
DEPTH- AND RANK-SELECTIVE MULTILINEAR TRANSFORMATION LAYERS FOR COMPACT CNNs

A PREPRINT

Van Tien Nguyen

The University of Texas at San Antonio
San Antonio, TX
tien.nguyen@utsa.edu

Mayur Dhanaraj*

Amazon
Arlington, VA
mayur.dhanaraj1994@gmail.com

Panagiotis Markopoulos

The University of Texas at San Antonio
San Antonio, TX
panagiotis.markopoulos@utsa.edu

Accepted to the 60th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, 2026

ABSTRACT

Convolutional Neural Networks (CNNs) have become indispensable in modern computer vision, yet their deeper layers are often heavily overparameterized and computationally inefficient. We present *Multilinear Transformation Layers (MTLs)*, a simple and effective alternative that replaces select convolutional layers with global multilinear transformations applied directly on activation tensors. By shifting from kernel-level factorization to activation-level projections, MTLs retain expressivity while offering substantial reductions in parameters and multiply-accumulate (MAC) operations. We introduce two complementary strategies: **D-MTL** (Depth-selective MTL), which replaces the T deepest convolutional layers, and **Rk-MTL** (Rank-selective MTL), which replaces layers whose normalized effective rank falls below a threshold τ . Both variants are fully differentiable and end-to-end trainable. Empirical studies on ResNet-50 (ImageNet-1K), ResNet-101 (CIFAR-10), and YOLOv11x (DOTA-v1) demonstrate consistent parameter reductions while matching or improving accuracy. On ImageNet-1K, D-MTL achieves up to 29% fewer parameters within 1.5 pp of baseline Top-1 accuracy, outperforming LR by 4–6 pp; Rk-MTL delivers up to 41.7% latency reduction in well-calibrated settings. On CIFAR-10, D-MTL reduces parameters by up to 30% while matching or surpassing accuracy. On DOTA-v1, D-MTL cuts parameters by up to 20% with commensurate MAC and latency reductions. In well-calibrated configurations, both variants consistently outperform prior tensor-based methods.

Keywords Convolutional Neural Networks, Multilinear Transformation, Parameter Efficiency, Effective Rank, Image Classification, Object Detection, Model Compaction

1 Introduction

Convolutional Neural Networks (CNNs) have become the workhorse of modern computer vision. From foundational image recognition systems [1, 2] to state-of-the-art image classification [3, 4] and object detection [5–8], CNNs underpin a remarkably broad range of applications. Yet this success comes at a cost: the same depth that makes CNNs expressive also makes them over-parameterized [9–11]. Redundant capacity wastes storage and energy, raises the

*Work was done prior to joining Amazon.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

risk of overfitting when labeled data is scarce, and makes deployment on resource-constrained hardware a persistent engineering challenge [8, 12, 13].

A natural response is to *compress* a trained network—through pruning, quantization, or knowledge distillation—or to *compact* the architecture from the start by replacing expensive building blocks with leaner alternatives. Compression is practical and widely used, but it is fundamentally bounded by the structure of the original network. Compaction, by contrast, avoids redundancy by design, which tends to yield better generalization especially under limited data. The central question we address is: *which specific building block, and which layers, should be replaced?*

Two complementary observations together point toward an answer. First, as depth increases in a CNN, spatial resolution decreases, channel dimensionality grows, and spatial locality becomes progressively less critical [14, 15]. Second, the weight tensors of deeper layers tend to exhibit low effective rank [16, 17]: their representational capacity is already confined to a low-dimensional subspace. Together, these suggest that the deepest convolutional layers are the natural candidates for compact replacement.

Contributions. We introduce *Multilinear Transformation Layers (MTLs)*, a fully differentiable, drop-in replacement for convolutions that applies three learned factor matrices—one per tensor mode—directly on activation tensors, eliminating spatial kernel dependence entirely. **D-MTL** (Depth-selective MTL) replaces the final T convolutional layers via a single hyperparameter before training begins, while **Rk-MTL** (Rank-selective MTL) targets layers whose normalized effective rank falls below a threshold τ —identified by a one-time SVD pass—and retraining the resulting hybrid from scratch. Experiments validate the approach across three benchmarks: on ImageNet-1K, D-MTL achieves up to 29% fewer parameters within 1.5 pp of baseline accuracy, outperforming LR [18] by 4–6 pp; Rk-MTL ($\tau=0.88$) delivers up to 41.7% latency reduction. On CIFAR-10, D-MTL reduces parameters by up to 30% while *improving* accuracy; on DOTA-v1, it cuts parameters by up to 20% with commensurate MAC and latency reductions. In well-calibrated settings, both methods consistently outperform prior tensor-based approaches [18, 19].

