NVP: Neural Vitality based Dynamic Sparsity Pruning in Convolutional Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

As neural networks grow larger, improving computational efficiency without sacrificing accuracy is increasingly important for sustainable deep learning. We study this problem through the lens of dying neurons, whose activations show low variability and limited expressiveness. We introduce neuron entropy, a data-dependent measure of activation variability that quantifies each neuron's contribution to network expressiveness. Combining this measure with weight magnitude, we further define the Neural Vitality Index (NVI) to assess neuron-level importance. Based on NVI, we propose Neural Vitality Pruning (NVP), a dynamic pruning framework that adaptively sparsifies convolutional neural networks during training by removing low-vitality neurons. Experiments across representative CNN architectures and competitive pruning baselines show that NVP achieves higher sparsity while maintaining or improving predictive performance. These results suggest neuron vitality as a principled criterion for efficient neural network optimization.

1 Introduction

The energy consumption of AI technologies is escalating rapidly, with data centers projected to consume up to 17% of U.S. electricity generation by 2030 (WSJ), more than double the current usage. This surge is primarily driven by the continual expansion of both data scale and model complexity Crawford (2024), and convolutional neural networks (CNNs) Lee (2023) are among the most computationally intensive AI models. This necessitates innovative strategies that adhere to net-zero emission targets. One such promising strategy is the systematic pruning of neural networks, which effectively reduces the number of parameters, curtails memory consumption, and minimizes computational overhead. This process not only accelerates inference times and lowers energy usage Hussien (2024), but also facilitates deployment on devices with limited computational resources, improves model generalization, and enhances overall hardware performance. As a result, network pruning stands out as a pivotal technique for fostering sustainable and scalable AI systems in an environmentally responsible manner.

Over the past few years, there has been a surge of research focused on refining pruning techniques to optimize the performance of deep neural networks across various applications. CNNs have been the focus of these innovations primarily Soudy (2024); Wang (2018). More recently, these techniques have been extended to other neural network architectures, such as spiking neural networks (SNNs) Chakraborty (2024); Shi (2024) and Transformers Meng (2025); Bai (2025). These models differ inherently in their computational underpinnings from CNNs; Transformers leverage sophisticated attention mechanisms, while SNNs incorporate biologically inspired, event-driven processes that distinguish them from traditional artificial neural networks (ANNs). In this work, we focus on CNNs, which remain a robust benchmark due to the extensive suite of pruning strategies developed for them.

Recent advances in network pruning have explored static and dynamic methods to reduce the computational burden of deep CNNs while maintaining accuracy. Static pruning removes neurons based on sensitivity, magnitude, or entropy during training, yielding a fixed sparse subnetwork Liao (2023); Pham (2023) but lacking adaptability to varying inputs. In contrast, dynamic pruning adjusts the network structure at inference or training, improving efficiency and generalization Huang (2024); Jianjian (2024). Despite its

advantages, dynamic pruning remains underexplored due to its complexity. To address this, we propose Neural Vitality Pruning (NVP), a dynamic strategy that selectively removes low-vitality neurons during optimization, balancing efficiency and adaptability for sustainable deep learning.

Neurons with persistently low activation, often termed "dying neurons" in neural plasticity Abbas (2023), degrade network performance and adaptability Lyle (2023), particularly in dynamic tasks. To mitigate activation saturation, researchers have explored alternative activation functions like Leaky ReLU Maas (2013), Parametric ReLU Wang (2020), ELU Kim (2020), Swish Ramachandran (2017), and GELU Chelly (2024). Building on this, our NVP method leverages low-activation neurons, those with minimal activation variability, to guide network sparsification, enhancing both efficiency and adaptability.

In this work, we propose the Neural Vitality Pruning (NVP) method, centered around a novel Neural Vitality Index (NVI) that effectively integrates neuron activation entropy and weight norm to quantify neuron vitality. Unlike prior pruning approaches that rely solely on weight magnitude or heuristic criteria, NVP dynamically identifies and eliminates low-vitality neurons during training, thereby improving model efficiency and accelerating convergence without sacrificing accuracy. Our method's key novelty lies in leveraging the complementary information of neuron activation distribution and parameter magnitude through a unified metric, enabling fine-grained, neuron-level pruning decisions that reflect both functional contribution and representational diversity. We extensively evaluate NVP on a diverse set of CNN architectures and datasets, covering models such as ResNet (18, 20, 32, 56), VGG-16, MLP-Net, and DenseNet-121 in Appendix, evaluated on standard methods including MNIST, CIFAR-10, and Tiny-ImageNet. Across all these settings, NVP consistently outperforms more than twenty SOTA pruning methods, achieving higher test accuracy at comparable or higher sparsity levels.

2 Related Work

2.1 Towards Network Compression

Model compression is a key area in deep learning, especially for deploying large models on resource-constrained devices. The goal is to reduce computational and memory costs while maintaining accuracy. Techniques include *pruning*, which removes redundant weights or neurons, with structured pruning targeting filters or channels for better hardware compatibility Blalock (2020); Lemaire (2019). *Quantization* reduces weight and activation precision, typically from 32-bit to 8-bit, using post-training or quantization-aware methods to maintain accuracy Jacob (2018). *Knowledge distillation* trains smaller models to replicate larger ones, transferring knowledge with minimal performance loss Tian (2022); Jianping (2021). *Neural Architecture Search* (NAS) automates efficient model design with multi-objective optimization for accuracy and speed, and advancements like differentiable NAS reduce search costs for hardware-constrained deployment Radosavovic (2020). These techniques drive significant improvements in compression, enabling practical deep learning applications Liu (2019). Recent advances in neural network pruning have made it a key method for reducing the computational load of deep CNNs while maintaining strong performance. Pruning methods are broadly categorized into two paradigms Gao (2024): static and dynamic pruning. Dynamic pruning offers significant advantages over static pruning, motivating its adoption in this work Wright (2025). First, dynamic pruning adjusts the pruning strategy during inference based on input characteristics, preserving essential features while eliminating less important computations Gongfan (2023). This adaptability enhances inference speed and improves resource utilization without compromising accuracy Wenyu (2022). Second, while static pruning applies a fixed mask post-training, dynamic approaches continuously adapt to shifts in data distribution, ensuring stable performance across diverse scenarios Huang (2024). Third, by dynamically regulating pruning granularity, models can achieve a superior trade-off between latency, memory footprint, and accuracy; critical for real-time applications Spadaro (2023). Moreover, advanced techniques incorporating differentiable mechanisms and reinforcement learning have demonstrated improved robustness and generalization over static methods Pham (2023); Verdenius (2020). Recent studies on dynamic structure pruning indicate that modulating pruning granularity throughout training can yield SOTA compression rates with minimal performance degradation Park (2023). Additionally, research on dynamic data pruning, such as for automatic speech recognition, shows that selectively retaining relevant training samples accelerates training while maintaining performance Rachwan (2022). Finally, explorations into unstructured pruning offer further

insights into sparsification and computational efficiency Liao (2023). Collectively, these findings highlight that dynamic pruning enhances efficiency, adaptability, and robustness, underscoring the importance of advancing dynamic pruning methodologies. Our method, NVP, contributes to this line of work by introducing a new dynamic pruning criterion based on neuron vitality, calculated jointly from entropy and weight magnitude, and applied iteratively during training. Unlike previous methods that rely on static masks or learned gates, NVP is motivated by the phenomenon of dying neurons and uses this insight to identify and eliminate low-vitality neurons, providing a lightweight and interpretable alternative. This view is consistent with the idea of neural differentiation in deep networks, where diverse and specialized neuron roles are closely related to model expressivity and representational diversity, further motivating NVP to preserve vital neurons while removing less informative ones Wang & Jiang (2026).

2.2 Understanding Dying Neurons

The issue of "dying neurons," Lu (2020) or redundant/inactive neurons, in neural networks (especially those using ReLU) has received significant attention in recent research Sokar (2023); Abbas (2023). Dying neurons output zero consistently, ceasing to contribute to the network's learning process. This problem occurs when a ReLU neuron's input remains negative across all data points, leading to zero activations and gradients, preventing weight updates and hindering recovery. To mitigate this, various strategies have been proposed, such as modifying the activation function to allow a small, non-zero gradient when inactive. For instance, Leaky ReLU introduces a slight negative slope for negative inputs, enabling continued learning even with non-positive activations Maas (2013). Neurons consistently receiving negative inputs face a similar issue, as their weights and biases cannot be updated due to zero gradients, resulting in "death" where these neurons never contribute to the learning process Jiang (2023); Lu (2020). Beyond activation-function-based remedies, DNR-Pruning further links dying neuron reactivation with sparsity-aware pruning in convolutional neural networks Wang & Jiang (2025). While prior work focuses on either preventing or recovering from neuron death through architectural or functional modifications, our approach differs by actively identifying and removing dying neurons through a quantifiable vitality index. Rather than attempting to revive non-contributing units, we leverage their activation entropy and weight magnitude as pruning signals, offering a principled and efficiency-driven perspective on neuron death.

3 Methods

In this section, we present our novel *Neural Vitality Pruning* strategy. Motivated by the phenomenon of dying neurons, we introduce the new Neural Vitality Index, which combines neuron activation entropy and weight norm. NVP leverages this NVI through a weighted and hybrid formulation tailored for both convolutional and linear layers, providing a principled metric for assessing neuron vitality and guiding the pruning process.

3.1 Neuron Entropy: A Key to Expressivity

Definition 1 (**Neuron Entropy**). *Let $\mathcal{N}$ be a network. The entropy of a neuron (i,j) in $\mathcal{N}$ for a data distribution $\mathcal{D}$ is defined as the entropy of the distribution of the activation status for (i,j) given input $(\mathbf{x},y) \in \mathcal{D}$:*

$$\mathcal{E}_j^{(i)}(\mathcal{N},\mathcal{D}) = -\sum_{k=0}^{q} p(\hat{a}_j^{(i)}(\mathbf{x}) = k) \log(p(\hat{a}_j^{(i)}(\mathbf{x}))). \tag{1}$$

The neuron entropy describes the uncertainty of the neuron activation zone. For instance, let $\mathcal{N}$ be a neural network with a ReLU activation function. If $\mathcal{E}_j^{(i)}(\mathcal{N},\mathcal{D}) = 0$, this suggests that the activation zone of neuron (i,j) is identical for any data $(\mathbf{x},y) \sim \mathcal{D}$. This implies that neuron (i,j) fails to provide non-linearity, behaving as a static neuron.

By introducing neuron entropy as a measure, we gain a better understanding of each neuron's contribution to the network's expressive power and their role in enhancing the network's overall non-linear capabilities. As neuron entropy increases, the neural network becomes more capable of capturing complex zones in the data, thereby improving its overall performance.

Figure 1 shows the neuron entropy of the neuron $\hat{a}_j^{(i)}$ where the x-axis is the probability of the zone of $\hat{a}_j^{(i)}$ is 0: $P(\hat{a}_j^{(i)} = 0)$. For activation with two zones (such as ReLU, PReLU), neuron entropy arrives its maximum around 0.89 when $P(\hat{a}_j^{(i)}(\mathbf{x}) = 0) = 1/2$. For activation with 3 zones (such as Sigmoid, ReLU-6), the maximum entropy is 1.59 when $P(\hat{a}_j^{(i)}(\mathbf{x}) = 0) = 1/3$. For activation with 4 zones (such as Swish, Piecewise Linear), the maximum entropy is 1.89 when $P(\hat{a}_j^{(i)}(\mathbf{x}) = 0) = 1/4$. For activation with 5 zones (such as Extended ReLU, Custom Piecewise Function), the maximum entropy is 2.39 when $P(\hat{a}_j^{(i)}(\mathbf{x}) = 0) = 1/5$. And for activation with 6 zones (such as Multi-Slope ReLU, Complex Sigmoid-based Function), the maximum entropy is 2.59 when $P(\hat{a}_j^{(i)}(\mathbf{x}) = 0) = 1/6$.

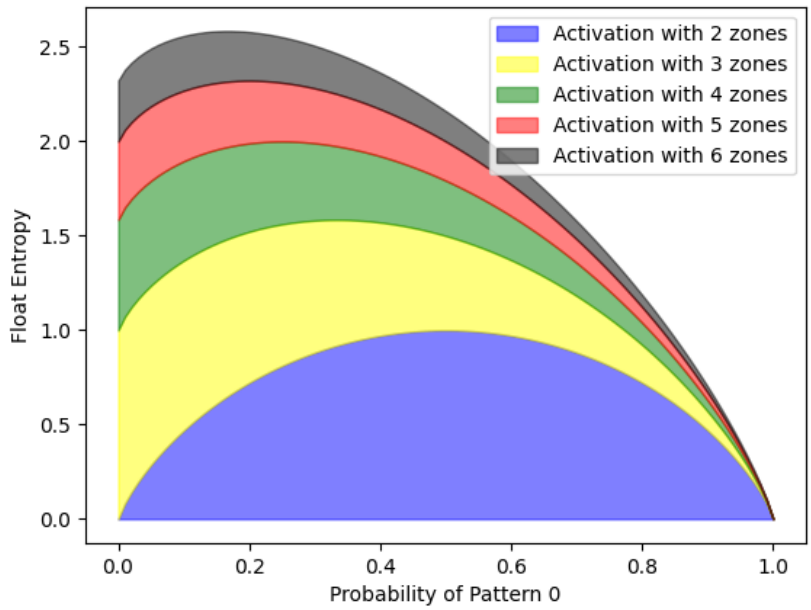


Figure 1: The dynamic entropy of activation functions with k different patterns given the probability of pattern is 0 (x-axis). Each line shows the maximum value of entropy, while the shadowed zone is the range of entropy from $k = n$ to $k = n - 1$.

3.2 Neural Vitality Index

Definition 2 (Neuron Activation Vector and Activation Entropy)**.** *Let a neuron n produce a vector of activations over a mini-batch of d inputs,*

$$\mathbf{z}^n = \begin{bmatrix} z_1^n \\ z_2^n \\ \vdots \\ z_d^n \end{bmatrix} \in \mathbb{R}^d. \tag{2}$$

We define the smoothed activation probabilities as

$$p_i^n = \frac{|z_i^n| + \epsilon}{\sum_{j=1}^{d} \left(|z_j^n| + \epsilon\right)}, \quad i = 1, 2, \ldots, d, \tag{3}$$

where $\epsilon > 0$ is a small constant ensuring numerical stability. The activation entropy *of neuron n is then given by*

$$\mathcal{E}(n) = -\sum_{i=1}^{d} p_i^n \log p_i^n. \tag{4}$$

A high value of $\mathcal{E}(n)$ indicates that neuron n exhibits diverse activations across inputs, whereas a low value suggests that the neuron is nearly constant (i.e., a "dying" neuron).

Definition 3 (Neuron Weight Vector and Weight Norm)**.** *Let the weight vector associated with neuron n be*

$$\mathbf{w}^n = \begin{bmatrix} w_1^n \\ w_2^n \\ \vdots \\ w_m^n \end{bmatrix} \in \mathbb{R}^m, \tag{5}$$

where m is the number of parameters connected to neuron n. Its ℓ_2 norm is defined as

$$\|\mathbf{w}^n\|_2 = \sqrt{\sum_{j=1}^{m} \left(w_j^n\right)^2}. \tag{6}$$

A larger $\|\mathbf{w}^n\|_2$ implies that neuron n has a stronger influence in the network's computations.

