SPvR: Structured Pruning via Ranking

Atif Hassan[1] **Jiaul H. Paik**[1] **Swanand Khare**[2]

[1]Department of Artificial Intelligence, IIT Kharagpur, Kharagpur, West Bengal, India
[2]Department of Mathematics, IIT Kharagpur, Kharagpur, West Bengal, India

Abstract

Deep neural networks have achieved state-of-the-art performance in multiple domains but are increasingly resource-intensive, limiting their deployment on constrained devices. We introduce Structured Pruning via Ranking (SPvR), a novel structured pruning approach to address this challenge for classification tasks. SPvR prunes pre-trained networks in terms of function composition and network width while adhering to a user-specified parameter budget. Our method leverages local grouping and global ranking modules to generate smaller yet effective networks tailored to a given dataset and model. Finally, we train the pruned networks from scratch, instead of fine-tuning. Our evaluations demonstrate that SPvR significantly surpasses existing state-of-the-art pruning methods on benchmark datasets, using standard architectures. Even with a 90% reduction in size, SPvR's sub-networks experience a minimal drop in test accuracy ($< 1\%$) while on ImageNet1K, we outperform all baselines by achieving $< 1\%$ Top-5 accuracy drop when pruning 70% of ResNet50 parameters. Additionally, when compared to MobileNetV3, an SPvR pruned network improves the Top-1 accuracy by 3.3% with 20% less parameters. Furthermore, we empirically show that SPvR achieves reduced inference latency, underscoring its practical benefits for deploying neural networks on resource-constrained devices.

1 INTRODUCTION

Highly over-parameterized, deep neural networks have shown remarkable proficiency in learning effective representations in diverse domains such as computer vision [Wang et al., 2023, Yuan et al., 2021, He et al., 2016], natural language processing [Wu et al., 2023, Radford et al., 2019, Devlin et al., 2018] and speech [Radford et al., 2023, Baevski et al., 2020]. However, their deployment on commercial, especially low-end, hardware is impeded by their substantial size, leading to large memory requirements and extended inference times. Consequently, recent deep-learning research has pivoted toward methods for reducing model size. These include network pruning [Fang et al., 2023, Blalock et al., 2020, Li et al., 2016, Molchanov et al., 2016, LeCun et al., 1989], low-rank weight approximation [Li et al., 2023a, Swaminathan et al., 2020, Denton et al., 2014], weight quantization [Li et al., 2023b, Gong et al., 2020, Courbariaux et al., 2016], and knowledge distillation [Liang et al., 2023, Pan et al., 2020, Hinton et al., 2015], with pruning receiving notable attention for its effective balance between size reduction and performance.

Network pruning is generally classified into two approaches: unstructured and structured. The former involves masking individual weights, leading to sparse models [Frankle and Carbin, 2018], while the latter prunes entire neurons or channels, resulting in dense sub-networks [Li et al., 2016]. Sparse models often necessitate specialized hardware for efficiency [Han et al., 2016], whereas dense sub-networks can reduce both inference time and storage requirements on conventional hardware. However, most structured pruning techniques are model-specific and tailored to particular network architectures such as Convolutional Neural Networks [Li et al., 2022, Sui et al., 2021, Luo and Wu, 2020] or language models [Ma et al., 2023, Hou et al., 2020, McCarley et al., 2019]. Many of these methods either overlook the dataset's role in pruning [He et al., 2019, Li et al., 2016] or require training the original large model to identify the optimal sub-network [Xia et al., 2022, Hou et al., 2020, Fan et al., 2019]. Furthermore, pruning algorithms need to generate optimal sub-networks tailored to user-defined parameter budgets, considering the varying sizes of the target deployment devices [Tiwari et al., 2021, Dupont et al., 2021].

In response to these challenges, we introduce SPvR (Struc-

tured Pruning via Ranking), a novel, structured pruning approach tailored for classification tasks. SPvR is applicable to any pre-trained network, without the need to train the original extensive model. It employs a local grouping module that partitions similar neurons layer by layer for efficient pruning alongside a global ranking module that assesses the overall importance of these groups. Based on a predefined parameter budget, the least significant groups are eliminated, resulting in a dense, smaller sub-network with reduced depth and parameter count. This layer reduction which typically occurs at high pruning rates is vital for lowering inference latency, as structured pruning often leads to irregular layer widths, which are not optimal for GPU utilization. After pruning, the resulting dense sub-network is re-initialized and trained from scratch. We choose to retrain instead of fine-tuning as the observations made in the seminal work by Liu et al. [2018] demonstrate that for structured pruning, the resultant architecture is more important than the retained weights.

Extensive evaluations on benchmark datasets, including CIFAR10 [Krizhevsky et al., 2009], Tiny ImageNet [Le and Yang, 2015], ImageNet Deng et al. [2009], and CityScapes [Cordts et al., 2016], using distinct architectures, namely, VGG16 [Simonyan and Zisserman, 2014], ResNet34 [He et al., 2016], ResNet50 [He et al., 2016], and SegNet [Badrinarayanan et al., 2017], demonstrate that SPvR's recommended shallower sub-networks significantly outperform other methods across all datasets and pruning rates, while also achieving notably lower inference latency due to their reduced depth. Furthermore, we exhibit the applicability of our method to resource-constrained devices by demonstrating that an SPvR pruned network with 20% fewer parameters significantly outperforms MobileNetV3 Howard et al. [2019], a model specifically designed for mobile phone CPUs.

1.1 OUR CONTRIBUTIONS:

- **Development of an Efficient Pruning Algorithm:** Our primary contribution is the development of Structured Pruning via Ranking (SPvR), a novel, structured pruning approach. SPvR is capable of effectively generating shallow and dense sub-networks tailored to specific datasets, pre-trained models, and user-defined parameter budgets. Unlike some state-of-the-art methods such as OTOv2Chen et al. [2023], SPvR only necessitates backpropagation on the pruned model (retraining) rather than the original, massive network.
- **Comprehensive Evaluation Demonstrating Enhanced Performance and Efficiency:** Through extensive experimental evaluations, we demonstrate that sub-networks generated by SPvR, when trained from scratch, consistently outperform existing pruning methods as well as hand-crafted architectures. This superior performance is observed across a range of benchmark datasets. Furthermore, a significant contribution of our work is the achievement of reduced model inference latency. This aspect is crucial for deploying neural networks in resource-constrained environments, aligning with the growing need for efficient and fast computational models.

