Leveraging the Graph Structure of Neural Network Training Dynamics

Fatemeh Vahedian
University of Michigan
vfatemeh@umich.edu

Ruiyu Li
University of Michigan
ruiyuli@umich.edu

Puja Trivedi
University of Michigan
pujat@umich.edu

Di Jin
University of Michigan
dijin@umich.edu

Danai Koutra
University of Michigan
dkoutra@umich.edu

ABSTRACT

Understanding the training dynamics of deep neural networks (DNNs) is important as it can lead to improved training efficiency and task performance. Recent works have demonstrated that representing the wirings of neurons in feedforward DNNs as graphs is an effective strategy for understanding how architectural choices can affect performance. However, these approaches fail to model training dynamics since a single, *static* graph cannot capture how DNNs change over the course of training. Thus, in this work, we propose a compact, expressive *temporal* graph framework that effectively captures the dynamics of many workhorse architectures in computer vision. Specifically, it extracts an informative summary of graph properties (e.g., eigenvector centrality) over a sequence of DNN graphs obtained during training. We demonstrate that our framework captures useful dynamics by accurately predicting trained, task performance when using a summary over early training epochs (<5) across four different architectures and two image datasets. Moreover, by using a novel, highly-scalable DNN graph representation, we also show that the proposed framework captures generalizable dynamics as summaries extracted from smaller-width networks are effective when evaluated on larger widths.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

deep learning, neural network training, dynamic graph mining

ACM Reference Format:

Fatemeh Vahedian, Ruiyu Li, Puja Trivedi, Di Jin, and Danai Koutra. 2022. Leveraging the Graph Structure of Neural Network Training Dynamics. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557628

1 INTRODUCTION

The impressive success of deep neural networks (DNNs) in machine learning tasks across a variety of domains [2, 6, 11, 19, 28] has led to considerable interest in understanding how the interplay between the training process, model architecture, hyper-parameters and other factors influences task performance [7, 9, 12, 23]. Key to this endeavor is a representation or framework for studying DNNs that is amenable to jointly analyzing these factors [13]. Recognizing the intuitive relationship between the wiring of neurons in DNNs and graphs, recent works [8, 24, 31] have proposed *graph* representations of DNNs and sought to leverage network science to understand how properties of the resulting representation are related to performance or behavior. Such graph representations are intuitive, provide a unified language (e.g., network science) for discussing properties of different architectures and have been shown to be effective at understanding static properties of DNNs.

However, existing graph representations are unable to capture how DNNs change throughout training and, therefore, cannot be effectively used to understand DNN training dynamics. Indeed, capturing dynamics is difficult as DNN graphs must not only be generated over time but also be expressive enough to meaningfully model how DNN parameters evolve. Existing graph representations are either unable to effectively scale for temporal settings due to untenable memory requirements [24] or are not expressive enough for a dynamic setting [31]. Therefore, we introduce a new graph representation and a corresponding framework that is able to model the training dynamics of many workhorse architectures in computer vision, while also preserving the benefits of a graph representation.

Present Work. In this paper, we propose a compact, graph representation and corresponding temporal framework for better understanding DNN training dynamics. Specifically, we first construct a series of graphs over the course of training, and then create informative summaries of graph properties (e.g., weighted degree, eigenvector centrality) to understand how the training induces structural changes in the DNN graph. To demonstrate the utility of the proposed framework, we use the temporal summaries extracted from a few early training epochs (<5) as effective features in the challenging task of predicting the final performance of fully-trained DNNs. Notably, reliably predicting performance is practically useful for early stopping [32]. Our main contributions are:

- **Compact, expressive graph representation of DNN:** We introduce a graph representation for convolutional layers that is significantly more compact than existing graph representations [24], enabling use in the analysis of the training dynamics.
- **Graph framework for NN performance prediction:** We propose a temporal framework that summarizes sequences of DNN graphs to effectively model structural changes during training.
- **Extensive empirical analysis:** Across several DNN architectures (AlexNet, VGG, LeNet, ResNet) and two image datasets, we verify the utility our framework in the challenging task of final performance prediction from early training epochs. Indeed, with temporal summaries over less than 5 epochs of training, our framework achieves classification accuracy of 90%. We also show that it captures generalizable dynamics by extracting summaries from smaller-width networks and predicting on large widths.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557628