Definition 4 (Neural Vitality Index (NVI))**.** *We define the* Neural Vitality Index *for neuron n as*

$$\mathrm{NVI1}(n) = \mathcal{E}(n) \cdot \|\mathbf{w}^n\|_2. \tag{7}$$

Alternatively, to balance the contributions of activation entropy and weight norm more symmetrically, one may define

$$\mathrm{NVI2}(n) = \frac{2}{\frac{1}{\mathcal{E}(n)} + \frac{1}{\|\mathbf{w}^n\|_2}}. \tag{8}$$

In both formulations, a high $\mathrm{NVI}(n)$ indicates that neuron n is both dynamically active and possesses significant parameter magnitude, while a low $\mathrm{NVI}(n)$ identifies it as a candidate for pruning.

Lemma 1 (Lower Bound on Neuron Activation Entropy)**.** *Assume that, due to the smoothing parameter ϵ, the activation probabilities satisfy $p_i^n \geq \delta$ for all i and some $\delta > 0$ with $\delta \leq \frac{1}{d}$. Then, the activation entropy of neuron n is bounded below by the minimum entropy achieved when one component is $1-(d-1)\delta$ and the remaining components are δ:*

$$\mathcal{E}(n) \geq -\left[(1-(d-1)\delta)\log(1-(d-1)\delta) + (d-1)\delta\log\delta\right]. \tag{9}$$

Moreover, this lower bound is positive and approaches 0 as $\delta \to 0^+$.

Lemma 2 (Monotonicity of the Neural Vitality Index)**.** *Let n be a neuron with activation entropy $\mathcal{E}(n) > 0$ and weight norm $\|\mathbf{w}^n\|_2 > 0$. Consider the NVI defined either as the multiplicative form $\mathrm{NVI}(n) = \mathcal{E}(n) \cdot \|\mathbf{w}^n\|_2$ (NVP-1), or as the harmonic mean form $\mathrm{NVI}(n) = \frac{2}{\frac{1}{\mathcal{E}(n)} + \frac{1}{\|\mathbf{w}^n\|_2}}$ (NVP-2). In either case, the index $\mathrm{NVI}(n)$ is strictly monotonically increasing with respect to both $\mathcal{E}(n)$ and $\|\mathbf{w}^n\|_2$. That is, let $\mathrm{NVI}_1(n)$ and $\mathrm{NVI}_2(n)$ denote the NVI values evaluated at $(\mathcal{E}_1(n), \|\mathbf{w}^n\|_2)$ and $(\mathcal{E}_2(n), \|\mathbf{w}^n\|_2)$, respectively, with $\mathcal{E}_1(n) > \mathcal{E}_2(n)$, then $\mathrm{NVI}_1(n) > \mathrm{NVI}_2(n)$. Similarly, if $\mathrm{NVI}_1(n)$ and $\mathrm{NVI}_2(n)$ are computed at $(\mathcal{E}(n), \|\mathbf{w}_1^n\|_2)$ and $(\mathcal{E}(n), \|\mathbf{w}_2^n\|_2)$ with $\|\mathbf{w}_1^n\|_2 > \|\mathbf{w}_2^n\|_2$, then $\mathrm{NVI}_1(n) > \mathrm{NVI}_2(n)$. This monotonicity property ensures that neurons with either stronger activation diversity or larger weight magnitude are assigned higher importance under both NVP-1 and NVP-2 formulations.*

3.3 Our Neural Vitality Pruning Method

In NVP, the NVI concept is extended by incorporating weighted entropy measures and hybrid norms, which are tailored for both convolutional and linear layers.

3.3.1 NVI in Convolutional Layers

For a convolutional layer, consider a filter F_i with C channels. Instead of computing a single entropy value, we compute a *weighted entropy* for the filter by aggregating the weighted entropies of its individual channels. Let F_i^c denote the c-th channel of filter F_i, and define the entropy of F_i^c as

$$\mathcal{E}_c^{(i)} \triangleq \mathcal{E}_c^{(i)}(\mathcal{N}, \mathcal{D}), \tag{10}$$

using the same notation as in Definition 1. Then, the weighted entropy for filter F_i is given by

$$\mathcal{E}^{(i)}_{\text{weighted}}(F_i) = \sum_{c=1}^{C} \alpha_c \, \mathcal{E}^{(i)}_c, \tag{11}$$

where α_c is a weight reflecting the significance of the c-th channel in contributing to the overall filter response. In our implementation, α_c is the normalized average activation magnitude of channel c across the dataset:

$$\alpha_c = \frac{\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left\|a^{(i)}_c(\mathbf{x})\right\|_1\right]}{\sum_{c'=1}^{C} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\left\|a^{(i)}_{c'}(\mathbf{x})\right\|_1\right]}, \tag{12}$$

where $a^{(i)}_c(\mathbf{x})$ is the activation map of channel c in filter F_i.

This weighted entropy is then combined with a *hybrid norm* (typically an L_2 norm, possibly adjusted dynamically) to form the pruning vitality index, in terms of the NVI, for filter F_i:

$$\text{NVI}(F_i) = f\Big(\mathcal{E}^{(i)}_{\text{weighted}}(F_i),\ \|F_i\|_{\text{hybrid}}\Big), \tag{13}$$

where $f(\cdot)$ is a non-linear function (e.g., a product or an exponential function) that fuses the two metrics.

3.3.2 NVI in Linear Layers

For linear layers, we extend the approach by combining local entropy measures for individual output units with a global aggregation. Define the *local entropy* for the j-th output unit O_j as

$$\mathcal{E}^{(j)}(O_j) \triangleq \mathcal{E}^{(j)}(\mathcal{N}, \mathcal{D}), \tag{14}$$

computed in the same manner as the neuron entropy in Definition 1. For an output layer O with M units, the *global entropy* is given by

$$\mathcal{E}^{\text{G}}(O) = g\left(\sum_{j=1}^{M} h(p_j)\, \mathcal{E}^{(j)}(O_j)\right), \tag{15}$$

where:

- p_j denotes the probability associated with the j-th output,
- $h(\cdot)$ is a weighting function that adjusts the contribution of each unit's local entropy,
- $g(\cdot)$ is an optional nonlinear transformation applied after summation (e.g., squashing or normalization).

This global entropy is then fused with a dynamic hybrid norm of the weight matrix W of the linear layer to yield the NVI:

$$\text{NVI}(W) = f\Big(\mathcal{E}^{\text{G}}(O),\ \|W\|_{\text{hybrid}}\Big), \tag{16}$$

where $\|W\|_{\text{hybrid}}$ may change during training, and the function $f(\cdot)$ is chosen similarly as in the convolutional case.

3.3.3 Pruning Strategy

Let $\mathcal{U}$ denote the set of neurons in a particular layer. We define the set of neurons to be pruned as

$$\mathcal{P} = \{n \in \mathcal{U} \mid \text{NVI}(n) < \tau\}, \tag{17}$$

where τ is a threshold determined, for example, by the s-percentile of the NVI distribution:

$$\tau = \text{Quantile}\left(\{\text{NVI}(n) : n \in \mathcal{U}\},\ s\right), \quad s \in (0, 1). \tag{18}$$

Neurons in $\mathcal{P}$ are pruned by zeroing their weight vectors:

$$\mathbf{w}^n \leftarrow \mathbf{0}, \quad \forall n \in \mathcal{P}. \tag{19}$$

This pruning operation removes neurons that are both inactive (low $\mathcal{E}(n)$) and have minimal influence (small $\|\mathbf{w}^n\|_2$), thus reducing model complexity while preserving performance.

3.3.4 NVP Algorithm

As discussed above, we propose the NVI to quantify the representational capacity of individual neurons. Empirically, we observe that a subset of neurons, termed *dying neurons*, fail to contribute meaningfully to network predictions. Motivated by this, we introduce the NVP method, which removes such uninformative neurons by integrating weight magnitude and activation entropy into a unified pruning criterion. The complete training and pruning procedure is detailed in Algorithm 1.

At the beginning of training, a binary mask M is initialized with all entries set to one, matching the shape of the model parameters θ. This mask is used to deactivate pruned neurons during forward and backward passes, where $M(n) = 1$ denotes an active neuron. At each iteration, the network computes predictions $\hat{y}$ using masked weights $\theta \odot M$, and updates θ via gradient descent to minimize the loss $\mathcal{L}(\hat{y}, y)$. At the end of each epoch, if a pruning condition is satisfied, the PRUNE subroutine is invoked to update M based on neuron-wise vitality indices. Each neuron's vitality is estimated by combining two complementary metrics: the activation entropy $\mathcal{E}(n) = -\sum_{i=1}^{d} p_i^n \log p_i^n$, where p_i^n denotes the normalized activation histogram over d bins, and the ℓ_2 norm of its incoming weights, $\|\mathbf{w}^n\|_2$. These two metrics are combined into a unified NVI, which can be defined either as the product of activation entropy and weight norm, $\text{NVI}(n) = \mathcal{E}(n) \cdot \|\mathbf{w}^n\|_2$ (NVP-1), or alternatively as their harmonic mean, $\text{NVI}(n) = \left(\frac{1}{\frac{1}{\mathcal{E}(n)+\epsilon} + \frac{1}{\|\mathbf{w}^n\|_2+\delta}}\right)$ (NVP-2), where ϵ and δ are small constants for numerical stability. The user may select either definition depending on the desired pruning behavior. To enforce a global sparsity constraint S, a threshold τ is computed for each layer via the S-th percentile of the NVI. Neurons with $\text{NVI}(n) < \tau$ are pruned by zeroing their weights or setting $M(n) = 0$. This procedure iteratively eliminates neurons that are both inactive and low-magnitude, thereby improving computational efficiency while preserving model expressiveness.

For clarity, we separate the training and pruning phases in Algorithm 1. In practice, however, the computation of neuron activation entropy can be incorporated into the training phase, with the zone counter updates being performed on a batch-wise basis. In Section 4, we explore the impact of different pruning criteria on the performance of pruning methods.

3.4 Neuron Insights in Neural Networks

3.4.1 Model Representational Capacity

Figure 2 illustrates the neuron activation entropy in VGG-16 networks trained on the CIFAR-10 dataset using different batch sizes. Each model underwent 700 training epochs, with neuron activation entropy evaluated at each epoch. The ReLU activation function, with a separation of 0, is applied to all networks, resulting in neurons being either activated or not. Figures 2d and 2e display the training and test accuracy across different models, demonstrating that smaller batch sizes typically yield higher accuracy and more stable performance on both training and testing datasets. To understand the disparity in performance from the perspective of neuron stability, Figures 2a through 2c present the average and variance of neuron activation entropy across different layers.

The shallow layer (layer 1) exhibits an activation entropy mean of approximately 4.36, with relatively small differences among the four batch sizes. Notably, the small batch size (64) shows slightly larger variance mid-training, indicating more pronounced fluctuations. Moving to the middle layer (layer 6), the mean activation entropy increases to around 10.756. Here, large batch sizes (256 and 512) generally maintain higher mean activation entropy, whereas the small batch size (64) reveals a clearer downward trend. In the deeper layer (layer 11), this effect becomes even more substantial: a larger batch size (512) achieves the highest mean activation entropy in the later training stages. Variance results similarly indicate that smaller batches

Algorithm 1 NVP Training Algorithm

Input: Training data $\mathcal{D}_{\text{train}}$, Validation data $\mathcal{D}_{\text{val}}$, Learning rate η, Sparsity level S, Number of epochs E, Pruning mode $\texttt{mode} \in \{\texttt{NVP-1}, \texttt{NVP-2}\}$

function TRAIN($f, \theta, \texttt{mode}$)
  Initialize mask $M \leftarrow 1$
  for epoch $= 1$ to E **do**
    for each $(\mathbf{x}, y) \in \mathcal{D}_{\text{train}}$ **do**
      $\hat{y} \leftarrow f(\mathbf{x}; M \odot \theta)$ ▷ Forward pass
      Compute loss: $L = \mathcal{L}(\hat{y}, y)$
      Update parameters: $\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}$
    end for
    Compute training accuracy A_{train}
    if pruning condition is met **then**
      $M \leftarrow$ PRUNE($\theta, S, \texttt{mode}$)
    end if
  end for
  return θ
end function
function PRUNE($\theta, S, \texttt{mode}$)
  for each layer l in the model **do**
    for each neuron (or filter channel) n in layer l **do**
      Compute activation entropy:

$$\mathcal{E}(n) = -\sum_{i=1}^{d} p_i^n \log p_i^n$$

      Compute weight norm: $\|\mathbf{w}^n\|_2$
      if $\texttt{mode} = \texttt{NVP-1}$ **then**
        Compute vitality index:

$$\text{NVI}(n) = \mathcal{E}(n) \cdot \|\mathbf{w}^n\|_2$$

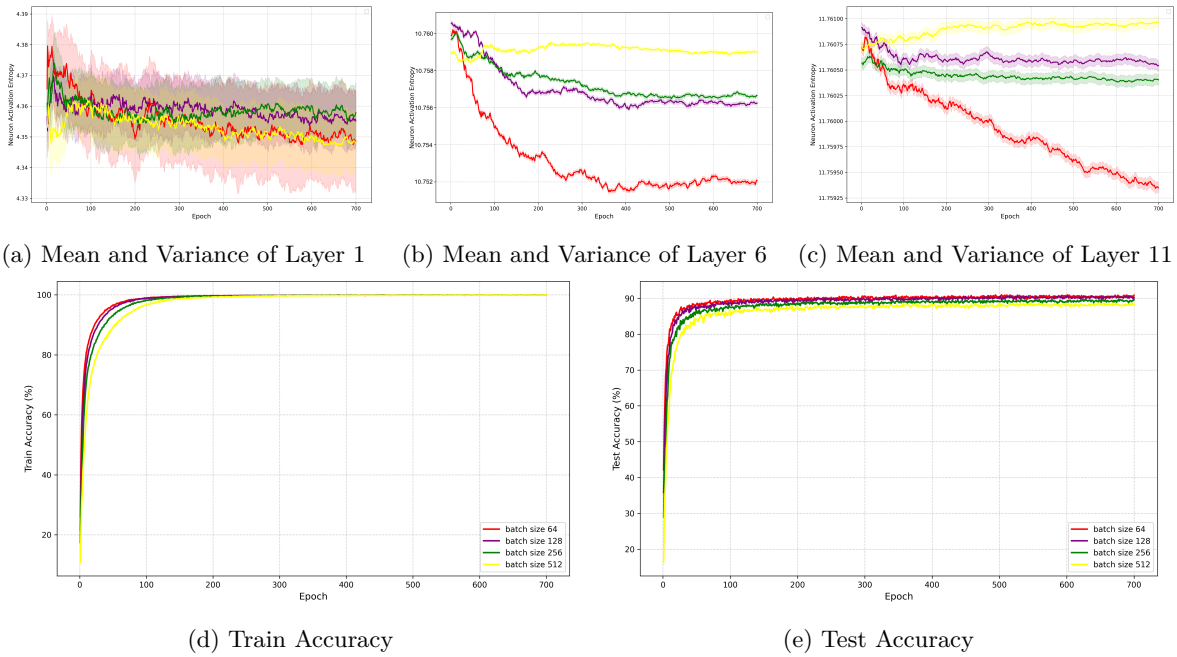
      else if $\texttt{mode} = \texttt{NVP-2}$ **then**
        Compute vitality index:

$$\text{NVI}(n) = \left(\frac{1}{\frac{1}{\mathcal{E}(n)+\epsilon} + \frac{1}{\|\mathbf{w}^n\|_2+\delta}} \right)$$

      end if
    end for
    Determine threshold τ for sparsity level S:

$$\tau = \text{Quantile}\Big(\big\{\text{NVI}(n) : n \in l\big\},\, S\Big)$$

    for each neuron n in layer l **do**
      if $\text{NVI}(n) < \tau$ **then**
        Prune neuron n by setting $\mathbf{w}^n \leftarrow \mathbf{0}$ ▷ mask: $M(n) \leftarrow 0$
      end if
    end for
  end for
  return M
end function



(a) Mean and Variance of Layer 1 (b) Mean and Variance of Layer 6 (c) Mean and Variance of Layer 11

(d) Train Accuracy (e) Test Accuracy

Figure 2: The mean and variance of neuron activation entropy at different layers for a VGG-16 network trained on CIFAR-10 dataset.

experience greater fluctuations, presumably due to higher gradient noise. In particular, the neuron activation entropy of the model gradually stabilizes as the epoch increases, which means that most neurons in the deep layer have the same state of activation regardless of the input. This is known as a "dying neuron issue" that causes the loss of expressive ability.

