You Only Live Twice - Investigating the Effects of Dying Neurons on Over-smoothing.

1089598

Abstract

Dying neurons are a well-known phenomenon that occurs when using ReLU activation functions. When a neuron is dead, it constantly outputs 0, is not trainable, and does not contribute meaningfully to the final output. Despite this drawback, ReLU activation functions remain a popular choice for many GNN architectures. We show that there is a direct link between over-smoothing in ReLU-based GCNs and dying neurons. We demonstrate this by showing that ReLU-based GCN models with fewer dying neurons suffer less from over-smoothing and generalize better. We prevent dying neurons in GCN layers by pruning the most negative weights connected to the dead neurons. We demonstrate the effectiveness of this method on the CORA dataset, achieving higher test accuracy for deep networks, fewer dead neurons, and reduced over-smoothing based on similarity measures. Our findings not only highlight an effective method to improve the performance of ReLU-based models but also show that avoiding dying neurons can mitigate over-smoothing, indicating a direct connection between these two phenomena.

1 Introduction

Graph Neural Networks (GNNs) are a popular tool for solving complex tasks on graphs. In recent years, there have been many advances in GNN architectures and their training. A key challenge that GNN architectures face is over-smoothing. Over-smoothing describes the phenomenon where the node representations within the graph's connected components converge to a constant value as the number of layers increases. This convergence has connections to Laplacian smoothing [Li et al., 2018].

The concept of dying neurons describes the event, closely tied to the ReLU activation function, in which a neuron only outputs 0 for any input [Lu Lu et al., 2020]. These neurons stop learning, and the weights connected to them remain fixed. Nevertheless, ReLU remains a simple and well-performing activation function, capable of outperforming other proposed solutions [Veličković et al., 2018].

In this research, we compare two approaches to preventing dying neurons and their effects on over-smoothing. First, we investigate possibilities for reviving these dead neurons to make them activate meaningfully again. We accomplish this by pruning the most negative weights leading to a neuron [Whitaker and Whitley, 2023]. Second, we compare this approach to LeakyReLU [Maas, 2013], a ReLU alternative with an αx slope in the negative domain. This modification introduces non-linearity and allows gradients to flow for negative inputs, given $\alpha \neq 0$ and $\alpha \neq 1$.

We evaluate the performance of both techniques on GCN models trained on the CORA dataset. Our results show that pruning is an effective technique, improving the generalization of GCNs as the number of layers increases. Remarkably, it outperforms LeakyReLU with $\alpha = 0.8$ on the deepest

model we trained (30 layers). Additionally, we demonstrate that pruning reduces the number of dead neurons across all model sizes and lowers the overall proportion of neurons outputting 0. Furthermore, pruning slows down the over-smoothing process in a 128-layer network compared to ReLU without pruning.

These results indicate that the same network can achieve higher performance and experience less over-smoothing simply by reviving dead neurons. This suggests that dead neurons in ReLU-based GCNs play a role in their over-smoothing behavior.

2 Related Work

Dying Neurons in Neural Networks There are many methods of preventing dying neurons in the deep learning literature. Some methods apply ReLU alternatives that are nonzero in the negative domain [Maas, 2013, Clevert et al., 2016, Hendrycks and Gimpel, 2023, Ramachandran et al., 2017]. Other approaches try to avoid dying neurons through weight initialization [He et al., 2015b] or by using regularization techniques, such as batch normalization [Ioffe and Szegedy, 2015] or dropout.

Recently, some approaches have tried to revive these neurons. Qiao et al. [2018] combines dying neurons with neural architecture search for Convolutional Neural Networks. They remove filters with many dying neurons and extend filters with fewer dying neurons to distribute their given compute budget. Whitaker and Whitley [2023] proposed reviving dying neurons in fully connected layers. They prune the most negative weights within a layer to allow the ReLU input to become positive again, hoping to revive the neuron.