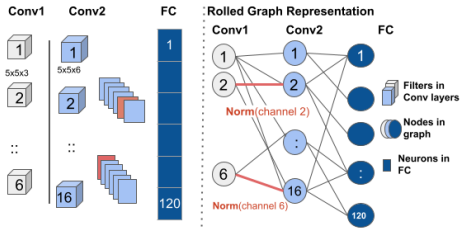


Figure 1: Rolled graph representation example: The resultant graph is a tri-partite graph with three node types (Conv1, Conv2, FC).

2 PROPOSED NN GRAPH REPRESENTATIONS

We now introduce our proposed compact representation of convolutional layers that can effectively and efficiently capture training dynamics. We begin by discussing existing graph representations.

DNNs as Graphs: Related Work & Limitations. Existing works primarily focus on representing DNNs which contain only fully connected (fc) and convolutional (conv) layers as these are the key components of many popular vision architectures (LeNet [17], VGG [27] and ResNet [9]). You et al. [31] propose representing DNNs as relational graphs where edges correspond to message passing between layers (nodes), and show that the graph clustering coefficient and average path length can identify a "sweet spot" for task performance. Rieck et al. [24] leverage weighted, stratified DNN graphs, where neurons correspond to nodes, and weights correspond to edges. They introduce a corresponding complexity measure which is empirically well-aligned with best training practices. Filan et al. [8] focus on weighted graph representation for MLPs only, where each neuron—including the input/output layers—corresponds to a node, and two neurons are linked if they appear in consecutive layers. They find that DNNs are "surprisingly modular."

While these representations provide interesting insights into DNNs, they are not suitable for understanding dynamics. Unweighted graphs cannot represent how convolutional filters or neurons change over training. Weighted edges in graph representations of fc layers [8, 24] suggest a mechanism for capturing dynamics. However, Rieck et al. "unroll" the convolution operator so each position in operation maps to a node in the graph and each multiplication maps to a weighted edge. Besides destroying the semantics of filters, unrolling leads to an explosion of nodes/edges and untenable memory requirements, as understanding dynamics requires creating DNN graphs over some epochs (time steps). Motivated by the inefficiencies and promise of the **unrolled graph representation** [24], we propose a compact, "rolled" representation for conv layers (Fig. 1), which is not only more interpretable than the unrolled model, but also highly effective despite its low footprint (§ 4).

Rolled Graph Representation of Conv Layers. Let tensor, $\mathcal{K}_i$, be a kernel in layer i with f_i filters, each with c_i channels and dimensions $h_i \times w_i$. We use bracket notation to index into the kernel: for example, $\mathcal{K}_i[l, :, :, :]$ indexes the l^{th} filter of kernel $\mathcal{K}_i$. We create f_i nodes representing each filter $\{v_1^{(i)}, v_2^{(i)}, ...v_{f_i}^{(i)}\}$, where node features are defined as the corresponding biases in that layer. Then, $\mathcal{K}_j$ is the next convolutional layer, defined analogously. While edges between neurons in fc layers are directly defined through neuron weights, each kernel contains multiple weights. To extract a weighted edge, we take the norm over each kernel's channels.

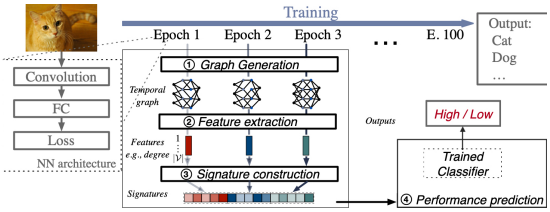


Figure 2: Our proposed framework for predicting NN performance in a downstream image classification task, shown for one test instance.

