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

No. 1089598 | GitHub: https://anonymous.4open.science/r/Dying_GNN_Neurons-1CD7

Abstract

Dying neurons are a well-known phenomenon within deep learning models that use the ReLU activation function. When a neuron is dead, it constantly outputs 0, making it neither trainable nor meaningful for the model's performance. Yet, ReLU is a popular activation function still used in many GNN architectures. We show that there could be a direct link between dying neurons and over-smoothing in GCN models. Our findings show that ReLU-based methods, with fewer dying neurons, suffer less from over-smoothing and generalize better. We demonstrate this by applying a pruning strategy to revive dead neurons, i.e., we prune the most negative weights connected to the dead neurons. We test our method on the CORA dataset, where pruning-based models are able to reach higher test accuracy, less dead neurons, and less over-smoothing compared to ReLU models without pruning. These findings point towards a highly effective method to improve the performance of ReLU-based GCN models and indicate that there might be a direct connection between the two phenomena dying neurons and over-smoothing in GNNs.

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 their architectures and training methods. Regardless of these advances, a major challenge that most of these architectures still 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 is connected to Laplacian smoothing [Li et al., 2018].

Dying neurons is a concept closely related to the ReLU activation function. Dead neurons are neurons that output 0 for any input [Lu Lu et al., 2020]. These neurons do not propagate gradients, and their weights remain fixed, effectively stopping them from learning. Nevertheless, ReLU remains a simple and well-performing activation function, that can outperform other activation functions [Veličković et al., 2018]. This makes it necessary to investigate the effects of dying neurons on over-smoothing, and whether preventing dying neurons can make ReLU-based models more resistant to over-smoothing.

In our research we compare two approaches for preventing dying neurons, with emphasis on their effects on over-smoothing. The first method we apply is reviving dead neurons. We revive a dead neuron by pruning the most negative weights connected to it. The second method is an alternative activation function that by definition prevents dying neurons. We choose LeakyReLU, a popular choice in the literature. This ReLU alternative has an αx slope in the negative domain. This allows for non-linearity as well as gradient flow for negative inputs, given $\alpha \neq 0, 1$.

We evaluate the performance of the two methods in GCN models on the CORA dataset. The pruning-based method is able to outperform the ReLU and LeakyReLU GCN with $\alpha = 0.8$ on our

deepest model (30 layers). Furthermore, pruning can reduce the amount of dead neurons and total 0 activations for ReLU. Moreover, pruning methods can decrease the node embedding similarity within a trained 128 layer GCN. Our results show that pruning is effective in improving the model's generalization capabilities and preventing over-smoothing as the number of layers increases.

2 Related Work

Dying Neurons in Neural Networks There are many approaches for preventing dying neurons in the literature. Some approaches propose 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 attempt to reduce the amount of dying neurons through weight initialization [He et al., 2015] or regularization techniques [Ioffe and Szegedy, 2015, Papp et al., 2021].

Recent work by Qiao et al. [2018] combines neural architecture search with dead neurons. They reduced the filter size for CNN layers with many dead neurons, and increased the filter size in layers with less dead neurons, attempting to redistribute their given compute budget. Whitaker and Whitley [2023] propose to revive a dead neurons within linear layers by pruning the most negative weights connected to them. Removing the most negative weights is supposed to make the ReLU input positive again, leading to nonzero outputs.

A distinction to our approach is that we apply these pruning techniques to GCNs. To our knowledge, we are the first to investigate the effects 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 prunes the weights [Gurevin et al., 2024] to reduce compute based on the lottery ticket hypothesis [Chen et al., 2021].

Over-Smoothing for GNNs The literature of GNNs offers many methods for preventing over-smoothing. These solutions include skip connections [Li et al., 2019] that add the input of a layer to its output, or jumping knowledge [Xu et al., 2018], passing on information of previous layers throughout the network. Other approaches use regularization methods, e.g. DropEdge by Rong et al. [2020] or DropGNN by Papp et al. [2021], to prevent models from over-smoothing. Zhao and Akoglu [2020] propose PairNorm, which normalizes the output of each layer, to reduce over-smoothing in GNNs. Other methods simply use shallow networks, avoiding deep architectures entirely [Xin et al., 2020].

Recent work by Wu et al. [2024] compared the performance of different activation functions on over-smoothing, concluding that ReLU-based GNNs suffer more from over-smoothing than other activation functions. Reserach by Kelesis et al. [2023] compared over-smoothing in GATs and GCNs with alternative activation functions to ReLU, further indicating that ReLU is prone to over-smoothing. Our work differs from previous approaches since we utilize weight pruning methods to reduce dying neurons. Furthermore, we specifically investigate the effects of dying neurons within ReLU-based models on their over-smoothing behavior.

