NEURAL IMPORTANCE-BASED TRAIN-TIME PRUNING

Akhil Krishna Reddy, Prajna Narayana Holla Kundapura, Rishabh Budhia & Ronak Jain

Thomas Lord Department of Computer Science
University of Southern California
{akhilkri,hollakun,budhia,ronakjai}@usc.edu

1 INTRODUCTION

Deep neural networks have been pivotal in advancing many fields, ranging from image and speech recognition to complex tasks in medicine, neuroscience, and competitive gaming arenas like chess, *StarCraft II*, and multiplayer online battles. Their ability to surpass human-level performance in specific tasks is a testament to their sophistication and effectiveness. Transfer learning has emerged as a critical strategy in specialized domains where data generation is challenging and expensive, such as medical image generation from advanced techniques like *Diffusion Tensor Imaging (DTI)* or *Magnetoencephalography (MEG)*. It leverages knowledge from source tasks to enhance CNN performance in target tasks with sparse data.

Despite their successes, the high predictive performance of these networks often incurs significant storage and computational burdens, directly impacting energy consumption. These deep architectures, characterized by their extensive parameterization, can lead to models having more number of parameters than training samples. For example, modern architectures like *EfficientNet-B3* and *DenseNet-121* have tens of millions of parameters, requiring substantial storage space and computational power. *EfficientNet-B3*, for instance, demands several billion floating-point operations (*FLOPs*) for a single image classification task.

Overparameterization, while beneficial during the training phase for efficient learning, becomes redundant post-training. Pruning eliminates extraneous network units (weights or filters) and can streamline these models without significant performance loss. This reduction is particularly crucial for deployment in resource-constrained environments such as mobile devices, edge computing, or embedded systems in autonomous vehicles. Our method deliberately eliminates non-essential *neurons* within the network, as determined by specific evaluative criteria. Such pruning makes the network sparse, optimizing its performance and significantly reducing the model's size and the computational time required for inference. This streamlined structure not only makes the network more efficient in resource utilization but also helps mitigate the risk of over-fitting, ensuring the model generalizes better to new, unseen data.

For an exhaustive and accurate assessment, our methodology extends beyond merely averaging the contribution ratio of a neuron across all forward passes in an epoch. We additionally average these ratios across the entirety of neurons in a given layer. This dual-natured averaging strategy — encompassing both the dimension of the forward passes and the breadth of neurons in the subsequent layer — offers a comprehensive perspective on a neuron's impact on network functionality. This approach meticulously calculates the average contribution ratio by considering the variance in data propagated across all forward passes and the neuron's influence relative to its peers in a given layer. By integrating these two dimensions of analysis, we achieve a more nuanced understanding of each neuron's role and importance. This refined evaluation process ensures that the pruning strategy is precise and context-sensitive, retaining neurons that are crucial for the network's learning efficacy while eliminating those with marginal utility. This technique optimizes the network, reducing its structural complexity and computational demands and enhancing the model's generalizability, thus contributing to an efficient and robust network architecture.

2 RELATED WORK

Numerous methodologies have been proposed in the landscape of deep neural network compression, each contributing unique insights to the delicate balance between model efficiency and performance.

Han et al. (2015) introduced a comprehensive approach involving pruning, trained quantization, and Huffman coding, achieving notable model compression. *Li et al. (2016)* advocated filter-level pruning, achieving substantial model size reduction without compromising accuracy. *He et al. (2017)* improved the efficiency of intense neural networks through channel pruning, reducing computational load without sacrificing accuracy. *Molchanov et al. (2017)* concentrated on inference efficiency, proposing a method for faster inference while maintaining accuracy. *Dong et al. (2017)* presented a comprehensive pruning framework with a novel regularization term. *Frankle and Carbin (2018)* introduced the lottery ticket hypothesis, identifying trainable sparse subnetworks through iterative pruning. *He et al. (2018)* proposed soft filter pruning, achieving efficient model acceleration. *Zhao et al. (2019)* contributed with *Pyramid Structured Filter Pruning (PSFP)*, optimizing the trade-off between model size and performance. Our proposed idea stands out by offering a dynamic and adaptive pruning strategy during training, allowing for a more nuanced identification and removal of the most minor "important" neurons while maintaining minimal impact on loss/accuracy.

