Beyond Filter Pruning: Top-K Spatial Selection for Efficient Neural Networks

Sarthak Ketanbhai Modi[1], Hans Farrell Soegeng[1], and Thomas Peyrin[1]

School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore
Sarthakk001@e.ntu.edu.sg

Abstract. Deep neural networks have shown remarkable performance across diverse domains, but their substantial model size and computational requirements hinder deployment in memory- and computation-constrained environments. Despite progress in pruning, existing methods primarily target inter-filter redundancy and leave significant spatial redundancy within convolutional kernels unaddressed. The resulting models retain many correlated spatial patterns, limiting achievable efficiency gains. We propose **Top-K Pruning (TKP)**, a pruning framework that extends sparsity to the spatial dimension by retaining only the most informative positions within each convolutional kernel. TKP introduces a two-stage procedure: an auxiliary regularization phase that concentrates information into dominant spatial locations, followed by deterministic Top-K selection that yields semi-structured sparsity. This transforms dense convolutions into efficient selective-sampling operations with strictly bounded computational cost. Across diverse architectures, including CNNs, vision transformers, vision-language models, and a large language model, TKP consistently achieves strong accuracy–efficiency trade-offs. On CIFAR and ImageNet models, TKP matches or exceeds the accuracy of prior structured pruning methods while delivering up to **8.9× theoretical FLOP reduction**. TKP remains robust under quantization-aware training, achieving **16–17× compression** with minimal accuracy loss on ResNet–18 and VGG–19. Moreover, TKP generalizes to BLIP-Base and LLaMA-2-7B, outperforming state-of-the-art pruning baselines. These results highlight TKP as a simple and effective approach for removing spatial redundancy in modern vision and large-scale models.

1 Introduction

Deep Neural Networks (DNNs) have achieved state-of-the-art performance across a wide range of domains, including computer vision [3,4,42,43], natural language processing [10, 41, 51, 53], graph learning [31, 45, 54, 59], and tabular data analysis [2, 15, 27, 44]. Despite their impressive accuracy, modern DNNs are often characterized by substantial computational and memory requirements, which pose significant challenges for deployment in resource-constrained or real-time environments [18, 20, 24, 30, 58, 60, 65].

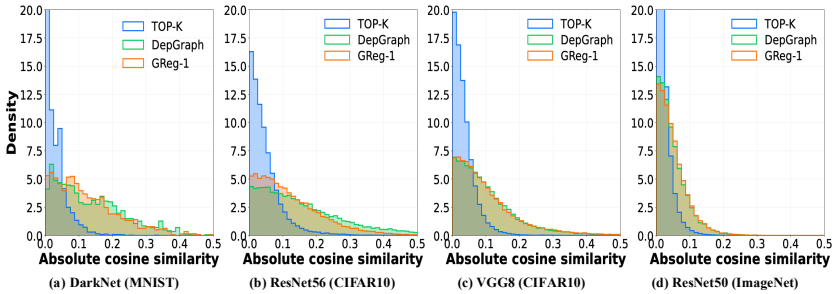

Fig. 1: Absolute cosine-similarity distributions after Top-K, DepGraph, and GReg across four backbone architectures.

Structured pruning methods, such as channel or filter pruning, have proven effective in reducing model size and computation by removing entire filters or feature maps [22, 23, 34, 37, 64]. However, these approaches primarily operate at the *inter-filter level* and may leave significant *intra-filter redundancy* unaddressed. After pruning entire filters, the remaining ones often exhibit high spatial correlation where neighboring weights within a local receptive field tend to learn correlated or linearly dependent patterns, as multiple filters still encode similar patterns (e.g., edges, textures, or orientations) at slightly shifted or rotated positions [8,9,20,29]. Consequently, the expressive capacity of the network remains over-parameterized within individual kernels, limiting the attainable compression and computational efficiency. Similar analyses in dilated convolution literature further suggest that densely sampled receptive fields lead to overlapping and redundant spatial coverage [5,6,16,63]. This is also observed empirically in Fig. 1, where we visualize the pairwise cosine similarity among layer features of models pruned using various structured pruning methods. Despite reducing inter-filter redundancy, existing approaches retain substantial intra-filter redundancy, as indicated by the high similarity among feature maps. In contrast, Top-K Pruning (TKP) produces lower pairwise cosine similarity across all examined architectures and datasets, demonstrating that its spatially selective pruning yields more diverse and decorrelated feature representations.

While several prior *unstructured pruning* techniques have attempted to achieve such sparsity by individually removing low-importance weights [20, 40], their fine-grained and irregular patterns are challenging to exploit for actual runtime acceleration. The resulting sparse tensors require additional indexing, scattering, and irregular memory access, which often offset theoretical speed-ups. In contrast, our approach enforces a *semi-structured spatial sparsity constraint* — retaining a single active connection per kernel according to an importance criterion. This yields a deterministic sparsity pattern, converting dense convolutional operations involving several multiplications and accumulations into a single multiply–accumulate per kernel. The resulting operation can be viewed as a *sparse spatial sampling mechanism*, efficiently realizable on modern accelerators, offering significant theoretical and practical speed-ups while maintaining model expressivity and generalization.

Through its simplicity in both implementation and deployment, we observe that **Top-K Pruning (TKP)** achieves remarkable improvements over previous state-of-the-art pruning methods across a wide range of model architectures. Moreover, TKP can be seamlessly applied to diverse architectures without modification. Unlike many existing approaches, TKP is also highly compatible with other model compression techniques such as quantization, enabling further efficiency gains without compromising performance. We observe that TKP consistently delivers substantial compression benefits while maintaining accuracy across various architectures.
Our key contributions are summarized as follows:

- We propose Top-K Pruning (TKP), a spatial pruning method that explicitly leverages intra-filter redundancy to reduce computation and memory usage while preserving performance.
- Extensive experiments across diverse vision and large-scale architectures demonstrate that TKP achieves favorable accuracy–efficiency trade-offs compared with existing pruning methods.
- TKP is simple to implement, compatible with quantization, and effective across CNNs, vision transformers, vision-language models, and large language models.