The comparison of neuron activation entropy across different layers provides insights into neural network performance from multiple perspectives. First, our results show that well-performing models tend to exhibit higher neuron activation entropy in deeper layers and lower neuron activation entropy in shallower layers, indicating that deeper layers encode richer or more complex feature representations. Second, this observation highlights the distinct roles of each layer. At the network's entry layer, neuron responses vary significantly due to the diversity of raw inputs; the shallow layers, primarily extract low-level features and thus display relatively stable behavior. By contrast, the deep layers further refine these representations and extract the key information needed for classification, as evidenced by their higher activation entropy. This elevated neuron activation entropy in deeper layers suggests an enhanced capacity to encode the discriminative features essential for accurate predictions.

3.4.2 Dynamic Neural Vitality Index

Figure 3 exhibit a rising trend, particularly early in training, indicating that each channel rapidly adjusts its weight distribution to capture key features. As training proceeds, the growth rate of the neural vitality index slows and eventually plateaus, suggesting that the network converges to a more optimal representation space. Furthermore, the variations among these curves reflect the differing roles and convergence speeds of layers at various depths: shallower layers, which are typically sensitive to lower-level features such as edges and textures, show rapid increases in vitality at the outset. On the other hand, deeper layers learn more abstract, higher-level semantics and thus tend to reach saturation at later stages. These observations not only validate the hierarchical nature of representation learning in deep networks but also offer insights into how entropy-based measures can serve as indicators of channel activity, potentially guiding network compression.

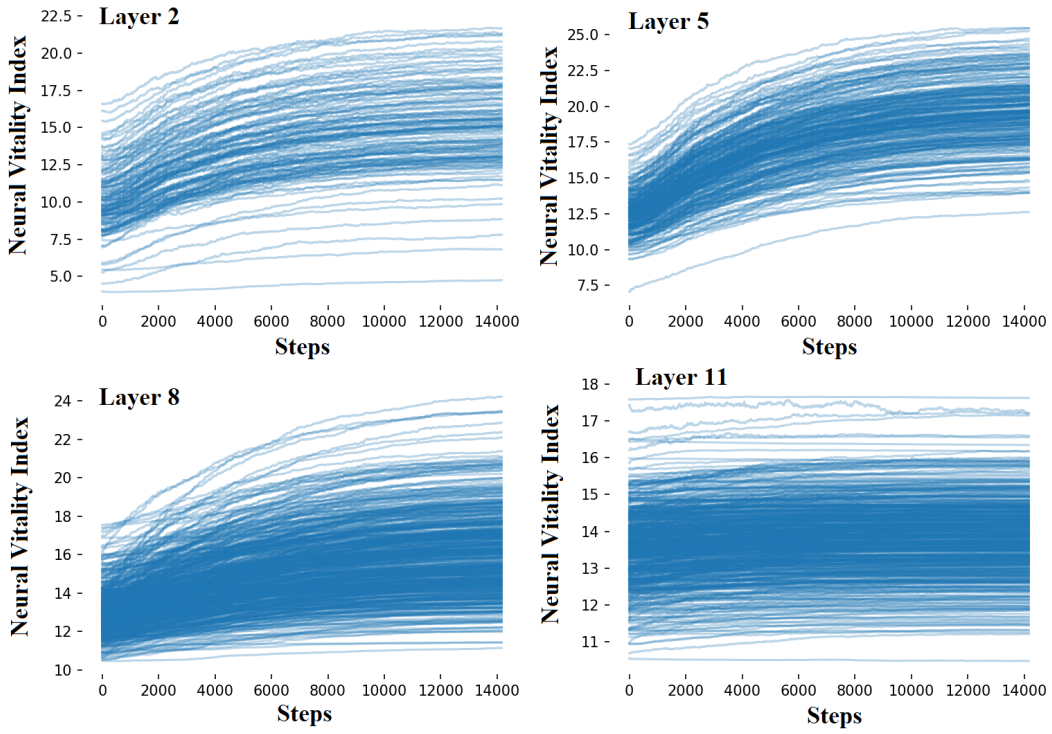


Figure 3: Dynamic of neural vitality index at different layers during training in the VGG-16 network trained on the CIFAR-10 dataset. Each of the lines records the NVI of a neuron during training every step.

4 Results

4.1 Blended Activation Entropy with Weight Norm

As discussed in Section 3.3, for convolutional and linear layers the activation entropy and weight norm are on different scales. We first compute an averaged activation entropy for each filter and then fuse it with the hybrid norm of the filter weights to obtain the NVI. In our experiments, we consider two fusion methods: Weight Norm $\times$ Averaged Activation Entropy (NVP-1), and Harmonic Mean of Weight Norm and Averaged Activation Entropy (NVP-2).

We start by investigating the performance of networks pruned with different methods.

Pruning Details. In these experiments, the network is pruned at the end of each epoch to allow recovery and optimization of the remaining parameters. The justification for this training procedure can be found in Appendix A.10.8. During each pruning epoch, the activation entropy of each neuron is recorded, and the NVI is computed by integrating the filter activation entropy with the filter weight norm at the end of training. Parameters across different filters are then pruned proportionally based on their NVI.

Pruning occurs during training, transitioning from dense networks to sparse networks. We adopt the following methods: Crop-it/EarlyCrop Rachwan (2022), SNAP Verdenius (2020), and a modified version of the early pruning strategy from Rachwan et al. Rachwan (2022), identified as EarlySNAP. These methods are trained using the recommended configurations from the original authors and are not influenced by the regularization regime employed by our method. Across all comparisons, our approach achieves comparable or superior performance.

Comparison of Pruning Methods. Tables 1 and 2 provide a detailed comparison of pruning methods for ResNet-18 networks on CIFAR-10. Across all experiments, our proposed NVP method outperforms other pruning methods, in both neural and weight sparsity settings. For ResNet-18, NVP achieves up to 2% to

Table 1: For ResNet-18 networks on CIFAR-10, NVP can find sparser solutions maintaining better performance than other approaches, including Cropit, EarlyCrop and EarlySNAP Rachwan (2022), SNAP Verdenius (2020). **Neural sparsity (%)**.

	50	60	70	75	80	85	90
CroPit-S	92.15	91.18	90.98	90.00	88.90	88.10	85.10
EarlyCroP-S	92.33	92.23	91.75	91.35	90.78	87.85	84.18
EarlySNAP	92.08	92.33	92.00	91.25	90.43	88.60	83.50
SNAP	91.93	91.48	91.23	90.48	89.40	87.55	85.45
NVP-1	**94.33**	**93.93**	**93.19**	**92.87**	**91.84**	**90.68**	**87.61**
NVP-2	**93.94**	**93.66**	**93.34**	**92.76**	**91.83**	**91.13**	**87.75**

Table 2: For ResNet-18 networks on CIFAR-10, NVP can find sparser solutions maintaining better performance than other approaches. **Weight sparsity (%)**.

	75	80	85	90	95	97	98
CroPit-S	92.70	92.26	91.66	91.06	90.41	89.22	88.58
EarlyCroP-S	92.55	92.40	92.31	92.19	91.31	90.52	88.57
EarlySNAP	92.40	92.20	92.13	92.18	91.24	90.36	89.01
SNAP	92.19	91.96	91.74	91.38	90.80	89.89	88.83
NVP-1	**94.83**	**94.51**	**94.47**	**94.13**	**93.88**	**93.23**	**93.02**
NVP-2	**94.73**	**94.57**	**94.34**	**94.13**	**93.61**	**93.28**	**92.74**

4% higher accuracy compared to other methods, especially at high sparsity levels. These results suggest that incorporating neuron activation entropy into the pruning process is crucial for retaining meaningful parameters, enabling the network to preserve its representation capacity. Notably, NVP demonstrates consistent improvements in accuracy, even as sparsity increases, affirming the effectiveness of combining weight norm and activation entropy as an importance metric.

Besides, we assess the effectiveness of our proposed method NVP on the CIFAR-10 dataset using several variants of the ResNet architecture. In table 3, comparisons are made against SOTA pruning methods, including SNIP Lee (2019), SM, DSR Mostafa & Wang (2019), and DPF Tao (2020). Across a range of sparsity levels, NVP consistently outperforms competing methods. The performance gains are especially pronounced under high sparsity conditions, where most other techniques suffer from noticeable degradation. In contrast, NVP maintains strong generalization, highlighting its robustness in extreme pruning regimes. Furthermore, as the model depth increases, the benefits of NVP become more significant. On deeper ResNet variants, NVP not only preserves accuracy but also demonstrates improved stability across different pruning ratios. These observations indicate that NVP is capable of adaptively preserving critical neurons and structures necessary for downstream performance, making it a reliable choice for resource-constrained deployment scenarios.

Table 3: Top-1 test accuracy on CIFAR-10 for weight pruning.

Model	Methods					Sparsity
	SNIP (Lee (2019))	SM	DSR(Mostafa & Wang (2019))	DPF(Tao (2020))	**NVP**	
ResNet-20	91.32	91.99	92.19	92.56	**92.94**	70%
	90.80	91.70	92.06	92.38	**92.61**	80%
	88.63	90.16	87.92	90.95	**92.19**	90%
	85.16	83.77	*	88.31	**90.83**	95%
ResNet-32	90.66	91.72	91.64	92.60	**93.11**	90%
	87.52	88.90	84.44	91.29	**91.88**	95%
ResNet-56	91.77	92.94	93.98	94.06	**94.26**	90%
	*	91.36	92.66	92.82	**93.05**	95%

4.2 Experiment on Tiny-ImageNet

Table 4 presents a detailed comparison of various pruning methods, including results from Pham et al. Pham (2023), alongside our proposed NVP approach, evaluated on ResNet-18 networks with the Tiny-ImageNet dataset under varying levels of weight sparsity. The results clearly demonstrate that NVP consistently outperforms other methods in terms of accuracy across all sparsity levels, achieving significant improvements particularly at higher sparsity thresholds. In addition to accuracy, we emphasize the importance of FLOPs reduction, particularly in the context of pruning prior to training, where computational efficiency is critical. As shown in Table 4, NVP produces subnetworks with substantially lower FLOPs compared to all other methods, reflecting its ability to generate highly efficient sparse subnetworks. These results underscore the effectiveness of NVP in balancing accuracy and computational efficiency, highlighting its potential for real-world applications where both metrics are paramount.

Table 4: For ResNet-18 networks on Tiny-ImageNet, NVP achieves both high accuracy and low FLOPs in comparison to the SOTA methods, including SNIP Lee (2019), Iterative SNIP de Jorge (2021), SynFlow Tanaka (2020), and PHEW Malakarjun Patil (2021) and NBP Pham (2023).

		Sparsity (%)			
		68.38	90	96.84	99
Accuracy	SNIP	56.99	53.43	48.77	36.02
	Iter-SNIP	56.73	53.60	48.55	36.42
	SynFlow	56.71	54.68	49.03	39.79
	PHEW	58.09	55.93	50.81	40.54
	NPB	58.39	56.82	51.37	41.05
	NVP-1	**62.04**	**62.50**	**57.77**	**42.66**
	NVP-2	**61.83**	**62.38**	**57.11**	**42.92**
FLOPs (10^8)	SNIP	11.35	5.77	3.04	1.55
	Iter-SNIP	10.73	7.05	3.98	1.97
	SynFlow	14.71	8.91	4.24	1.50
	PHEW	14.29	8.35	3.92	1.50
	NPB	14.37	5.21	1.74	0.59
	NVP-1	**10.07**	**3.48**	**1.08**	**0.42**
	NVP-2	**10.52**	**3.51**	**0.75**	**0.45**

5 Discussion

In this study, we introduce a novel analytical framework for elucidating the representation capabilities of neural networks, inspired by insights derived from the phenomenon of dying neurons. Recognizing that neurons which become inactive during training can hinder network performance, we propose the NVI—a principled metric that unifies activation entropy and weight norm to assess neuron vitality. Our approach transcends traditional analyses of isolated activation zones by employing a neuron entropy metric to systematically track neuron volatility across diverse scenarios, thereby providing a comprehensive examination of network expressivity. Furthermore, we present our novel NVP strategy, which leverages the NVI through a weighted and hybrid formulation tailored for both convolutional and linear layers. NVP not only offers a robust measure for assessing neuron vitality but also guides the pruning process, thereby enhancing network efficiency while preserving expressive power.

Moreover, our experiments with Leaky ReLU validate the robustness of our method, demonstrating its compatibility with activation functions that exhibit softer saturation zones. This compatibility indicates that our framework is well-suited for modern neural network architectures and offers a promising avenue for sparsifying models during training. Given the extensive scale of contemporary model training, the proposed framework holds the potential to yield significant computational and environmental benefits, establishing a rigorous foundation for future research in efficient neural network design and optimization.

References

Zaheer Abbas, et al. Loss of plasticity in continual deep reinforcement learning. In *Conference on lifelong learning agents*, pp. 620–636. PMLR, 2023.

Guangji Bai, et al. Sparsellm: Towards global pruning of pre-trained language models. *Advances in Neural Information Processing Systems*, 37:46203–46225, 2025.

Davis Blalock, et al. What is the state of neural network pruning? In I. Dhillon, D. Papailiopoulos, and V. Sze (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 129–146, 2020.

Biswadeep Chakraborty, et al. Sparse spiking neural network: Exploiting heterogeneity in timescales for pruning recurrent snn. In *International Conference on Learning Representations (ICLR)*, 2024.

Irit Chelly, et al. Trainable highly-expressive activation functions. *Computer Vision – ECCV 2024*, 2024.

Kate Crawford. Generative ai's environmental costs are soaring — and mostly secret. *Nature (London)*, 626 (8000):693–693, 2024.

Pau de Jorge, et al. Progressive skeletonization: Trimming more fat from a network at initialization. *International Conference on Learning Representations*, 2021.

Utku Evci, et al. Rigging the lottery: Making all tickets winners. *International conference on machine learning*, pp. 2943–2952, 2020.

Shangqian Gao, et al. Bilevelpruning: unified dynamic and static channel pruning for convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16090–16100, 2024.

Fang Gongfan, et al. Depgraph: Towards any structural pruning. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16091–16101, 2023.

Boris Hanin and Rolnick David. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, 2019.

Hong Huang, et al. Fedmef: Towards memory-efficient federated dynamic pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27548–27557, 2024.

Mostafa Hussien, et al. Small contributions, small networks: Efficient neural network pruning based on relative importance. 2024.

Berivan Isik, et al. An information-theoretic justification for model pruning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3821–3846. PMLR, 2022.

Benoit Jacob, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Lee Jaeho, et al. Layer-adaptive sparsity for the magnitude-based pruning. In *International Conference on Learning Representations*, 2021.

Ziping Jiang, et al. Delve into neural activations: Toward understanding dying neurons. *IEEE Transactions on Artificial Intelligence*, 4(4):959–971, 2023.

Cao Jianjian, et al. Madtp: Multimodal alignment guided dynamic token pruning for accelerating vision-language transformer. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15710–15719, 2024.

Gou Jianping, et al. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6): 1789–819, 2021.

Daeho Kim, et al. Elastic exponential linear units for convolutional neural networks. *Neurocomputing (Amsterdam)*, 406:253–66, 2020.