Guo et al. (2016) advanced the field with dynamic network surgery, enabling adaptive pruning and regrowing during training. *Liu et al. (2017)* explored channel pruning during training with "network slimming," systematically reducing network channels for efficient model compression. In addition, *Seul-Ki Yeom et al. (2021)* presented a novel criterion for deep neural network pruning using *Layer-wise Relevance Propagation (LRP)*. Our solution provides a more comprehensive approach to dynamic rank calculation, incorporating running averages and maximum contributions. This adaptability offers finer control over the pruning process, providing a more nuanced evaluation of neuron importance.

3 Approach

3.1 Task

This project[1] introduces a novel approach to prune feed-forward neural layers, aiming to remove the least significant neurons with minimal impact on the model's accuracy and loss. This technique addresses the shortcomings of existing methods, such as activation-based pruning, which often lacks attention in the nuanced effects of neuron interactions with subsequent layers. Furthermore, recent importance-based pruning methods such as *LRP* need to address the 'vanishing importance' issue in deeper networks, akin to vanishing gradients, leading to inefficient pruning. Our approach measures the contribution of a neuron to the network's learning representations, as a quantifiable entity, while making pruning decisions. Preliminary evaluations suggest that our method exceeds existing state-of-the-art pruning techniques, particularly in scenarios where fine-tuning is impractical.

3.2 Datasets

As we aimed for rapid experimentation and inference generation, we chose these datasets from the default classes on Pytorch -

- *MNIST* and *FashionMNIST* - These datasets contributed to the systematic optimization for the rank calculation hyperparameters, given their size and complexity
- *CIFAR-10* and *CIFAR-100* - These datasets helped in the rank methodology's adaptability to low resolution datasets. They contributed to the development and testing of different Importance-weighting techniques within the rank functionality
- As we aimed to increase complexity (and the number of parameters) for *VAEs*, we chose the *CelebA* dataset over *MNIST* for the *VAE* implementation. The presence of distinct samples i.e., no two images were of the same celebrity greatly helped the variance of this dataset. Training and optimizing *VAEs* for this complex dataset helped in elucidating "special case" samples, helping the development of *max rank* functionality within the model

[1] https://github.com/akhil-reddy/activation-based-pruning

Algorithm 1 Importance based Pruning

Input: Initialized Deep neural *net*, training data x_t
Output: Pruned network model
Train for a few iterations until network attains initial maturity:
while Pruning Target met **do**
  for each forward pass **do**
    for each layer in *net* **do**
      for each neuron in layer **do**
        Compute running-average and max contributions of neuron
      end for
    end for
  end for
  Supply contribution matrix (C_{avg} and C_{max}) to the Rank Calculator.
  for each layer in *net* **do**
    Assign ranks based on their contribution to the network
  end for
  Supply the rank matrix to Network Reinit module
  for each layer in *net* **do**
    Determine the bottom $x\%$ neurons of each layer to prune
  end for
  Initialize the new network while excluding the pruned neurons
  Import weights from the previously trained model for the non-pruned neurons
  return Pruned Model
  Continue training to recover the lost representations from the pruned neurons and use them to implicitly improve the learning density of the other neurons
end while
Measure the train-performance of the model

3.3 APPROACH

In our approach to optimizing network efficiency, we employ a sophisticated method for calculating the contribution/importance of each neuron, which forms the basis for selective pruning. This process begins by assessing the contribution of a neuron to the *affine combinations* of neurons in the following layer. For each neuron, we calculate a contribution ratio, reflecting the proportion of its weighted activation in relation to the sum of weighted activations from all neurons that feed into a specific neuron in the next layer. We then extend this analysis over all forward passes and across all neurons in a given layer, thereby obtaining two key metrics Average Contribution C_{avg} and Maximum Contribution C_{max}.