2 RELATED WORK

Structured pruning strategies for convolutional neural networks have seen a variety of approaches [He and Xiao, 2023]. Weight-dependent methods, including norm-based filter pruning (ℓ_1 and ℓ_2 norms), focus on filter removal based on their norms Li et al. [2016], while Filter Pruning via Geometric Median (FPGM) targets filters near the geometric median of a layer He et al. [2019]. However, these methods overlook the data's influence on the final architecture. Activation-based pruning, such as HRank Lin et al. [2020] and CHIP Sui et al. [2021], remove filters by analyzing activation ranks or cross-channel correlations. Techniques like ThiNet Luo et al. [2017] and NISP Yu et al. [2018] determine filter importance through reconstruction error or feature ranking, respectively. CURL Luo and Wu [2020] employs KL-divergence for channel masking and global filter removal. These methods, however, suffer from being time-consuming (e.g., HRank), neglecting output layer changes (CHIP), or using suboptimal ranking (ThiNet, CURL). Regularization approaches like Network Slimming Liu et al. [2017] target filters with minimal scaling factors in batch normalization layers, whereas optimization-based methods, such as those using Taylor Expansion Molchanov et al. [2019], rank filters by weight and gradient impacts. Random Channel Pruning (RCP) integrates the Lottery Ticket Hypothesis, selecting pruned models for further training [Li et al., 2022, Frankle and Carbin, 2018]. Both regularization and optimization require initial full model training. OTOv2 (Only Train Once) Chen et al. [2023] stands out by pruning both vision and language models. However, these techniques necessitate training the original, large network from scratch to identify pruned sub-networks. Unlike most structured pruning methods our proposed technique focuses on depth reduction to lower inference latency, offering a significant advantage in resource efficiency by retraining only the pruned networks.

3 SPVR: AN EFFICIENT MODEL PRUNING ALGORITHM

Structured Pruning via Ranking (SPvR) is a novel, network pruning algorithm that assesses the global importance of neurons/filters through forward passes on the original network, targeting the least important ones for pruning. This approach reduces parameter count and the network's depth,

leading to faster sub-networks during training and inference compared to those from other pruning methods. SPvR comprises two key components: a local grouping module that clusters similar neurons/filters within layers and a global ranking module that prioritizes these groups for efficient pruning.

3.1 RANKING MODULE

Let f_θ be an L layer neural network parameterized by θ where $\theta = \{\theta^1, \theta^2, \cdots, \theta^L\}$. Here, θ^i represents the parameters of layer i while θ^i_j denotes the j-th neuron/filter at layer i. Given a dataset $\mathcal{D} = \{(x_0, y_0), \cdots, (x_n, y_n)\}$ composed of input and output pairs x_k and y_k, respectively, the task of training f_θ is solving the following minimization problem:

$$\min_\theta \frac{1}{n} \sum_{k=1}^{n} E(y_k, f_\theta(x_k)) \tag{1}$$

where E is the error function, $f_\theta(x_k) \in \mathbb{R}^c$ is the softmax final output of f_θ for a given input x_k and c is the number of classes. A neuron is important if its removal significantly changes the output of f_θ. Let $f_{\mathcal{M}^i_j(\theta)}$ denote the network after masking j-th neuron/filter in the i-th layer from the original network. More precisely, $\mathcal{M}^i_j$ is defined as follows:

$$\mathcal{M}^i_j(\theta) = \begin{cases} \theta^m_r & \text{if } m \neq i \vee r \neq j \\ 0 & \text{otherwise} \end{cases}$$

Masking θ^i_j may lead to one of the following cases:

1) $\|f_\theta(x_k) - f_{\mathcal{M}^i_j(\theta)}(x_k)\|_2 < \epsilon$
$\underset{p}{\text{argmax}}\ f_\theta(x_k)_p = \underset{p}{\text{argmax}}\ f_{\mathcal{M}^i_j(\theta)}(x_k)_p$

2) $\|f_\theta(x_k) - f_{\mathcal{M}^i_j(\theta)}(x_k)\|_2 \geq \epsilon$
$\underset{p}{\text{argmax}}\ f_\theta(x_k)_p = \underset{p}{\text{argmax}}\ f_{\mathcal{M}^i_j(\theta)}(x_k)_p$

3) $\|f_\theta(x_k) - f_{\mathcal{M}^i_j(\theta)}(x_k)\|_2 < \epsilon$
$\underset{p}{\text{argmax}}\ f_\theta(x_k)_p \neq \underset{p}{\text{argmax}}\ f_{\mathcal{M}^i_j(\theta)}(x_k)_p$

4) $\|f_\theta(x_k) - f_{\mathcal{M}^i_j(\theta)}(x_k)\|_2 \geq \epsilon$
$\underset{p}{\text{argmax}}\ f_\theta(x_k)_p \neq \underset{p}{\text{argmax}}\ f_{\mathcal{M}^i_j(\theta)}(x_k)_p$

Here $\epsilon \to 0$. If removing θ^j_i results in case 1, it is considered the least important neuron/filter. On the other hand, if the removal of θ^i_j leads to case 4, it is considered the most important neuron/filter. Following the above cases, under the i.i.d. assumption [Molchanov et al., 2019], the importance of the j-th neuron in the i-th layer is determined by:

$$\mathcal{I}^i_j = \sum_{k=1}^{n} L\left(f_\theta(x_k), f_{\mathcal{M}^i_j(\theta)}(x_k)\right) \tag{2}$$

where, $L = I + \left|f_\theta(x_k)_q - f_{\mathcal{M}^i_j(\theta)}(x_k)_q\right|$

$$I = \begin{cases} 1 & \text{if } \underset{p}{\text{argmax}}\ f_\theta(x_k)_p \neq \underset{p}{\text{argmax}}\ f_{\mathcal{M}^i_j(\theta)}(x_k)_p \\ 0 & \text{otherwise} \end{cases}$$

Here, $q = \underset{p}{\text{argmax}}\ f_\theta(x_k)_p$ indicates the class predicted by $f_\theta(x_k)$. The inclusion of I in the scoring function L is motivated by the idea that if masking a neuron/filter leads to a misclassification, it should be considered crucial for the task and assigned a higher importance value. Given that $\left|f_\theta(x_k)_q - f_{\mathcal{M}^i_j(\theta)}(x_k)_q\right| \leq 1$, assigning a value less than 1 diminishes the significance of misclassification, while any value above 1 has the same effect on the final ranking (the scores may vary, but the rank remains consistent). Therefore, in our experiments, we assign 1 for misclassifications. If two different neurons/filters result in the same number of misclassifications including no misclassifications, as indicated by I, the tie is resolved by $\left|f_\theta(x_k)_q - f_{\mathcal{M}^i_j(\theta)}(x_k)_q\right|$, which assesses the deviation in the predicted class's probability before and after masking.