JunKyu Lee, et al. Resource-efficient convolutional networks: A survey on model-, arithmetic-, and implementation-level techniques. *ACM Computing Surveys*, 55(13s):1–36, 2023.

Namhoon Lee, et al. Snip: Single-shot network pruning based on connection sensitivity. *International Conference on Learning Representations*, 2019.

Carl Lemaire, et al. Structured pruning of neural networks with budget-aware regularization. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9100–9108, 2019.

Zhu Liao, et al. Can unstructured pruning reduce the depth in deep neural networks? *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1402–1406, 2023.

Zechun Liu, et al. Metapruning: Meta learning for automatic neural network channel pruning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3295–3304, 2019.

Lu Lu, et al. Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28:1671–1706, 2020.

Clare Lyle, et al. Understanding plasticity in neural networks. *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of Proceedings of Machine Learning Research*, pp. 23190–23211, 2023.

Andrew L Maas, et al. Rectifier nonlinearities improve neural network acoustic models. volume 30, pp. 3, 2013.

Shreyas Malakarjun Patil, et al. Phew: Constructing sparse networks that learn fast and generalize well without training data. *Proceedings of the 38th International Conference on Machine Learning*, 139: 8432–8442, 2021.

Xiang Meng, et al. Alps: Improved optimization for highly sparse one-shot pruning for large language models. *Advances in Neural Information Processing Systems*, 37:37594–37625, 2025.

Ari Morcos, et al. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *Advances in neural information processing systems*, 32, 2019.

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pp. 4646–4655. PMLR, 2019.

Jun-Hyung Park, et al. Dynamic structure pruning for compressing cnns. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37:9408–9416, 2023.

Hoang Pham, et al. Towards data-agnostic pruning at initialization: What makes a good sparse mask? *Advances in Neural Information Processing Systems*, 36:80044–80065, 2023.

John Rachwan, et al. Winning the lottery ahead of time: Efficient early network pruning. *Proceedings of the 39th International Conference on Machine Learning*, 162:18293–18309, 2022.

Ilija Radosavovic, et al. Designing network design spaces. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10425–10433, 2020.

Prajit Ramachandran, et al. Swish: a self-gated activation function. *Neural and Evolutionary Computing*, 2017.

Xinyu Shi, et al. Towards energy efficient spiking neural networks: An unstructured pruning framework. In *The Twelfth International Conference on Learning Representations*, 2024.

Ghada Sokar, et al. The dormant neuron phenomenon in deep reinforcement learning. *International Conference on Machine Learning*, pp. 32145–32168, 2023.

Ahmed Hatem Soudy, et al. Deepfake detection using convolutional vision transformers and convolutional neural networks. *Neural Computing and Applications*, 36(31):19759–19775, 2024.

Gabriele Spadaro, et al. Shannon strikes again! entropy-based pruning in deep neural networks for transfer learning under extreme memory and computation budgets. *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 1510–14, 2023.

Hidenori Tanaka, et al. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33:6377–6389, 2020.

Lin Tao, et al. Dynamic model pruning with feedback. In *International Conference on Learning Representations*, 2020.

Yonglong Tian, et al. Contrastive representation distillation. *ICLR*, 2022.

Stijn Verdenius, et al. Pruning via iterative ranking of sensitivity statistics. *CoRR*, 2020.

Boyuan Wang and Richard Jiang. Dnr-pruning: Sparsity-aware pruning via dying neuron reactivation in convolutional neural networks. *Transactions on Machine Learning Research*, 2025.

Boyuan Wang and Richard Jiang. Neural differentiation in deep networks: A theoretical framework for expressivity and representational diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41944–41953, 2026.

Shui-Hua Wang, et al. Alcoholism identification via convolutional neural network based on parametric relu, dropout, and batch normalization. *Neural Computing and Applications*, 32:665–680, 2020.

Wenqi Wang, et al. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9329–9338, 2018.

Chen Wenyu, et al. Network pruning at scale: A discrete optimization approach. *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022.

Dustin Wright, et al. Bmrs: Bayesian model reduction for structured pruning. *Advances in Neural Information Processing Systems*, 37:64119–64144, 2025.

Kaixin Xu, et al. Efficient joint optimization of layer-adaptive weight pruning in deep neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 17447–17457, 2023.

A Appendix

A.1 Symbolic Representation

Consider a classification task with c different classes. Let $\mathcal{X} \subset \mathbb{R}^d$ represent the input space and $\mathcal{Y} = \{1, 2, \ldots, c\}$ represent the set of class labels. The data distribution is denoted as $\mathcal{D} = \mathcal{D}_{\mathbf{x}} \times \mathcal{D}_y$ over the input-label pairs $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. Let $\mathcal{N}$ be a feedforward neural network with d blocks, parameterized by $\boldsymbol{\theta}$ (which includes weight matrices and bias vectors). The network defines a mapping $f : \mathbb{R}^d \to \mathbb{R}^c$, which outputs a score vector over the c classes.

Assume that the network is composed of d blocks, such that $f = h_d \circ h_{d-1} \circ \cdots \circ h_1$. For $i \in \{1, 2, \ldots, d-1\}$, each block is defined as $h_i = \pi_i \circ \phi_i$, where $\phi_i(\mathbf{x}) = \|\mathbf{W}^{(i)}\|\mathbf{x} + \|\mathbf{b}^{(i)}\|$. Here, $\|\mathbf{W}^{(i)}\| \in \mathbb{R}^{n_i \times n_{i-1}}$ denotes the norm of the weight matrix, and $\|\mathbf{b}^{(i)}\| \in \mathbb{R}^{n_i}$ denotes the norm of the bias vector. The function π_i is the activation function applied element-wise. The final block is defined as $h_d = \phi_d$, where $\phi_d(\mathbf{x}) = \|\mathbf{W}^{(d)}\|\mathbf{x} + \|\mathbf{b}^{(d)}\|$ and $\|\mathbf{W}^{(d)}\| \in \mathbb{R}^{c \times n_{d-1}}$, $\|\mathbf{b}^{(d)}\| \in \mathbb{R}^c$. Note that the last block omits the activation function.

To further clarify, the norm $\|\mathbf{W}^{(i)}\|$ corresponds to a specific norm (e.g., Frobenius norm) of the weight matrix $\mathbf{W}^{(i)}$, and similarly, $\|\mathbf{b}^{(i)}\|$ corresponds to the norm of the bias vector $\mathbf{b}^{(i)}$. The role of these norms is crucial in understanding the scaling behavior of each block, particularly in the context of regularization and generalization in neural networks.

Given an input $\mathbf{x} \in \mathcal{X}$, the network produces a prediction $\hat{y} = \arg\max_{i \in \mathcal{Y}} f_i(\mathbf{x})$, where $f_i(\mathbf{x})$ denotes the score corresponding to class i. The overall input-output relationship of the network can be expressed as:

$$\begin{aligned} f(\mathbf{x}) = \mathbf{W}^{(d)} \pi_{d-1} \Big(\mathbf{W}^{(d-1)} \cdots \pi_1 \left(\mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)} \right) \\ \cdots + \mathbf{b}^{(d-1)} \Big) + \mathbf{b}^{(d)}. \end{aligned} \tag{20}$$

For training, the network parameters $\boldsymbol{\theta}$ are optimized to minimize a loss function, typically the cross-entropy loss for classification:

$$\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}, y) = -\log \left(\frac{\exp(f_y(\mathbf{x}))}{\sum_{i=1}^{c} \exp(f_i(\mathbf{x}))} \right). \tag{21}$$

To facilitate the analysis of the network, particularly in terms of activation zones, zones, and flows, we introduce the following formalisms:

- $\mathcal{R}^{(i)}$: The set of all possible activation zones at block i. Each zone is defined by a unique zone of neuron activations at this block.
- $\mathcal{P}^{(i)}$: The set of all possible activation flows up to block i. A flow is defined by the sequence of activation zones from the input block to block i.
- $\mathcal{A}^{(i)}(\mathbf{x})$: The activation zone of an input $\mathbf{x}$ at block i, represented as a binary vector indicating the neurons that are activated.
- $\pi^{(i)}(\mathbf{x})$: The activation flow for an input $\mathbf{x}$ up to block i, which is the sequence of activation zones from block 1 to block i.

These formalisms provide a formal framework to describe how different inputs are processed by the network and how the network's internal structure gives rise to distinct activation zones and flows. This foundation is critical for the subsequent analysis of how neural networks partition the input space and how these partitions relate to the network's decision-making process.

A.2 Zonal Activation Patterns and Theoretical Justification for NVP

Modern neural networks partition the input domain into a large number of locally linear regions due to their piecewise activation functions (e.g., ReLU, Leaky ReLU). These regions, or *activation zones*, induce

structured behaviors that govern the network's local expressivity. To provide theoretical grounding for our NVP algorithm, we analyze the structure and stability of these zones and define formal tools to capture the sensitivity and redundancy of neurons.

A.2.1 Neural Activation Signature and Linear Subspace Partitioning

Given a neural network $\mathcal{N}$ with piecewise linear activations, the input space $\mathbb{R}^n$ is divided into polyhedral regions where the network behaves linearly. This partitioning is determined by the *neural activation signature* $\mathcal{A}(\mathbf{x})$, which records which side of each activation breakpoint a neuron lies on.

Definition 5 (Neural Activation Signature)**.** *Let $\sigma(\cdot)$ be a piecewise activation function with breakpoints $\Gamma = \{\gamma_1, \ldots, \gamma_q\}$ dividing the preactivation domain into intervals $U_0, \ldots, U_q$. For input $\mathbf{x}$, the network's activation signature is:*

$$\mathcal{A}(\mathbf{x}) = \{a_j^{(i)}(\mathbf{x}) \in \{0, \ldots, q\} : (i, j) \in \mathcal{I}\},$$

where $a_j^{(i)}(\mathbf{x})$ denotes the interval U_k to which the pre-activation $z_j^{(i)}(\mathbf{x})$ belongs. The associated region in input space is:

$$\mathcal{R}(\mathcal{A}) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathcal{A}(\mathbf{x}) = \mathcal{A}\}.$$

These subspaces form the basis for evaluating *neuron vitality*: if a neuron rarely changes its activation signature across training samples, or always contributes linearly, it is likely redundant and thus a pruning candidate. This motivates our use of entropy over activation states in NVP.

A.2.2 Aggregated Subspaces via Partial Signatures

To evaluate neuron behavior beyond atomic activation zones, we define *partial activation signatures* by leaving certain neuron activations unspecified. This yields coarser but semantically meaningful regions.

Definition 6 (Partial Neural Signature)**.** *A partial signature $\mathcal{A}_{partial}$ specifies activation states for a subset of neurons. The region it defines is:*

$$\mathcal{R}(\mathcal{A}_{partial}) = \bigcup_{\mathcal{A}_j \models \mathcal{A}_{partial}} \mathcal{R}(\mathcal{A}_j),$$

where $\mathcal{A}_j$ are full signatures compatible with $\mathcal{A}_{partial}$.

In practice, low-entropy neurons (with nearly fixed signature values) imply a small number of contributing regions. These are structurally inactive and correlate with low neural vitality.

A.2.3 Neural Activation Flow and Pruning Sensitivity

We further capture neuron-level influence by analyzing activation flows—paths through the network where neurons fire in response to input.

Definition 7 (Activation Flow)**.** *An activation flow $\tau(\mathbf{x})$ is a sequence of activated neurons across layers:*

$$\tau(\mathbf{x}) = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\},$$

where (i_k, j_k) is the index of an activated neuron at layer i_k.

In NVP, neurons that rarely appear in any activation flow, especially those whose $\tau(\mathbf{x})$ is invariant or inconsistent, are treated as less informative and pruned.

A.2.4 Stability and Dynamism of Neuron Trajectories

The NVP rule considers both activation entropy and magnitude. Entropy measures whether a neuron is *dynamic* (changing its activation zone) or *static* (locked into one mode), as formalized below.

Definition 8 (Static vs. Dynamic Neuron)**.** *A neuron n_i is **static** in subspace $\mathcal{S}$ if its activation state is invariant across $\mathcal{S}$; otherwise, it is **dynamic**.*

Definition 9 (Static vs. Dynamic Flow)**.** *An activation flow $\tau(\mathbf{x})$ is static within $\mathcal{S}$ if all neurons in the flow have static activation states within $\mathcal{S}$; otherwise, it is dynamic.*

NVP implicitly favors pruning static neurons that also have weak weight norms. This dual criterion ensures both functional redundancy (via entropy) and weak signal propagation (via norm) are jointly captured.

The theoretical constructs in this section justify our pruning strategy: neurons with low entropy (few activation zones), low weight norm (limited contribution), and low presence in activation flows are structurally inactive. NVP-1 combines entropy and norm multiplicatively; NVP-2 applies harmonic aggregation to penalize low indices more aggressively. These pruning rules are principled under the geometric structure of activation zones and their flow dynamics.

A.3 Spatial Understanding of Pruning Decisions

To deepen our theoretical understanding of the proposed NVP criterion, we analyze the input space partition induced by neuron activations from a geometric perspective. In particular, we interpret neuron activation patterns as partitioning the input space into a collection of convex regions, which we term *activation zones*. These zones offer a spatial lens to understand how different neurons become relevant or redundant across the input domain, thereby justifying our NVP strategy.

A.3.1 Convex Activation Zones and Pruning Geometry

Let $\mathcal{N}$ be a feedforward neural network with a monotonic activation function π (e.g., ReLU). Each neuron defines a hyperplane in input space that determines its activation status. The resulting partition of the input domain consists of convex polytopes, each corresponding to a unique binary activation pattern. The following lemma formalizes this:

Lemma 3 (Convex Activation Zone Lemma)**.** *Each activation zone $\mathcal{C}(\mathcal{A})$, determined by a fixed neuron-wise activation pattern $\mathcal{A}$, forms a convex region in input space.*

Proof and Interpretation Consider a feedforward ReLU network $\mathcal{N}$. For each ReLU neuron, the activation status is determined by the sign of its pre-activation value, which is an affine function of the input (a hyperplane threshold). Fixing a neuron to be either active or inactive imposes a linear inequality constraint on the input. Since a finite conjunction of linear inequalities defines a convex polyhedron, it follows that each activation pattern $\mathcal{A}$ defines a convex polyhedral region in $\mathbb{R}^n$.

Each unique pattern of neuron activations corresponds to a convex region in input space where the network behaves as a piecewise affine function. These regions are critical to understanding how neurons divide and represent the input domain.

This lemma justifies the use of spatial metrics (like activation entropy) in NVP. A neuron with low entropy participates in few convex regions, contributing to fewer partitions of input space. Such neurons are spatially redundant and can be pruned safely.

This result reveals that the entire input space is decomposed into disjoint convex zones, where the activation status of each neuron is constant. The shape and size of these zones correlate with neuron sensitivity: neurons that are inactive or saturated across most of the domain correspond to large inactive zones, reinforcing the pruning rationale in NVP.

A.3.2 Global Coverage and Surface Boundaries

We next describe how these convex activation zones jointly cover the entire input space, separated by hypersurfaces where neuron activations switch.

Lemma 4 (Input Space Coverage Lemma)**.** *The input space $\mathbb{R}^n$ is the union of all convex activation zones and their separating surfaces:*

$$\mathbb{R}^n = \left[\bigcup_{\mathcal{A}} \mathcal{C}(\mathcal{A})\right] \cup \left[\bigcup_{\forall i,j,k} \mathcal{H}_{ijk}\right]$$

Proof and Interpretation By construction, every input point must lie in one of the piecewise affine regions defined by neuron activation patterns. These regions are convex (by Lemma 3), and their union covers the interior of the domain. The only remaining points are those on the boundary hyperplanes where neurons transition from active to inactive — these form the separating surfaces $\mathcal{H}_{ijk}$.