2 The Top-K Pruning (TKP) Method

In this section, we introduce the detailed design and motivation of our method.

2.1 Problem formulation

We consider a convolutional weight tensor

$$W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times K \times K},$$

where C_{out} and C_{in} denote the number of output and input channels, respectively, and K is the spatial kernel size. Each slice $W_{ij} \in \mathbb{R}^{K \times K}$ connects input channel j to output channel i.

Our goal is to retain only the most informative spatial coefficients within each kernel by introducing a binary spatial mask $\mathcal{M}_{ij} \in \{0,1\}^{K \times K}$ that preserves exactly K_{keep} elements:

$$\|\mathcal{M}_{ij}\|_0 = K_{\text{keep}}.$$

The pruned kernel is then defined as

$$\tilde{W}_{ij} = W_{ij} \odot \mathcal{M}_{ij},$$

where $\odot$ denotes element-wise multiplication. The network parameters after pruning are denoted $\tilde{\theta} = \{\tilde{W}, \text{BN}, \text{FC}, \dots\}$.

This formulation seeks to impose structured sparsity within each kernel, reducing intra-filter redundancy while maintaining representational capacity. However, a naive Top-K sparsification,

$$\mathcal{M}_{ij} = \mathrm{TopK}(|W_{ij}|^2, K_{\mathrm{keep}}),$$

where $\mathrm{TopK}(X, K) : \mathbb{R}^{m\times n} \times \mathbb{N} \to \{0,1\}^{m\times n}$ returns a binary mask with $\mathcal{M}_{ij} = \mathbf{1}\{X_{ij} \geq X_{(K)}\}$, and $X_{(K)}$ denotes the K-th largest element in X, introduces abrupt, non-smooth pruning decisions. The resulting hard selection discards small but informative weights that contribute to spatial coherence within the receptive field. Consequently, the pruned update $\tilde{W}_{ij} = W_{ij} \odot \mathcal{M}_{ij}$ creates a discontinuous optimization landscape, leading to unstable fine-tuning and pronounced accuracy degradation.

From an optimization standpoint, the hard constraint $\|\mathcal{M}_{ij}\|_0 = K_{\mathrm{keep}}$ is non-differentiable and provides no gradient signal for redistributing information among spatial locations. As a result, kernels often converge to inconsistent or noisy sparsity patterns, causing loss of local structure. To address this, we introduce an auxiliary regularization stage that gradually encourages each kernel to concentrate its weight into a small number of dominant spatial positions before applying the hard Top-K mask. As illustrated in Fig. 2, this stage amplifies the relative difference between the top-ranked and lower-ranked positions, ensuring that the most salient connection clearly emerges. This prevents information loss during pruning and yields more stable fine-tuning.

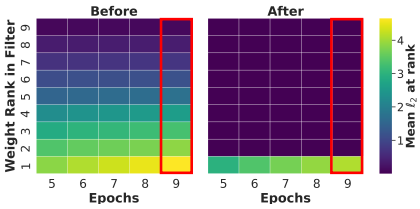

Fig. 2: Effect of auxiliary regularization. Without regularization (left), multiple spatial coefficients exhibit similar magnitudes, leading to information loss under hard Top-K pruning. After regularization (right), energy is concentrated in a few positions, enabling more stable pruning with minimal loss.

2.2 Stage 1: Information concentration via auxiliary regularization

Before applying a discrete Top-K selection, it is desirable that each kernel exhibits a clear hierarchy among its spatial coefficients. To achieve this, we introduce an auxiliary *information-concentration* regularizer that encourages each kernel to localize its informative content to a few dominant positions.

For each kernel W_{ij}, we form the vector of squared magnitudes

$$\mathbf{s}_{ij} = \mathrm{sort}_{\downarrow}\Big(\mathrm{vec}(W_{ij} \odot W_{ij})\Big) \in \mathbb{R}^{K^2},$$

sorted in descending order. The auxiliary loss penalizes coefficients ranked below the top K_{keep} according to an exponentially decaying weight:

$$\mathcal{L}_{\mathrm{aux}}(W) = \tfrac{1}{C_{\mathrm{out}}C_{\mathrm{in}}} \sum_{i,j} \sum_{r=K_{\mathrm{keep}}+1}^{K^2} \exp\big(-\alpha(r - K_{\mathrm{keep}})\big)\, \mathbf{s}_{ij}[r],$$

where $\alpha > 0$ controls the rate of decay, assigning progressively smaller penalties to lower-ranked coefficients. Larger α values enforce stronger concentration of information within the top-ranked spatial locations.

The complete Stage 1 objective combines the standard task loss with the auxiliary term:

$$\min_{\theta} \; \mathcal{L}_{\mathrm{CE}}\big(f_\theta(x), y\big) + \beta\, \mathcal{L}_{\mathrm{aux}}(W),$$

where $\mathcal{L}_{\mathrm{CE}}$ denotes the cross-entropy loss and $\beta > 0$ controls the strength of regularization.

This stage progressively shapes each kernel such that its information content becomes highly concentrated in a small subset of spatial coordinates. Consequently, when the hard Top-K mask is later applied, the selected coefficients correspond to well-separated, high-saliency positions rather than arbitrarily large weights, leading to smoother optimization and minimal degradation in representational capacity.

2.3 Stage 2: Hard Top-K mask and fine-tuning

Following the auxiliary regularization in Stage 1, we freeze the spatial sparsity pattern by retaining the K_{keep} most informative coefficients within each kernel. The binary mask $\mathcal{M}_{ij} \in \{0,1\}^{K\times K}$ is defined as

$$\mathcal{M}_{ij}(u,v) = \begin{cases} 1, & \text{if } W_{ij}(u,v)^2 \in \mathrm{TopK}\big(W_{ij}^2, K_{\mathrm{keep}}\big), \\ 0, & \text{otherwise.} \end{cases}$$

During fine-tuning, this mask is applied to both the forward and backward passes:

$$W_{ij} \leftarrow W_{ij} \odot \mathcal{M}_{ij}, \qquad \frac{\partial \mathcal{L}}{\partial W_{ij}} \leftarrow \mathcal{M}_{ij} \odot \frac{\partial \mathcal{L}}{\partial W_{ij}}.$$

Hence, gradient flow is restricted to the retained positions, ensuring that the spatial structure established in Stage 1 remains fixed throughout optimization.