C_{avg} : This is a measure of a neuron's average importance across all forward passes and for all neurons in a given layer. It's calculated by averaging the contribution ratios over time and across different neurons in the subsequent layer. This metric provides insight into the consistent, overall impact of a neuron within the network, reflecting how integral it is to the network's functioning, on average.

C_{max} : This metric assesses whether a neuron has ever been highly activated during any of the forward passes. It represents the peak influence of a neuron for the given training data. A high value of C_{max} indicates that at some point, the neuron had a significant impact on the activation of neurons in the next layer, even if it may not contribute consistently (as measured by C_{avg}).

By considering both C_{avg} and C_{max}, the pruning strategy gains a comprehensive view of each neuron's role in the network: C_{avg} provides a picture of consistent, average performance, while C_{max} identifies the potential peak influence a neuron can have. This dual analysis allows for a more nuanced approach to pruning, ensuring that neurons are evaluated based on both their steady contributions and their peak performance capabilities. For a given neuron *k*,

$$C_{avg} = \frac{1}{F * M} \sum_{f=1}^{F} \sum_{j=1}^{M} \frac{w_{kj} a_{kf}}{\sum_{i=1}^{N} w_{ij} a_{if}} \tag{1}$$

$$C_{max} = \max(\frac{w_{kj}a_{kf}}{\sum_{i=1}^{N} w_{ij}a_{if}}), \forall i \in [1, N], \forall j \in [1, M], \forall f \in [1, F] \tag{2}$$

where *M* is the number of neurons is the next layer, *N* is the number of neurons in the current layer and *F* is the number of forward passes in the epoch.

Both C_{avg} and C_{max} are weighted by hyperparameters λ_1 and λ_2 of the learning environment, respectively. The aggregate contribution is thus calculated as:

$$\lambda_1 C_{\text{avg}} + \lambda_2 C_{\text{max}}$$

When λ_1 is set near 1, the emphasis is placed more on the consistent average contribution of neurons. This is ideal for scenarios where maintaining a steady and reliable performance across all forward passes is crucial. Conversely, by setting λ_2 near 1, the focus shifts to the peak performance of neurons, targeting those that may not have shown high activation levels but have the potential for significant impact in specific instances.

The key is to find a balance between these two parameters that best suits the task. This may involve a process of experimentation and iterative adjustment, where different combinations of λ_1 and λ_2 are tested to observe their effects on task adaptation, and network efficiency and accuracy.

For instance, a balanced approach with $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$ might be a starting point, penalizing neurons only if they are consistently underperforming in both average and maximum contributions. However, if the network is prone to overfitting, a higher weight on λ_1 might be more effective. Ultimately, the optimal setting of these hyperparameters is contingent upon the specific architecture and task of the neural network. Continuous monitoring and adjustment of λ_1 and λ_2 are therefore crucial to prune effectively while aligning with the desired outcome.

This methodical and adjustable approach ensures that the pruning process is both precise and adaptable, effectively identifying and removing neurons that contribute minimally to the network's performance at inference time.

4 RESULTS

4.1 PRUNING FOR CLASSIFICATION TASKS

The baseline models of comparison for the classification tasks are described in the related work section above. For simpler data sets such as *MNIST* (or its variation i.e., *FashionMNIST*), most papers utilize fully connected feed-forward networks such as *LeNet*. For the preliminary evaluation of our rank calculation methodology, we build a similar feedforward network for the comparison in Table 1. Similarly, for colour datasets (*CIFAR-10* and *CIFAR-100*) with moderately complex features, our baseline models were popular basic neural networks with convolutional layers, like *VGG-16*.

The metric that naturally suited these classification tasks was accuracy and we've compared the train-time accuracy of our model with the train-test-time accuracy of baseline models. Given that our methodology involves pruning and recovery at train-time, we believe that accuracy at train-time best elucidates the different phases of the methodology.