3 Method

Dying Neurons The phenomenon of dying neurons in neural networks occurs because of the ReLU activation function. The ReLU function is defined as $\text{ReLU}(x) = \max(0, x)$, which makes the gradient of ReLU 0, if $x \leq 0$. Therefore, ReLU stops the gradient flow through neurons that

output 0, due to its derivative. This makes dying neurons an event that is specific to the ReLU function, since other functions like sigmoid or tanh are nonzero in the negative domain. A neuron is considered as dead, if it has a 0 output for any given input. As a consequence, no gradients can flow through this neuron anymore, stopping the weights from being updated and the neuron will never be reactivated. Therefore, a dead neuron remains dead. This can reduce the amount of active parameters in a neural network, which in turn potentially limits the model's expressive power.

Detecting Dying Neurons Our method focuses on dying neurons in GCNs. One layer of a GCN is as follows:

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

where N is the number of nodes, X', X are node embeddings, $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ is the symmetric normalized adjacency with self-loop, and Θ is the weight matrix [Kipf and Welling, 2017]. A dead neuron is a column in X' with only 0s. This means that the same feature is 0 for every node.

Synaptic Stripping Synaptic Stripping, proposed by Whitaker and Whitley [2023], proposed to revive a dead neuron by pruning its most negative weights. Their pruning strategy prunes weights by setting them to 0. This works because ReLU in previous layers guarantees non negative activations and therefore only negative weights contribute to negative ReLU inputs. We prune 10% of the most negative weights, rounding up to the nearest integer.

Measuring Over-smoothing To evaluate over-smoothing we use two methods. Firstly, we train models of different depth and compare their test accuracy. We hypothesize that less over-smoothing should lead to less performance decrease when scaling up the number of layers. Secondly, we evaluate over-smoothing within the models using two similarity metrics. The first metric, proposed by Wu et al. [2024]: $\mu_{\text{DTM}}(X) = \|X - \frac{\mathbf{1}^T X}{N}\|_F$, 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]: $\mu_{\text{Dirichlet}}(X) = \sqrt{\frac{1}{N}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\|X_i - X_j\|_2^2}$, 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 in GCNs The relevance of dying neurons for over-smoothing in GNNs can be motivated twofold. Firstly, a method that can revive dead neurons can increase the amount of active parameters and potentially increase the model's expressive power. An increase in expressive power in turn can lead to better generalization and less over-smoothing [Jin and Zhu, 2024]. Secondly, previous work on the relationship between activation functions and over-smoothing [Kelesis et al., 2023] has shown that the probability of the ReLU output being 0 can be used to construct an upper bound on the distance of node embeddings to the subspace where over-smoothing is prevalent. They derived this bound utilizing singular values of the weight matrices. Whilst reviving dead neurons can not offer theoretical guarantees on reducing this probability, it can help to decrease this probability in practice. Therefore, reviving dead neurons could potentially help to reduce over-smoothing. For more details, refer to Kelesis et al. [2023].

Our methodology consists of training a GCN with ReLU activation function. We chose GCNs, because it is a simple architecture with many experimental and theoretical results on over-smoothing [Wu et al., 2024, Kelesis et al., 2023, Lu Lu et al., 2020]. After every epoch, we detect dead neurons on the validation set, which can act as an extra regularizer compared to using the training set. We then prune 10% of the most negative weights in the weight matrix Θ (Eq. 1) by setting them to 0, other re-initialization strategies failed to show promising results in preliminary experiments.

4 Results

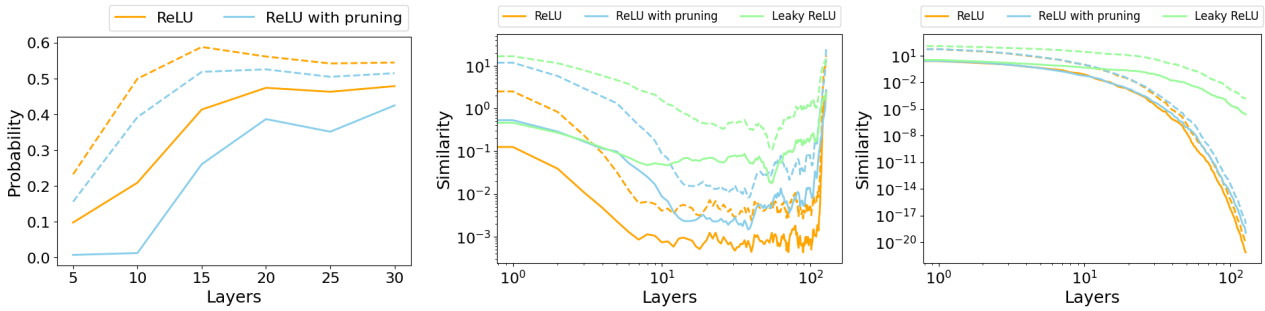
Table 1: Mean and standard deviation of test accuracies of the GCN models. We trained each model 10 times over 100 epochs with hidden dimension of 32, a learning rate of 0.005, and a weight decay of 0.0005. We test the model that performed best on the validation dataset.