A distinction of our work is that we utilize these pruning techniques to revive dying neurons in GCN layers. To our knowledge, we are the first to investigate the effect of dying neurons on over-smoothing in GCNs. Our approach is different from common GNN pruning, which either prunes the graph itself [Hossain et al., 2024, Jamadandi et al., 2024] or attempts to prune the weights [Gurevin et al., 2024] to reduce compute based on the lottery ticket hypothesis [Chen et al., 2021].

Over-Smoothing for GNNs There are a variety of attempts to prevent over-smoothing in GNNs in the literature. Previous methods include skip connections [He et al., 2015a, Li et al., 2019], which add the layer input to its output, or jumping knowledge [Xu et al., 2018], which passes on information of previous layers. Other approaches avoid using deep networks and rely on shallow networks [Xin et al., 2020].

Regularization methods like DropEdge [Rong et al., 2020] or DropGNN [Papp et al., 2021] are also common techniques to prevent models from early over-smoothing, as well as normalization techniques like pair-norm, which normalizes the output of each layer [Zhao and Akoglu, 2020].

Recent work by Wu et al. [2024] conducted experiments on over-smoothing behavior. They compared different activation functions and found that ReLU-based GCNs start over-smoothing faster than networks with other activation functions. Further work by Kelesis et al. [2023] compared the over-smoothing behavior of GCNs and GAT models when using alternative activation functions to ReLU. Compared to these two approaches, we specifically investigate the relationship between dying neurons in ReLU and over-smoothing. Moreover, we try to verify this by using a method based on weight pruning to revive the dead neurons.

3 Method

Dying Neurons The phenomenon of dying neurons in neural networks occurs when the input for the ReLU activation function is negative, and therefore the ReLU output is 0. This is specific to the ReLU activation function because many alternative activation functions, like sigmoid or tanh, are nonzero in the negative domain. The ReLU activation function is defined as $\text{ReLU}(x) = \max(0, x)$. This makes the gradient of ReLU 0 if $x \leq 0$, preventing the gradient flow through neurons that have a 0 activation. This becomes a problem when previous gradient updates push x into the negative domain. A dead neuron has a 0 output for any given input. This stops all gradients and leads to this neuron never being activated again because the weights are no longer updated. Therefore, a dead neuron will remain dead. Furthermore, this results in a reduced number of active parameters in the network. This could potentially limit the model's expressive power.

Detecting Dying Neurons Our approach works with dying neurons in GCN networks. One layer of our GCN is as follows:

$$X' = \text{ReLU}(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}X\Theta) \tag{1}$$

Assuming $X' \in \mathbb{R}^{N \times d}$, where N is the number of nodes in the graph, a dead neuron is a column in $f(X')$ with only zeros. This means that the same feature is 0 for every node.

Synaptic Stripping The synaptic stripping approach of Whitaker and Whitley [2023] proposes reviving dead neurons by pruning the most negative weights, thereby setting them to 0. Given that the output of the previous layer is nonnegative due to the ReLU activation, only negative weights contribute to a negative ReLU input, thereby leading to dead neurons. We prune 10% of the most negative weights, rounding up to the nearest integer.

Measuring Over-smoothing To evaluate our results, we use two methods. Firstly, we train models with different depths and compare their accuracy on the test set. We hypothesize that less over-smoothing should lead to better performance when scaling up. Secondly, we utilize the following metric proposed by Wu et al. [2024]:

$$\mu_{\text{DTM}}(X) = \|X - \frac{\mathbf{1}^T X}{N}\|_F, \tag{2}$$

where N is the number of nodes, $X \in \mathbb{R}^{N \times d}$ is the node embedding matrix, and $\mathbf{1} \in \mathbb{R}^N$ is the all-ones vector. We call this metric Distance To Mean (DTM). We also utilize the Dirichlet energy for graphs [Cai and Wang, 2020, Rusch et al., 2023]:

$$\mu_{\text{Dirichlet}}(X) = \sqrt{\frac{1}{N}\sum_{i \in \mathcal{V}}\sum_{j \in \mathcal{N}_i}\|X_i - X_j\|_2^2}, \tag{3}$$

where $\mathcal{N}_i$ is the set of all nodes connected to node i. For more information on over-smoothing, please refer to Rusch et al. [2023].