4.1.1 RESULTS

To evaluate the advantages of our pruning/ranking methodology, we compare three curves as shown in Figure 1. The blue and green curves are to illustrate the adaptability of the pruning methodology over the random pruning of neurons, essentially making the removal of neurons stochastic instead of systematic. Each trough in the graph shows the loss of accuracy immediately after a pruning epoch. However, as training progresses, there is a near-perfect recovery in performance up until we remove about 70% neurons from the fully connected layers. It is important to note that our pruning method

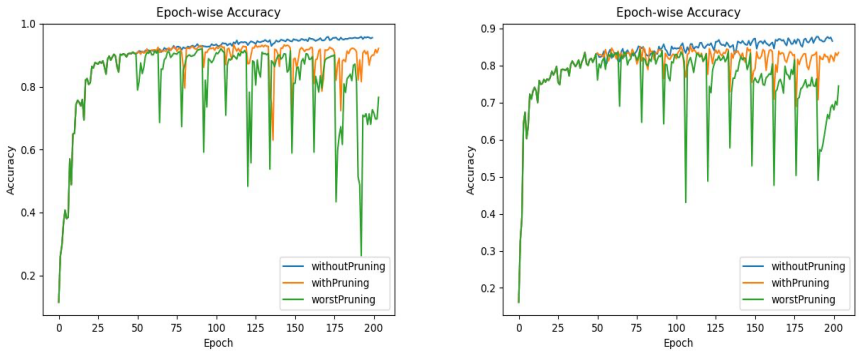

Figure 1: The left figure compares the accuracy of a basic feedforward network for *MNIST* over random and rank-based pruning styles. The right figure is the same for the *FashionMNIST* dataset.

Table 1: Accuracy comparison for classification tasks. A datapoint $A@P$ shows A accuracy at P % neurons pruned.

Pruning Methodology	**MNIST**	**CIFAR-10**	**CIFAR-100**
PQH	0.9842	-	-
NS	0.9851	0.9221	0.6419
FPRL	-	0.9407	0.7224
LRP	-	0.9323@30	-
PFEC	-	0.9294	-
Rank-based	0.9217@68	0.9140@41	0.7625@41

ensures minimal impact to accuracy in that scenario as well. When comparing the *CIFAR* results, our methodology outperforms *LRP* on an accuracy-per-pruned-neuron basis (with recovery).

An open question from our results is on the subjectivity of determining "important" neurons. While our theory suggests that activation is a reasonable proxy for importance/contribution, further analysis on different layer types is required to strengthen/weaken that hypothesis.

4.2 Pruning for generative tasks

After a survey of related literature, we couldn't find a relevant comparator for comparing the pruning performance of *Variational Autoencoders (VAE)*, especially at train-time. Instead, our experimentation involved the comparison of the *VAE* Loss (the proportional summation of *Kullback-Leibler Divergence (KLD)* and *Binary Cross-Entropy (BCE)* losses) at different pruning intervals to infer the loss behaviour after prune epochs having a significant drops (between 8-10%) in the number of neurons.

In the contrast to the classification tasks above, generative models are typically evaluated on loss metrics. After multiple pruning trails, we've consistently observed that the loss degradation follows the same path, irrespective of the ratio of *KLD* to *BCE* losses in the total loss. Hence, we've fine-tuned this parameter to elucidate subtle variations (by "magnifying" the *KLD* coefficient, with a higher proportion) in behaviour over time.

