Neural Growth Factor-Inspired Pruning for Efficient Neural Networks

Julian Weaver
The University of Texas at Austin
julian.weaver@utexas.edu

Abstract

The computational demands of large deep neural networks (DNNs) hinder their deployment in resource-constrained settings. This motivates model compression techniques like pruning (2; 4). This work introduces and evaluates a pruning technique inspired by Neural Growth Factor (NGF)-mediated neuronal apoptosis. We monitor neuron activation magnitudes during a "critical period" early in training and prune neurons with persistently low activation. This process mimics biological competition for limited trophic support (9). This single-shot, activation-based NGF-inspired method is compared against standard magnitude-based pruning (3) and an unpruned baseline. Evaluations span supervised learning (FashionMNIST (12) with MLPs and CNNs (6) and reinforcement learning (CartPole-v1) (10) with policy networks trained via REINFORCE (11). We assess the efficacy of this strategy in reducing parameter count while preserving performance across different tasks and architectures.

1 Introduction

Deep Neural Networks (DNNs), while powerful, often contain millions or billions of parameters, leading to high computational, memory, and energy costs (2; 4). This limits deployment on platforms with limited resources, like mobile devices or embedded systems. Neural network pruning addresses this by removing redundant network components to create smaller, efficient models while aiming to maintain performance (4).

Magnitude-based pruning is a common baseline, removing weights with small absolute values under the heuristic that they contribute less (3; 5). While simple and often effective in reducing parameters, the resulting sparsity often yields limited practical speedups on standard hardware without specialized libraries (2; 4). This motivates exploring alternative criteria, such as neuron activation statistics (8; 1) or methods inspired by biological processes.

Biological nervous systems undergo developmental refinement, in which neurons compete for limited neurotrophic factors like Neural Growth Factor (NGF). Neurons failing to secure sufficient trophic support, often correlated with lower activity or less effective connectivity, undergo apoptosis (programmed cell death) (9; 7). This selective survival prunes less effective connections, optimizing neural circuitry.

Inspired by this, we propose an NGF-inspired pruning method. We hypothesize that neuron activation magnitude during an early "critical period" reflects functional importance. By pruning neurons with low average activation during this period, we aim to mimic biological competition and achieve efficient networks. Unlike magnitude pruning, this is neuron-level and activation-based.

Our goals are:

- Implement a single-shot, NGF-inspired, activation-based pruning algorithm.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

- Evaluate its effectiveness on supervised (FashionMNIST (12)) and reinforcement learning (CartPole-v1 (10)) tasks.
- Compare performance (sparsity, accuracy/reward) against unpruned and magnitude-pruned baselines.

2 Methods

We compare our proposed NGF-inspired pruning against standard magnitude pruning.

2.1 Pruning Algorithms

NGF-Inspired Pruning: This method mimics NGF-mediated survival based on activity during a critical period.

- *Monitoring Phase:* During a predefined critical period (e.g., specific epochs or episodes early in training), we track the average absolute activation value for each neuron in designated prunable layers (custom 'PrunableLinear', 'PrunableConv2d').
- *Pruning Phase:* At the end of the critical period, neurons in each layer are ranked by their average activation. Neurons below a threshold determined by a 'keep_fraction' hyperparameter (e.g., keeping the top 80%) are pruned.
- *Implementation:* Pruning involves applying a persistent binary mask to the neuron's output and setting its associated weights (incoming, outgoing) and bias to zero. This performs neuron-level pruning, potentially offering structured sparsity benefits (2).
- *Justification:* Activation magnitude serves as a proxy for functional contribution, analogous to trophic factor competition. The critical period mirrors developmental windows. Neuron-level pruning is chosen for potential hardware efficiency.

Magnitude Pruning (Baseline): We use standard global, unstructured L1 magnitude pruning (3).

- *Mechanism:* All weight parameters (excluding biases) across prunable layers are pooled.
- *Timing:* Pruning occurs once at a specific epoch/episode.
- *Criterion:* Weights are ranked by absolute magnitude ($|w|$).
- *Implementation:* A fraction ('prune_fraction') of weights with the lowest magnitude are set to zero using standard library functions (e.g., 'torch.nn.utils.prune.l1_unstructured').
- *Justification:* Chosen as a widely used, simple baseline for comparison (2).