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 3 GCN models, one with ReLU activation function, one with our proposed method of reviving dead neurons in GCN layers, and one with LeakyReLU. We set $\alpha = 0.8$ for LeakyReLU-based on previous results by Wu et al. [2024], indicating that LeakyReLU suffers remarkably less from over-smoothing compared to smaller values for α. We trained the models on the CORA dataset for node classification, a citation network of 2708 machine learning papers [McCallum et al., 2000]. Table 1 gives several key insights. The results show that LeakyReLU almost always performs better than its ReLU counterparts, especially for 15 and 20 layers. Indicating good generalization and resistance to over-smoothing for LeakyReLU-based models. Yet, ReLU with pruning is able to achieve the highest accuracy on the deepest model with 30 layers. Additionally, ReLU with pruning achieves higher accuracy than the ReLU-based model for 20, 25, and 30 layers. This indicates that pruning can effectively increase the performance of GCN models, especially for deeper networks. Furthermore, we can see a turning point, where the performance of a model begins to drop. This turning point is accompanied by high variance, indicating that some models are still able to learn properly, while others fail, likely due to over-smoothing. ReLU-based models have this turning point at 10 and 15 layers as compared to 20 and 25 layers for LeakyReLU, indicating that LeakyReLU can effectively prevent early over-smoothing. The lower variance of pruning-based models around this turning point shows that pruning can make the models more robust.

Figure 1a shows that our proposed pruning method can effectively reduce the amount of dead neurons for all layer sizes. This effect is especially visible for models up to 10 layers, where a 10 layer model with pruning has an average of 1% dead neurons, compared to 20.5% without pruning, effectively increasing the amount of active parameters. Moreover, we can see that the overall number of 0 outputs from ReLU was reduced for every model. This empirically shows, that pruning can reduce the probability of a 0 ReLU output in practice, which impacts the over-smoothing properties of ReLU-based GNNs.

In our second experiment, we trained each model with 128 layers and the same hyperparameters as in the first experiment. We average results over 3 runs and pick the model with the highest accuracy on the validation set. Figure 1b compares the Dirichlet energy, as well as the DTM metric for all three models. The results show that pruning helps the model to consistently keep its internal node representations more dissimilar compared to ReLU without pruning. This strongly indicates



(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).

that ReLU-based methods are able to effectively reduce over-smoothing within GCNs. We also compare over-smoothing of randomly initialized GCNs in Figure 1c. The results show that the LeakyReLU architecture seems to suffer less from over-smoothing, likely because it does not cut off all negative values. Furthermore, this shows that trained GCN models increase dissimilarity within their internal representations towards the last layers, likely to generate meaningful logits.

5 Conclusion

We have shown that pruning strategies similar to synaptic stripping can be applied to GCN architectures. Reviving dead neurons in GCN models helped them outperform their ReLU, as well as LeakyReLU counterparts for a depth of 30 layers. Furthermore, reviving dead neurons helped the GCN to consistently have higher dissimilarity in its internal representations, compared to ReLU-GCNs without pruning. These findings indicate that dying neurons contribute to over-smoothing within GNNs. Our methodology is straightforward to implement and can be easily adapted for existing ReLU-based GCN models.

Limitations Our experiments pointed towards several limitations. First, reviving dead neurons showed inferior performance to LeakyReLU in most comparisons, which indicates that preventing dead neurons entirely is still more effective than reviving them. Second, whilst our method was able to reduce the number of dead neurons, we still found around 40% dead neurons in deep models. There might be more effective ways of reviving neurons. Lastly, even though the adaptation of our method in current GCN models can be easily done, detecting dead neurons and pruning weights has additional overhead which scales with model size, and could lead to training time increases.

Future Work We acknowledge that our study was limited by low compute resources, and we propose further investigation to verify our hypothesis of the relation of dying neurons and over-smoothing on 1) different datasets, 2) different GNN architectures, and 3) larger GNNs. Reducing the number of dead neurons with even more effective reviving methods could give further valuable insights. Another shortcoming is that there is no theoretical analysis of our results, as pruning is based on a heuristic; finding mathematical guarantees 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. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. 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.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. URL https://arxiv.org/abs/1609.02907.

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, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax, 2018. URL https://arxiv.org/abs/1809.10341.

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.