4.2.1 Results

The graphs in Figure 2 are to be viewed holistically, to discern the generative results; due to the lack of a comparator as described earlier. When comparing the regenerative capabilities (after a few epochs of recovery) of an *85* million parameter model to that of a *58* million one (i.e., approximately 30% neurons pruned), we can observe that pruning the model improves learning density and representation - the cohesiveness of facial features and the orientation of background details -

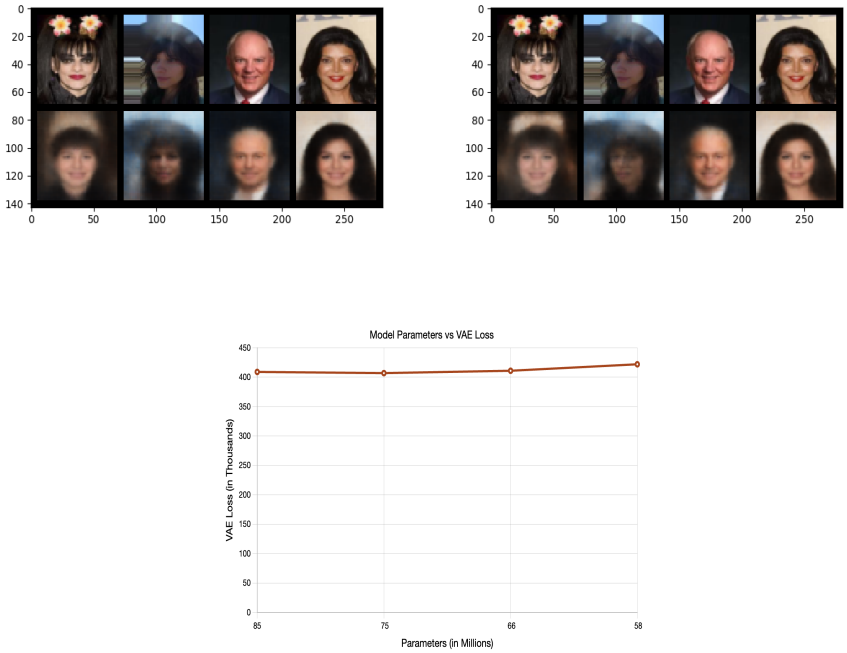

Figure 2: The top two figures show the reconstruction capabilities of *VAE* before and after pruning 30% of the network. The bottom figure shows the comparable loss in *VAE* for different parameter sizes.

substantially in a few epochs (15) of pruning. While these representational improvements can be attributed to the natural progression of training, our null hypothesis tests indicate otherwise.

An open question from the generative results is on the uncertainty of the above improvements. As our accessible hardware allowed limited experimentation on *VAEs*, further analysis on said representational improvements is required.

5 Conclusion and Future Work

We pruned about 40% of neurons for feedforward and *CNN* models without an impact to accuracy, resulting in a more compact and resource-efficient network. Further pruning has a minor impact on loss and accuracy but performs better than randomized pruning of neurons (worst-case scenario in the graphs). For *Variational Autoencoders (VAE)*, pruning more than 30% of neurons in an 85 million parameter model had little-to-no impact on the regenerative capabilities of the *VAE*. This inferential result is crucial as it helps in generating *LoRA* models for targeted applications.

Our proposed approach provides a novel perspective to neural network compression by introducing a dynamic pruning strategy during train-time. Unlike static methods, our approach maintains the model performance while achieving compression goals. By decoupling rank calculation between layers and considering weighted activations, we accurately assess neuron significance, mitigating the 'vanishing importance' problem. Our dynamic rank calculation method, guided by hyperparameters λ_1 and λ_2, comprehensively explains each neuron's impact, allowing flexible adaptation to different scenarios. The ability to switch between average and max rank calculation enhances flexibility, crucial for mission-critical tasks. Future work should explore hyperparameter trade-offs, generalizability across architectures, and optimization for computational efficiency, promising continued exploration and refinement in neural network compression.

6 REFERENCES

Han, S., Mao, H. and Dally, W.J., 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

Guo, Y., Yao, A. and Chen, Y., 2016. Dynamic network surgery for efficient dnns. Advances in neural information processing systems, 29.

Li, H., Kadav, A., Durdanovic, I., Samet, H. and Graf, H.P., 2016. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710.

He, Y., Zhang, X. and Sun, J., 2017. Channel pruning for accelerating very deep neural networks. In Proceedings of the IEEE international conference on computer vision (pp. 1389-1397).