2 Related Work

Post-training compression. Pruning eliminates redundant weights or filters [20, 21]. The Lottery Ticket Hypothesis [22] showed sparse subnetworks can match dense counterparts. Quantization [23] reduces numerical precision, and Knowledge Distillation [24] transfers knowledge from a large teacher to a smaller student. These approaches are bounded by the original network’s structure.

Pre-training compaction. MobileNet [8] and EfficientNet [3] redesign convolutional blocks for efficiency; NAS [25] automates efficient architecture search. Many such architectures still rely on convolutional kernels that dominate parameter counts in deeper layers.

Tensor-based compaction. Low-rank and CP-inspired decompositions factorize convolutional filters [18] and evolved into multilinear filter mappings [19]. Tucker decomposition compresses full layers [26] and inspired Tensor Contraction Layers [27]. Tensor Train [28, 29] and Tensor Ring [30] achieve high compression but can impose sequential dependencies. In contrast, our MTLs act on *activation tensors* rather than kernels, offering global projections and explicit control over the accuracy–compactness trade-off.

Rank analysis. Sedghi *et al.* [16] characterized rapid spectral decay in convolutional layers. The effective rank [17] provides a continuous measure of effective dimensionality. Idelbayev and Carreira-Perpiñán [31] showed rank-aware compression outperforms uniform-rank approaches; Schotthöfer *et al.* [32] framed low-rank discovery as a differential equation problem. Our **Rk-MTL** uses effective rank directly as a layer-selection criterion.

3 Proposed Method

3.1 Multilinear Transformation Layer (MTL)

Scalars, vectors, matrices, and tensors are denoted by non-bold, bold lowercase, bold uppercase, and bold calligraphic letters, respectively (e.g., x ; $\mathbf{x}$; $\mathbf{U}_1$; $\mathcal{X}, \mathcal{Y}$). In a standard convolutional layer $l + 1$, the input $\mathcal{X}^{(l)} \in \mathbb{R}^{D \times D \times C}$ is convolved with filters $\mathcal{F}^{(l+1)} \in \mathbb{R}^{k \times k \times C \times O}$ to produce $\mathcal{X}^{(l+1)} \in \mathbb{R}^{d \times d \times O}$:

$$\mathcal{Y}(i, j, p) = \langle \mathcal{X}_{i,j}, \mathcal{F}(:, :, p) \rangle. \quad (1)$$

An MTL replaces this with trainable multilinear projections across all tensor modes:

$$\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3, \quad (2)$$

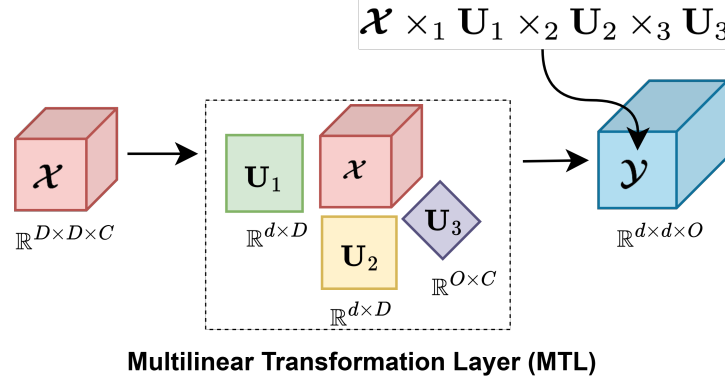


Figure 1: Multilinear Transformation Layer (MTL). All trainable parameters reside in $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$, eliminating dependence on a spatial kernel k .

where $\mathbf{U}_1, \mathbf{U}_2 \in \mathbb{R}^{d \times D}$ project spatial modes and $\mathbf{U}_3 \in \mathbb{R}^{O \times C}$ mixes channels (Fig. 1). The element-wise form is given in (3):

$$\mathcal{Y}(j_1, j_2, j_3) = \sum_{i_1=1}^D \sum_{i_2=1}^D \sum_{i_3=1}^C \left[\mathcal{X}(i_1, i_2, i_3) \prod_{n=1}^2 U_n(j_n, i_n) \right] U_3(j_3, i_3), \quad \forall j_1, j_2, j_3. \quad (3)$$

All factor matrices are learned via standard backpropagation without specialized procedures.