Formally, the edge between node $v_k^{(i)}$ representing the k^{th} filter in layer i (i.e., $\mathcal{K}_i[k, :, :, :]$) and node/filter $v_l^{(j)}$ in layer $j = i + 1$ (i.e. $\mathcal{K}_j[l, :, :, :]$) has weight $w_{v_k^{(i)}, v_l^{(j)}} = \text{norm}(\mathcal{K}_j[l, k, :, :])$, which is the norm of the k^{th} channel of the l^{th}filter in the j^{th} layer. While alternative edge weight configurations can be supported, we focus on the filter norm as other studies, including those on pruning NNs [18], have demonstrated that this value has strong correlation with filter importance. As shown in Fig. 1, in the case of two conv layers (i.e., w/o the fc layer), the resultant graph is an attributed bipartite graph with $f_i + f_j$ nodes and $f_i \times f_j$ edges, where node attributes include flattened weight vectors or filter maps. Other information such as average gradients can also be used as node features.

3 PROPOSED TEMPORAL FRAMEWORK

Successful performance prediction based on only a few epochs could be used for early stopping [32], and thus, more efficient NN training. We show the utility of our proposed graph representation, by investigating the relationship between the temporal graph structure of NNs in early training stages and NN performance in downstream tasks. Formally:

Task 1 (NN Performance Prediction). *Let $\mathcal{N} = \{N_{tr_1}, \ldots, N_{tr_n}\}$ be a training set of n NNs trained for T epochs and $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ their corresponding downstream task accuracies (e.g., for image classification). We seek to predict the accuracy α_{tst} of a new instance N_{tst} trained for a very small number of $t \ll T$ epochs by using t epochs for the trained NNs in $\mathcal{N}$.*

Building on temporal graph mining and summarization [3, 10, 20, 25], to tackle this problem, we introduce a temporal graph-based framework (Fig. 2), which consists of four steps:

Step (S1) Graph generation. This step involves converting the training process of each input NN $N_{tr_i} \in \mathcal{N}$ into a time-evolving graph where each static snapshot corresponds to a different epoch (time step). The output is a set of n temporal graphs $\{\mathcal{G}_{tr_1}, ..., \mathcal{G}_{tr_n}\}$, where $\mathcal{G}_{tr_i} = \{\text{G}_i^1, ..., \text{G}_i^t\}$ corresponds to the i^{th} original NN in $\mathcal{N}$.

Step (S2) Feature extraction. Next, the goal is to capture the structural dynamics of the NN training process. We aim to select graph measures that can capture changes during the training process, take into account the edge weight of graphs and can be calculated efficiently. In order to do that in an interpretable way, we extract two well-known node centralities from each snapshot of each generated time-evolving graph $\mathcal{G}_i$: weighted degree centrality and eigenvector centrality [5]. The weighted degree is a simple function of the learnable weight matrix $\mathbf{W}$ during the training phase of NN, therefore it gives us insights into the training dynamics at the node/neuron/filter level. The eigenvector centrality is an extension of the degree centrality, which captures the highly influential nodes, and has been successfully used in neuroscience to capture

the dynamic changes of real neural networks (or connectomes) [21]. Eigenvector centrality can be used to capture importance and connectivity of filters/neurons (i.e., the nodes in our graph representation). It has also been used for detecting communities [22] or clusters [30], and thus provides structural information about the clusterability of the NN, which is complementary to that provided by the simpler and more efficient-to-compute degree centrality. Our choice of features is guided by the inherent k-partite structure of our proposed graph representation, which cannot be captured well by other commonly-used graph features (e.g., clustering-based features like triangles, transitivity, clustering coefficient are 0).

Step (S3) Graph signature construction. In order to be able to compare DNN graphs (with different number of nodes and edges) [14, 15], we summarize the structural changes in the generated temporal graphs at the *graph* level (rather than the *node* level, as in (S2)), and construct a statistical summary of the extracted node centralities (signature) per time-evolving graph $\mathcal{G}_i$. For each snapshot $\mathbf{G}_i^{(\tau)}$ of $\mathcal{G}_i$, we create a signature vector using five node feature aggregators, which were introduced in [4] for graph similarity: *median, mean, standard deviation, skewness, and kurtosis*, where all but the median are moments of the corresponding distribution. Thus, $\mathbf{G}_i^{(\tau)}$ is mapped to a (static) signature vector $s_i^{(\tau)} \in \mathbb{R}^5$, representing the statistical summary of its node features (i.e., degree or eigenvector centrality) at time τ. To put more emphasis on the most recent timesteps, we can redefine the signature at time τ as the linear weighted average of the signatures up to that point, $\mathbf{s}_i^{(\tau)} \leftarrow \frac{\sum_{j=1}^{\tau} j * \mathbf{s}^{(j)}}{\sum j}$, or an exponential function of the previous signatures, $\mathbf{s}_i^{(\tau)} \leftarrow \alpha \mathbf{s}_i^{(\tau)} + (1-\alpha)\mathbf{s}_i^{(\tau-1)}$. To obtain the temporal signature of the evolving graph $\mathcal{G}_i$, we aggregate the (static) signatures up to timestamp/epoch t, $\mathbf{s}_i^t = \mathbf{s}_i^1 \oplus ... \oplus \mathbf{s}_i^t$, where $\oplus$ denotes concatenation.