The resulting fine-tuning objective is expressed as

$$\min_{W} \; \mathcal{L}_{\mathrm{CE}}\big(f_{W\odot\mathcal{M}}(x), y\big) \quad \text{s.t. } \|\mathcal{M}_{ij}\|_0 = K_{\mathrm{keep}},\ \forall\, i, j,$$

where $\mathcal{M}$ remains constant during this phase. This yields a deterministic optimization process in which only the selected connections are trainable.

The complete two-stage optimization pipeline can therefore be summarized as

Stage 1: $\min_{\theta} \; \mathcal{L}_{\mathrm{CE}}\big(f_\theta(x), y\big) + \beta\, \mathcal{L}_{\mathrm{aux}}(W),$

Stage 2: $\mathcal{M}_{ij} \leftarrow \mathrm{TopK}\big(W_{ij}^2, K_{\mathrm{keep}}\big),$

$$\min_{W} \; \mathcal{L}_{\mathrm{CE}}\big(f_{W\odot\mathcal{M}}(x), y\big), \text{ s.t. } \|\mathcal{M}_{ij}\|_0 = K_{\mathrm{keep}}.$$

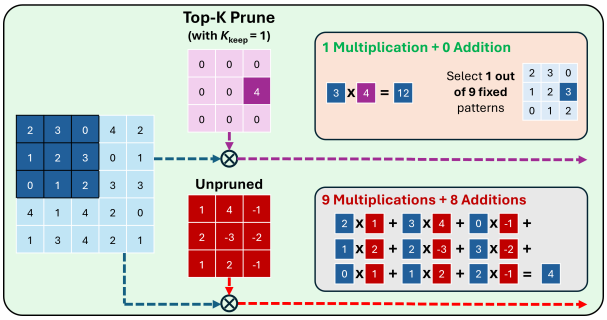

Fig. 3: Illustration of Top-K spatial pruning mechanism. (**Bottom**) Dense convolution (**Top**) Top-K pruning; $K_{keep} = 1$

This second stage transforms the soft, information-concentrated weights obtained from Stage 1 into a fixed, structured sparsity pattern. Because $\mathcal{M}$ is deterministic and shared across all training samples, the fine-tuned network exhibits reproducible behavior and stable convergence.

2.4 Sparsity and speed-up

As illustrated in Fig. 3, conventional dense convolution (bottom) performs spatial aggregation over all K^2 receptive-field samples, involving several multiplications followed by their accumulation. In contrast, the Top-K mechanism for $K_{keep} = 1$ (top) removes both redundant sampling and aggregation: each kernel samples a single spatial location and performs one multiplication to produce its response. This transformation converts the convolution from a weighted local integration into a selective sampling operation that retains only the most informative spatial connection while discarding spatially correlated neighbors. Hence, Top-K pruning enforces a fixed and deterministic spatial sparsity pattern within each convolutional kernel, characterized by the kept ratio $\rho = \frac{K_{\text{keep}}}{K^2}$. Post-training, weights are stored in a sparse format where each kernel retains only the top-k active weights alongside their spatial position indices. Each position index requires $\lceil \log_2(K^2) \rceil$ bits, where K is the kernel size. For example: for a 3x3 kernel, the initial memory footprint would be 9×32 bits $= 288$ bits. However after top-1 pruning: it would be 1×32 bits $+ \lceil \log_2(9) \rceil = 36$ bits. Hence we observe a compression of $288/36 = 8\times$. **Limitation 1.** There exists a theoretical upper limit to the amount of speed-up attainable by TKP. However we address this in Sec. 3.4.

2.5 Extension to Transformer architectures

TKP extends to transformer architectures by applying group-wise Top-K selection to linear projection matrices in attention and feed-forward layers. Consider a projection matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Each row $\mathcal{W}_i$ is partitioned into non-overlapping

groups $\{\mathcal{W}_{i,g}\}_{g=1}^{d_{\text{in}}/G}$ of size G. A binary mask $\mathcal{M}_{i,g} \in \{0,1\}^G$ retains only the top-K_{keep} magnitudes:

$$\mathcal{M}_{i,g}(r) = \begin{cases} 1, & \text{if } \mathcal{W}_{i,g}(r)^2 \in \text{TopK}\big(\mathcal{W}_{i,g}^2, K_{\text{keep}}\big), \\ 0, & \text{otherwise.} \end{cases}$$

The pruned projection is $\tilde{W} = W \odot \mathcal{M}$, yielding deterministic sparsity ratio $\rho = K_{\text{keep}}/G$ (e.g., 2:4 with G=4, K_{keep}=2), compatible with accelerators supporting N:M structured sparsity. We use the same two-stage method as described earlier, where Stage 1 encourages clearer separation between retained and removed weights within each group, while Stage 2 applies the hard Top-K selection.

3 Experiments

To evaluate the effectiveness of our method, we conduct experiments on various model architectures, including ResNet, VGG, DenseNet, GoogleNet, MobileNet and ConvNeXt. In the remainder of this section, we first introduce the detailed experimental setup, and then we answer three research questions (RQs) to lead our discussion, which are as follows:

(RQ1) Speed-up improvement: How effective is our method in improving the models' speed-up while retaining a low performance drop?

(RQ2) Combination with quantization: Can the models retain performance when combined with other compression techniques?

(RQ3) Effectiveness on LLMs and VLMs: Can our method effectively compress large-scale models?