Novelty and distinction from prior work. Unlike Kossaifi *et al.* [27], whose Tensor Contraction Layers (TCLs) replace *fully connected* layers acting on flattened feature vectors, our MTL replaces *convolutional* layers directly, acting on three-dimensional activation tensors $\mathcal{X} \in \mathbb{R}^{D \times D \times C}$ via mode-wise projections. This is non-trivial: replacing convolutions requires preserving spatial structure for both classification and detection. To the best of our knowledge, this is the first work to systematically replace convolutional layers with activation-level multilinear transformations, complemented by data-driven rank analysis for layer selection.

3.2 D-MTL: Depth-Selective Replacement

D-MTL replaces the final T convolutional layers with MTLs. The hyperparameter T is a single, interpretable knob: small T preserves expressive power while larger T increases compaction. This is motivated by the well-established observation that deeper CNN layers encode increasingly abstract representations with higher representational redundancy [15], where spatial locality becomes less critical and global correlations dominate.

3.3 Rk-MTL: Rank-Selective Replacement

Depth alone is an imperfect proxy for compressibility—some layers encode information in a low-dimensional subspace regardless of their position [16]. **Rk-MTL** uses the *effective rank* of each layer’s weight tensor as a principled, data-driven criterion.

For layer ℓ with weight tensor $\mathcal{F}^{(\ell)} \in \mathbb{R}^{k \times k \times C \times O}$, reshape to $\mathbf{W}^{(\ell)} \in \mathbb{R}^{O \times k^2 C}$ with singular values $\sigma_1, \dots, \sigma_m$ and normalized weights $p_i = \sigma_i / \sum_j \sigma_j$. The effective rank [17] is:

$$r_{\text{eff}}(\mathbf{W}^{(\ell)}) = \exp \left(- \sum_{i=1}^m p_i \log p_i \right), \quad (4)$$

then the normalized effective rank is defined as:

$$\bar{r}_{\text{eff}}(\mathbf{W}^{(\ell)}) = \frac{r_{\text{eff}}(\mathbf{W}^{(\ell)})}{m} \in (0, 1]. \quad (5)$$

Values of $\bar{r}_{\text{eff}} \ll 1$ indicate that information content is concentrated in a low-dimensional subspace—precisely where MTL replacement is most effective. Given threshold τ , Rk-MTL: (1) trains the baseline CNN to convergence; (2) computes $\bar{r}_{\text{eff}}(\mathbf{W}^{(\ell)})$ via a single SVD pass per layer; (3) forms $\mathcal{S}_\tau = \{\ell : \bar{r}_{\text{eff}}(\mathbf{W}^{(\ell)}) < \tau\}$; and (4) retrain the

Table 1: Parameters and computational cost: standard convolution vs. the proposed MTL.

Method	# Params	MACs
Conv	k^2CO	$\mathcal{O}(k^2D^2CO)$
MTL	$2dD + CO$	$\mathcal{O}(\min\{D^2Cd + DCd^2 + Cd^2O, D^2Cd + DCdO + Dd^2O, D^2CO + D^2dO + Dd^2O\})$