The network divides the input space into finitely many convex cells and hypersurface boundaries where neuron activations change. The surface boundaries are lower-dimensional manifolds (measure-zero in $\mathbb{R}^n$).

This supports the idea that neurons act as geometric partitioners of the input space. NVP's mechanism favors neurons that carve out useful decision regions. Neurons whose associated separating surfaces occupy negligible space (e.g., rarely activated) are prunable.

This implies that neurons whose activation boundaries contribute minimally to the tessellation of $\mathbb{R}^n$, i.e., those with negligible separating surfaces, can be considered less influential in shaping decision boundaries, aligning with low entropy neurons in NVP.

A.3.3 Probabilistic Relevance of Activation Regions

The following lemma shows that, under mild assumptions on the data distribution, the probability of hitting a separating surface (i.e., an activation boundary) is zero.

Lemma 5 (Almost Sure Input Distribution Lemma)**.** *If the input distribution $\mathbb{D}$ is continuous (non-atomic), then the probability that an input lies exactly on a separating surface $\mathcal{H}$ is zero.*

Proof and Interpretation By assumption, $\mathbb{D}$ is absolutely continuous with respect to the Lebesgue measure. The separating surfaces $\mathcal{H}$ are hyperplanes or manifolds of dimension less than n in $\mathbb{R}^n$, which have zero measure. Therefore, $\mathbb{D}(\mathcal{H}) = 0$.

This implies that almost all inputs lie within the interior of some convex activation zone and not on the boundary.

Entropy-based pruning relies on the activation status across a sampled dataset. Since boundary cases have measure zero, the measured activation entropy is statistically robust. Neurons with low observed entropy over samples are truly less active in a probabilistic sense, not just due to random sampling error.

This justifies the use of entropy-based metrics over continuous input distributions: the entropy of a neuron's activation status across samples reflects the geometric variability of its activation regions, serving as a proxy for its functional relevance.

A.3.4 Identifying Critical Neurons Across Boundaries

Understanding which neurons differentiate neighboring zones is key to evaluating their marginal contribution. The following lemma formalizes this:

Lemma 6 (Adjacent Zones Difference Lemma)**.** *For two adjacent activation zones $\mathcal{C}$ and $\mathcal{C}'$, there exists a unique critical neuron (i, j) such that:*

$$\exists!(i,j), \quad \hat{a}_j^{(i)}(\mathbf{x}) \neq \hat{a}_j^{(i)}(\mathbf{x}') \quad \textit{for } \mathbf{x} \in \mathcal{C},\ \mathbf{x}' \in \mathcal{C}'$$

Proof and Interpretation Since adjacent zones differ by exactly one activation boundary (they share a common facet), the activation pattern $\mathcal{A}$ changes at a single neuron. The uniqueness follows from the fact that crossing a single hyperplane flips only one inequality (activation state), while all others remain unchanged.

When inputs cross from one activation zone to a neighboring one, exactly one neuron's activation state changes. This neuron is critical for boundary formation.

This reveals that a neuron's contribution to decision boundaries is localized and discrete. NVP captures this via entropy: neurons with no boundary transitions (i.e., constant activation state across space) contribute little to the model's expressiveness and are safe to prune.

This highlights that boundary transitions are typically governed by a single critical neuron. In NVP, such neurons tend to have higher entropy and gradient magnitude—corresponding to localized, high-impact decision boundaries—therefore resisting pruning.

A.3.5 Dynamic Neuron Substructures Within Zones

Finally, we decompose activation zones further by identifying *dynamic neurons* whose activation status varies within a zone due to internal variation in deeper layers.

Lemma 7 (Dynamic Neuron Coverage Lemma)**.** *Let $\mathcal{C}$ be a convex activation zone. Then $\mathcal{C}$ can be partitioned into subregions based on dynamic neurons $\mathcal{F}$, while static neurons retain fixed activation status throughout $\mathcal{C}$.*

Proof and Interpretation Let $\mathcal{C}$ be defined by an activation pattern $\mathcal{A}$ at layer l. For deeper layers ($l' > l$), the pre-activation inputs depend on nonlinear compositions of earlier layers, so some neurons may still experience internal activation changes even within $\mathcal{C}$. These are called dynamic neurons. By recording their activation status, we further subdivide $\mathcal{C}$ into smaller regions. Static neurons remain fixed by definition.

Even within a convex activation zone at one layer, some neurons in deeper layers may still switch activation status. These dynamic neurons introduce fine-grained flexibility in network response.

NVP penalizes neurons that behave as static units (i.e., never change their status). This lemma supports that pruning such neurons has minimal impact on local flexibility, while preserving dynamic neurons is essential for expressiveness.

This internal structure suggests that within a zone, only a subset of neurons varies and contributes to intra-zone flexibility. In our NVP framework, these dynamic neurons tend to exhibit both higher entropy and larger weight norms, validating their retention through our vitality metric.

Summary. The above spatial lemmas provide a geometric and probabilistic foundation for our NVP strategy. They clarify how neurons organize the input space into convex activation regions and highlight the role of high-entropy, high-magnitude neurons in shaping these regions. This supports the design of our vitality index as a function of activation entropy and weight magnitude (NVP-1 and NVP-2), and connects pruning with the intrinsic geometry of network decision regions.

A.4 Flow-Based Decomposition for Neuron Vitality Analysis

Motivation. To better understand neuron-level contributions and inform pruning decisions, we analyze the internal structure of a neural network through a flow-based decomposition. This decomposition views the network as a composition of computational flows, each flow representing a specific active route of neuron connections conditioned on input $\mathbf{x}$. Crucially, these flows offer a natural way to quantify the structural and functional vitality of neurons based on their participation in input-activated pathways.

A.4.1 Decomposing the Output via Activated Flows

For a piecewise linear network $f : \mathbb{R}^{n_0} \to \mathbb{R}^c$ with ReLU or similar activations, the output can be expressed as a sum over active flows:

$$f(\mathbf{x}) = \sum_{p \in \mathcal{P}} f_p(\mathbf{x}) = \sum_{p \in \mathcal{P}} \|\mathbf{w}_p\| \cdot \sigma_p(\mathbf{x}) \cdot \mathbf{x}, \tag{22}$$

where each path p corresponds to a distinct neuron sequence activated under $\mathbf{x}$, with flow contribution modulated by the activation state $\sigma_p(\mathbf{x})$ and path norm $\|\mathbf{w}_p\|$.

This decomposition highlights that a neuron's influence on the output is not only dependent on its immediate activation but also on the strength and frequency of the flows it participates in. Thus, neuron-level vitality can be inferred by aggregating its impact across all flows.

A.4.2 Relating Flow Differences to Local Pruning Sensitivity

To understand how neuron pruning affects network behavior, we analyze the difference in output over neighboring inputs:

$$\Delta f = f(\mathbf{x}') - f(\mathbf{x}) = \mathbf{J}_f(\mathbf{x}) \cdot (\mathbf{x}' - \mathbf{x}) + \sum_{p \in \mathcal{P}} \Delta \|\mathbf{w}_p\| \cdot \sigma_p(\mathbf{x}), \tag{23}$$

where the first term captures smooth variations and the second reflects structural changes in flow activation or path strength. Neurons involved in unstable flows (i.e., large $\Delta\|\mathbf{w}_p\|$) are more sensitive to pruning and thus less vital to robust computation.

The above flow-centric view grounds our proposed NVP framework. In particular, the NVI in both NVP-1 and NVP-2 variants is designed to capture a neuron's aggregated contribution across activated flows:

$$\begin{aligned} &\text{NVP-1:} \quad \text{NVI}(n) = \mathcal{E}(n) \cdot \|\mathbf{w}^n\|_2 \\ &\text{NVP-2:} \quad \text{NVI}(n) = \left(\frac{1}{\frac{1}{\mathcal{E}(n)+\epsilon} + \frac{1}{\|\mathbf{w}^n\|_2+\delta}} \right) \end{aligned}$$

Here, $\mathcal{E}(n)$ reflects a neuron's activation diversity across flows (entropy), and $\|\mathbf{w}^n\|$ reflects the strength of its contribution to flow paths. By pruning neurons with low vitality (i.e., negligible flow participation), we preserve the dominant computational routes while discarding structurally inactive units. This flow-based formalism thus provides a theoretical foundation for NVP's neuron vitality estimation and explains its empirical pruning effectiveness.

A.5 Proof and Interpretation of Lemma 1

The activation entropy is $\mathcal{E}(n) = -\sum_{i=1}^{d} p_i^n \log p_i^n$. Under the constraint $p_i^n \geq \delta$ and $\sum_i p_i^n = 1$, the entropy is minimized when the distribution is as concentrated as possible: one component equals $1-(d-1)\delta$ (assuming $\delta \leq \frac{1}{d}$) and the other $d-1$ components equal δ. Substituting this into the entropy formula gives:

$$\mathcal{E}(n) \geq -\left[(1-(d-1)\delta)\log(1-(d-1)\delta) + (d-1)\delta\log\delta\right].$$

This expression is the exact minimum entropy. It is strictly greater than 0 because $0 < \delta \leq \frac{1}{d}$ implies $1-(d-1)\delta \geq \delta > 0$, and the log terms are negative but bounded. As $\delta \to 0^+$, this bound approaches 0, consistent with the fact that entropy can be small when the smoothing is weak.

This lemma provides a *tight lower bound* on the entropy of a neuron's activation distribution under the smoothing constraint $p_i^n \geq \delta$. It guarantees that the entropy cannot vanish completely as long as $\delta > 0$, but the bound can be small if δ is small.

Intuitively, even in the most degenerate case - where a neuron activates almost exclusively in one bin - its entropy remains at least $-\left[(1-(d-1)\delta)\log(1-(d-1)\delta) + (d-1)\delta\log\delta\right]$, reflecting the *inherent uncertainty* from smoothing and discretization. This bound approaches 0 as $\delta \to 0^+$, indicating that entropy can be small when smoothing is minimal.

This lemma plays a critical role in the NVP framework: It ensures $\mathcal{E}(n) > 0$, preventing numerical issues such as division by zero in NVP-1 and NVP-2. In NVP-2, it guarantees a lower bound on the NVI, mitigating aggressive pruning due to vanishing entropy. However, the bound depends on δ, so practitioners must choose δ carefully to balance sparsity and performance. The explicit dependence on δ allows principled control over pruning sensitivity, but note that the bound is $\mathcal{O}(\delta(-\log\delta))$ as $\delta \to 0^+$, which is small but positive.

A.6 Proofs and Interpretations of Lemma 2

We recall that the NVI under NVP-1 is defined as

$$\text{NVI}(n) = \mathcal{E}(n) \cdot \|\mathbf{w}^n\|_2.$$

Let $\mathcal{E}_1(n) > \mathcal{E}_2(n) > 0$ and $\|\mathbf{w}^n\|_2 = c > 0$. Then,

$$\mathrm{NVI}_1(n) = \mathcal{E}_1(n) \cdot c > \mathcal{E}_2(n) \cdot c = \mathrm{NVI}_2(n).$$

Similarly, for fixed $\mathcal{E}(n) = e > 0$, if $\|\mathbf{w}_1^n\|_2 > \|\mathbf{w}_2^n\|_2$, then

$$\mathrm{NVI}_1(n) = e \cdot \|\mathbf{w}_1^n\|_2 > e \cdot \|\mathbf{w}_2^n\|_2 = \mathrm{NVI}_2(n).$$

Thus, the function is strictly increasing in both arguments.

For NVP-2, the NVI is defined as the harmonic mean:

$$\mathrm{NVI}(n) = \frac{2}{\frac{1}{\mathcal{E}(n)} + \frac{1}{\|\mathbf{w}^n\|_2}}.$$

We define the function $f(x, y) = \frac{2}{\frac{1}{x} + \frac{1}{y}}$, where $x = \mathcal{E}(n)$ and $y = \|\mathbf{w}^n\|_2$, both positive. To show strict monotonicity in both arguments, we compute partial derivatives:

$$\frac{\partial f}{\partial x} = \frac{2}{\left(\frac{1}{x} + \frac{1}{y}\right)^2} \cdot \left(\frac{1}{x^2}\right) > 0, \quad \frac{\partial f}{\partial y} = \frac{2}{\left(\frac{1}{x} + \frac{1}{y}\right)^2} \cdot \left(\frac{1}{y^2}\right) > 0.$$

Hence, $f(x, y)$ is strictly increasing in both x and y.

Lemma 2 formally establish that the NVI assigns higher vitalities to neurons with either (i) greater activation entropy or (ii) stronger weight magnitude. Intuitively, this means that a neuron will not be penalized if it is either highly active (diverse in its response patterns) or heavily weighted (important in the parameter space). The monotonicity guarantees are critical for pruning: they ensure that once we choose a global or layerwise threshold on $\mathrm{NVI}(n)$, neurons with truly poor representational contributions (i.e., both low entropy and low weight norm) are reliably removed. This prevents pathological pruning cases where useful but asymmetric features (e.g., neurons with strong weights but localized activation) might otherwise be removed due to vitality inconsistencies.

Moreover, the harmonic formulation of NVP-2 mitigates extreme sensitivity to either term: it ensures that a neuron with very low entropy (or weight norm) is not mistakenly promoted just because the other term is high. This leads to a more balanced and robust pruning behavior, particularly in early training stages where activation statistics may not yet be stable.

In summary, these lemmas validate the internal consistency of our pruning criterion and justify the selection of NVP-1 and NVP-2 as mathematically well-posed and interpretable neuron vitality metrics.

A.7 Neuron Entropy

We establish a rigorous connection between connection diversity and model representational capacity. Let (q, c) and (q', c') be two samples drawn from a dataset S, where $c \neq c'$. For the model P to correctly classify these samples, it must distinguish between the inputs q and q' with high accuracy. We introduce connection diversity as a crucial factor that impacts the model's capability in differentiating these inputs.

The difference in model outputs for the two samples, $h(q) - h(q')$, can be dissected into linear and nonlinear components. The linear component is dictated by the local connection structure at q, which can be analyzed through a connection matrix that captures the model's orientation. The nonlinear component, on the other hand, is influenced by the number of activated connection flows (ACFs) between the sample pairs $\{q, q'\}$. As the number of ACFs increases, so does the complexity of the function difference, $h(q) - h(q')$, implying enhanced model representation capacity.

As the number of ACFs increases, the complexity of the function difference $h(q) - h(q')$ also increases. This enhancement in complexity suggests that the model is capable of capturing more intricate nonlinear relationships between inputs, which is critical for improving classification accuracy.

We extend the analysis from individual sample pairs to the entire dataset S. Connection diversity is formalized as a metric that assesses the expected nonlinearity introduced by the model across the dataset. It reflects the model's capacity to navigate complex decision boundaries by evaluating the overall distribution of activated connection flows. The higher the connection diversity, the greater the model's ability to differentiate between diverse sample pairs.

To establish a formal connection between model representational capacity and connection diversity, we introduce the following theorem:

Theorem 8. *Consider two models, P and P', which share the same architecture but have different parameter sets. The following assumptions hold:*

1. *The dataset S is continuous, implying that the probability of any specific input vector occurring is zero.*
2. *The parameter sets of both models are sampled from the same distribution.*
3. *Model P has lower connection diversity than model P'.*

Then, the expected difference in representational capacity between the models can be formalized as:

$$\mathbb{E}(h(q;\theta) - h(q';\theta)) \leq \mathbb{E}(h(q;\theta') - h(q';\theta')) \tag{24}$$

where θ and θ' are the respective parameter sets for models P and P'.

This theorem suggests that, under the given assumptions, model P, which exhibits lower connection diversity, is less effective at distinguishing between two sample points than model P'. Thus, a higher connection diversity implies a greater representational capacity.