3.1 Experimental setup

Datasets and benchmarks. We evaluate TKP across three standard image classification benchmarks: CIFAR-10, CIFAR-100 [32], and ImageNet-1k [7]. CIFAR-10/100 contain 50,000 training and 10,000 test images of size 32×32 across 10/100 classes respectively. ImageNet-1k consists of 1.28M training and 50K validation images across 1,000 categories. For CIFAR experiments, we apply standard data augmentation, including random cropping with 4-pixel padding and horizontal flipping. For ImageNet, we use random resized cropping to 224×224 and horizontal flipping during training, with center cropping during evaluation. We also provide preliminary results of TKP on VLMs and LLMs on Flickr30k and WikiText-2 respectively. Flickr30k [62] contains 31,783 images, each paired with five human-annotated captions describing salient visual concepts and relationships. We use 29K images for training, 1K for validation, and 1K for testing. Images are resized to 384×384 and tokenized using the BLIP processor with a maximum caption length of 30 tokens. WikiText-2 [38] is a language modeling benchmark containing over 2 million tokens from high-quality

Wikipedia articles, with a vocabulary of approximately 33K words. Text is tokenized with sequences of up to 2048 tokens, and we report perplexity on the validation set.

Model architectures. To demonstrate the universality of Top-K Pruning (TKP) across modalities, we evaluate its performance on a diverse suite of convolutional, transformer, and language architectures. For convolutional networks, we include VGG-8, VGG-19 [46], ResNet-18, ResNet-50, ResNet-56 [21], GoogleNet [50], DenseNet-121 [26], MobileNetV3-Large [25], and ConvNeXt-Large [36] on CIFAR-10, CIFAR-100, and ImageNet-1k benchmarks. For transformer-based architectures, we validate TKP on ViT-small and ViT-B/16 [11]. To further validate cross-domain generality, we extend our study to the Large Language and Vision–Language domain using the LLaMA-2-7B [52] model and BLIP-Base [35] model. For each architecture, we start from pretrained models achieving competitive baseline accuracies to ensure fair comparison with prior work. The detailed experimental setup, implementation details, and codebase are provided at our GitHub repository: `https://github.com/sarthak2003-modi/Top-K-Pruning`.

Compared methods. We compare TKP against state-of-the-art pruning methods spanning both vision backbones and large language models. For CNNs and vision transformers, we include DepGraph [13], which constructs dependency graphs to prune structurally coupled parameters across layers; Torque Pruning [17], which ranks weights based on torque-inspired saliency scores; Exponential Torque Pruning (ETP) [39], which extends torque pruning using exponential weighting to amplify importance separation; Growing Regularization (GReg) [56], which progressively increases sparsity-inducing penalties during training; MDP [48], which performs multi-dimensional pruning by jointly considering structural dependencies; and Isomorphic Pruning [12], which preserves architectural symmetry while removing redundant parameters.

For LLMs and VLMs, we benchmark against Wanda++ [61], which performs activation-aware magnitude pruning using lightweight second-order approximations; Probe Pruning (PP) [33], which estimates parameter importance using probing-based sensitivity analysis; FLAP [1], which prunes weights based on fluctuation-based importance estimation; and SparseGPT [14], a one-shot pruning method that approximates second-order reconstruction error using blockwise optimization.

Evaluation metrics.

Speed-up and Accuracy Drop (RQ1). For convolutional and transformer architectures, we report both the *FLOP Reduction* and the corresponding *Accuracy Drop* after pruning. This aligns with the state-of-the-art in the compression domain [13, 39, 56]. FLOPs provide an architecture-agnostic proxy for computational cost and correlate strongly with inference latency on modern accelerators. The *Accuracy Drop* (Δ) denotes the performance degradation relative to the dense baseline and quantifies the trade-off between efficiency and accuracy. In addition, we measure **inference latency** represented as **Speed-Up** and **energy consumption** to assess practical hardware efficiency.

Compression Ratio (RQ2). To analyze TKP's compatibility with other compression techniques, such as quantization, we compute the *Compression Ratio* as the ratio of memory footprint between baseline and pruned model. This metric reflects the overall reduction in model storage footprint and is particularly relevant for deployment under memory constraints.
Perplexity and BLEU-4 (RQ3). For LLaMA-2-7B, we evaluate the Perplexity [14, 47] on WikiText-2 dataset and for BLIP, we evaluate the BLEU-4 score on the Flickr30k dataset.

3.2 Ablation study

To assess the effectiveness of our Stage 1 regularization, we perform an ablation study across multiple CNN architectures and pruning configurations. The compared strategies include: (1) **No Regularization:** Stage 1 is omitted, and pruning is directly performed in Stage 2. (2) **L2 Regularization:** The exponential regularization term is replaced by a standard ℓ_2 norm penalty on the convolutional weights. (3) **Exponential Regularization:** Our proposed scheme as described in Stage 1, which penalizes weights in a progressively non-linear manner to promote sharper separation between important and unimportant connections.

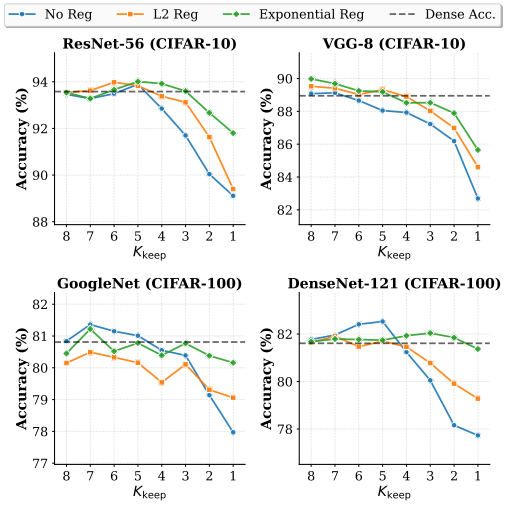

Fig. 4: Pruned accuracy vs. kept spatial connections (K_{keep}) across three regularizations and four architectures.

As summarized in Fig. 4, exponential regularization consistently yields superior Top-1 accuracy in **high-sparsity regimes** (i.e., smaller K_{keep}), indicating that it facilitates more structured weight separation and better information preservation after pruning. The non-linear scaling of the exponential term encourages early suppression of uninformative filters while maintaining strong gradients for dominant weights, resulting in more discriminative sparse representations.