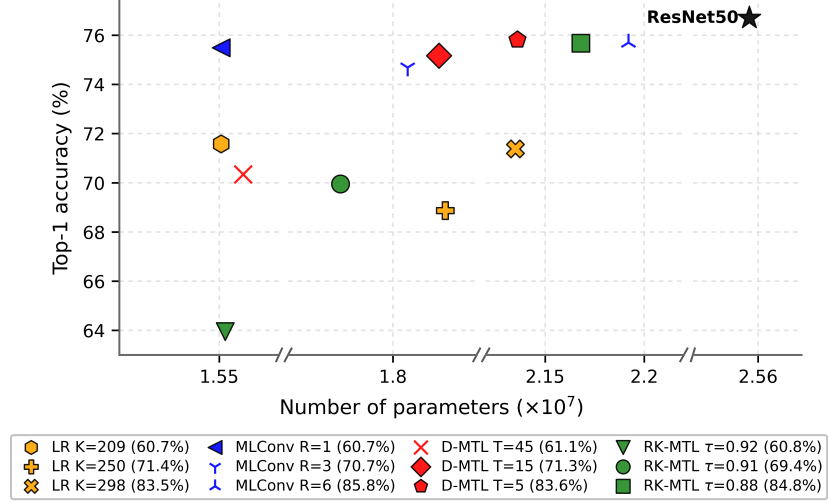


Figure 2: Exp. 1: Accuracy–parameter trade-off for ResNet-50 on ImageNet-1K. The Pareto frontier of D-MTL dominates LR across all parameter budgets; Rk-MTL ($\tau=0.88$) achieves competitive accuracy at a comparable parameter count.

resulting hybrid from scratch. Lower τ replaces only the most compressible layers; higher τ admits more replacements. Rk-MTL relaxes D-MTL’s depth assumption and directly targets layers with low intrinsic dimensionality, complementing D-MTL for architectures with irregular rank profiles.

3.4 Model Size and Computational Complexity

Table 1 compares Conv and MTL. MTL eliminates kernel dependence: parameter count reduces to $2dD + CO$, scaling linearly with spatial dimensions rather than quadratically in k . Contractions in (2) are evaluated in order $\times_3, \times_1, \times_2$ (channel mixing first), which minimizes MACs when $C, O \gg D$ —a condition satisfied in every deep layer targeted by D-MTL and Rk-MTL across all three architectures. Whether MAC reductions translate to lower wall-clock latency depends on input resolution and hardware kernel availability, as discussed in Section 4.4.

4 Experimental Studies

We evaluate MTLs on three benchmarks: ResNet-50 on ImageNet-1K (Exp. 1), ResNet-101 on CIFAR-10 (Exp. 2), and YOLOv11x on DOTA-v1.0 (Exp. 3). All contractions are performed in the order $\times_3, \times_1, \times_2$.

4.1 Experiment 1: ImageNet-1K Classification

4.1.1 Settings

ResNet-50 [33] on ImageNet-1K [2] (1.28M training / 50K validation, 1,000 classes). SGD with momentum 0.9, initial LR 0.1 with cosine decay, weight decay 10^{-4} , batch size 256, 90 epochs. D-MTL replaces $T \in \{5, 15, 45\}$ layers; Rk-MTL uses $\tau \in \{0.88, 0.91, 0.92\}$ computed from baseline weights. Comparisons: LR [18] ($K \in \{209, 250, 298\}$) and MLconv [19] ($R \in \{1, 3, 6\}$).

Table 2: Exp. 1: Top-1/Top-5 accuracy (%) and model size (M) for ResNet-50 on ImageNet-1K. N = number of layers replaced.

Config	Top-1	Top-5	# Params	N
<i>ResNet-50 (baseline)</i>				
–	76.70	93.22	25.56	–
<i>LR [18]</i>				
$K = 209$	71.58	90.38	15.51	–
$K = 250$	68.87	88.83	18.25	–
$K = 298$	71.38	90.40	21.35	–
<i>MLconv [19]</i>				
$R = 1$	75.49	92.55	15.51	–
$R = 3$	74.69	92.09	18.07	–
$R = 6$	75.70	92.56	21.92	–
<i>D-MTL (Ours)</i>				
$T = 45$	70.34	89.05	15.62	45
$T = 15$	75.17	92.33	18.22	15
$T = 5$	75.82	92.83	21.36	5
<i>Rk-MTL (Ours)</i>				
$\tau = 0.92$	63.95	84.96	15.53	17
$\tau = 0.91$	69.95	89.22	17.75	14
$\tau = 0.88$	75.68	92.67	21.68	8

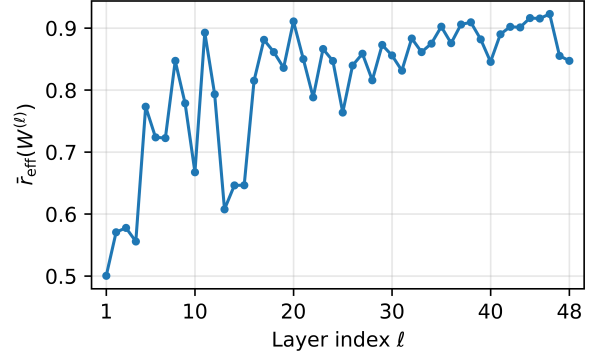


Figure 3: Exp. 1: Per-layer $\bar{r}_{\text{eff}}(\mathbf{W}^{(\ell)})$ for ResNet-50 on ImageNet-1K. Unlike ResNet-101, no contiguous low-rank collapse occurs; most layers maintain $\bar{r}_{\text{eff}} > 0.5$, explaining why the practical operating point is $\tau=0.88$.