2.2 Experimental Setup

Supervised Learning (FashionMNIST):

- *Models:* Simple MLP and LeNet-5 CNN (6).
- *Dataset:* FashionMNIST (12), a 10-class grayscale image dataset (28x28 pixels), used as a standard benchmark.
- *Training:* Standard supervised loop with Adam optimizer, CrossEntropy loss, batch processing, and early stopping based on validation accuracy. Pruning applied during training.

Reinforcement Learning (CartPole-v1):

- *Model:* Simple MLP policy network (PolicyMLP).
- *Environment:* CartPole-v1 from Gymnasium (10), a classic control task requiring balancing a pole on a cart.
- *Training:* REINFORCE algorithm (Monte Carlo Policy Gradient) (11). Policy parameters θ updated via gradient ascent, with a discounted return. Pruning applied at a specific episode number.

3 Experiments and Results

We evaluated the proposed NGF-inspired pruning method against baseline (unpruned) and standard magnitude pruning approaches across supervised and reinforcement learning tasks. Key hyperparameters for NGF pruning, including the fraction of neurons to keep (`keep_fraction`) and the timing/duration of the critical period, were varied to assess their impact.

3.1 Supervised Learning: FashionMNIST

Experiments were conducted using both MLP and LeNet-5 architectures on the FashionMNIST dataset.

MLP Performance vs. Sparsity The relationship between neuron sparsity and model accuracy for the MLP is shown in Figure 1. NGF pruning maintains performance comparable to the baseline up to approximately 15% neuron removal (`keep_fraction` ≈ 0.85). Beyond this threshold, accuracy drops sharply, and performance becomes sensitive to hyperparameter choices, particularly the critical period start epoch. A later start allows the network to achieve higher accuracy before pruning potentially causes divergence. This contrasts with the expected smoother degradation from magnitude pruning.

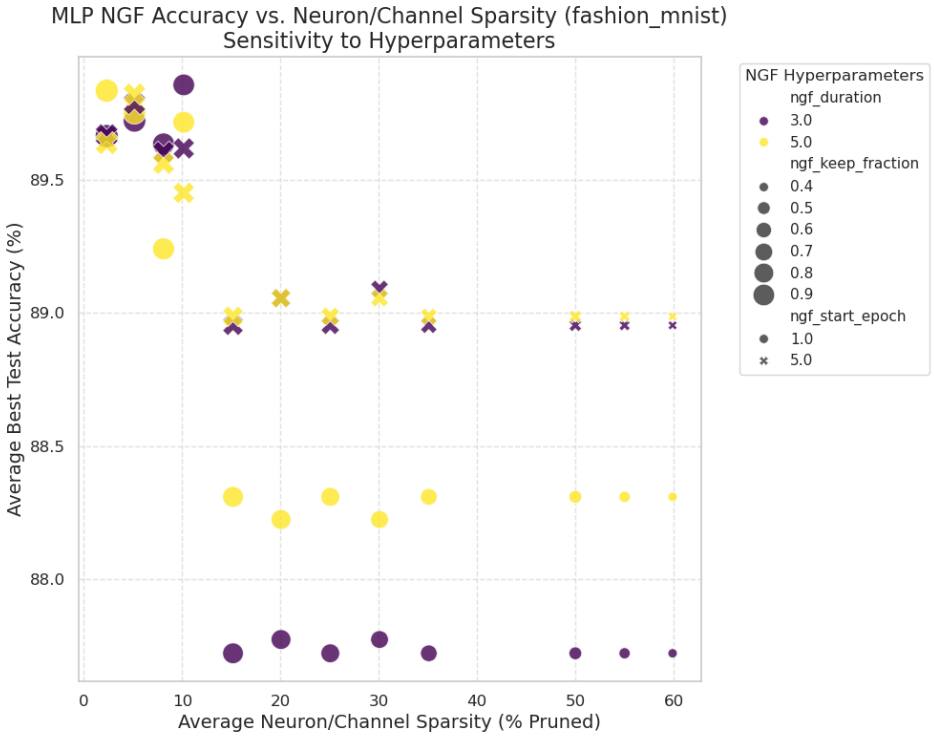

Figure 1: MLP on FashionMNIST: Average best test accuracy vs. NGF neuron sparsity. Different markers indicate NGF hyperparameter settings. Accuracy is stable up to $\sim$15% neuron removal before a sharp decline.