Preventing Dying Neurons for GNNs The relevance of dying neurons for GNN over-smoothing can be motivated in two ways. As already mentioned, the expressiveness of GNNs can be limited by dying neurons. Therefore, alleviating the problem of dying neurons could lead to more expressive models, which, in turn, can have better generalization as well as less over-smoothing [Jin and Zhu, 2024]. Furthermore, previous work on the relationship between activation functions and over-smoothing [Kelesis et al., 2023] shows that the probability of the ReLU output being 0 can be used to bound the distance of node embeddings to the subspace where over-smoothing is prevalent. This bound can be derived using singular values of the weight matrices. While the approach of synaptic

stripping doesn't provide a theoretical guarantee of increasing this probability, actively preventing dying neurons, which always output zero, can help decrease this probability in practice. Therefore, reviving dead neurons can increase the upper bound, resulting in less over-smoothing. For more details, refer to Kelesis et al. [2023].

In our approach, we train a GCN with the ReLU activation function. After every epoch, we detect dead neurons on the validation set, which can act as an extra regularizer compared to detecting them on the training set. We then prune 10% of the negative weights and reinitialize them to 0. We prune weights in the weight matrix Θ of every GCN layer (eq. 1). We chose a GCN since GCNs are a simple architecture with many theoretical and experimental results on over-smoothing [Wu et al., 2024, Kelesis et al., 2023, Lu Lu et al., 2020].

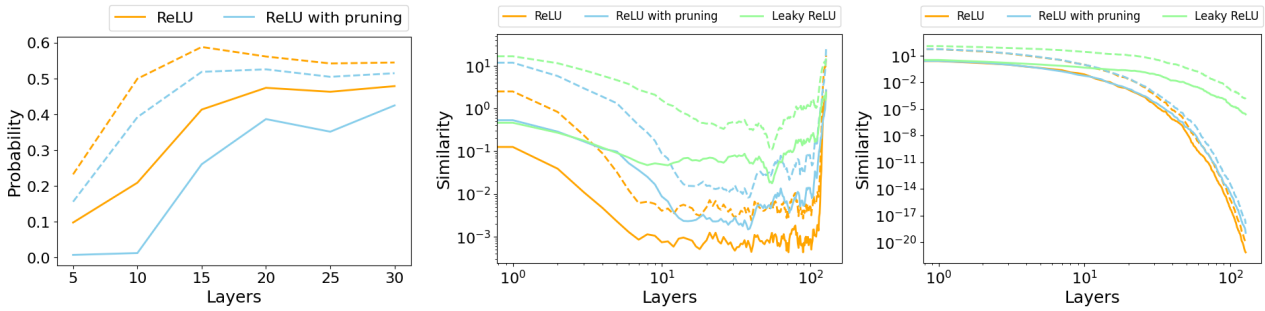
4 Results

Table 1: Mean and standard deviation of test accuracies of a GCN with ReLU, ReLU with pruning, and LeakyReLU $\alpha = 0.8$. We trained each model 10 times over 100 epochs with hidden dimension of 32.