Step (S4) Performance prediction. For the performance prediction step, we consider a classification task: We train a classifier (e.g., SVM, MLP) using the training graphs $\{\mathcal{G}_{tr_1}, \mathcal{G}_{tr_2}, ..., \mathcal{G}_{tr_n}\}$ represented by their temporal signatures $\{\mathbf{s}_{tr_1}^t, \mathbf{s}_{tr_2}^t, ..., \mathbf{s}_{tr_n}^t\}$, and their corresponding accuracies $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ mapped to labels $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ (e.g., high/low accuracy) based on some threshold. Test NN instances are then classified by the trained model.

4 EMPIRICAL ANALYSIS

In this section, we empirically evaluate our framework in the NN performance prediction task. We seek to answer three questions: **(Q1)** How effective is our framework when predicting the performance per DNN architecture? **(Q2)** Can our framework generalize to unseen architectures? **(Q3)** How efficient is it?

Data. We investigate NN dynamics using two well-known image classification datasets, CIFAR-10 [16] and ImageNet [26]. CIFAR-10 consists of 50K training images and 10K test images. For ImageNet, we use a sample that has 50K training images and 5K validation images used as the test set [1].

Baselines. We compare our framework with variants of a graph-agnostic approach that was proposed in [29] to predict the accuracy of NNs directly from the learned weights. We consider three baseline methods using different feature vectors as input to our step (**S4**): **(1)** B_{W_l} leverages the flattened parameters (weights/kernels and biases) of the last layer, since, according to [29], the parameters learned in the last dense layer are as informative as those from all the layers for predicting NN performance[1]; **(2)** $B_{\hat{W}}$ applies 7 aggregators to the flattened vector of the last layer: the mean, the variance, and q^{th} percentiles for $q \in \{0, 25, 50, 75, 100\}$; **(3)** $B_{W'}$ applies our 5 aggregators (from step (**S3**)) on the flattened vector of the last layer: median, mean, standard deviation, skewness, and kurtosis. We also compare our proposed rolled graph representation to **(4)** the previously proposed unrolled representation [24].

Table 1: Description of NNs: early stopping epoch range, accuracy, and accuracy threshold defining the classes of the classification task.

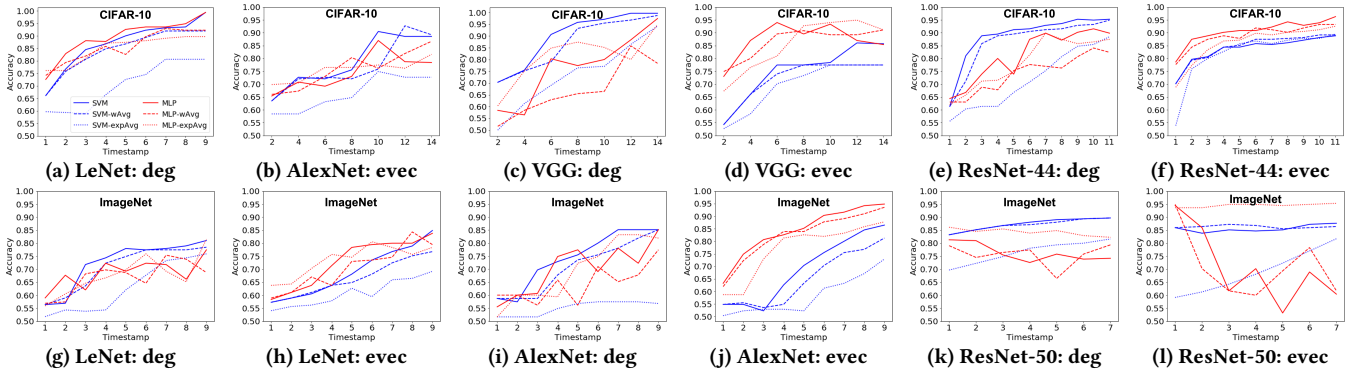
	CIFAR-10					ImageNet		
	LeNet	AlexNet	VGG	ResNet-32	ResNet-44	LeNet	AlexNet	ResNet-50
Early stop.	11~50	30~50	45~50	16~120	16~120	16~50	16~50	16~120
Acc. range	9.4~73.8	5.5~82.4	8.8~87.6	8.4~90.0	9.9~89.8	0.6~14.4	0.6~20.1	0.86~41.66
Acc. thres.	40	40	40	40	40	9	10	25