MLP Efficiency and Loss Comparison Figure 2 compares the computational efficiency (training time) and final performance (minimum test loss) between the baseline, magnitude pruning, and NGF pruning across various hyperparameter settings for the MLP. NGF pruning generally results in the shortest training times, partly attributed to faster convergence or divergence leading to early stopping. It also often yields higher minimum test loss compared to the baseline and magnitude pruning, although specific NGF configurations can match or slightly improve upon the baseline loss. Magnitude pruning also reduces training time relative to the baseline, albeit less dramatically than NGF, while generally achieving better test loss than most NGF configurations.

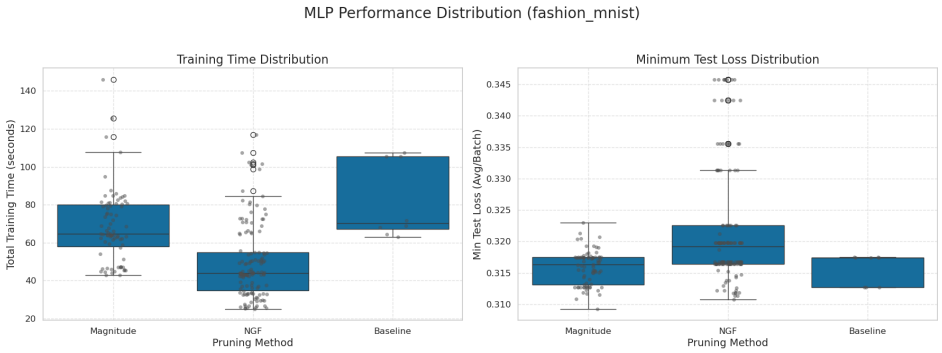

Figure 2: MLP on FashionMNIST performance comparison: Training time (left) and minimum test loss (right) via box plots for baseline, magnitude pruning, and various NGF configurations.

MLP Learning Dynamics The learning curves in Figure 3 illustrate the training and test loss progression for the MLP under different NGF settings. These plots visually confirm the threshold behavior: runs where a high fraction of neurons are removed (`keep_fraction` < 0.85) exhibit divergence in both training and test loss relatively early in training.

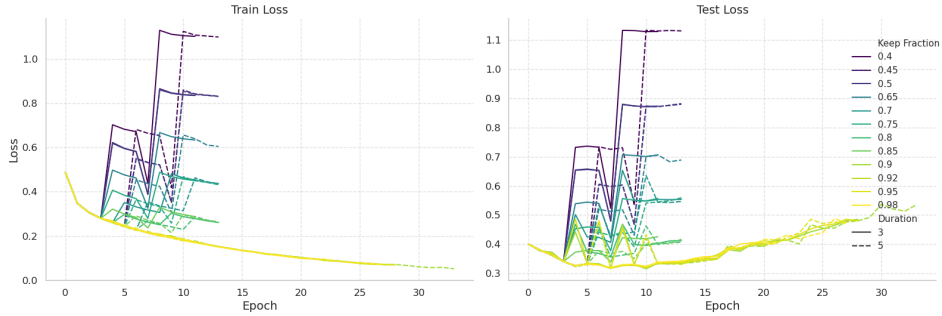

Figure 3: MLP on FashionMNIST learning dynamics with NGF pruning: Training loss (left) and test loss (right) over epochs for varying NGF `keep_fraction` (color intensity) and critical period duration (line style).