Layer	ReLU		ReLU with Pruning		LeakyReLU $\alpha = 0.8$	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
0	0.4688	0.0042	<u>0.4696</u>	0.0037	**0.4707**	0.0096
5	<u>0.7615</u>	0.0131	0.7587	0.0142	**0.7649**	0.0107
10	0.6447	0.0957	<u>0.6656</u>	0.0594	**0.7564**	0.0162
15	<u>0.2708</u>	0.1043	0.2471	0.0898	**0.7322**	0.0190
20	0.1867	0.0227	<u>0.2063</u>	0.0366	**0.5501**	0.2023
25	0.1879	0.0239	<u>0.2094</u>	0.0335	**0.2622**	0.1444
30	<u>0.1946</u>	0.0414	**0.2074**	0.0294	0.1703	0.0690

We trained a GCN model with the ReLU activation function, utilizing our proposed approach of pruning the weights to revive dead neurons. Furthermore, we trained a GCN with the LeakyReLU activation function, choosing $\alpha = 0.8$ based on the results of Wu et al. [2024]. They showed that LeakyReLU with this parameter choice performs exceptionally well compared to smaller values for α. We trained the models on the CORA dataset for node classification, a citation network of 2,708 machine learning papers [McCallum et al., 2000]. Table 1 shows that the GCN utilizing LeakyReLU outperforms the ReLU-based GCNs and maintains good performance even for depths of 15 and 20 layers, before being outperformed by the pruning method at 30 layers. Furthermore, ReLU with pruning outperforms ReLU for 20 and 25 layers and achieves the highest performance overall at 30 layers. These findings indicate that pruning can increase the performance of GCNs for deep networks. Furthermore, we can see that some of ReLU's tendency to over-smooth could be attributed to the phenomenon of dying neurons. We also notice a turning point where accuracy begins to drop for all models: at 10 to 15 for ReLU-based models and at 20 to 25 for LeakyReLU-based models. This turning point is accompanied by high variance, indicating that some models are still able to learn properly, while others fail. Lower variance around the turning point for pruning-based methods shows higher levels of robustness due to pruning, while the offset in LeakyReLU's turning point demonstrates that it can effectively prevent early over-smoothing.

In Figure 1a, we can see that the pruning technique consistently results in fewer dying neurons than



(a) Comparison of dead neurons (solid) and 0 ReLU output (dotted).

(b) Trained models. Dirichlet energy (solid) and DTM (dotted).

(c) Random initialized models. Dirichlet energy (solid) and DTM (dotted).

Figure 1: (a) Probability of a neuron being completely dead (0 output for all nodes) and probability of a neuron having 0 output after ReLU. We compare the similarity of node embeddings across layers measured in Dirichlet energy (solid) and DTM (dotted). We compare trained models (b) and randomly initialized models (c).

the network without pruning. This is especially visible for networks with up to 10 layers, where the amount of dead neurons is particularly low—1% compared with 20.5%. Figure 1a shows that pruning can effectively reduce the number of 0 outputs for ReLU, empirically demonstrating that this approach can decrease the probability of ReLU outputting 0, which impacts the over-smoothing properties of the network [Kelesis et al., 2023].

In our second experiment, we trained each model with 128 layers and hidden dim of 32 for 100 epochs, repeating the experiment 3 times. Figure 1b shows the Dirichlet energy for graphs, as well as DTM across the model's layers. The results show that our method is able to consistently maintain a higher dissimilarity across layers, indicating that pruning helps alleviate over-smoothing. The plotted values start rising towards the end because we trained the networks. Plots for randomly initialized models (Fig 1c) highlight that the LeakyReLU architecture seems to suffer less from over-smoothing, likely because this approach does not cut off all negative values. Furthermore, we can show that training GCNs forces them to increase dissimilarity towards the last layers. probably in order to generate the logits.

5 Conclusions

We have shown that pruning strategies similar to synaptic stripping can be applied to GCN architectures and reviving dead neurons in ReLU-based models helps to prevent over-smoothing. We demonstrated that pruning-based models can outperform ReLU and LeakyReLU-based models with 30 layers, and their internal representations are consistently further apart and converge slower than those of ReLU-based models. Our methodology can easily be implemented in current ReLU-based GCN models to prevent over-smoothing and improve performance. We acknowledge that our results are limited by low compute resources, and we propose further investigation of dying neurons and over-smoothing on 1) different datasets and 2) deeper networks. It would be interesting to see if ReLU with pruning can outperform other methods on even deeper architectures beyond 30 layers. Another shortcoming is the theoretical analysis of this approach, as pruning is based on a heuristic; finding mathematical guarantees seems to be more difficult and requires further investigation.