4.1.2 Results and Discussion

D-MTL. Table 2 shows D-MTL consistently outperforms LR. At $\sim 21\text{M}$ parameters, D-MTL ($T=5$) achieves 75.82% Top-1 vs. LR ($K=298$) at 71.38%—a 4.4 pp improvement. At $\sim 18\text{M}$, D-MTL ($T=15$) at 75.17% outperforms LR ($K=250$) at 68.87% by 6.3 pp. The accuracy gap is at most 1.5 pp at $T=15$, and the D-MTL Pareto frontier dominates LR across all budgets (Fig. 2).

Rk-MTL. Fig. 3 shows ResNet-50 does not exhibit a contiguous low-rank collapse; most layers maintain $\bar{r}_{\text{eff}} > 0.5$. At $\tau=0.88$ (8 layers), Rk-MTL achieves 75.68% Top-1 with 21.68M parameters—within 0.14 pp of D-MTL ($T=5$)—and reduces inference latency by 41.7% (Table 5), as these rank-selected layers overlap with computationally dominant layers at 224×224 . Aggressive thresholds ($\tau \geq 0.91$) replace layers lacking genuine low-rank structure, causing significant accuracy loss. All variants converge smoothly without special optimization procedures.

4.2 Experiment 2: CIFAR-10 Classification

4.2.1 Settings

ResNet-101 [33] on CIFAR-10 [34], 200 epochs, batch 128, LR 10^{-3} , weight decay 10^{-5} . D-MTL: $T \in \{3, 15, 42\}$. Rk-MTL: $\tau \in \{0.01, 0.3, 0.9\}$, ranks computed post-convergence; hybrid retrained from scratch. Comparisons: LR ($K \in \{229, 260, 300\}$) and MLconv ($R \in \{4, 6, 8\}$). Data-limited regime: 2/5 training subset.

4.2.2 Results and Discussion

Fig. 4 plots the accuracy–parameter trade-off.

D-MTL. D-MTL consistently reduces parameters while preserving or improving accuracy. With $T=42$, it reaches 87.30%—above the 86.07% baseline—with only 30.46M parameters ($\sim 28\%$ fewer). $T=3$ achieves 87.33% at $\sim 5\%$ compaction. On the 2/5 subset, the baseline drops 7.22 pp; D-MTL ($T=15$) declines by only 6.02 pp, demonstrating stronger data efficiency.

Rk-MTL. Fig. 5 shows that under full data, layers $\ell \in [28, 65]$ collapse to near-zero effective rank. Under 2/5 data, the low-rank region expands to earlier layers, directly explaining the larger N (e.g., 22 vs. 32 at $\tau=0.9$). At moderate thresholds ($\tau=0.3$, $\tau=0.01$), Rk-MTL yields 87.01% and 86.95% with 27.84M and 33.61M parameters—competitive