LeNet Layer-wise Sparsity To understand where different methods prune, Figure 4 compares the distribution of pruned elements across layers in LeNet-5. Magnitude pruning primarily removes weights in the central fully connected layer (FC1). In contrast, NGF pruning removes neurons most heavily from the initial convolutional layer (Conv1) and the final fully connected layer (FC3). This indicates that weight magnitude and neuron activation identify different structural elements as least important according to their respective criteria.

LeNet Learning Dynamics Similar to the MLP, the LeNet-5 architecture pruned with NGF exhibits a comparable threshold effect, as shown by the training and test loss curves in Figure 5. High levels of neuron pruning lead to divergence and poor generalization.

3.2 Reinforcement Learning: CartPole-v1

We evaluated NGF pruning using a PolicyMLP on the CartPole-v1 environment, comparing a specific NGF configuration (`keep_fraction` $= 0.9$, critical period episodes 100-200) against the baseline. Figure 6 presents the results. While the NGF-pruned agent initially achieves lower rewards per episode compared to the baseline, its learning trajectory appears more linear and stable. Notably, the average computation time per episode is consistently lower for the NGF-pruned agent throughout training. This suggests that while NGF pruning might slow initial reward acquisition, it could lead to faster convergence in terms of wall-clock time by simplifying the policy network early on, potentially

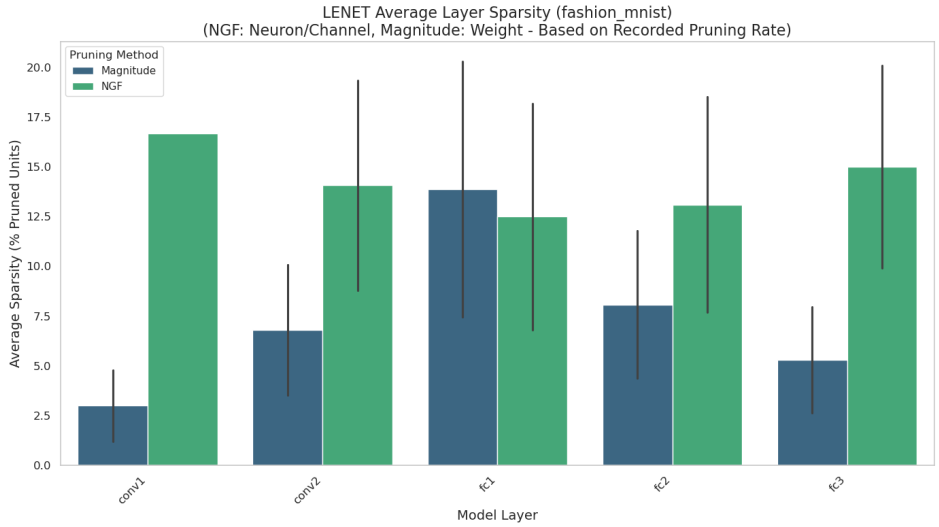

Figure 4: Layer-wise sparsity distribution in LeNet-5 on FashionMNIST after magnitude pruning (weights) vs. NGF pruning (neurons).

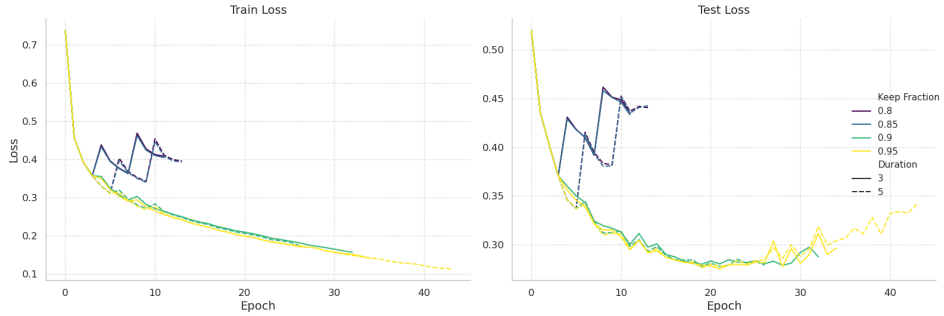

Figure 5: LeNet-5 on FashionMNIST learning dynamics with NGF pruning: Training loss (left) and test loss (right) over epochs for varying NGF `keep_fraction`.

reaching near-optimal performance more efficiently. Late in training, the NGF agent's performance approaches or potentially exceeds the baseline.