3.1.1 SPvR Ranking Function vs KL Divergence

Sub-optimal ranking criteria have previously been employed to gauge the impact of masking a neuron/filter on a network's final output, thereby estimating its importance. CURL [Luo and Wu, 2020], a state-of-the-art method, uses a criterion based on KL-divergence. We demonstrate the advantage of our approach over KL-divergence with an example, further supported by a detailed empirical evaluation in Appendix A.

Consider a three class classification task where a pretrained network's final layer has three neurons representing classes 0, 1 and 2, respectively, with a softmax function applied to the output. For a sample input x yielding $f_\theta(x) = [0.1, 0.3, 0.6]^T$ from the model, the sample is classified as belonging to class 2. Now, for three neurons indexed by $j = 1$, $j = 2$ and $j = 3$ in the i-th layer, let $f_{\mathcal{M}^i_1(\theta)}(x) = [0.1, 0.6, 0.3]^T$, $f_{\mathcal{M}^i_2(\theta)}(x) = [0.01, 0.1, 0.89]^T$ and $f_{\mathcal{M}^i_3(\theta)}(x) = [0.1, 0.8, 0.1]^T$. The neuron at $j = 3$ is the most crucial since its removal leads to misclassification along with a large change in output followed by $j = 1$ (misclassification) and $j = 2$ (no effect on the network's classification accuracy). Let $\theta' = \mathcal{M}^i_1(\theta)$, $\theta'' = \mathcal{M}^i_2(\theta)$ and $\theta''' = \mathcal{M}^i_3(\theta)$. We calculate the importance of neurons at $j = 1$, $j = 2$ and $j = 3$ using both KL divergence and our proposed method.

$$\mathrm{KL}\left(f_\theta(x) \parallel f_{\theta'}(x)\right) = 0.20 \quad L\left(f_\theta(x), f_{\theta'}(x)\right) = 1.3$$
$$\mathrm{KL}\left(f_\theta(x) \parallel f_{\theta''}(x)\right) = 0.32 \quad L\left(f_\theta(x), f_{\theta''}(x)\right) = 0.29$$
$$\mathrm{KL}\left(f_\theta(x) \parallel f_{\theta'''}(x)\right) = 0.78 \quad L\left(f_\theta(x), f_{\theta'''}(x)\right) = 1.5$$

In this scenario, the KL divergence criterion incorrectly assigns greater importance to the neuron at $j = 2$ in comparison to $j = 1$, whereas our proposed ranking function accurately identifies the correct ranking.

3.2 GROUPING MODULE

Determining $\mathcal{I}_j^i$ for individual neurons/filters in a wide and deep network is computationally expensive. In order to make SPvR more compute efficient, we group layerwise similar neurons/filters so that $\mathcal{I}_j^i$ estimates the importance of the j-th group in the i-th layer where the size of the group is a hyperparameter denoted by d. Let $S \in \mathbb{R}^{n \times m}$ be a network's intermediate layer output. The similarity between two neurons/filters is measured by the correlation between their output activations/channels given by the following correlation matrix, $C \in \mathbb{R}^{m \times m}$:

$$C = \hat{S}^T \hat{S} \tag{3}$$

$$\text{where, } \hat{S} = \bar{S} \frac{1}{\sqrt{n}} \mathrm{Diag}\left(\bar{S}^T \bar{S}\right)^{-1/2}$$
$$\text{and, } \bar{S} = S - \frac{1}{n} \mathbb{1}_n \mathbb{1}_n^T S$$

Algorithm 1 Grouping Module

Input: Correlation Matrix C, group-size d
Let $\mathcal{G} = \{\}$ (set of sets) , $\mathcal{Q} = \{1, \cdots, m\}$
for $j \in \mathcal{Q}$ **do**
  $\mathcal{T} =$ Indices corresponding to top d values in $C_{(j,:)}$
  $\mathcal{G}.\mathrm{append}(\mathcal{T})$
  $C_{(:,\mathcal{T})} = -\infty$
  $\mathcal{Q} = \mathcal{Q} \setminus \mathcal{T}$
end for
Output: $\mathcal{G}$

Here, $\mathrm{Diag}\left(\bar{S}^T \bar{S}\right)$ is a diagonal matrix where the diagonal entries are equal to the diagonal entries of $\bar{S}^T \bar{S}$. Using C, the neurons are partitioned into mutually exclusive groups using Algorithm 1 where, $C_{(j,:)}$ denotes the j-th row and $C_{(:,\mathcal{T})}$ denotes all columns indexed in $\mathcal{T}$. In the case of CNNs, the output of the i-th layer is denoted by a tensor $S^i \in \mathbb{R}^{n \times m \times w \times h}$ where n is the number of samples, m is the number of output channels and w, and h are the width and height of each output channel, respectively. In such a scenario, the tensor is first reduced to a matrix, $S_{pq}^i \in \mathbb{R}^{n \times m}$ as follows:

$$S_{pq}^i = \sum_{r=1}^{w} \sum_{t=1}^{h} |S_{pqrt}^i| \tag{4}$$
$$\forall p \in \{1, 2, \cdots, n\} \text{ and } \forall q \in \{1, 2, \cdots, m\}$$

Applying Eqn. 3 and Algorithm 1 on S_{pq}^i yields the groups for the current layer. Although Eqn. 3 describes a linear correlation, we find that it works well in practice while being lightweight to compute.

