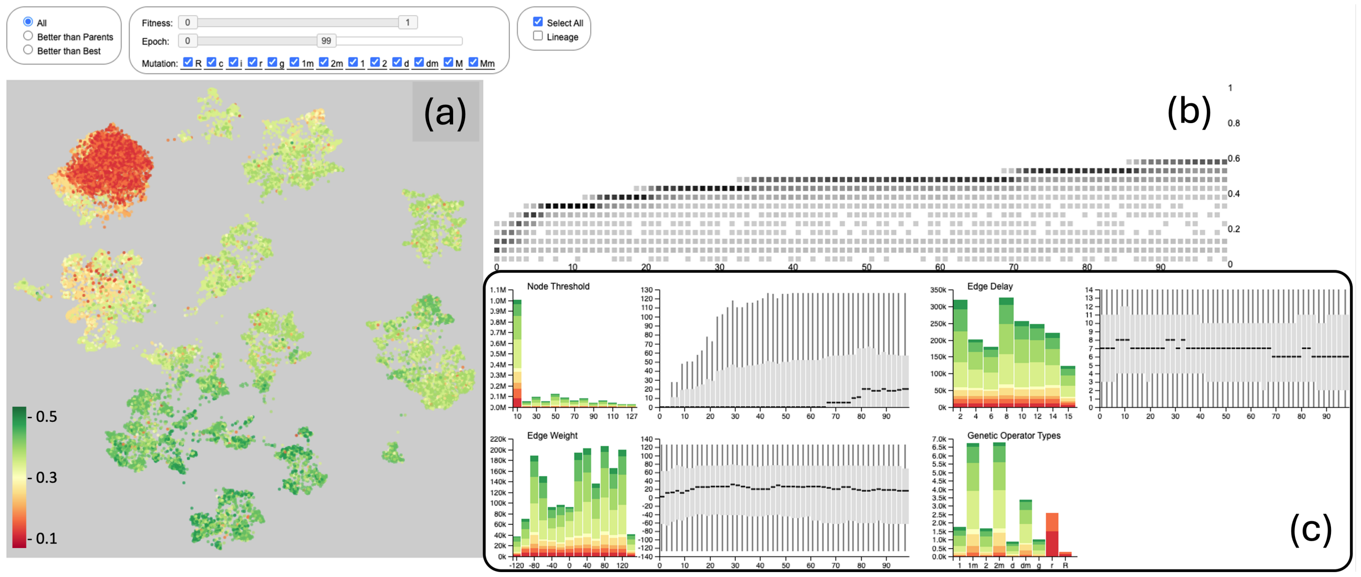

Fig. 2. SNNVis consists of (a) graph embedding view, for showing a graph embedding result and classifying SNN clusters, (b) performance heatmap view, showing the distribution of fitness scores for each generation, and (c) network feature view, for analyzing network features and genetic operator types of selected SNNs. To investigate the evolutionary process and evolving SNNs, users are able to filter and select SNN populations of interest using multiple interactive analytics features and tightly coordinated views.

consists of multiple coordinated visualization components including the graph embedding view (a), the performance heatmap (b), and the network feature view (c). Each view visualizes different aspects of evolving SNNs. The interconnected views also provide highly interactive capabilities for effective analysis. The following subsections describe data and the details of each view.

A. Data

We used SNN populations generated by using EONS for a classification task of Digits data [7]. Digits data is optical recognition of handwritten digits dataset (10 classes, about 180 images per class, 1,797 images in total). The dataset includes 100 generations, 250 SNN per generation, and 25,000 networks in total. The best fitness score is 0.56, the average is 0.39 with the STDEV of 0.13, and the median is 0.43.

B. Visualizing Graph Embedding

Dimensionality reduction: We project the embedding results for each network at a 2D latent space using dimensionality reduction that allows high-dimensional data to be visually interpreted revealing clusters and outliers. In this work, we use the technique for evaluating our graph embedding method, clustering SNN, and investigating the evolving process. There are many different dimensionality reduction techniques. We tested the three most popular methods including principle component analysis (PCA) [8], T-distributed stochastic neighbor embedding (t-SNE) [9], and uniform manifold approximation and projection (UMAP) [10]. While PCA is a linear technique, t-SNE and UMAP are non-linear dimensionality reduction techniques. We tested the techniques on multiple datasets. UMAP often outperforms other techniques in terms of speed and scalability. Also, UMAP often produces results that better reflect the global structure of the data while still maintaining the local relationships between nearby points.

Visual representation: The graph embedding view in Figure 2 (a) shows the projection results of the 25,000 SNNs in the 2D latent space. This view is a major visualization component of the visual analytics workspace and a starting point of the analytics process. Users can evaluate embedding results, investigate clusters and outliers, and understand the evolving patterns. Each dot corresponds to a network and its color represents the performance (fitness score, f) of the network ($1 \geq f \geq 0$). Dark red means a low fitness score and yellow and dark green do medium and high fitness scores respectively. There are many clusters where instances (networks) are close together and have similar colors. This reflects each cluster has similar network characteristics.

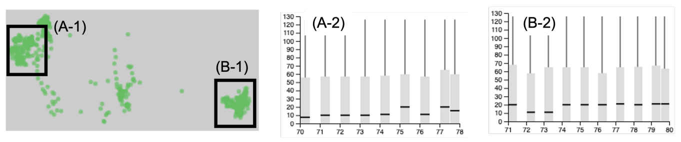

Fig. 3. Comparing clusters: The two separate clusters (A-1 and B-1) have the same fitness score, but their node threshold distributions are different.

C. Visualizing Network Performance and Characteristics

Performance heatmap: The grid-based heatmap in Figure 2 (b) shows how the performance changes as they evolve and its distribution for each generation. The generation is mapped to the x-axis and the performance value to the y-axis. The heatmap is constructed by partitioning the visible portion of the data into two-dimensional bins in both the y- and x-axis dimensions (dark black means more networks).