4 Discussion

Our experiments provide initial insights into the behavior of the proposed NGF-inspired neuron pruning technique compared to baseline and standard magnitude pruning.

Performance Threshold in Supervised Learning: A key finding on FashionMNIST (Figures 1, 3, 5) is the existence of a relatively sharp performance threshold for NGF pruning. Removing up to 15% of neurons based on activation during a critical period had minimal impact on final accuracy for both MLP and LeNet architectures. Beyond this threshold, performance degraded rapidly, and results became highly sensitive to hyperparameters like the critical period timing. This suggests that while a fraction of neurons identified by low activation might be redundant, removing too many destabilizes the network. This contrasts with magnitude pruning, which exhibits a more gradual performance decline as more individual weights are removed. The abrupt failure mode of NGF pruning might stem from the removal of entire functional units (neurons) rather than fine-grained connections.

Efficiency and Trade-offs: NGF pruning consistently reduced training time (Figure 2), often significantly. However, this speedup in supervised learning was frequently associated with poorer final test loss, potentially linked to the network diverging and triggering early stopping. Magnitude

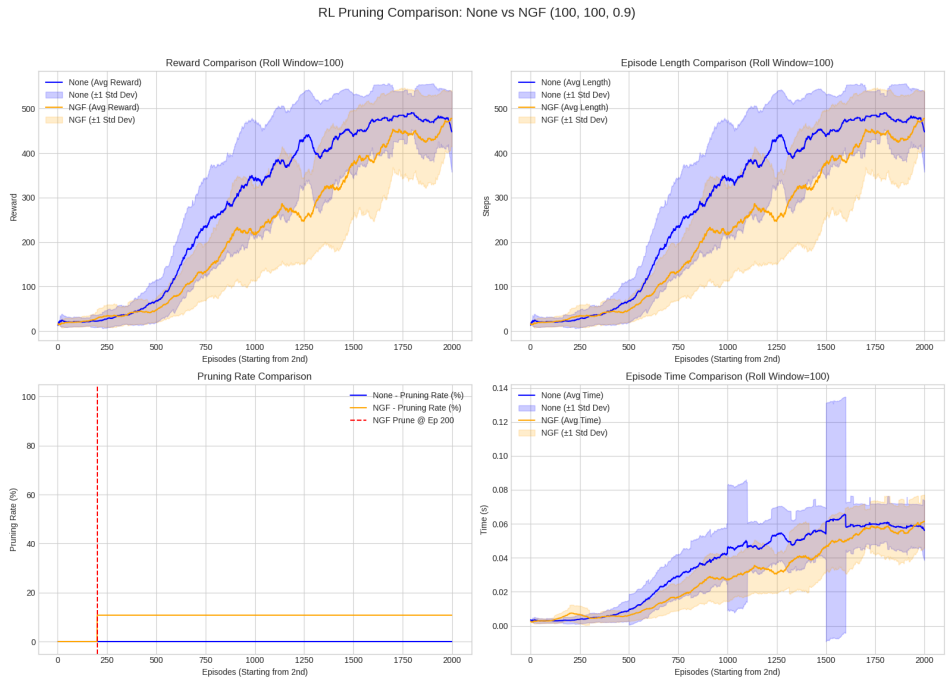

Figure 6: Reinforcement learning on CartPole-v1: Comparison of baseline vs. NGF pruning (`keep_fraction=0.9`). Shows reward/episode length (top), pruning rate (bottom left, post-critical period), and episode computation time (bottom right).

pruning offered a more balanced trade-off, providing speedup over baseline with less performance degradation than many NGF configurations.

Structural Impact: The two pruning methods target different parts of the network structure in LeNet-5 (Figure 4). Magnitude pruning focused on the central layers, while NGF pruning preferentially removed neurons from the input and output layers. This suggests that weight magnitude identifies redundancy differently than neuron activation magnitude. The functional significance of removing early-layer sensory processing neurons versus later-layer integrative neurons under NGF pruning warrants further study.