Once the neurons/filters are grouped, the importance of each group across all layers is assessed using the ranking module. These groups are globally sorted throughout the model, and the least important ones are pruned away until the desired parameter count is reached. At high pruning rates, this process may result in the pruning of entire layers, as we do not impose a minimum threshold for layer pruning. Such a scenario, known as layer collapse, leads to entire layers being removed, rendering a network untrainable [Tanaka et al., 2020]. This poses a challenge for methods that rely on preserved weights for subsequent fine-tuning. In contrast, our approach views the pruned sub-network as a new model and initializes (re-initialization) its parameters through standard initialization techniques. After re-initialization, the pruned networks undergo training from scratch, a strategy recommended by Liu et al. [2018]. Consequently, layer collapse is a beneficial feature of our method contributing significantly to reducing inference latency in the pruned models at high pruning rates. Ultimately, SPvR autonomously identifies the optimal number of layers and the precise layerwise width of the final pruned architecture for the specific dataset, model, and parameter budget.

SPvR Pruning Time Complexity: The grouping module requires only a single forward pass through the original large model to compute layerwise groups across the entire network while the ranking module requires $\sum_{i=1}^{L} \left\lceil \frac{m_i}{d} \right\rceil$ number of forward passes where m_i is the number of neurons/filters at the i-th layer of an L layer neural network and d is the group size, a hyper-parameter. Hence, the time complexity of our pruning algorithm, similar to most other pruning algorithms is $O(n)$ where n is the number of samples.

4 EXPERIMENTAL SETUP

4.1 DATASETS AND MODELS

We evaluate our method on four datasets ranging from small to large scale: CIFAR10 [Krizhevsky et al., 2009] ($50K$ training samples, $10K$ test samples and 10 classes), Tiny ImageNet ($100K$ training samples, $10K$ test samples and 200 classes) [Le and Yang, 2015] and ImageNet1K Deng et al. [2009] ($1200K$ training samples, $50K$ validation samples and 1000 classes). For CIFAR10, Tiny ImageNet and ImageNet1K, we utilize VGG16 with batch normalization.

Table 1: Comparison of the **Top-1** and **Top-5** accuracy scores on the CIFAR10, Tiny ImageNet and ImageNet1K datasets, for multiple pruning methods at different levels of pruning for the VGG16, ResNet34 and ResNet50 networks, respectively. Higher values are better. Bold values indicate the best score. The *Param* column indicates the percentage of parameters removed from the original model.

			Methods										
Dataset	Model	Param	Base	ℓ_1	ℓ_2	Taylor	FPGM	RCP	HRank	CURL	NISP	OTOv2	SPvR
CIFAR 10	VGG16 (Top-1)	00%	94.25	-	-	-	-	-	-	-	-	-	-
		70%	-	93.53	93.45	93.23	92.72	86.68	92.84	94.18	92.51	93.20	**94.21**
		80%	-	92.60	92.99	92.87	91.63	85.90	93.09	93.89	91.48	92.70	**94.33**
		90%	-	92.05	91.64	91.55	90.85	84.00	92.36	93.49	90.28	91.07	**94.43**
		95%	-	90.20	90.36	90.37	88.41	84.09	91.44	92.14	88.77	91.03	**93.64**
		98%	-	87.32	87.64	87.22	87.09	83.78	91.10	91.66	87.00	87.88	**92.60**
Tiny Image Net	Res Net34 (Top-1)	00%	63.02	-	-	-	-	-	-	-	-	-	-
		70%	-	60.95	60.50	60.65	60.73	58.18	57.90	58.38	57.06	58.72	**62.89**
		80%	-	58.73	58.51	59.40	59.14	56.52	55.30	56.85	54.76	59.05	**62.55**
		90%	-	56.42	56.78	57.06	56.22	54.21	52.85	55.13	52.62	55.94	**60.36**
		95%	-	55.07	54.96	55.16	52.88	50.65	50.55	48.33	50.82	52.10	**58.95**
		98%	-	52.15	51.18	51.59	50.42	45.46	46.83	39.49	45.10	47.73	**55.84**
Tiny Image Net	Res Net34 (Top-5)	00%	83.22	-	-	-	-	-	-	-	-	-	-
		70%	-	81.59	81.28	81.68	81.68	80.51	80.27	80.68	79.39	82.04	**83.08**
		80%	-	80.65	80.58	80.61	80.39	79.26	78.85	80.22	77.56	82.18	**82.73**
		90%	-	79.30	79.40	79.79	78.70	78.08	77.02	79.81	76.78	81.00	**82.34**
		95%	-	79.16	78.32	78.86	76.46	75.57	75.68	74.80	75.35	78.66	**81.59**
		98%	-	76.68	77.42	77.42	76.13	72.19	73.26	67.46	71.89	74.66	**80.72**
Image Net1K	Res Net50 (Top-1)	00%	76.32	-	-	-	-	-	-	-	-	-	-
		40%	-	73.82	73.89	71.69	74.83	75.13	74.98	-	75.43	-	**75.58**
		70%	-	70.07	70.91	-	-	-	69.10	73.39	-	72.20	**73.70**
		80%	-	68.11	69.00	-	-	-	-	-	-	70.10	**72.18**
Image Net1K	Res Net50 (Top-5)	00%	92.89	-	-	-	-	-	-	-	-	-	-
		40%	-	91.76	91.82	91.01	92.32	92.52	92.33	-	92.45	-	**92.69**
		70%	-	89.23	89.57	-	-	-	89.58	91.46	-	90.70	**91.91**
		80%	-	88.06	88.43	-	-	-	-	-	-	89.30	**90.58**