For **smaller architectures** such as VGG-8 and ResNet-56 on CIFAR-10, this effect is especially pronounced. These models exhibit limited representational redundancy; therefore, aggressive pruning without a guiding regularization often leads to abrupt accuracy degradation. The exponential term counteracts this by inducing stronger sparsity patterns aligned with task-relevant weights, leading to smoother accuracy decay as sparsity increases. In contrast, for **larger networks** such as DenseNet-121 and GoogleNet on CIFAR-100, the performance gap between regularization strategies narrows as K_{keep} increases. We attribute

Table 1: Pruning results on multiple vision benchmarks. Rows highlighted in light green are our methods.

Architecture	Method	Base (%)	Pruned (%)	Acc. Drop (%)	FLOP Reduc.(×)	Speed-Up (×)	Energy Usage (%)
ResNet56 (CIFAR-10)	DepGraph	93.58	88.26	-5.32	8.92	2.38	49.8
	Torque (p)	93.58	87.81	-5.77	8.92	2.41	48.7
	ETP	93.58	90.46	-3.12	8.90	2.64	47.2
	GReg-1	93.58	90.12	-3.46	8.80	2.44	46.8
	Isomorphic	93.58	90.90	-2.68	8.90	2.40	49.9
	MDP	93.58	90.55	-3.03	8.85	2.97	42.2
	TKP (Ours)	93.58	91.80	-1.78	8.83	3.94	30.6
VGG-19 (CIFAR-100)	DepGraph	73.50	70.39	-3.11	8.92	2.87	35.3
	Torque (r)	73.03	65.87	-7.16	8.88	2.84	37.1
	ETP	73.50	71.30	-2.20	9.03	3.03	32.2
	GReg-2	73.50	67.64	-5.86	8.84	2.56	42.0
	Isomorphic	73.50	71.85	-1.65	9.00	2.66	39.9
	MDP	73.50	71.20	-2.30	8.95	2.81	38.1
	TKP (Ours)	73.50	72.77	-0.73	8.90	5.15	25.3
ResNet50 (ImageNet-1k)	DepGraph	76.15	75.83	-0.32	2.07	1.71	61.2
	Torque (p)	76.07	74.67	-1.40	2.34	1.94	51.8
	ETP	76.15	75.62	-0.53	2.30	1.94	52.1
	GReg-2	76.13	75.36	-0.77	1.49	1.14	91.1
	Isomorphic	76.15	73.94	-2.21	2.05	1.79	57.1
	MDP	76.15	75.18	-0.97	2.12	1.77	56.6
	TKP (Ours)	76.15	76.49	+0.34	3.51	3.01	43.5
ViT-B/16 (ImageNet-1k)	CP-ViT	77.91	77.36	-0.55	1.50	1.21	83.4
	DepGraph+EMA	81.07	79.58	-1.49	1.69	1.34	78.7
	DepGraph	81.07	79.17	-1.90	1.69	1.31	79.8
	GReg-2	81.07	77.50	-3.57	1.74	1.42	76.9
	Isomorphic	81.07	79.85	-1.22	1.74	1.56	71.6
	MDP	81.07	80.12	-0.95	1.78	1.48	72.3
	TKP (Ours)	81.07	80.93	-0.14	1.78	1.52	71.4

this to two interacting factors: (1) The inherent redundancy in high-capacity models that cushions the impact of pruning, and (2) The optimization overhead introduced by additional regularization, which may hinder convergence when the effective pruning ratio is low. **Limitation 2.** In low-sparsity scenarios (large K_{keep}), exponential regularization may slightly underperform compared to simpler baselines. This behavior likely stems from over-regularization, introducing an unnecessary penalty when redundancy remains high. Nevertheless, this limitation has minimal practical significance, as pruning methods are primarily deployed to achieve high sparsity for real efficiency gains.

3.3 Speed-Up improvement (RQ1)

Tables 1 and 2 summarize the pruning results across multiple CNN and ViT architectures. Rows highlighted in green correspond to TKP. Overall, TKP consistently achieves the most favorable accuracy–efficiency trade-off and, importantly, translates theoretical FLOP reductions into strong *measured* latency and energy improvements on a single NVIDIA A100 GPU.

CIFAR CNNs. On ResNet56 (CIFAR-10), TKP achieves the highest pruned accuracy (91.80%) with the smallest degradation from the dense model (-1.78%)

Table 2: Accuracy–efficiency trade-offs on ImageNet-1k for modern architectures.

Method	Base Acc. (%)	Pruned Acc. (%)	Theor. FLOP (×)	Speed-Up (×)
MobileNetV3 (Large)				
TKP (Ours)	75.2	**72.8**	2.00	1.79
MDP	75.2	71.6	2.00	1.80
Isomorphic	75.2	70.8	2.00	1.83
DepGraph	75.2	70.7	2.00	1.70
ConvNeXt (Large)				
TKP (Ours)	84.4	**81.4**	6.33	**4.11**
MDP	84.4	78.1	6.30	2.51
Isomorphic	84.4	78.4	6.31	2.72
DepGraph	84.4	77.1	6.35	2.70

at a comparable theoretical reduction ($\sim 8.8\times$). In contrast, competing methods exhibit substantially larger accuracy drops, typically between -5.77% and -2.68%, as shown in Table 1. Beyond accuracy retention, TKP also delivers the strongest hardware gains, achieving $3.94\times$ measured latency speed-up while reducing energy consumption to 30.6% of the dense baseline, significantly lower than competing approaches.

A similar trend is observed on VGG-19 (CIFAR-100). TKP attains the best accuracy retention, reaching 72.77% with only a -0.73% drop at $8.90\times$ theoretical FLOP reduction. In comparison, existing methods incur noticeably larger degradation, with accuracy drops ranging from -7.16% to -1.65%. Moreover, TKP provides the largest realized acceleration, achieving a $5.15\times$ latency speed-up and reducing energy consumption to 25.3% of the dense model, whereas baselines achieve smaller practical gains and higher energy usage.