Network feature view: The network feature view in Figure 2 (c) allows users to analyze the node and edge features to understand the relationships between the features and network performance. For the three features: node threshold, edge delay and weight, each feature has a pair of a stacked bar histogram and a boxplot. The histogram shows the distribution of feature values, with the color of the stacked bars indicating performance (the color scheme is the same as of the graph embedding view). The boxplots show the change in feature values as evolution progresses. For example, the node threshold histogram displays the distribution of the threshold values of entire nodes of selected networks. Also, the stacked bars of the histograms show the portions of corresponding networks. The boxplot shows the degree of dispersion in node thresholds for each generation. Unlike the histograms of the three network features, the histogram at the bottom-right shows the distribution of the selected networks with respect to the types of genetic operators. In Figure 2, the histograms and boxplots show the results when selecting the entire SNN population. In this case, the node threshold for most networks is low, below 10, but after generation 65, the number of nodes with high values increased. For the delay and weight, no drastic change can be seen. For the genetic operator type, large number of networks are generated by a first or second child of crossover and mutation. Also, we can realize the randomly generated networks have low fitness scores.

D. Interactive visual analytics:

Network filtering: *SNNVis* provides multiple filtering options for interactive analysis using the control panel at the top-left in Figure 2. Users can select SNNs that have a specific range of generations and fitness scores and select SNNs generated by specific genetic operator types. Genetic operator types used in this study and their descriptions are in Table I. This is useful when filtering randomly generated SNNs and analyzing the relationship between the genetic operator types and optimization progress. Also, users can select networks that have better performance than their direct parents for each generation. Similarly, users can select better networks than the best ones in the previous generation. This is a feature that hides the intermediate results of finding the optimal value. For example, users can see how the best fitness score changes, how network features are optimized, and what types of genetic operators produce better SNNs. Selecting the filtering options shows the corresponding information of the selected SNNs across the three views.

Cluster analysis on the embedding view: Users can use region selection using mouse dragging in the embedding view and also apply a combination of the filtering options (e.g., generation, fitness score, and genetic operator type). As other visualizations are connected to the embedding view, the views make it easier to see the corresponding information of the selected SNNs. For example, once selecting networks, the performance heatmap shows only the selected networks, and the histograms and the boxplots show the feature values of the selected networks. Cluster analysis of SNNs, finding specific sets of SNNs that deliver good or bad performance is critical to optimize the SNNs. The SNNs do not have specific labels for classification. The generations or the fitness scores of the networks cannot be used as labels for classification, because the networks with totally different generations or characteristics can achieve similar fitness scores. An example is shown in Figure 3. We select two different clusters (A-1 and B-1) in the graph embedding view where the two clusters include networks with the same fitness score. The boxplot (A-2) shows the distribution of node thresholds of the cluster (A-1) and the boxplot (B-2) is for the cluster (B-1). The median and upper quartile values of most generations of B-1 are higher than A-1. Network clustering via graph embeddings is therefore essential for revealing hidden relationships between network characteristics and performance. For example, in the embedding view, there is a big red (low fitness) cluster at the upper-left of the view, while a couple of green and dark green clusters are located at the bottom-middle. This view also allows users to detect outliers with fitness scores that are significantly different from the dominant score of the cluster.

TABLE I
GENETIC OPERATOR TYPES

Label	Description
R	Randomly generated at the initial epoch
dm	Duplicate and mutation
1	First child of crossover
2	Second child of crossover
1m	First child of crossover and mutation
2m	Second child of crossover and mutation

V. CONCLUSION AND DISCUSSION

We introduce *SNNVis*, visual analytics with novel graph embedding for evolutionary SNNs. We introduce a novel graph embedding for preserving important network features. We demonstrate *SNNVis* to analyze and interpret the evolutionary process and get insights into improving evolutionary SNNs. While we focused on analyzing evolving SNNs in terms of specific network features, we realized analyzing spiking signal activities would help to optimize SNNs. Even though visualizing the behind continuous activities will be a challenging problem, for future work, we will investigate techniques to understand and interpret the spiking activities of SNNs.

REFERENCES

[1] C. D. Schuman, J. S. Plank, A. Disney, and J. Reynolds, “An evolutionary optimization framework for neural networks and neuromorphic architectures,” in *International Joint Conference on Neural Networks*. Vancouver: IEEE, July 2016.

[2] C. D. Schuman, J. P. Mitchell, R. M. Patton, T. E. Potok, and J. S. Plank, “Evolutionary optimization for neuromorphic systems,” in *Proceedings of the Neuro-inspired Computational Elements Workshop*, 2020, pp. 1–9.

[3] M. Drouhard, C. D. Schuman, J. D. Birdwell, and M. E. Dean, “Visual analytics for neuroscience-inspired dynamic architectures,” in *2014 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, 2014, pp. 106–113.

[4] A. Kasiński, J. Pawłowski, and F. Ponulak, “‘snn3dviewer’ - 3d visualization tool for spiking neural network analysis,” in *Computer Vision and Graphics*, L. Bolc, J. L. Kulikowski, and K. Wojciechowski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 469–476.

[5] A. Disney, J. Reynolds, C. D. Schuman, A. Klibisz, A. Young, and J. S. Plank, "Danna: A neuromorphic software ecosystem," *Biologically Inspired Cognitive Architectures*, vol. 17, pp. 49–56, 2016.

[6] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[7] E. Alpaydin and C. Kaynak, "Optical Recognition of Handwritten Digits," UCI Machine Learning Repository, 1998, DOI: https://doi.org/10.24432/C50P49.

[8] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[9] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[10] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.