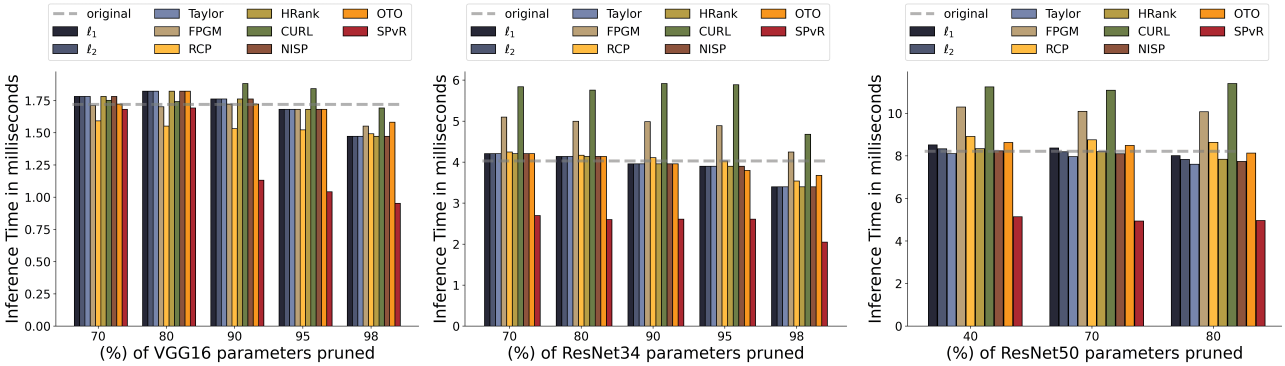
[Simonyan and Zisserman, 2014], ResNet34 and ResNet50 [He et al., 2016] models, respectively. The selection of datasets, models, and their specific combinations is based on the structured pruning literature [Hoang and Liu, 2023, Goyal et al., 2020, Krishnan et al., 2019].

4.2 BASELINES

For CIFAR10 and Tiny ImageNet, we compare against ℓ_1 norm, ℓ_2 norm, Taylor expansion, HRank, FPGM, CURL, Random Channel Pruning (RCP), NISP, and OTOv2. For ImageNet1K, we compare against the same methods but report the results as detailed in their respective publications.

4.3 IMPLEMENTATION DETAILS

Our experiments were conducted using PyTorch v2.3.1 [Paszke et al., 2019] on an NVIDIA A100 GPU. Since no pre-trained models are available for CIFAR10 and Tiny-ImageNet, we generate our own "pre-trained" versions of VGG16 and ResNet34 by training them from scratch to achieve the maximum reported accuracy on their respective datasets. On the other hand, we use an ImageNet1K pre-trained ResNet50 model for pruning. For training VGG16, ResNet34 and ResNet50, we employed SGD with a momentum of 0.9 as the optimizer along with the cosine annealing scheduler [Loshchilov and Hutter, 2016]. The learning rate for each experiment was determined through a grid search within the range $[0.0001, 1.0]$. Training durations were set at 200 epochs for VGG16 with a batch size of 128, 100 epochs for ResNet34 with a batch size of 512 and 100 epochs for ResNet50 with a batch size of 256. The group size, d, was set to 2, 4 and 8 for CIFAR10, Tiny ImageNet1K and ImageNet experiments, respectively. The pruning library by Fang et al. [2023] was utilized to implement the baseline methods, except for OTOv2. On the ImageNet1K dataset, we compare SPvR's performance against the results reported



(a) (b) (c)

Figure 1: Inference latency for (a) VGG16, (b) ResNet34 and (c) ResNet50. The results are reported in terms of milliseconds averaged over 1500 runs with a standard deviation of $0.01 - 0.001$. Smaller values are better.

by the baseline methods. Further implementation information is provided in Section D in the Appendix. Details on the architectures recommended by SPvR and their corresponding size on disk at various pruning percentages are available in Section C in the Appendix while architecture-specific pruning implementations are provided in Section B in the Appendix.

5 RESULTS

5.1 PERFORMANCE AGAINST BASELINES

Table 1 showcases the top-1 accuracy for CIFAR10 and both top-1 and top-5 accuracies for Tiny ImageNet and ImageNet1K. The results clearly indicate that SPvR surpasses all other pruning methods at every pruning stage across all three models by a significant margin. Notably, our method enhances the performance of the VGG16 model when 90% of its parameters are pruned while maintaining a minimal accuracy drop of less than 1% even after a 95% reduction in parameters. On the Tiny ImageNet dataset, our pruned ResNet34 networks exhibit less than a 1% drop in top-5 accuracy, even with 90% of the model pruned, far surpassing every other pruning baseline. Similarly, our 70% pruned ResNet50 achieves less than 1% drop in Top-5 accuracy on the ImageNet1K dataset while also displaying the least drop in Top-1 accuracy with increasing pruning rate. As observed by Li et al. [2022], ℓ_1 and ℓ_2 pruning methods are incredibly tough-to-beat baselines when trained using the correct set of hyper-parameters.

5.2 RETRAINING VS FINE-TUNING

Is retraining critical for structured pruning? Although Liu et al. [2018] answer in the affirmative, we further investigate this phenomenon by comparing the performance of training from scratch against fine-tuning on all chosen datasets and models. For fair evaluation, we choose the highest pruning rate where depth reduction has not occurred since layer removal, in the case of fine-tuning, requires learning new connections from scratch. Table 2 demonstrates that fine-tuning performs worse than training from scratch for all networks when trained for the same number of epochs with the best learning rates selected using grid-search. The gap is more pronounced for higher pruning rates which is in line with the observations of Liu et al. [2018].

Table 2: A comparison between Fine-Tuning (FT) a pruned model versus Training From Scratch (TFS). All accuracy scores are reported in $\%$.

Dataset	Model	Params	FT	TFS
CIFAR10	VGG16	60%	93.40	**94.25**
TinyImageNet (Top-1)	ResNet34	60%	61.60	**62.91**
TinyImageNet (Top-5)	ResNet34	60%	82.70	**83.11**
ImageNet1K (Top-1)	ResNet50	90%	67.77	**69.37**
ImageNet1K (Top-5)	ResNet50	90%	87.91	**89.08**

5.3 INFERENCE LATENCY

We also examine model inference latency, defined as the time it takes for a model to make a prediction for a single sample. The inference latency is significantly affected by the network's layerwise width, where non-standard layer structures (not a power of two) minimally impact the prediction time for a single sample. We chose not to focus on FLOPs count since models of similar sizes can have identical FLOPs but vastly different inference latencies [Liu et al., 2021]. The inference latency, measured in milliseconds and averaged over 1500 runs, is displayed in Fig. 1. Techniques that globally rank and remove neurons or filters

Table 3: Comparison of the **Top-1** and **Top-5** accuracy scores on the CIFAR10, Tiny ImageNet and ImageNet1K datasets, for multiple pruning methods at a single level of pruning for the VGG16, ResNet34 and ResNet50 networks, respectively. Each pruned model is trained from scratch. Higher values are better. Bold values indicate the best score. The *Param* column indicates the percentage of parameters removed from the original model.