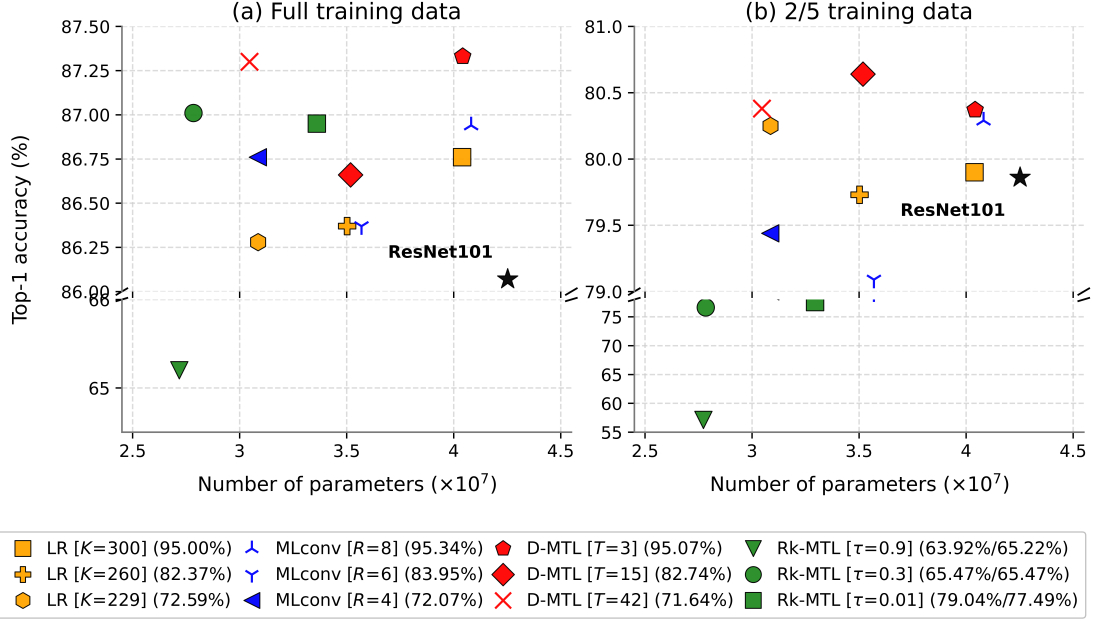


Figure 4: Exp. 2: Accuracy vs. parameters for ResNet-101 on CIFAR-10 (full and 2/5 reduced data). **D-MTL** and **Rk-MTL** in well-calibrated configurations lie on a more favorable frontier than the baseline and prior methods (LR, MLconv). Best viewed in color.

Table 3: Exp. 2: Validation accuracy (%) and model size (M) for ResNet-101 on CIFAR-10. For Rk-MTL, N is reported as Full / 2/5.

Config	Val Acc (Full / 2/5)	# Params	N
<i>ResNet-101 (baseline)</i>			
–	86.07 / 79.86	42.52	–
<i>LR [18]</i>			
$K = 229$	86.28 / 80.52	30.86	–
$K = 260$	87.22 / 79.73	35.03	–
$K = 300$	86.76 / 79.90	40.39	–
<i>MLconv [19]</i>			
$R = 4$	86.76 / 79.44	30.85	–
$R = 6$	86.31 / 79.09	35.22	–
$R = 8$	86.94 / 80.29	40.82	–
<i>D-MTL (Ours)</i>			
$T = 42$	87.30 / 80.38	30.46	42
$T = 15$	86.66 / 80.64	35.18	15
$T = 3$	87.33 / 80.37	40.42	3
<i>Rk-MTL (Ours)</i>			
$\tau = 0.9$	65.20 / 57.11	27.18 / 27.73	22 / 32
$\tau = 0.3$	87.01 / 76.63	27.84 / 27.84	19 / 28
$\tau = 0.01$	86.95 / 77.55	33.61 / 32.95	11 / 19