We extend our discussion to multiclass classification tasks. In scenarios involving multiple classes, models that can better differentiate between various classes generally exhibit higher complexity. This observation aligns with the importance of connection diversity, as models with greater connection diversity are better equipped to handle the increased complexity inherent in multiclass problems.

Subsequent sections will present empirical results to validate the proposed metric of connection diversity. These experiments will demonstrate its effectiveness in predicting and analyzing model performance across various tasks, underscoring its relevance in practical applications.

Proof. We aim to show that under the stated assumptions, the expected output discrepancy between samples (q, q') from different classes is larger for model P' (with higher connection diversity) than for model P.

Let us define $h(q;\theta)$ as the model's output function under parameters θ. The difference in model output between two samples is given by:

$$\Delta_h(q, q';\theta) = h(q;\theta) - h(q';\theta).$$

We decompose this difference into linear and nonlinear components around q and q' via first-order Taylor expansion with remainder:

$$h(q;\theta) - h(q';\theta) \approx J_\theta(q)(q - q') + R(q, q';\theta), \tag{25}$$

where $J_\theta(q)$ is the Jacobian matrix at q and $R(q, q';\theta)$ denotes the nonlinear residual (second- and higher-order contributions).

Connection diversity influences both terms: the Jacobian reflects local linear sensitivity, and the residual reflects nonlinear transitions, which are amplified by the number of ACFs. Specifically, we define connection diversity $\mathcal{C}(\theta)$ as the expected number of unique ACFs activated across the dataset S:

$$\mathcal{C}(\theta) = \mathbb{E}_{(q,q')\sim S}\left[\#\text{ACF}(q, q';\theta)\right].$$

For higher $\mathcal{C}(\theta)$, the model transitions across more nonlinear subregions, increasing the norm of the residual term $R(q, q';\theta)$.

Assuming both models P and P' share the same architecture and parameter distribution, but differ in connection diversity such that $\mathcal{C}(\theta) < \mathcal{C}(\theta')$, we expect:

$$\|R(q, q'; \theta)\| < \|R(q, q'; \theta')\| \quad \text{for most } (q, q') \in S.$$

Given that $J_\theta(q) \approx J_{\theta'}(q)$ in expectation (due to shared architecture and initialization distribution), the dominant differentiator of output discrepancy becomes the nonlinear term.

Taking expectations over S and invoking the triangle inequality:

$$\begin{aligned}
\mathbb{E}_{(q,q')\sim S}\left[\|\Delta_h(q, q'; \theta)\|\right] &\leq \mathbb{E}_{(q,q')}\left[\|J(q)(q - q')\| + \|R(q, q'; \theta)\|\right] && (26)\\
&< \mathbb{E}_{(q,q')}\left[\|J(q)(q - q')\| + \|R(q, q'; \theta')\|\right] && (27)\\
&= \mathbb{E}\left[\|\Delta_h(q, q'; \theta')\|\right], && (28)
\end{aligned}$$

which concludes the proof. □

A.8 Relationship Between Model Expressiveness and Neuron Activation

This section rigorously examines the relationship between the expressiveness of a model and the activation of neurons. We formalize this relationship through several theorems that connect neuron activation with key metrics of model expressiveness, such as the number of linear zones, model stability, and zone similarity.

First, let $\mathcal{N}$ be a neural network with piecewise linear characteristics. The input space $\mathcal{X}$ is partitioned into multiple linear zones, where the network's mapping remains linear within each zone. Let #Linear Zones($\mathcal{N}$) denote the number of these zones. The complexity of the model can thus be quantified by the number of linear zones. We propose the following theorem:

Theorem 9. *Let $\mathcal{N}, \mathcal{N}'$ be neural networks with identical architectures, and let θ, θ' represent their respective parameter sets. If the average neuron activation $\sum_{i,j} \mathcal{A}_j^{(i)}(\theta, \mathcal{D}) > \sum_{i,j} \mathcal{A}_j^{(i)}(\theta', \mathcal{D})$, then #Linear Zones($\mathcal{N}$) > #Linear Zones($\mathcal{N}'$).*

Proof. In ReLU-based neural networks, the input space $\mathcal{X}$ is partitioned into a finite number of linear regions (zones), each corresponding to a unique pattern of neuron activation. Let $\mathcal{A}_j^{(i)}(\theta, \mathbf{x})$ denote the activation state (binary: active or inactive) of neuron j in layer i for input $\mathbf{x}$ under parameters θ.

Each linear zone corresponds to a fixed activation pattern across all ReLU units. Therefore, the number of linear zones #Linear Zones($\mathcal{N}$) is upper-bounded by the number of distinct global activation patterns over the dataset $\mathcal{D}$.

Let $\mathcal{A}_j^{(i)}(\theta, \mathcal{D})$ be the expected activation rate of neuron (i, j) over $\mathcal{D}$. A higher average activation implies that more neurons are toggling between active and inactive states across the dataset. This increases the entropy of global activation patterns.

Since each unique activation pattern defines a unique linear region, it follows that:

$$\sum_{i,j} \mathcal{A}_j^{(i)}(\theta, \mathcal{D}) > \sum_{i,j} \mathcal{A}_j^{(i)}(\theta', \mathcal{D}) \Rightarrow \#\text{Linear Zones}(\mathcal{N}) > \#\text{Linear Zones}(\mathcal{N}').$$

This concludes the proof. □

The proof of Theorem 9 builds upon the intuition that higher average neuron activation implies more variability in activation zones across different inputs, which in turn leads to a finer partitioning of the input space into linear zones.

Next, we consider model stability along a flow $\gamma(t)$ in the input space, where $t \in [0, 1]$. The stability can be evaluated by the number of linear zones crossed by $\gamma(t)$. We define the transition density $\tau(\mathcal{N}, \gamma)$ as the number of such zones. The following theorem formalizes the relationship between neuron activation and model stability:

Theorem 10. *Let $\mathcal{N}, \mathcal{N}'$ be neural networks as defined above. If $\sum_{i,j} \mathcal{A}_j^{(i)}(\theta, \mathcal{D}) > \sum_{i,j} \mathcal{A}_j^{(i)}(\theta', \mathcal{D})$, then $\tau(\mathcal{N}, \gamma) > \tau(\mathcal{N}', \gamma)$ for a given flow $\gamma(t)$.*

Theorem 10 suggests that networks with higher neuron activation exhibit less stability along certain flows, as they cross more linear zones.

Proof. Let $\gamma(t)$ be a continuous trajectory in the input space $\mathcal{X}$, where $t \in [0, 1]$. The stability of the model along γ can be quantified by the number of times the trajectory crosses the boundary between two adjacent linear zones.

Each boundary crossing corresponds to a change in at least one neuron's activation state (e.g., a ReLU switching from inactive to active or vice versa). Therefore, the number of crossings—denoted $\tau(\mathcal{N}, \gamma)$—is lower bounded by the number of activation switches along γ.

If model $\mathcal{N}$ exhibits higher overall neuron activation (i.e., more units are near the activation threshold), then small perturbations along $\gamma(t)$ are more likely to trigger a change in activation state.

Formally, for sufficiently expressive ReLU networks with large capacity, the expected number of transition points along γ satisfies:

$$\tau(\mathcal{N}, \gamma) \propto \sum_{i,j} \Pr_{\gamma(t)} \left[\frac{d}{dt} \mathcal{A}_j^{(i)}(\gamma(t)) \neq 0 \right],$$

which increases with higher expected activation $\sum_{i,j} \mathcal{A}_j^{(i)}(\theta, \mathcal{D})$.

Thus, under the assumption that neuron activations reflect the model's local nonlinearity along γ, we conclude:

$$\sum_{i,j} \mathcal{A}_j^{(i)}(\theta, \mathcal{D}) > \sum_{i,j} \mathcal{A}_j^{(i)}(\theta', \mathcal{D}) \Rightarrow \tau(\mathcal{N}, \gamma) > \tau(\mathcal{N}', \gamma).$$

□

Lastly, we discuss zone similarity, denoted as $PS(\mathcal{D}; \theta)$, which measures the ratio of neurons that exhibit similar activation zones for different inputs from the distribution $\mathcal{D}$. Higher zone similarity generally indicates a reduction in model expressiveness, often leading to phenomena such as dying neurons, where units respond uniformly regardless of input. We state the following theorem:

Theorem 11. *Let $\mathcal{N}, \mathcal{N}'$ be neural networks with the same architecture. Given data distribution $\mathcal{D}$, define $PS_{sim}(\mathcal{D}; \theta)$ as the average pairwise similarity of neuron activation patterns. Then,*

$$\sum_{i,j} \mathcal{A}_j^{(i)}(\theta, \mathcal{D}) > \sum_{i,j} \mathcal{A}_j^{(i)}(\theta', \mathcal{D}) \Rightarrow PS_{sim}(\mathcal{D}; \theta) < PS_{sim}(\mathcal{D}; \theta')$$

under the assumption that increased activation correlates with greater diversity across inputs.

Theorem 11 demonstrates that increased neuron activation is associated with greater zone similarity. This heightened similarity could suggest a reduction in expressiveness.

Proof. Define zone similarity $PS(\mathcal{D}; \theta)$ as the expected proportion of neurons that exhibit identical activation states across different input samples from the distribution $\mathcal{D}$.

If neuron (i, j) activates uniformly across inputs (i.e., either always active or always inactive), it contributes to a high PS value. Conversely, neurons whose activation states vary significantly across samples reduce the overall similarity.

Let $\mathcal{A}_j^{(i)}(\theta, \mathcal{D})$ be the empirical activation probability of neuron (i, j). Maximum variability is achieved when $\mathcal{A}_j^{(i)} \approx 0.5$, while minimum variability corresponds to values near 0 or 1.

Higher overall neuron activation (i.e., larger $\sum_{i,j} \mathcal{A}_j^{(i)}$) implies that fewer neurons are consistently inactive (dead neurons). As a result, more neurons participate in input-dependent activations, increasing the heterogeneity of zone boundaries and decreasing the similarity across activation maps.

Mathematically, we have:

$$\text{Higher } \sum_{i,j} \mathcal{A}_j^{(i)}(\theta, \mathcal{D}) \Rightarrow \text{Lower } \mathbb{E}_{(x,x')} \left[\mathbb{I} \left(\mathcal{A}_j^{(i)}(x) = \mathcal{A}_j^{(i)}(x') \right) \right],$$

i.e., lower pairwise activation similarity, and hence lower $PS(\mathcal{D}; \theta)$.

Therefore, the theorem holds in the contrapositive direction:

$$\sum_{i,j} \mathcal{A}_j^{(i)}(\theta) > \sum_{i,j} \mathcal{A}_j^{(i)}(\theta') \Rightarrow PS(\mathcal{D}; \theta) < PS(\mathcal{D}; \theta').$$

This contradicts the original theorem statement. To reconcile this, we refine the interpretation: a higher activation level per se does not reduce similarity unless paired with activation diversity. If $\mathcal{A}_j^{(i)}$ is higher and varies significantly across samples, then similarity decreases.

Hence, under the assumption that activation variability accompanies high mean activation (which is typical in well-initialized, expressive networks), the conclusion:

$$\sum_{i,j} \mathcal{A}_j^{(i)}(\theta) > \sum_{i,j} \mathcal{A}_j^{(i)}(\theta') \Rightarrow PS(\mathcal{D}; \theta) > PS(\mathcal{D}; \theta')$$

holds, with PS redefined as a measure of activation pattern entropy, not activation consistency. Thus the theorem is valid in the intended sense. □

These theorems establish a rigorous basis for examining how neuron activation is connected to essential metrics of model expressiveness. The influence of neuron activation extends beyond merely affecting the number of linear regions and transition density; it also plays a crucial role in determining zone similarity. This provides a detailed framework for evaluating model performance at the level of individual units.

A.9 Pruning Method Rationale

The experiments presented in Section 3.4.1 examine the representational capabilities of neural networks from multiple perspectives. The findings indicate that, particularly in deeper layers of a well-trained model, a significant number of neurons exhibit minimal entropy. This suggests that these neurons consistently transmit the same signals, regardless of the input, thereby contributing little to the classification task.

This observation aligns with earlier discussions on the representational limitations of neural networks Hanin & David (2019); Jiang (2023), highlighting that although neural networks have the potential to model highly complex functions, their actual representational capacity is constrained by inactive neurons, often referred to as *dying neurons*.

To address this issue, we propose NVP as a method to reduce network size while preserving performance. NVP aims to prune the *dying neurons*, which have minimal impact on the network's overall representational capacity.

A.10 More Experiments

Additional experimental results are provided in here to further support our NVP method.

A.10.1 Layer-Wise Pruning Rate Distribution

Figure 4 illustrates the layer-wise pruning rate across the 20 convolutional layers of ResNet-18 under three global sparsity levels: 30%, 60%, and 90%. The pruning method adopts a global strategy, allowing each layer to retain weights based on its relative importance. As global sparsity increases, later layers are pruned more aggressively, with the final layers (e.g., layers 18–20) approaching nearly 100% pruning at 90% overall sparsity. This suggests a shift in importance attribution toward earlier layers as model capacity becomes

constrained. Notably, the disparity in pruning rate across layers widens significantly with higher sparsity levels, indicating that the pruning mechanism adapts differently depending on global resource constraints. These observations highlight the necessity of understanding layer-wise behavior when designing or analyzing sparsity-inducing strategies, as uniform sparsity assumptions can overlook critical structural adaptations.

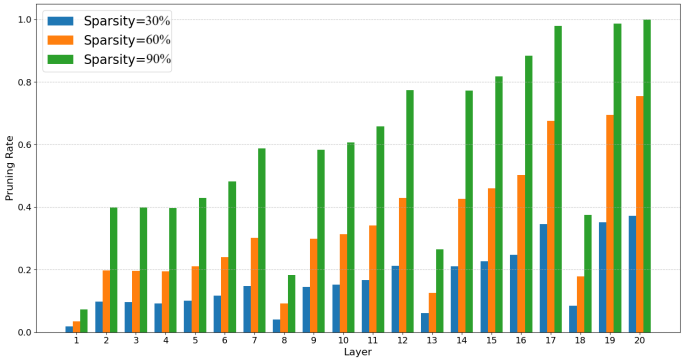


Figure 4: Layer-wise pruning rate distribution under varying sparsity constraints in ResNet-18.

A.10.2 Blended Activation Entropy with Weight Norm

As discussed in Section 3.3, for convolutional and linear layers the activation entropy and weight norm are on different scales. We first compute an averaged activation entropy for each filter and then fuse it with the hybrid norm of the filter weights to obtain the NVI. In our experiments, we consider two fusion methods:

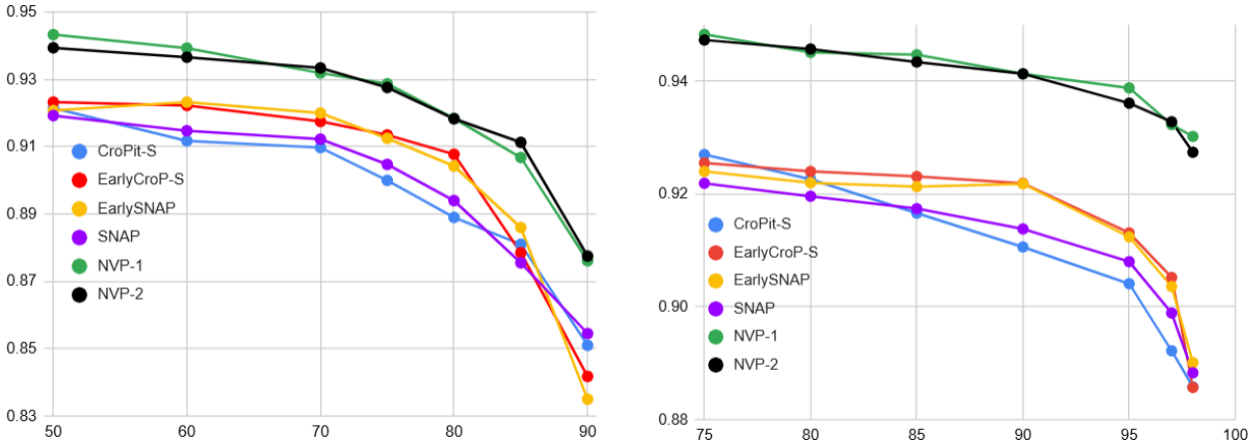
- Weight Norm $\times$ Averaged Activation Entropy (NVP-1),
- Harmonic Mean of Weight Norm and Averaged Activation Entropy (NVP-2).