Molchanov, P., Tyree, S., Karras, T., Aila, T. and Kautz, J., 2016. Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440.

Dong, J., Zheng, H. and Lian, L., 2017. Activation-based weight significance criterion for pruning deep neural networks. In Image and Graphics: 9th International Conference, ICIG 2017, Shanghai, China, September 13-15, 2017, Revised Selected Papers, Part II 9 (pp. 62-73). Springer International Publishing.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S. and Zhang, C., 2017. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE international conference on computer vision (pp. 2736-2744).

Frankle, J. and Carbin, M., 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635.

He, Y., Kang, G., Dong, X., Fu, Y. and Yang, Y., 2018. Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:1808.06866.

Feng, Y., Huang, C., Wang, L., Luo, X. and Li, Q., 2022. A Novel Filter-Level Deep Convolutional Neural Network Pruning Method Based on Deep Reinforcement Learning. Applied Sciences, 12(22), p.11414.

Yeom, S.K., Seegerer, P., Lapuschkin, S., Binder, A., Wiedemann, S., Müller, K.R. and Samek, W., 2021. Pruning by explaining: A novel criterion for deep neural network pruning. Pattern Recognition, 115, p.107899.

LeCun Y., Bottou L., Bengio Y. and Haffner P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324

Simonyan K. and Zisserman A., 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556

7 ACKNOWLEDGEMENTS

The authors would like to thank Oliver Liu, Joshua Robinson, Tejas Srinivasan and Dr. Dani Yogatama from the Thomas Lord Department of Computer Science at USC for their feedback on the feasibility of our ideas and methods. We would also like to thank Moshe Sipper for their Medium article on the implementation of *Variational Autoencoders* with the *CelebA* dataset.

8 AUTHOR CONTRIBUTIONS

This section lists the author contributions of each author.

- All authors contributed equally to the idea generation and brainstorming tasks
- All authors contributed equally to running the initial tests and collecting the results for the feasibility of the idea
- Rishabh Budhia contributed to the data preprocessing and the structural skeleton of the model implementation

- Ronak Jain contributed to the rank calculator, including exploration and experimentation
- Ronak Jain and Prajna Holla contributed to the development and implementation of the *CNN* models
- Akhil Krishna Reddy contributed to the ideation of pruning & reinitialization, along with the development and implementation of the generative *VAE* model
- Prajna Holla contributed to the aggregation and visualization of results
- Rishabh Budhia contributed to maintaining the *GitHub* codebase for various implementations, during experimentation
- All authors contributed equally to the poster and presentation
- All authors contributed equally for the writing of the report

A Appendix

A.1 Models for Classification Tasks

We've implemented the below models -

- For the simpler task of classifying *MNIST* images, we've used a four layer fully connected neural network with two hidden layers of sizes 50 and 30 respectively. Faster initial experimentation was the rationale behind this configuration as it helped in determining the various hyper-parameters of the rank calculation methodology. A simple model also helped in determining the adverse effects of over-pruning
- For the colour images in the *CIFAR* data sets, the features were dense and blurry. With low accuracy models, the experiments wouldn't produce the required definitive drops in accuracy as training progresses. Hence, we chose a simple CNN with two convolutional blocks and three hidden layers with a systematic reduction in layer size from a fully-connected-input of 2048 to $10/100$ output neurons

A.2 Models for Generative Tasks

We chose a variational autoencoder as our most advanced model (for experimentation) because of its relevance in today's model design. From generative models (such as *Stable Diffusion*) to foundational ones (in the newly released *DeepMind Gemini* and other multimodal architectures), VAEs are crucial for imploding and exploding higher representations of data and information. However, they are expensive to train and are presented as the perfect candidate for train-time pruning.

For the implementation, we've utilized a typical setup of two convolutional encoder-decoder blocks supplemented with two or three hidden layers. From a simple setup of ten layers, the model had *85* million parameters, further reinforcing the need for pruning at scale.