Reinforcement Learning Potential: The results on CartPole-v1 (Figure 6), while preliminary (testing only one NGF setting), suggest a potentially different dynamic in RL. Although initial reward gain was slower, the NGF-pruned agent exhibited more linear learning and significantly faster computation per episode. This faster episode time, combined with steady learning, might indicate that NGF pruning helps discover a simpler, more generalizable policy faster in terms of wall-clock time, even if asymptotic reward is similar or slightly lower initially. The stability and efficiency observed suggest NGF pruning could be particularly well-suited for RL scenarios where computational resources or interaction time are constrained.

Limitations and Future Work: This study used a single-shot pruning approach. Iterative NGF pruning might yield better results by allowing the network to recover between pruning steps. The activation metric used (average absolute activation) is simple; exploring other statistics (e.g., variance, frequency) could be beneficial. The experiments were limited to relatively simple architectures and datasets. Thus, evaluation on larger models and more complex tasks is necessary to validate the dynamics of NGF-Prune. Further exploration in RL is necessary, including varying hyperparameters, testing on more complex environments, and directly comparing against magnitude pruning in the RL context. Finally, understanding the reasons why NGF and magnitude pruning affect different network layers could prove valuable.

Use of Language Models Throughout this project, Large Language Models (LLMs) served as assistants for various tasks. In software development, they aided in generating visualization code, debugging errors, and refining code design and documentation. For manuscript preparation, LLM contributions were editorial, focusing on improving clarity, grammar, and overall structure. LLMs also assisted in compiling and summarizing information for the model card and datasheet, using browsing capabilities to gather external resources related to the FashionMNIST dataset.

References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017.

[2] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of Machine Learning and Systems*, 2:129–146, 2020.

[3] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1135–1143, 2015.

[4] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research (JMLR)*, 22(241):1–124, 2021.

[5] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 2, pages 598–605, 1989.

[6] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[7] Rita Levi-Montalcini. The nerve growth factor 35 years later. *Science*, 237(4819):1154–1162, 1987.

[8] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11264–11272, 2019.

[9] Dale Purves. *Body and Brain: A Trophic Theory of Neural Connections*. Harvard University Press, 1988.

[10] Matt Towers, Jordan Kunzmann, Geoffrey Raposo, Antonin Raffin, Maximilian Ernestus, et al. Gymnasium: An API standard for single-agent reinforcement learning environments. *NeurIPS 2023 Datasets and Benchmarks Track*, 2023.

[11] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[12] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A Model Card for NGF-Inspired Neuron Pruning

This model card describes an exploration of a neuron pruning technique inspired by Neurotrophic Growth Factors (NGF), comparing it against baseline (no pruning) and standard magnitude pruning on MLP and LeNet-5 architectures for supervised learning (FashionMNIST) and reinforcement learning (CartPole-v1) tasks.

A.1 Model Details

A.1.1 Model Description

This work investigates a novel neuron pruning method where neurons are removed based on their average absolute activation during a defined "critical period" of training. This is inspired by biological processes where less active neurons are eliminated during development. The technique was applied in a single-shot manner after the critical period to MLP and LeNet-5 models trained on FashionMNIST and a Policy MLP trained on CartPole-v1. Key findings include a sharp performance threshold in supervised learning (~15% neuron removal), significant training time reduction often coupled with higher final test loss (SL), preferential pruning of input/output layers (LeNet-5), and promising stability and wall-clock time efficiency in preliminary RL experiments.

Developed by: Julian Weaver

Model type: Neural Network Pruning Technique (applied to MLP, LeNet-5, Policy MLP)

A.2 Uses

A.2.1 Direct Use

Research and experimentation to understand the characteristics, efficiency, and performance trade-offs of NGF-inspired activation-based neuron pruning compared to other methods like magnitude pruning. Analysis of model behavior under different pruning thresholds and critical period settings.

A.2.2 Downstream Use

Potential application in scenarios prioritizing training or inference speed, especially where a slight reduction in peak performance is acceptable (e.g., certain RL tasks, resource-constrained deployments). Requires further validation for specific use cases.