ImageNet CNNs. On ResNet50 (ImageNet-1k), TKP achieves the largest theoretical reduction ($3.51\times$) and uniquely *improves* accuracy over the dense model, reaching 76.49% ($+0.34\%$). In contrast, competing approaches either maintain or reduce accuracy while achieving smaller speed-ups. These results indicate that TKP effectively removes spatial redundancy in deep CNNs while preserving (or even enhancing) representational capacity.

Vision Transformers. TKP also generalizes well to transformer architectures. On ViT-B/16 (ImageNet-1k), TKP achieves a $1.78\times$ FLOP reduction with only a -0.14% accuracy drop (80.93% top-1). Competing methods incur larger degradation under similar sparsity constraints, highlighting the robustness of TKP when applied to attention-based architectures. Furthermore, TKP achieves a $1.52\times$ measured latency improvement and reduces energy usage to 71.4% of the dense baseline.

Modern architectures. Tab. 2 further demonstrates the effectiveness of TKP on recent ImageNet architectures. On MobileNetV3-Large, where all methods operate under the same theoretical $2.00\times$ FLOP reduction, TKP achieves the highest top-1 accuracy (72.8%) while maintaining competitive measured speed-up. On ConvNeXt-Large, TKP obtains 81.4% accuracy together with a $6.33\times$ theoretical FLOP reduction and a $4.11\times$ measured speed-up, substantially outperforming competing methods in both accuracy retention and practical ac-

celeration. These results indicate that the proposed spatial selection strategy generalizes well to both lightweight mobile networks and modern high-capacity convolutional architectures.

Practical speed-up and energy. Across all evaluated architectures, TKP consistently translates theoretical FLOP reductions into tangible hardware gains while maintaining strong predictive performance. For example, on ResNet56 and VGG-19, TKP reduces energy consumption to 30.6% and 25.3% of the dense baseline while achieving 3.94× and 5.15× measured latency speed-ups, respectively. Similar trends are observed on ImageNet CNNs, Vision Transformers, and the additional modern architectures in Tab. 2, demonstrating that TKP provides consistent practical acceleration across diverse network designs.

3.4 Effectiveness with quantization (RQ2)

We observe from Limitation 1 that there is a maximum threshold on the compression achievable using TKP. This creates a limit for deployment in resource-constrained devices. Deploying DNNs on edge devices is heavily constrained by memory footprint, which is consistently identified as one of the primary bottlenecks in on-device inference due to limited storage and high memory-access cost [19, 23, 49, 60]. Hence, in RQ2, we address this limitation of TKP by exploring its performance when combined with quantization. Quantization helps in reducing the memory footprint of models by reducing the bit width of each parameter. When combined with TKP, quantization provides up to 17× compression in terms of model size.

Table 3 evaluates the interaction between Top–K Pruning (TKP) and Quantization-Aware Training [28] (QAT) on ResNet–18 and VGG–19 for CIFAR–100. We compare QAT-INT8 alone with QAT combined with existing structured pruning baselines (DepGraph, GReg–1), and with our proposed QAT + TKP pipeline.

For ResNet–18, QAT-INT8 closely matches the dense baseline, incurring only a 0.14% accuracy drop while achieving a 3.96× reduction in model size. When QAT is combined with structured pruning, however, accuracy degrades sharply: QAT + DepGraph drops by 5.07%, and QAT + GReg–1 suffers a substantial 12.95% decline. In contrast, QAT + TKP achieves 81.54% accuracy with a 0.49% drop while reaching a 16× compression ratio, demonstrating that TKP produces sparsity patterns that remain stable under quantization. Unlike other pruning methods that remove spatial connections aggressively and amplify quantization-induced noise, TKP preserves the most influential spatial structures, leading to significantly stronger performance under QAT.

A similar trend appears for VGG–19. QAT-INT8 provides moderate degradation (–1.36%) with a 3.99× model size reduction. When combined with pruning baselines, accuracy reductions become substantially larger: –3.85% for QAT + DepGraph and –5.16% for QAT + GReg–1, despite comparable compression ratios. In comparison, QAT + TKP attains 71.45% accuracy at a 17.17× compression ratio, matching the compactness of the pruning baselines while retaining performance. These results highlight that TKP introduces structured sparsity

Table 3: Benchmarking Top–K Pruning (TKP) and Quantization on CIFAR-100. "Acc. Drop" is relative to the dense baseline.

ResNet-18 (CIFAR-100)					
Method	**Accuracy**	**Acc. Drop**	**#Params (M)**	**Model Size**	**Compr.**
Base (Dense)	82.03%	0.00	11.24	42.87 MB	1.00×
QAT-INT8	81.89%	–0.14	11.25	10.83 MB	3.96×
QAT + DepGraph	76.96%	–5.07	2.83	2.72 MB	15.76×
QAT + GReg-1	69.08%	–12.95	2.81	2.70 MB	15.88×
QAT + TKP (Ours)	**81.54%**	**–0.49**	**2.71**	**2.68 MB**	**16.00×**
VGG-19 (CIFAR-100)					
Method	**Accuracy**	**Acc. Drop**	**#Params (M)**	**Model Size**	**Compr.**
Base (Dense)	73.50%	0.00	20.08	76.58 MB	1.00×
QAT-INT8	72.14%	–1.36	20.10	19.17 MB	3.99×
QAT + DepGraph	69.65%	–3.85	4.56	4.35 MB	17.60×
QAT + GReg-1	68.34%	–5.16	4.59	4.38 MB	17.48×
QAT + TKP (Ours)	**71.45%**	**–2.05**	**4.53**	**4.46 MB**	**17.17×**

that is inherently more quantization-friendly, preserving spatial information that QAT can effectively adapt to. Across both architectures, the empirical evidence shows that TKP integrates smoothly with quantization, maintaining strong accuracy even at extreme compression levels while outperforming existing structured pruning baselines by large margins under the same quantization regime. This demonstrates that TKP provides an effective pathway for jointly achieving high sparsity and low-bitwidth efficiency.