References

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks, 2020. URL https://arxiv.org/abs/2006.13318.

Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks, 2021. URL https://arxiv.org/abs/2102.06790.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016. URL https://arxiv.org/abs/1511.07289.

Deniz Gurevin, Mohsin Shan, Shaoyi Huang, MD Amit Hasan, Caiwen Ding, and Omer Khan. Prunegnn: Algorithm-architecture pruning framework for graph neural network acceleration. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 108–123, 2024. doi: 10.1109/HPCA57654.2024.00019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015a. URL https://arxiv.org/abs/1512.03385.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015b. URL https://arxiv.org/abs/1502.01852.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Tanvir Hossain, Khaled Mohammed Saifuddin, Muhammad Ifte Khairul Islam, Farhan Tanvir, and Esra Akbas. Tackling oversmoothing in gnn via graph sparsification: A truss-based approach, 2024. URL https://arxiv.org/abs/2407.11928.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL https://arxiv.org/abs/1502.03167.

Adarsh Jamadandi, Celia Rubio-Madrigal, and Rebekka Burkholz. Spectral graph pruning against over-squashing and over-smoothing, 2024. URL https://arxiv.org/abs/2404.04612.

Yufei Jin and Xingquan Zhu. Atnpa: A unified view of oversmoothing alleviation in graph neural networks, 2024. URL https://arxiv.org/abs/2405.01663.

Dimitrios Kelesis, Dimitrios Vogiatzis, Georgios Katsimpras, Dimitris Fotakis, and Georgios Paliouras. Reducing oversmoothing in graph neural networks by changing the activation function. In *ECAI 2023*, pages 1231–1238. IOS Press, 2023.

Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns?, 2019. URL https://arxiv.org/abs/1904.03751.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning, 2018. URL https://arxiv.org/abs/1801.07606.

Lu Lu Lu Lu, Yeonjong Shin Yeonjong Shin, Yanhui Su Yanhui Su, and George Em Karniadakis George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5):1671–1706, January 2020. ISSN 1815-2406. doi: 10.4208/cicp.oa-2020-0165. URL http://dx.doi.org/10.4208/cicp.OA-2020-0165.

Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013. URL https://api.semanticscholar.org/CorpusID:16489696.

Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000. URL https://api.semanticscholar.org/CorpusID:349242.

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks, 2021. URL https://arxiv.org/abs/2111.06283.

Siyuan Qiao, Zhe Lin, Jianming Zhang, and Alan Yuille. Neural rejuvenation: Improving deep network training by enhancing computational resource utilization, 2018. URL https://arxiv.org/abs/1812.00481.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Swish: a self-gated activation function. *arXiv: Neural and Evolutionary Computing*, 2017. URL https://api.semanticscholar.org/CorpusID:196158220.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification, 2020. URL https://arxiv.org/abs/1907.10903.

T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks, 2023. URL https://arxiv.org/abs/2303.10993.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

Tim Whitaker and Darrell Whitley. Synaptic stripping: How pruning can bring dead neurons back to life, 2023. URL https://arxiv.org/abs/2302.05818.

Xinyi Wu, Amir Ajorlou, Zihui Wu, and Ali Jadbabaie. Demystifying oversmoothing in attention-based graph neural networks, 2024. URL https://arxiv.org/abs/2305.16102.

Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. Graph highway networks, 2020. URL https://arxiv.org/abs/2004.04635.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks, 2018. URL https://arxiv.org/abs/1806.03536.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns, 2020. URL https://arxiv.org/abs/1909.12223.