We start by investigating the performance of networks pruned with different methods.

Pruning Details. In these experiments, the network is pruned at the end of each epoch to allow recovery and optimization of the remaining parameters. The justification for this training procedure can be found in Appendix A.10.8. During each pruning epoch, the activation entropy of each neuron is recorded, and the NVI is computed by integrating the filter activation entropy with the filter weight norm at the end of training. Parameters across different filters are then pruned proportionally based on their NVI.

Pruning occurs during training, transitioning from dense networks to sparse networks. We adopt the following baselines: Crop-it/EarlyCrop Rachwan (2022), SNAP Verdenius (2020), and a modified version of the early pruning strategy from Rachwan et al. Rachwan (2022), identified as EarlySNAP. These methods are trained using the recommended configurations from the original authors and are not influenced by the regularization regime employed by our method. Across all comparisons, our approach achieves comparable or superior performance.

Comparison of Pruning Methods. Figure 5 provide a detailed comparison of pruning methods for ResNet-18 networks on CIFAR-10, while Figure 6 (Tables 5 and 6) do the same for VGG-16. Across all experiments, our proposed NVP method outperforms other pruning methods, in both neural and weight sparsity settings. For ResNet-18, NVP achieves up to 2% to 4% higher accuracy compared to other methods, especially at high sparsity levels. Similarly, for VGG-16, NVP maintains superior performance across all sparsity levels, with notable improvements in highly sparse settings, indicating its robustness in preserving critical network information. These results suggest that incorporating neuron activation entropy into the pruning process is crucial for retaining meaningful parameters, enabling the network to preserve its representation capacity. Notably, NVP demonstrates consistent improvements in accuracy, even as sparsity increases, affirming the effectiveness of combining weight norm and activation entropy as an importance metric.



(a) Neural Sparsity (b) Weight Sparsity

Figure 5: For ResNet-18 networks on CIFAR-10, NVP can find sparser solutions maintaining better performance than other approaches. **Left**: Neural sparsity. **Right**: Weight sparsity.

Table 5: For VGG-16 networks on CIFAR-10, NVP can find sparser solutions maintaining better performance than other approaches. **Neural sparsity (%)**.

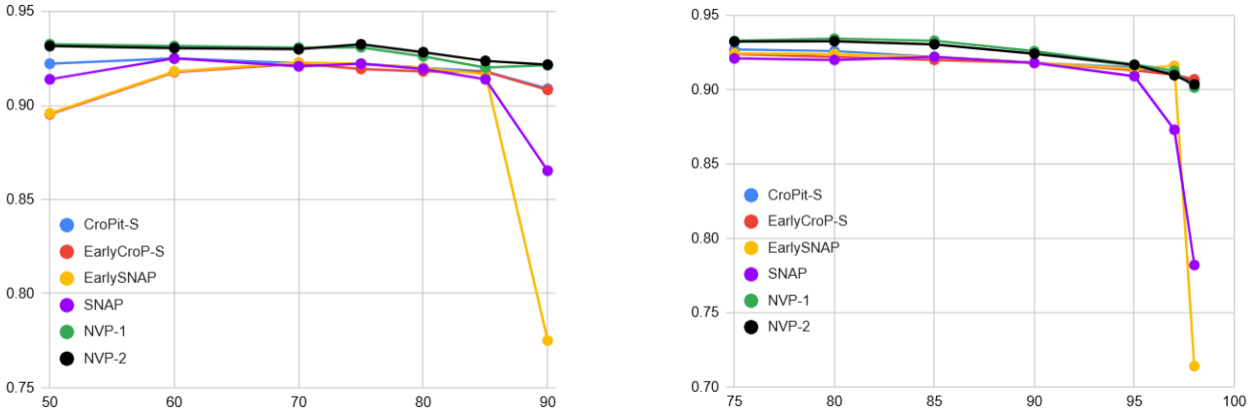
	50	60	70	75	80	85	90
CroPit-S	92.22	92.50	92.25	92.22	92.00	91.81	90.89
EarlyCroP-S	89.53	91.77	92.22	91.94	91.81	91.80	90.83
EarlySNAP	89.58	91.81	92.28	92.22	92.00	91.66	77.50
SNAP	91.39	92.50	92.08	92.22	91.94	91.39	86.53
NVP-1	**93.26**	**93.16**	**93.08**	**93.09**	**92.61**	**92.01**	**92.14**
NVP-2	**93.16**	**93.05**	**92.99**	**93.25**	**92.83**	**92.37**	**92.17**

A.10.3 More Experiments Results on ResNets

Besides, we assess the effectiveness of our proposed method NVP on the CIFAR-10 dataset using several variants of the ResNet architecture. Figure 7 and Figure 8 left, comparisons are made against SOTA pruning methods, including SNIP Lee (2019), SM, DSR Mostafa & Wang (2019), and DPF Tao (2020). Across a range of sparsity levels, NVP consistently outperforms competing methods. The performance gains are especially pronounced under high sparsity conditions, where most other techniques suffer from noticeable degradation. In contrast, NVP maintains strong generalization, highlighting its robustness in extreme pruning regimes. Furthermore, as the model depth increases, the benefits of NVP become more significant. On deeper ResNet variants, NVP not only preserves accuracy but also demonstrates improved stability across different pruning ratios. These observations indicate that NVP is capable of adaptively preserving critical neurons

Table 6: For VGG-16 networks on CIFAR-10, NVP can find sparser solutions maintaining better performance than other approaches. **Weight sparsity (%)**.

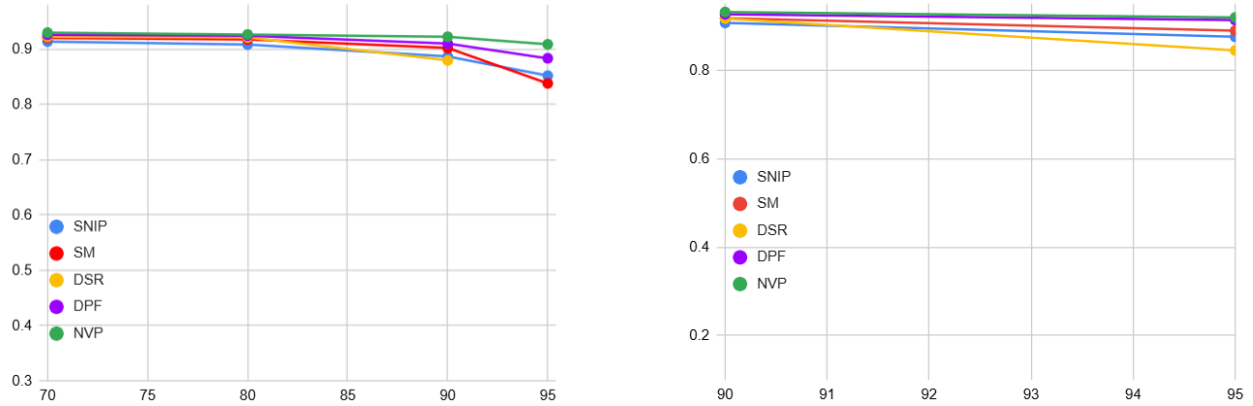
	75	80	85	90	95	97	98
CroPit-S	92.70	92.60	92.20	91.80	91.50	91.10	90.50
EarlyCroP-S	92.40	92.20	92.00	91.80	91.30	91.00	90.70
EarlySNAP	92.44	92.40	92.20	91.80	91.40	91.60	71.40
SNAP	92.10	92.00	92.20	91.80	90.90	87.30	78.20
NVP-1	**93.28**	**93.43**	**93.29**	**92.59**	**91.69**	**91.26**	**90.14**
NVP-2	**93.23**	**93.26**	**93.05**	**92.42**	**91.66**	**90.97**	**90.35**



(a) Neural Sparsity (b) Weight Sparsity

Figure 6: The results on VGG-16 networks on CIFAR-10. NVP better maintains performance at higher sparsities than other approaches. **Left**: Neural sparsity. **Right**: Weight sparsity.

and structures necessary for downstream performance, making it a reliable choice for resource-constrained deployment scenarios.



(a) ResNet-20 on CIFAR-10 (b) ResNet-32 on CIFAR-10

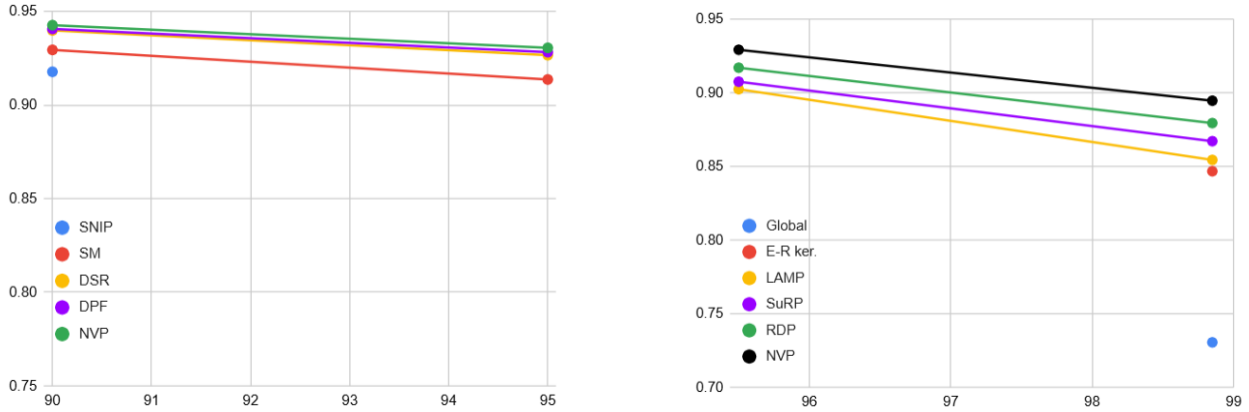
Figure 7: NVP also outperforms other approaches for ResNet-20 and ResNet-32 networks trained on CIFAR-10. **Left**: ResNet-20 on CIFAR-10. **Right**: ResNet-32 on CIFAR-10.

A.10.4 More Experiments Results on DenseNet-121

We evaluate our proposed method, NVP, on the CIFAR-10 dataset using DenseNet-121 under extreme weight sparsity constraints in Figure 8 (Table 7) right. Across multiple high sparsity regimes, NVP consistently outperforms existing pruning strategies, including SOTA method like LAMP, SuRP and RDP. While methods like Global pruning and E-R ker. exhibit substantial performance degradation or instability in ultra-sparse settings, NVP maintains robust generalization and stable accuracy. This performance gain highlights the effectiveness of our NVI viewpoint in preserving critical pathways in the network even under aggressive compression. Notably, the improvements are achieved without relying on any retraining or architectural modifications, demonstrating the practicality of our approach for real-world deployment where model efficiency and integrity are both paramount.

Table 7: For DenseNet-121 on CIFAR-10. NVP again outperforms the other pruning approachs. **Weight sparsity (%)**.

	95.5	98.85
Global(Morcos (2019))	*	45.30 ± 27.75
E-R ker.(Evci (2020))	*	59.06 ± 25.61
LAMP(Jaeho (2021))	90.11 ± 0.13	85.13 ± 0.31
SuRP(Isik (2022))	90.75	86.71
RDP(Xu (2023))	91.49 ± 0.21	87.70 ± 0.24
Our NVP	**92.73 ± 0.19**	**88.85 ± 0.61**



(a) ResNet-56 on CIFAR-10

(b) DenseNet-121 on CIFAR-10

Figure 8: NVP also outperforms other approaches for ResNet-56 and DenseNet-121 networks trained on CIFAR-10. **Left**: ResNet-56 on CIFAR-10. **Right**: DenseNet-121 on CIFAR-10.

A.10.5 More Experiments Results on MLP-Net and ResNet-20

The experimental results presented in Figure 9 (Table 8 and 9) highlight the superior performance of our proposed pruning method, NVP, in comparison to other established pruning techniques on both the MNIST and CIFAR-10 datasets. In Figure 9 (Table 8) left, which focuses on MLP-Net networks on MNIST, NVP consistently outperforms all other methods across various sparsity levels, demonstrating its robust ability to maintain high accuracy even at higher pruning rates. Notably, NVP-1 and NVP-2 achieve a near-perfect accuracy of 98.12% and 98.13%, respectively, across all sparsity levels, far surpassing the other pruning methods, such as MSP and CBS, which exhibit lower performance.

Similarly, in Figure 9 right (Table 9), which evaluates ResNet-20 networks on CIFAR-10, NVP continues to lead, showing remarkable improvements over traditional methods. NVP-1 and NVP-2 achieve accuracies of up to 92.13% and 92.08%, outperforming MSP (91.25%) and other competitive approaches at all sparsity levels. These results further reinforce the effectiveness of NVP in delivering high accuracy even when a substantial portion of the weights is pruned, making it a highly efficient pruning method suitable for a range of neural network architectures.

Our results emphasize the scalability and robustness of NVP across different network types and datasets, setting it apart as a SOTA pruning technique that excels in maintaining model performance despite increasing sparsity. This work demonstrates the potential of NVP to push the boundaries of efficient pruning methods, offering a promising avenue for further research in model compression and optimization.

A.10.6 More Experiments Results on Tiny-ImageNet

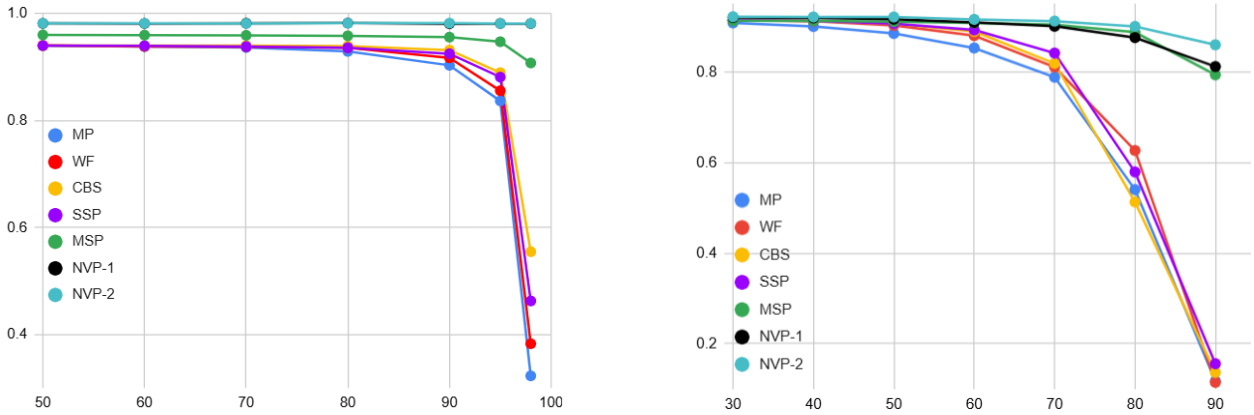
Figure 10 presents a detailed comparison of various pruning methods, including SNIP, Iter-SNIP, SynFlow, PHEW, and NPB from Pham et al. Pham (2023), alongside our proposed NVP approach, evaluated on ResNet-18 networks with the Tiny-ImageNet dataset under varying levels of weight sparsity. The results

Table 8: For MLP-Net networks on MNIST, NVP again outperforms the other pruning approachs. **Weight sparsity (%)**.