3.5 Effectiveness on Large Models (RQ3)

To evaluate the scalability of TKP to modern large-scale architectures, we conduct experiments on both large language models (LLMs) and vision–language models (VLMs). Specifically, we evaluate pruning performance on the LLaMA-2-7B model using the WikiText-2 benchmark and on the BLIP captioning model using the Flickr30k dataset. These experiments assess whether TKP can preserve model capability under both unstructured and semi-structured sparsity in large transformer-based systems.

Large Language Models. Table 4 reports perplexity results for LLaMA-2-7B on WikiText-2 under three sparsity settings: unstructured 50% sparsity and two commonly used semi-structured patterns (4:8 and 2:4). We compare TKP against several representative pruning baselines, including SparseGPT, FLAP, Probe Pruning (PP), and Wanda++. All methods are evaluated under identical training budgets; since TKP employs a two-stage optimization process, its training budget is divided equally between the regularization and fine-tuning stages. Across all sparsity configurations, TKP consistently achieves the lowest perplexity. Under 50% sparsity, TKP reduces perplexity to 5.59, outperforming all competing

Table 4: LLaMA-2-7B perplexity (↓) on WikiText-2. Lower is better.

Method	**50%**	**4:8**	**2:4**
Dense	6.00	–	–
SparseGPT	11.06	14.31	28.74
FLAP	8.62	11.37	15.64
PP	6.72	8.21	11.03
Wanda++	6.44	7.38	10.15
TKP (Ours)	**5.59**	**6.14**	**9.31**

methods and even improving upon the dense baseline. Competing approaches incur noticeably larger degradation, with perplexity values ranging from approximately 6.44 to 11.06. The advantage of TKP becomes even more pronounced under structured sparsity. For the 4:8 pattern, TKP achieves a perplexity of 6.14, significantly lower than alternative approaches. Under the more aggressive 2:4 constraint, TKP maintains strong language modeling performance with a perplexity of 9.31, while competing methods exhibit substantially higher degradation. These results indicate that TKP effectively preserves the most informative connections during pruning, enabling robust performance even under strict sparsity constraints.

Vision–Language Models. We further evaluate TKP on multimodal architectures using the BLIP model on the Flickr30k image captioning benchmark. Table 5 reports BLEU-4 scores under semi-structured sparsity. Wanda++ and PP are given the same total training budget as TKP, while TKP's budget is split equally between the regularization and fine-tuning stages. Under the 2:4 sparsity pattern, TKP achieves a BLEU-4 score of 29.93, outperforming both Wanda++ and PP and exhibiting the smallest performance degradation relative to the dense model. Hence, TKP introduces sparsity patterns that remain stable even in multimodal architectures, preserving both linguistic and visual performance.

Table 5: BLIP captioning performance (BLEU-4 ↑) on Flickr30k. Higher is better.

Method	Pattern	BLEU-4
Dense	–	32.85
Wanda++ [61]	2:4	28.93
PP [33]	2:4	28.71
TKP (Ours)	2:4	**29.93**

4 Related Works

In this section, we discuss different structured pruning techniques that have been proposed in the community. Structured pruning removes entire filters, channels, or blocks to yield efficient models with regular memory access. Classic approaches rank filters by magnitude or importance, while Soft Filter Pruning (SFP) temporarily zeros filters but allows them to regrow during training, mitigating capacity loss. HRank prunes filters producing low-rank feature maps, reflecting limited information content. Growing Regularization (GReg) gradually increases sparsity-inducing penalties to stabilize pruning dynamics. DepGraph explicitly models inter-layer dependencies to prune structurally coupled parameters consistently across diverse architectures. Torque-based pruning assigns saliency scores inspired by physical torque, and its exponential variant (ETP) further separates important and unimportant filters. These methods successfully eliminate redundant filters but operate exclusively at the inter-filter or channel level, leaving substantial spatial redundancy inside surviving kernels. To address redundancy within convolution kernels, intra-filter pruning removes uninformative spatial positions. Structured Sparsity Learning [57] (SSL) groups weights at the same kernel location across filters, yielding shared spatial masks that prune entire kernel positions. Structured Sparse Convolutions [55] (SSC) enforce predefined

sparse kernel patterns (e.g., checkerboard masks) to reduce computation without iterative pruning. Sub-kernel pruning removes rows, columns, or spatial blocks of kernels, offering coarse but structured spatial sparsity. However, these approaches often rely on fixed masks, limiting adaptability and fine-grained selection. In contrast, our method (TKP) learns to concentrate importance into a few spatial locations and selects the most informative positions per filter.

5 Conclusion

In this work, we introduced **Top-K Pruning (TKP)**, a universal pruning framework that targets the spatial redundancy remaining within convolutional kernels after conventional filter or channel pruning. By combining an auxiliary exponential regularization stage with a deterministic Top-K selection, TKP concentrates information into a small number of salient spatial positions and produces fixed sparsity patterns that translate directly into FLOPs reduction and a smaller memory footprint.

Extensive experiments across modern vision and large-scale architectures, including CNNs on CIFAR and ImageNet, ViT-B/16 on ImageNet, LLaMA-2-7B on WikiText-2, and BLIP-Base on Flickr30k, demonstrate that TKP achieves strong accuracy–efficiency trade-offs. TKP delivers up to 8.9× theoretical FLOP reduction and up to 5.15× measured latency speed-up on CNNs with minimal accuracy degradation, preserves or improves accuracy on deeper models such as ResNet-50, and remains effective under quantization, achieving 16–17× compression with limited loss. TKP further generalizes to large language and vision-language models, outperforming representative pruning baselines under the evaluated sparsity settings.