(Q1) NN Performance Prediction - Per Architecture. *Task setup.* We cast the NN performance prediction as a classification task where the generated temporal graphs are labeled as high and low accuracy based on the performance of their corresponding NNs. Table 1 lists the threshold value chosen for low and high accuracy labels based on the final accuracy range of trained NNs, as well as the early stopping epochs for each architecture. Five-fold cross validation is used to predict the label of the test graphs using SVM and MLP, where the input is the set of temporal signatures $\{\mathbf{s}_{tr_1}^t, \mathbf{s}_{tr_2}^t, ..., \mathbf{s}_{tr_n}^t\}$. In Fig. 3, per classifier we use the three signatures described in Step (S3), denoted as: no suffix, -wAvg, -expAvg.

Results. We present the results for both types of signatures for the temporal graphs corresponding to the training dynamics of different architectures on CIFAR-10 and ImageNet in Fig. 3 (top and bottom, resp). In all the cases, classification accuracy of 80-95% is achieved in less than 10 training epochs. For degree-based signatures, SVM tends to outperform MLP, while the trend is reversed for eigenvector-based signatures. For example, for both VGG and AlexNet for the CIFAR-10 image classification task, MLP can predict the performance with accuracy ~95% using the eigenvector-based signatures from the first 6 training epochs; the same trend is observed on ImageNet for the LeNet and AlexNet architectures. In all the cases, both classifiers reach performance over 80%-90% significantly before the early stopping point for all the architectures.

Comparison to baselines. Table 2 summarizes the classification accuracy of the last timestamp t of our proposed framework (e.g., $t = 9$ for LeNet) and the baselines. For all the architectures except for VGG, both signature types of our graph-aware framework achieve significantly higher accuracy than the three graph-agnostic baselines. For VGG (CIFAR-10), our degree-based graph signature outperforms the baselines, while the eigenvector-based signature achieves comparable performance to $B_{W'}$, which replaces our graph features with the flattened weights in the last NN layer. The consistently high accuracy of our framework in this task compared to the graph-agnostic baselines illustrates the utility of our graph representation and feature extraction, and the insufficiency of leveraging directly the learned weights. Finally, replacing our efficient rolled graph representation with the baseline unrolled representation leads to significantly worse performance and longer runtime,

[1] This observation was consistent with our experiments: the entire flattened vector across layers did not lead to higher prediction accuracy.



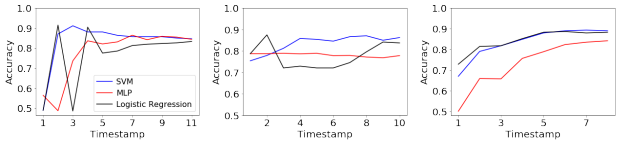
(a) LeNet: deg (b) AlexNet: evec (c) VGG: deg (d) VGG: evec (e) ResNet-44: deg (f) ResNet-44: evec

(g) LeNet: deg (h) LeNet: evec (i) AlexNet: deg (j) AlexNet: evec (k) ResNet-50: deg (l) ResNet-50: evec

Figure 3: NN performance classification on CIFAR-10 (a-f) and ImageNet (g-l): 'deg' for degree-based and 'evec' for eigenvector centrality-based temporal signature. For ImageNet, eigenvector-based signatures yield higher performance compared to the degree-based ones.