A.2.3 Out-of-Scope Use

- Deployment in safety-critical systems without extensive validation due to observed performance cliffs and hyperparameter sensitivity.
- Direct application to significantly different model architectures (e.g., Transformers) or data modalities without re-evaluation.
- Assuming robustness or fairness properties without specific testing, especially after pruning.

A.3 Bias, Risks, and Limitations

- **Bias:** The FashionMNIST dataset may primarily reflect European fashion, potentially leading to biased model performance on underrepresented clothing styles. The pruning technique's impact on fairness across subgroups was not evaluated.
- **Risks:**
  - **Reliability:** NGF pruning showed a sharp performance cliff in SL experiments; pruning beyond ~15% led to rapid degradation and instability. Applying the technique requires careful threshold tuning to avoid unexpected failures.
  - **Hyperparameter Sensitivity:** Performance, especially beyond the threshold, is highly sensitive to the critical period timing and pruning percentage.
- **Limitations:**
  - Single-shot pruning was used; iterative pruning might yield better results.
  - Only average absolute activation was used as a metric; other statistics could be explored.

- Limited evaluation scope (simple datasets/architectures).
- RL evaluation was preliminary (single NGF setting).
- Theoretical understanding of layer-specific effects is incomplete.

A.3.1 Recommendations

Users (both direct and downstream) should be made aware of the risks, biases and limitations of the model. Specifically: * Carefully tune the pruning threshold and critical period hyperparameters, benchmarking performance rigorously. * Be aware of the potential performance cliff associated with NGF pruning compared to the more gradual degradation of magnitude pruning (in SL). * Evaluate fairness implications if used in applications affecting people. * Conduct further testing for specific downstream tasks and architectures before deployment.

A.4 Training Details

A.4.1 Training Data

- **FashionMNIST:** The standard training split provided by `torchvision.datasets.FashionMNIST` (with `train=True`) was used.
- **CartPole-v1:** Training data was generated through agent interaction with the standard `CartPole-v1` environment from the Gymnasium library, using its default dynamics.

A.4.2 Training Procedure

Preprocessing

- **FashionMNIST:** Images converted to PyTorch Tensors (scaling pixels to [0.0, 1.0]), then normalized using approximate dataset mean (0.2860) and std dev (0.3530).
- **CartPole-v1:** Raw 4-dimensional state vector (cart position, cart velocity, pole angle, pole angular velocity) used directly as input.

Speeds, Sizes, Times

NGF pruning consistently reduced training time (wall-clock) compared to baseline, significantly in some cases (e.g., Figure \ref{fig:mlp*perf*comparison}). RL episodes were computationally faster with NGF pruning (Figure \ref{fig:rl_comparison}). Precise speedups depend on configuration and hardware.

A.5 Evaluation

A.5.1 Testing Data, Factors & Metrics

Testing Data

- **FashionMNIST:** Standard test split from `torchvision.datasets.FashionMNIST` (`train=False`).
- **CartPole-v1:** Performance evaluated during training interaction episodes within the standard environment.

Factors

- Pruning Method (NGF, Magnitude, None)
- Pruning Threshold / Keep Fraction
- Critical Period Timing (Start/Duration - Epoch for SL, Episode for RL)
- Model Architecture (MLP, LeNet-5, Policy MLP)
- Random Seed

Metrics

- **Supervised Learning (FashionMNIST):** Final Test Accuracy, Final Test Loss, Training Time, Layer-wise Sparsity.

- **Reinforcement Learning (CartPole-v1):** Reward per Episode, Computation Time per Episode, Learning Curve Stability (Avg. reward over window).

A.5.2 Results

- **SL Performance Threshold:** Minimal accuracy impact up to ~15% NGF neuron pruning (FashionMNIST MLP/LeNet), then sharp decline.
- **SL Efficiency:** NGF reduced training time but often worsened final test loss vs. baseline/magnitude.
- **Structural Impact (LeNet):** NGF pruned input/output layers more; Magnitude pruned central layers more.
- **RL Potential:** Preliminary results show slower initial reward gain but more linear learning and faster episode computation time with NGF. Potential for faster policy discovery (wall-clock time).