TKP is simple to implement, introduces minimal training overhead, and does not depend on architecture-specific heuristics, making it widely applicable to modern vision and vision–language systems. While its theoretical speed-up is bounded by kernel size, our results show that TKP offers substantial practical gains, even under tight sparsity constraints. We believe TKP provides a strong foundation for future research on spatially structured sparsity, hybrid pruning–quantization pipelines, and efficient transformer, LLM and VLM compression.

References

1. An, Y., Zhao, X., Yu, T., Tang, M., Wang, J.: Fluctuation-based adaptive structured pruning for large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 10865–10873 (2024)
2. Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., Kasneci, G.: Deep neural networks and tabular data: A survey. IEEE transactions on neural networks and learning systems **35**(6), 7499–7519 (2022)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)

4. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
5. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
6. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
8. Denil, M., Shakibi, B., Dinh, L., Ranzato, M., De Freitas, N.: Predicting parameters in deep learning. Advances in neural information processing systems **26** (2013)
9. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. Advances in neural information processing systems **27** (2014)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). pp. 4171–4186 (2019)
11. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
12. Fang, G., Ma, X., Mi, M.B., Wang, X.: Isomorphic pruning for vision models. In: European Conference on Computer Vision. pp. 232–250. Springer (2024)
13. Fang, G., Ma, X., Song, M., Mi, M.B., Wang, X.: Depgraph: Towards any structural pruning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16091–16101 (2023)
14. Frantar, E., Alistarh, D.: Sparsegpt: Massive language models can be accurately pruned in one-shot. In: International conference on machine learning. pp. 10323–10337. PMLR (2023)
15. Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting deep learning models for tabular data. Advances in neural information processing systems **34**, 18932–18943 (2021)
16. Gupta, A., Rush, A.M.: Dilated convolutions for modeling long-distance genomic dependencies. arXiv preprint arXiv:1710.01278 (2017)
17. Gupta, A., Bau, T., Kim, J., Zhu, Z., Jha, S., Garud, H.: Torque based structured pruning for deep neural network. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 2711–2720 (2024)
18. Gupta, M., Modi, S.K., Zhang, H., Lee, J.H., Lim, J.H.: Is bio-inspired learning better than backprop? benchmarking bio learning vs. backprop. arXiv preprint arXiv:2212.04614 (2022)
19. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
20. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems **28** (2015)

21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
22. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:1808.06866 (2018)
23. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.J., Han, S.: Amc: Automl for model compression and acceleration on mobile devices. In: Proceedings of the European conference on computer vision (ECCV). pp. 784–800 (2018)
24. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
25. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1314–1324 (2019)
26. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
27. Huang, X., Khetan, A., Cvitkovic, M., Karnin, Z.: Tabtransformer: Tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678 (2020)
28. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2704–2713 (2018)
29. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866 (2014)
30. Javaheripi, M., Bubeck, S., Abdin, M., Aneja, J., Bubeck, S., Mendes, C.C.T., Chen, W., Del Giorno, A., Eldan, R., Gopi, S., et al.: Phi-2: The surprising power of small language models. Microsoft Research Blog **1**(3), 3 (2023)
31. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
32. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
33. Le, Q., Diao, E., Wang, Z., Wang, X., Ding, J., Yang, L., Anwar, A.: Probe pruning: Accelerating llms through dynamic pruning via model-probing. arXiv preprint arXiv:2502.15618 (2025)
34. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)
35. Li, J., Li, D., Xiong, C., Hoi, S.: Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: International conference on machine learning. pp. 12888–12900. PMLR (2022)
36. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11976–11986 (2022)
37. Luo, J.H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision. pp. 5058–5066 (2017)
38. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843 (2016)
39. Modi, S.K., Lim, Z.P., Kuchhal, S., Cao, Y., Cheng, Y., Teo, Y.S., Lin, S.W., Li, Z.: Towards universal & efficient model compression via exponential torque pruning. arXiv preprint arXiv:2506.22015 (2025)

40. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440 (2016)
41. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
42. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
43. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
44. Shan, Y., Hoens, T.R., Jiao, J., Wang, H., Yu, D., Mao, J.: Deep crossing: Web-scale modeling without manually crafted combinatorial features. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 255–262 (2016)
45. Simonovsky, M., Komodakis, N.: Graphvae: Towards generation of small graphs using variational autoencoders. In: International conference on artificial neural networks. pp. 412–422. Springer (2018)
46. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
47. Sun, M., Liu, Z., Bair, A., Kolter, J.Z.: A simple and effective pruning approach for large language models. arXiv preprint arXiv:2306.11695 (2023)
48. Sun, X., Lakshmanan, B., Shen, M., Lan, S., Chen, J., Alvarez, J.M.: Mdp: multi-dimensional vision model pruning with latency constraint. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 20113–20123 (2025)
49. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE **105**(12), 2295–2329 (2017)
50. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
51. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
52. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
53. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
54. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
55. Verma, V.K., Mehta, N., Si, S., Henao, R., Carin, L.: Pushing the efficiency limit using structured sparse convolutions. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 6503–6513 (2023)
56. Wang, H., Qin, C., Zhang, Y., Fu, Y.: Neural pruning via growing regularization. arXiv preprint arXiv:2012.09243 (2020)
57. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. Advances in neural information processing systems **29** (2016)

58. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4820–4828 (2016)
59. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
60. Yang, X., Ye, J., Wang, X.: Factorizing knowledge in neural networks. In: European Conference on Computer Vision. pp. 73–91. Springer (2022)
61. Yang, Y., Zhen, K., Ganesh, B., Galstyan, A., Huybrechts, G., Müller, M., Kübler, J.M., Swaminathan, R.V., Mouchtaris, A., Bodapati, S.B., et al.: Wanda++: Pruning large language models via regional gradients. arXiv preprint arXiv:2503.04992 (2025)
62. Young, P., Lai, A., Hodosh, M., Hockenmaier, J.: From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Transactions of the association for computational linguistics **2**, 67–78 (2014)
63. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
64. Yu, J., Huang, T.: Autoslim: Towards one-shot architecture search for channel numbers. arXiv preprint arXiv:1903.11728 (2019)
65. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6848–6856 (2018)