			Methods								
Dataset	Model	Param	ℓ_1	ℓ_2	Taylor	FPGM	RCP	HRank	CURL	NISP	SPvR
CIFAR10	VGG16 (Top-1)	90%	92.70	92.12	91.65	91.00	88.07	93.10	93.69	90.01	**94.43**
TinyImageNet	ResNet34 (Top-1)	90%	56.20	56.88	57.16	56.23	54.25	52.35	55.43	52.93	**60.36**
	ResNet34 (Top-5)	90%	79.10	79.30	79.89	78.60	78.28	77.34	80.00	76.97	**82.34**
ImageNet1K	ResNet50 (Top-1)	80%	71.50	71.45	-	-	-	-	-	-	**72.18**
	ResNet50 (Top-5)	80%	90.30	90.40	-	-	-	-	-	-	**90.58**

tend to create more irregular layer widths compared to layerwise pruning methods, leading to slower inference times such as in the CURL-based sub-networks compared to the original, unpruned networks. Conversely, SPvR-generated sub-networks exhibit significantly reduced inference latency, benefiting from decreased depth even with irregular layer widths. Specifically, at the 98% pruning level for VGG16, SPvR-generated sub-networks demonstrate remarkably low latency. For ResNet34 and ResNet50, our recommended sub-networks consistently show lower inference latency than those generated by other methods across all pruning stages.

Table 4: A 94% pruned ResNet50 against MobileNetV3 on ImageNet1K. Param - model parameters in Million.

Model	Param (M)	Top-1 Acc.
MobileNetV3-small	2.4	65.40
MobileNetV3-minimal	2.0	61.90
ResNet50-pruned	1.6	65.20

5.4 APPLICATION TO MOBILE PHONES

The MobileNet [Howard et al., 2017] family of networks are highly efficient models specifically designed to run on mobile phone CPUs with MobileNetV3 [Howard et al., 2019] being the latest model. These networks have been developed through hours of careful research combined with automated architecture search methods such as Neural Architecture Search. Instead, we advocate that SPvR can be used to quickly find an efficient architecture for a given dataset and parameter budget. To demonstrate this, we prune a ResNet50 model down from 25.6M parameters to 1.6M, train it on the ImageNet dataset and compare the results against the MobileNetV3 networks. Table 4 shows that our pruned network performs at par with MobileNetV3-small while having 33% fewer parameters and achieves 3.3% better Top-1 accuracy than MobileNetV3-minimal while having 20% fewer parameters.

5.5 APPLICATION TO IMAGE SEGMENTATION

We trained a SegNet[1] model having a VGG16 encoder and decoder backbone on the CityScapes dataset [2]. The CityScapes dataset contains 5000 annotated images with 20 labels. We consider 20 samples per class and a group size, $d = 4$ to generate the groupings and rankings. Since a semantic segmentation task is essentially a classification task over each pixel, we sum the L term in Eqn. 2 over all pixels, i.e.,

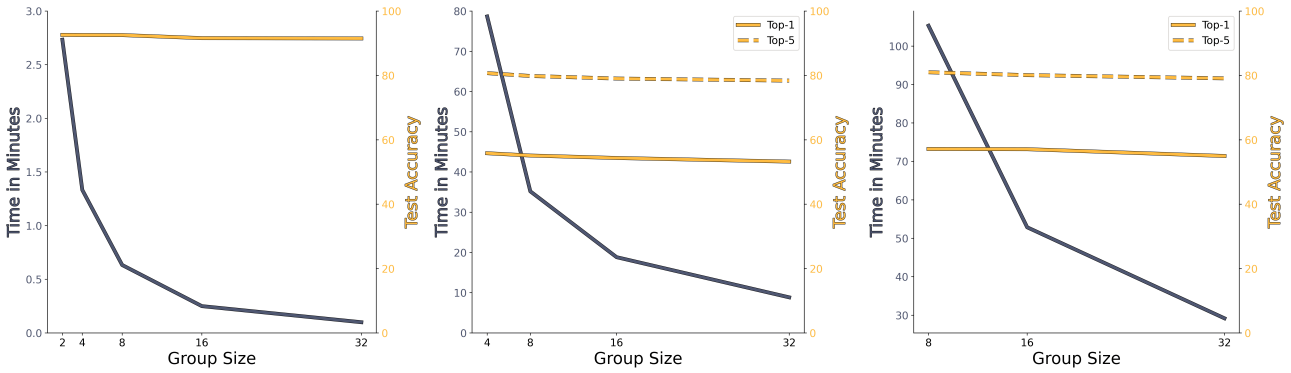
$$L = \sum_{u=1}^{w \times h} I + \left| f_\theta(x_k)_q - f_{\mathcal{M}_j^i(\theta)}(x_k)_q \right|$$

where w and h are the width and height of the image. The original SegNet model achieves 51.2% IoU whereas its 80% pruned version using SPvR achieves 46.5% IoU.

6 ABLATION STUDY

6.1 RETRAINING VS ARCHITECTURE

Does SPvR's success stem from the architecture of the pruned sub-networks or retraining them from scratch? To answer this question and disentangle the performance benefits



(a) (b) (c)

Figure 2: (a) Change in Top-1 accuracy and time required to prune 98% of the VGG16 network with increasing group size, (b) Change in Top-1, Top-5 accuracy and time required to prune 98% of the ResNet34 network with increasing group size (c) Change in Top-1, Top-5 accuracy and time required to prune 98% of the ResNet50 network with increasing group size

of training from scratch from our generated sub-networks, we re-run all experiments for each baseline, training them from scratch except for OTOv2 which already re-trains the original pre-trained model. We produce the results on a single but high pruning rate for all datasets as it should be enough to demonstrate the efficacy of our sub-networks.