Table 2: Comparison of classification accuracy between baselines and our graph-based framework. OOM: Out Of Memory.

		CIFAR-10				ImageNet	
		LeNet	AlexNet	VGG	ResNet-44	AlexNet	ResNet-50
SVM	B_{W_l}	0.63	0.61	0.65	0.48	0.64	0.53
	$B_{\hat{W}}$	0.8	0.74	0.51	0.74	0.66	0.56
	$B_{W'}$	0.75	0.82	0.88	0.72	0.67	0.6
	unroll-deg	0.92	0.49	OOM	OOM	OOM	OOM
	Ours-deg	**0.99**	0.87	**0.97**	**0.95**	0.85	**0.88**
	Ours-evec	0.81	**0.94**	0.86	0.88	**0.86**	**0.88**
MLP	B_{W_l}	0.68	0.83	0.67	0.47	0.64	0.68
	$B_{\hat{W}}$	0.55	0.57	0.58	0.61	0.62	0.55
	$B_{W'}$	0.58	0.75	0.52	0.66	0.67	0.58
	unroll-deg	0.93	0.89	OOM	OOM	OOM	OOM
	Ours-deg	**0.98**	0.91	**0.96**	0.88	0.85	**0.74**
	Ours-evec	0.91	**0.96**	0.85	**0.96**	**0.94**	**0.70**



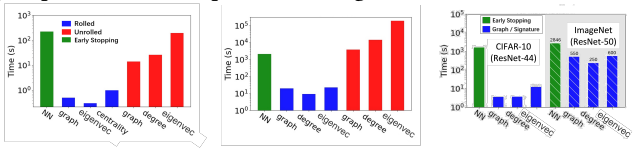
(a) CIFAR, ResNet44 (b) ImgNet, ResNet50 (c) CIFAR, ResNet32

Figure 4: NN classification to generalize degree signatures to unseen architectures. (a) Train: ResNet-32, test:ResNet-44. (b) Train: ResNet-34, test: ResNet-50. (c) Train: LeNet, AlexNet, VGG, test: ReNet-32.

which justifies the utility of our representation. Due to extensive memory and space requirements of the unrolled representation, we report results only for LeNet (t=9) and AlexNet (t=14) on CIFAR-10.

(Q2) NN Performance Prediction - Generalizing to Unseen Architectures. *Task setup.* To show that our proposed graph representation and signatures are general, we fully train a small set of different NN architectures (and hyperparameters), and predict the performance on unseen NN architectures. Two sets of experiments were considered: **(1)** Train our classifier on the smaller ResNet architectures (ResNet-32 for CIFAR-10, and ResNet-34 for ImageNet), and test the performance of the larger ResNet architectures (ResNet-44 for CIFAR-10, and ResNet-50 for ImageNet); **(2)** Train our classifier on the combination of old architectures (VGG, AlexNet and LeNet) to predict the accuracy of a newer architecture (ResNet).

Results. In Figs. 4a and 4b (exp. 1), we observe that our proposed framework is able to accurately predict the performance level of the previously unseen, large ResNet architectures. Our results show that the proposed temporal signatures can be used in a generalized scenario to predict the accuracy level of the same architecture with different numbers of layers (on the same dataset). This generalization from small to bigger architectures is important since it is faster to train the smaller architectures. In Fig. 4c (exp. 2), we see that our proposed method also successfully predicts the performance level of a new architecture (i.e. ResNet) when the training set is a combination of older architectures (LeNet, VGG, AlexNet).



(a) LeNet (b) VGG (c) ResNet

Figure 5: Avg runtime for the NN training, the graph generation (S1) and feature extraction (S2) on: (a) CIFAR-10 with LeNet; (b) CIFAR-10 with VGG; and (c) CIFAR-10 (ResNet-44) and ImageNet (ResNet-50).