Summary

NGF-inspired pruning shows potential for computational efficiency, especially in RL, but exhibits sharp performance thresholds and hyperparameter sensitivity in supervised learning compared to magnitude pruning. It impacts network structure differently than weight magnitude pruning. Further research into iterative application, metrics, and complex tasks is needed.

A.6 Technical Specifications

A.6.1 Model Architecture and Objective

- **Architectures:** Simple MLP (784-512-256-10), LeNet-5 (Conv/Pool/Conv/Pool/FC/FC/FC), Policy MLP (4-128-2).
- **Objective:** Demonstrate and compare NGF-inspired pruning against baselines on image classification (cross-entropy loss) and reinforcement learning (policy gradient / maximizing cumulative reward).

A.6.2 Compute Infrastructure

Software

PyTorch, Gymnasium, torchvision, numpy, pandas, tqdm, imageio [Potentially others - More Information Needed]

A.7 Glossary

- **NGF:** Neurotrophic Growth Factor. Biological molecules supporting neuron survival; inspiration for the pruning approach.
- **Neuron Pruning:** Removing entire neurons (units) from a network.
- **Magnitude Pruning:** Removing individual weights based on their magnitude (absolute value).
- **Critical Period:** A specific phase during training where neuron activations are measured to decide which neurons to prune.
- **Single-shot Pruning:** Pruning performed once during training.

B Datasheet

FashionMNIST Datasheet Summary

- **Motivation**:
  - Developed by Zalando Research (Han Xiao, Kashif Rasul, Roland Vollgraf) in 2017 specifically as a benchmark replacement for the original MNIST dataset.
  - MNIST had become "too easy" for modern ML algorithms (achieving >99.7% accuracy), limiting its usefulness for differentiating models. Fashion-MNIST provides a greater challenge while retaining MNIST's format (28x28 grayscale images, 10 classes, 60k train/10k test split) for easy adoption.

- **Composition**:
  - Consists of 70,000 total 28x28 pixel grayscale images of fashion items.
  - Split into a 60,000-image training set and a 10,000-image test set (6,000 train / 1,000 test images per class).
  - Features 10 balanced classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. Labels derived from internal Zalando "silhouette codes" assigned by fashion experts.
- **Collection Process**:
  - Images were sampled from Zalando's existing database of "front look thumbnail" product images.
  - Products appearing predominantly white were excluded due to low contrast issues.
  - Sampling aimed for class balance, drawing from men's, women's, and kids' items available pre-2017.
- **Preprocessing**:
  - A specific pipeline was applied to source images: format conversion, edge trimming, resizing (longest edge to 28px), sharpening, padding/centering to 28x28, negating pixel intensities (to match MNIST's dark-on-light convention where high values are darker), and converting to 8-bit grayscale.
  - The low 28x28 resolution is a key characteristic contributing to classification difficulty (human accuracy ~83.5%). The raw, higher-resolution source images are not distributed.
- **Uses**:
  - Primarily used for benchmarking image classification algorithms, comparing CNNs, ViTs, and other models. Common in academic research, ML tutorials, and prototyping.
  - Also utilized for evaluating generative models (AEs, VAEs), representation learning, anomaly detection, federated learning, and robustness studies.
  - **Limitations**: Low resolution hinders real-world fidelity; potential bias from single-source (Zalando, pre-2017, European focus, white items excluded); unsuitable for fine-grained attribute recognition or tasks needing demographic data (which isn't included).
- **Distribution**:
  - Publicly released under the MIT License, allowing free use, modification, and redistribution.
  - Available via the official GitHub repository (`zalandoresearch/fashion-mnist`), direct download links, and integrated within major ML libraries (TensorFlow, PyTorch, etc.) for easy access.
- **Maintenance**:
  - Maintained by the original creators (Zalando Research) through the official GitHub repository.
  - Treated as a static dataset (no new versions planned). The community can contribute benchmark results by submitting issues to the GitHub repository. Its integration into ML libraries ensures continued practical availability.