As per Table 3, the sub-networks generated by each method show a slight improvement in accuracy when trained from scratch except for RCP on the CIFAR10 dataset where its score increases by 4%. On the ImageNet1K dataset, we report results for all baselines as made available by the corresponding authors. For this particular experiment, we only train the models generated by ℓ_1 and ℓ_2 norm methods as they form one of the strongest baselines. We find that both sub-networks benefit from retraining by up to 3% but are yet unable to outmatch the performance of SPvR. In general, none of the methods are able to achieve comparable performance to SPvR indicating the importance of the sub-networks generated by our method.

6.2 IMPACT OF GROUP SIZE ON PERFORMANCE AND PRUNING TIME

To understand the impact of group size d, we observe the change in the accuracy of pruned networks and the time required for pruning under varying values of d. We perform this ablation study for only the maximum level of pruning as it is the worst-case scenario regarding both accuracy and time. Hence, we ablate the VGG16, ResNet34 and ResNet50 networks at a pruning rate of 98% for $d = \{2, 4, 8, 16, 32\}$, $d = \{4, 8, 16, 32\}$ and $d = 8, 16, 32$, respectively. It is expected that values of d closer to 1 produce fine-grained pruning results but at the cost of slower rankings. According to Figs. 2a, 2b and 2c, our hypothesis is indeed validated with SPvR generally being robust to group size as the top-1 accuracy drop for $d = (2-32)$ for VGG16 is about 1%, the top-5 accuracy drop for $d = (4-32)$ for ResNet34 is about 2% and the top-5 accuracy drop from $d = (8-32)$ for ResNet50 is about 1%. At the same time, for $d = 32$, SPvR can prune VGG16, ResNet34 and ResNet50 under **6 seconds**, **8 minutes** and **30 minutes**, respectively.

7 CONCLUSION

We introduced Structure Pruning via Ranking (SPvR), an efficient model pruning algorithm that efficiently prunes vision and language models without requiring backpropagation on the original, pre-trained models. Our approach leverages a novel combination of local layerwise grouping and global ranking to prune less significant neuron or filter groups guided by user-defined parameter budgets. This process results in the generation of compact sub-networks with reduced depth and parameter counts. Our comprehensive evaluation across various benchmark datasets and models confirms SPvR's superior performance. The algorithm outpaces both existing pruning methods as well as hand-crafted architectures in terms of accuracy and achieves significant reductions in inference latency. SPvR is particularly well-suited for deployment in resource-constrained environments, such as edge devices and mobile platforms, where model size and inference speed are critical. For example, lightweight image classification models that can run not only on the latest flagship mobile devices but also on older generation phones. It also holds promise for large-scale ML services (MLaaS), where reducing computational overhead can lead to significant cost savings. For example, faster and lighter image segmentation models. Thus, our findings underscore the potential of SPvR in addressing the deployment challenges of large neural networks on resource-constrained devices.

References

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

Tianyi Chen, Luming Liang, Tianyu Ding, Zhihui Zhu, and Ilya Zharkov. Otov2: Automatic, generic, user-friendly. *arXiv preprint arXiv:2303.06862*, 2023.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Robin Dupont, Hichem Sahbi, and Guillaume Michel. Weight reparametrization for budget-aware network pruning. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 789–793. IEEE, 2021.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101, 2023.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Cheng Gong, Yao Chen, Ye Lu, Tao Li, Cong Hao, and Deming Chen. Vecq: Minimal loss dnn model compression with vectorized weight quantization. *IEEE Transactions on Computers*, 70(5):696–710, 2020.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3690–3699. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/goyal20a.html`.

Isabelle M Guyon. Design of experiments for the nips 2003 variable selection benchmark. 2003. URL `https://api.semanticscholar.org/CorpusID:115452637`.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44 (3):243–254, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey, 2023.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Duc NM Hoang and Shiwei Liu. Revisiting pruning at initialization through the lens of ramanujan graph. *ICLR 2023*, 2023.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9782–9793. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6f5216f8d89b086c18298e043bfe48ed-Paper.pdf.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Gokul Krishnan, Xiaocong Du, and Yu Cao. Structural pruning in deep neural networks: A small-world approach. *arXiv preprint arXiv:1911.04453*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. Revisiting random channel pruning for neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 191–201, 2022.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. *arXiv preprint arXiv:2306.11222*, 2023a.

Zhuo Li, Hengyi Li, and Lin Meng. Model compression for deep neural networks: A survey. *Computers*, 12(3):60, 2023b.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, pages 20852–20867. PMLR, 2023.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1529–1538, 2020.

Jiaqiang Liu, Jingwei Sun, Zhongtian Xu, and Guangzhong Sun. Latency-aware automatic cnn channel pruning with gpu runtime analysis. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 1(1):100009, 2021.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1458–1467, 2020.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*, 2023.

JS McCarley, Rishav Chakravarti, and Avirup Sil. Structured pruning of a bert-based question answering model. *arXiv preprint arXiv:1910.06360*, 2019.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019.

Haojie Pan, Chengyu Wang, Minghui Qiu, Yichang Zhang, Yaliang Li, and Jun Huang. Meta-kd: A meta knowledge distillation framework for language model compression across domains. *arXiv preprint arXiv:2012.01266*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 2023.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Yang Sui, Miao Yin, Yi Xie, Huy Phan, Saman Aliari Zonouz, and Bo Yuan. Chip: Channel independence-based pruning for compact neural networks. *Advances in Neural Information Processing Systems*, 34:24604–24616, 2021.

Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frederic Andres. Sparse low rank factorization for deep neural network compression. *Neurocomputing*, 398:185–196, 2020.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6377–6389. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf`.

Rishabh Tiwari, Udbhav Bamba, Arnav Chavan, and Deepak K Gupta. Chipnet: Budget-aware pruning with heaviside continuous approximations. *arXiv preprint arXiv:2102.07156*, 2021.

Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10 (5):1122–1136, 2023.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9194–9203, 2018.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

SPvR: Structured Pruning via Ranking (Supplementary Material)

Atif Hassan[1] **Jiaul H. Paik**[1] **Swanand Khare**[2]

[1]Department of Artificial Intelligence, IIT Kharagpur, Kharagpur, West Bengal, India
[2]Department of Mathematics, IIT Kharagpur, Kharagpur, West Bengal, India

A SPVR RANKING FUNCTION VS KL DIVERGENCE: EMPIRICAL EVALUATION

In order to show the superiority of our ranking function in a more practical setting, we train a single hidden layer feed-forward neural network on a binary classification synthetic dataset and compare the neuron rankings produced by the KL divergence criterion and our method against the ground truth. Specifically, the data is generated as per the algorithm in [Guyon, 2003] using the scikit-learn library [Pedregosa et al., 2011]. The number of features and samples are set to 100 and 1000, respectively. To introduce noise into the dataset, the binary labels corresponding to each sample are flipped to either 1 or 0 with a probability of 0.02 which also introduces imbalance into the dataset. A single hidden layer feed-forward neural network with 64 neurons in the hidden layer and ReLU activation along with 2 neurons in the output layer and softmax activation is trained on the synthetic dataset using the Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.001. Once the network is trained, the ground truth ranked list is generated by measuring the number of misclassifications that occur when masking individual neurons with higher misclassifications being attributed to more important neurons. We choose this as the ground truth since the main philosophy of pruning is to remove parameters that do not hurt the final accuracy of a model. Two more ranked lists are generated using our ranking function and the KL divergence criterion, respectively. The Kendall Tau rank correlation metric [Kendall, 1938], a non-parametric rank similarity measure, is used to evaluate the rank performance of both methods in comparison to the ground truth where a value of 1 indicates exact rank association while a value of 0 indicates no association. Table 5 demonstrates that our ranking criterion is much better suited for the task of computing neuron importance for pruning in comparison to the KL divergence criterion.

Table 5: Evaluation of the neuron ranks produced by our proposed ranking function and the KL divergence criterion against the generated ground truth ranked list in terms of the Kendall Tau rank correlation metric. Bold values indicate the best performance with 1 being the highest achievable score.

Ranking Methods	Kendall Tau Rank Correlation
SPvR Ranking criterion	**0.861**
KL Divergence criterion	0.586

Remark A.1. Measuring only the number of misclassifications does not provide the complete picture of a neuron's importance as the change induced in the final output layer needs to be taken into account in order to correctly rank neurons with the same number of misclassifications.

B PRUNING STRATEGIES

Pruning VGG16: Pruning networks without skip connections, such as VGG16, is relatively straightforward. The ranking module provides a sorted list of groups that must be pruned. One can iterate over the list and discard the least important groups until the user-supplied target parameter budget is reached. The remaining groups form the smaller sub-network.

Pruning ResNets: The ResNet type architectures have two sets of skip connections, known as identity and projection

shortcuts [He et al., 2016]. Layers with the same number of filters share the identity shortcut, while layers with different numbers of filters require a projection. When iterating over the sorted list provided by the ranking module, if a group from a particular layer is discarded then the least significant group from each subsequent layer with an identity shortcut is discarded until a group from a layer after a projection shortcut is encountered.

C PRUNED ARCHITECTURES

C.1 VGG16

Table 6: The number of channels per layer for each pruning percentage. Here, *"M"* denotes the position of the max-pooling layer. The *Param* column indicates the percentage of parameters removed from the original model. The *Size* column denotes the actual size of the model on disk in megabytes.

Param	Model Size	Architecture
00%	112	64, 64, "M", 128, 128, "M", 256, 256, 256, "M", 512, 512, 512, "M", 512, 512, 512, "M"
70%	33.7	58, 64, "M", 126, 128, "M", 238, 224, "M", 192, 192, 94, "M", 56, 482, 512, "M"
80%	22.5	58, 64, "M", 126, 128, "M", 238, 224, "M", 192, 192, 94, "M", 56, 196, 512, "M"
90%	11.2	54, 64, "M", 124, 128, "M", 224, 220, "M", 174, 110, "M"
95%	5.56	42, 64, "M", 110, 126, "M", 170, 138, 92, "M"
98%	2.25	36, 62, "M", 78, 112, "M", 92, 44, 46, "M"

C.2 RESNET34

Table 7: The number of channels per layer per block with the number of blocks being denoted by $\times$ and each block being denoted by *[.]*. The *Param* column indicates the percentage of parameters removed from the original model. The *Size* column denotes the actual size of the model on disk in megabytes.

Param	Size	Architecture
00%	163	[64, 64]$\times$3, [128, 128]$\times$4, [256, 256]$\times$6, [512, 512]$\times$3
70%	46.8	[60, 60]$\times$3, [112, 112]$\times$4, [328, 328]$\times$3
80%	30.6	[60, 60]$\times$3, [104, 104]$\times$4, [256, 256]$\times$3
90%	15.2	[60, 60]$\times$3, [100, 100]$\times$4, [152, 152]$\times$3
95%	8.03	[60, 60]$\times$3, [76, 76]$\times$4, [88, 88]$\times$3
98%	2.72	[60, 60]$\times$3, [36, 36]$\times$4, [44, 44]$\times$3

C.3 RESNET50

Table 8: The number of channels per layer per block with the number of blocks being denoted by $\times$ and each block being denoted by *[.]*. The *Param* column indicates the percentage of parameters removed from the original model. The *Size* column denotes the actual size of the model on disk in megabytes.

Param	Size	Architecture
0%	195	[64, 64, 64]×3, [128, 128, 128]×4, [256, 256, 256]×6, [512, 512, 512]×3
40%	118	[64, 64, 64]×3, [120, 120, 120]×4, [240, 240, 240]×6, [328, 328, 328]×3
70%	60.3	[64, 64, 64]×3, [112, 112, 112]×4, [200, 200, 200]×6, [152, 152, 152]×6
80%	41.5	[56, 56, 56]×3, [112, 112, 112]×4, [168, 168, 168]×6, [96, 96, 96]×3
95%	12	[56, 56, 56]×3, [88, 88, 88]×4, [64, 64, 64]×6

D IMPLEMENTATION DETAILS

Hyper-parameters for each method were adopted from their respective publications, except RCP, where we sampled 20 sub-architectures instead of 100 due to resource constraints. Consistency was maintained across all methods, including SPvR, in terms of optimizer, scheduler, batch size, and training duration. In the ranking phase of SPvR, similar to CURL, only a subset of the dataset was used. Specifically, 50 training samples per class were randomly selected for both CIFAR10 and Tiny ImageNet and 20 samples per class for the ImageNet datasets.