(Q3) Time efficiency and early stopping. Figure 5 depicts the total average runtime of NN training for early stopping of each architecture (green bars) in comparison to the average runtime of graph generation, degree calculation and eigenvector centrality calculation for the number of epochs needed for that architecture to achieve the highest accuracy in classification task. For all the architectures, our proposed rolled graph representation (in blue) can achieve a high-accuracy prediction much faster than the early stopping approach of NN training. But for the baseline unrolled representation (in red), this is only true for LeNet. As the size of NN increases, the unrolled graph generation gets slower and the corresponding graph-based approach is slower than early stopping.

5 CONCLUSION

We investigated the early training dynamics of NNs from a time-evolving graph perspective. To the best of our knowledge, we are the first to model the NN training dynamics and structure as a temporal graph. We coupled this representation with a new, compact graph model for convolutional layers. Then, we showed that a simple, temporal graph signature based on summary statistics of the degree or eigenvector centrality distributions over *only a few epochs* can be used as a strong predictor variable to estimate the accuracy of NNs in downstream tasks. Exploring the role of our efficient proposed framework for early stopping is a promising future direction.

Acknowledgements: This material is based upon work supported by the NSF under Grant No. IIS 1845491, and Amazon and Facebook faculty awards.

REFERENCES

[1] Tiny imagenet. https://www.kaggle.com/c/tiny-imagenet/data, 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2015.

[3] Caleb Belth, Xinyi Zheng, and Danai Koutra. Mining persistent activity in continually evolving networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, pp. 934–944, 2020.

[4] Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Network similarity via multiple social theories. In *Proc. Advances in Social Networks Analysis and Mining 2013, ASONAM '13*, 2013.

[5] Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *J math soc*, 2(1):113–120, 1972.

[6] Yue Cao, Thomas Andrew Geddes, Jean Yee Hwa Yang, and Pengyi Yang. Ensemble deep learning in bioinformatics. *Nat Mach Intell*, 2(9):500–508, 2020.

[7] Supriyo Chakraborty and et al. Interpretability of deep learning models: A survey of results. In *SmartWorld*, pp. 1–6, 2017. doi: 10.1109/UIC-ATC.2017.8397411.

[8] Daniel Filan, Stephen Casper, Shlomi Hod, Cody Wild, Andrew Critch, and Stuart Russell. Clusterability in neural networks. *arXiv preprint arXiv:2103.03386*, 2021.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B: Condensed Matter and Complex Systems*, 88(9):1–30, 2015.

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2015.

[13] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2018.

[14] Danai Koutra and Christos Faloutsos. *Individual and Collective Graph Mining: Principles, Algorithms and Applications*. Synthesis Lectures on Data Mining and Knowledge Discovery, Morgan Claypool, 2017.

[15] Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Node and graph similarity: Theory and applications. In *IEEE International Conference on Data Mining (ICDM) tutorial*, 2014.

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[18] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2017.

[19] Yu Li, Chao Huang, Lizhong Ding, Zhongxiao Li, Yijie Pan, and Xin Gao. Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. *Methods*, 166:4–21, 2019.

[20] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *ACM Comput. Surv.*, 51(3), 2018.

[21] Gabriele Lohmann, Daniel S Margulies, Annette Horstmann, Burkhard Pleger, Joeran Lepsien, Dirk Goldhahn, Haiko Schloegl, Michael Stumvoll, Arno Villringer, and Robert Turner. Eigenvector centrality mapping for analyzing connectivity patterns in fmri data of the human brain. *PloS one*, 5(4):e10232, 2010.

[22] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[23] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2017.

[24] Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2019.

[25] Polina Rozenshtein and Aristides Gionis. Mining temporal networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, pp. 3225–3226. ACM, 2019. URL https://rozensp.github.io/KDD19-tutorial-temporal/.

[26] Olga Russakovsky, Jia Deng, and et al. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2015.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2014.

[29] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020.

[30] Qin Wu, Xingqin Qi, Eddie Fuller, and Cun-Quan Zhang. "follow the leader": A centrality guided clustering and its application to social network analysis. *The Scientific World Journal*, 2013, 2013.

[31] Jiaxuan You, Jure Leskovec, Kaiming He, and Saining Xie. Graph structure of neural networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2020.

[32] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.