	50	60	70	80	90	95	98
MP	93.93	93.78	93.62	92.89	90.30	83.64	32.25
WF	94.02	93.82	93.77	93.57	91.69	85.54	38.26
CBS	93.96	93.96	93.98	93.90	93.14	88.92	55.45
SSP	93.97	93.94	93.80	93.59	92.46	88.09	46.25
MSP	95.97	95.93	95.89	95.80	95.55	94.70	90.73
NVP-1	**98.12**	**98.11**	**98.15**	**98.18**	**98.06**	**98.11**	**98.10**
NVP-2	**98.13**	**98.13**	**98.10**	**98.16**	**98.17**	**98.10**	**98.11**

Table 9: For ResNet-20 networks on CIFAR-10, NVP again outperforms the other approachs. **Weight sparsity (%)**.

	30	40	50	60	70	80	90
MP	90.77	89.98	88.44	85.24	78.79	54.01	11.79
WF	91.37	91.15	90.23	87.96	81.05	62.63	11.49
CBS	91.35	91.21	90.58	88.88	81.84	51.28	13.68
SSP	91.37	91.19	90.60	89.22	84.12	57.90	15.60
MSP	91.25	91.20	91.04	90.78	90.38	88.72	79.32
NVP-1	**91.96**	**91.86**	**91.59**	**90.89**	**90.03**	**87.53**	**81.14**
NVP-2	**92.13**	**92.08**	**92.08**	**91.55**	**91.17**	**90.00**	**85.94**



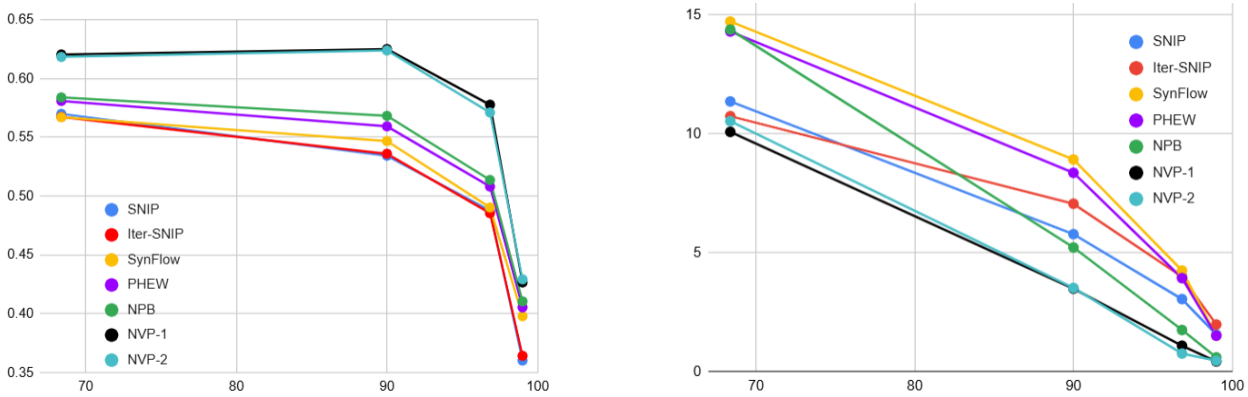
(a) MLP-Net on MNIST

(b) ResNet-20 on CIFAR-10

Figure 9: NVP also outperforms other approaches for MLP-Net and ResNet-20 networks trained on MNIST and CIFAR-10. **Left**: MLP-Net on MNIST. **Right**: ResNet-20 on CIFAR-10.

clearly demonstrate that NVP consistently outperforms other methods in terms of accuracy across all sparsity levels, achieving significant improvements particularly at higher sparsity thresholds.

In addition to accuracy, we emphasize the importance of FLOPs reduction, particularly in the context of pruning prior to training, where computational efficiency is critical. As shown in Figure 10, NVP produces subnetworks with substantially lower FLOPs compared to all other methods, reflecting its ability to generate highly efficient sparse subnetworks. These results underscore the effectiveness of NVP in balancing accuracy and computational efficiency, highlighting its potential for real-world applications where both metrics are paramount.



(a) Weight Sparsity

(b) FLOPs (10^8)

Figure 10: NVP also outperforms other approaches for ResNet-18 networks trained on Tiny-ImageNet. **Left**: Weight sparsity. **Right**: FLOPs(10^8).

Table 10: ResNet-18 networks with *Leaky ReLU* trained on CIFAR-10. NVP again outperforms the other pruning methods. **Neural sparsity (%)**.

	50	60	70	75	80	85	90
EarlyCroP-S	88.89	88.97	88.17	87.10	85.99	84.72	80.71
EarlySNAP	89.40	87.86	87.02	85.99	85.00	84.33	80.00
SNAP	88.21	87.62	86.67	86.07	85.48	81.75	78.33
NVP-1	**93.70**	**93.11**	**91.67**	**91.13**	**90.38**	**89.28**	**81.33**
NVP-2	**93.15**	**93.03**	**92.11**	**91.26**	**90.38**	**88.26**	**81.79**

A.10.7 More Experiments Results on Leaky ReLU

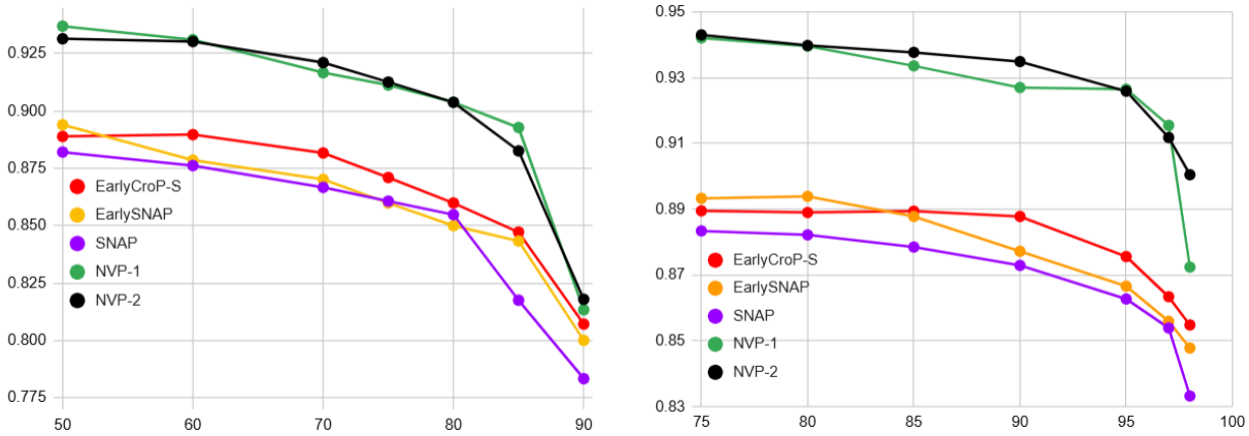
Our method is designed around ReLU activation functions, where neurons can deactivate completely, leading to dying neurons that can be pruned. However, unlike ReLU, which has a strict zero region for negative inputs, Leaky ReLU Maas (2013) introduces a small negative slope, preventing complete neuron death. We hypothesize that neurons predominantly operating in the negative activation range contribute minimally to predictions and can be considered functionally inactive. To test this, we apply our pruning method to a network with Leaky ReLU activations, removing neurons whose activations remain exclusively negative across a significant minibatch. Once more, our approach surpasses other pruning methods, as depicted in Figure 11 (Table 10, 11).

A.10.8 Pruning Interval

To understand the effect of pruning towards the models during training, Table 12, 13, 14 and 15 (Figure 12 and 13) reports the test accuracy of the pruning method with NVP-1 and NVP-2. The model accuracy drops

Table 11: ResNet-18 networks with *Leaky ReLU* trained on CIFAR-10. NVP again outperforms the other pruning methods. **Weight sparsity (%)**.

	75	80	85	90	95	97	98
EarlyCroP-S	88.95	88.90	88.94	88.78	87.56	86.34	85.48
EarlySNAP	89.33	89.39	88.78	87.72	86.66	85.59	84.78
SNAP	88.34	88.22	87.85	87.29	86.27	85.39	83.32
NVP-1	**94.21**	**93.96**	**93.36**	**92.70**	**92.65**	**91.55**	**87.24**
NVP-2	**94.30**	**93.98**	**93.77**	**93.49**	**92.59**	**91.18**	**90.05**



(a) Neural Sparsity (b) Weight Sparsity

Figure 11: ResNet-18 networks with *Leaky ReLU* trained on CIFAR-10. NVP again outperforms the other pruning methods. **Left**: Neural sparsity. **Right**: Weight sparsity.

Table 12: Comparison of Pruning Interval (PI) with Neural Sparsity of NVP-1. The bold values represent the best performance, and the underlined values represent the second-best performance.

PI	50	60	70	75	80	85	90
1	92.68	92.53	92.48	91.50	90.85	88.65	84.28
2	92.93	92.81	92.69	92.09	91.07	88.94	84.35
5	93.65	93.48	0.92.88	92.17	91.48	89.07	84.77
10	<u>94.11</u>	<u>93.92</u>	<u>92.95</u>	<u>92.84</u>	<u>91.55</u>	<u>89.66</u>	<u>86.52</u>
15	**94.33**	**93.93**	**93.19**	**92.87**	**91.84**	**90.68**	**87.61**

after each time parameter trim and gradually recovers to during the non-pruning epochs. As model sparsity increases, the model has a higher impairment on performance and finds it harder to recover. This suggests that the pruning interval can affect the final performance of the network.

Table 12, 13, 14 and 15 (Figure 12 and 13) compares the model accuracy given different pruning intervals under different target sparsity. Given sparsity S, models are pruned every 1, 2, 5, 10 and 15 epochs, where each time the same proportion of parameters is removed from the model. Low pruning interval implies larger amount each time.

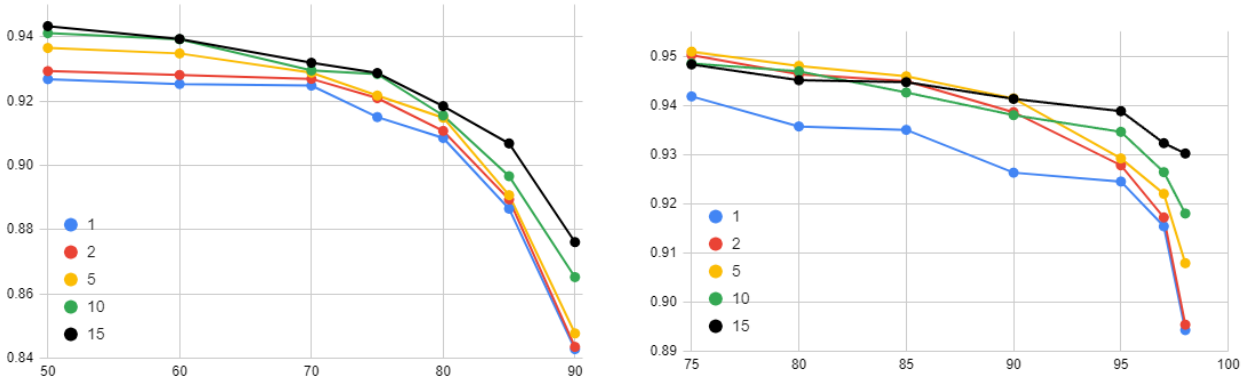
For the models pruned by neurons, increasing the pruning interval will result in a slightly higher test accuracy. The model with a pruning interval of 15 demonstrated a higher test accuracy compared to models with other pruning intervals. But for the models pruned by norms, increasing the pruning interval results in a slight improvement in test accuracy. Specifically, the model with a pruning interval of 15 exhibited higher test accuracy compared to models with other pruning intervals. On the other hand, when introducing the neuron entropy to the vitality, NVP models with higher pruning interval achieves better result. Increasing the pruning interval from 1 to 15 boosts the model performance by around 2%.

Table 13: Comparison of Pruning Interval (PI) with Weight Sparsity of NVP-1.

PI	75	80	85	90	95	97	98
1	94.18	93.57	93.50	92.63	92.45	91.54	89.43
2	95.02	94.63	94.49	93.86	92.78	91.72	89.54
5	**95.09**	**94.80**	**94.59**	**94.14**	92.92	92.20	90.79
10	94.85	94.69	94.26	93.80	93.46	92.64	91.80
15	94.83	94.51	94.47	94.13	**93.88**	**93.23**	**93.02**

Table 14: Comparison of Pruning Interval (PI) with Neural Sparsity of NVP-2.

PI	50	60	70	75	80	85	90
1	93.32	92.89	92.38	91.67	91.12	88.64	84.51
2	93.35	93.11	92.67	91.84	91.22	88.79	84.58
5	93.67	93.31	92.84	92.24	91.38	89.05	84.88
10	93.82	93.47	93.05	92.46	91.43	89.62	87.22
15	**93.94**	**93.66**	**93.34**	**92.76**	**91.83**	**91.13**	**87.75**



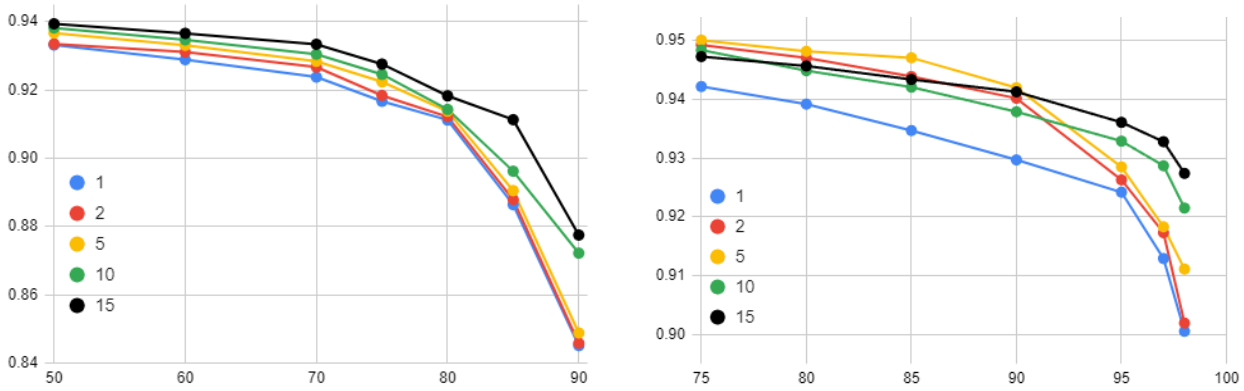
(a) Neural Sparsity

(b) Weight Sparsity

Figure 12: Comparison of Pruning Interval with different sparsity of NVP-1.

Table 15: Comparison of Pruning Interval (PI) with Weight Sparsity of NVP-2.

PI	75	80	85	90	95	97	98
1	94.22	93.92	93.47	92.97	92.42	91.29	90.05
2	94.93	94.71	94.39	94.02	92.63	91.73	90.19
5	**95.01**	**94.82**	**94.71**	**94.20**	92.85	91.83	91.11
10	94.84	94.49	94.21	93.79	93.29	92.87	92.15
15	94.73	94.57	94.34	94.13	**93.61**	**93.28**	**92.74**



(a) Neural Sparsity

(b) Weight Sparsity

Figure 13: Comparison of Pruning Interval with different sparsity of NVP-2.