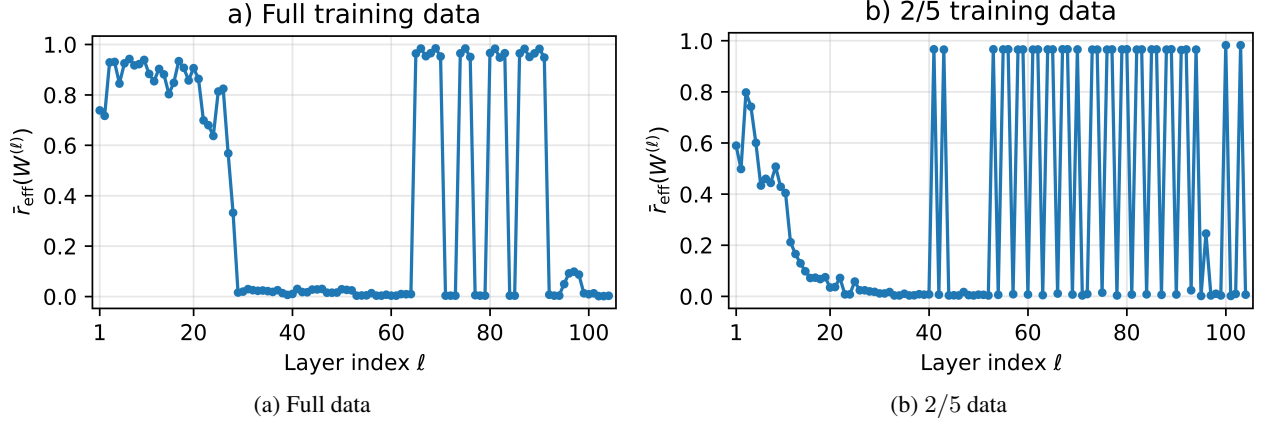


Figure 5: Exp. 2: Per-layer $\bar{r}_{\text{eff}}(\mathbf{W}^{(\ell)})$ for ResNet-101 baseline. (a) Layers $\ell \in [28, 65]$ collapse to near-zero rank. (b) Low-rank region expands under reduced data, explaining larger N and stronger Rk-MTL gains.

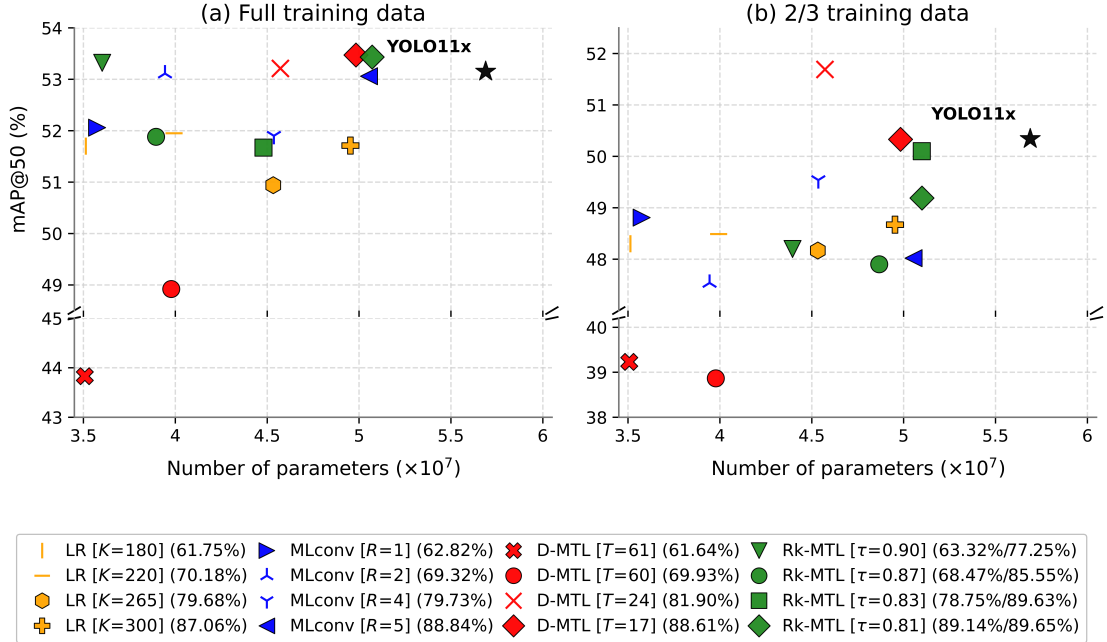


Figure 6: Exp. 3: mAP@50 vs. parameters for YOLOv11x on DOTA-v1 (full and 2/3 reduced data). D-MTL and Rk-MTL in well-calibrated configurations outperform all LR and MLconv baselines. Best viewed in color.

with D-MTL. The aggressive $\tau=0.9$ replaces layers without genuinely low rank, causing accuracy collapse to 65.20%, underscoring the importance of calibrating τ below the architecture’s actual rank concentration level.

4.3 Experiment 3: DOTA-v1.0 Object Detection

4.3.1 Settings

YOLOv11x [35, 36] on DOTA-v1.0 [37], 300 epochs, batch 16, Adam. D-MTL replaces $T \in \{17, 24, 60, 61\}$ convolutional layers (excluding detection head). Rk-MTL uses $\tau \in \{0.81, 0.83, 0.87, 0.90\}$. Comparisons: LR ($K \in \{180, 220, 265, 300\}$), MLconv ($R \in \{1, 2, 4, 5\}$). Both full and 2/3 data splits are evaluated.

4.3.2 Results and Discussion

Fig. 6 plots the mAP@50–parameter trade-off.

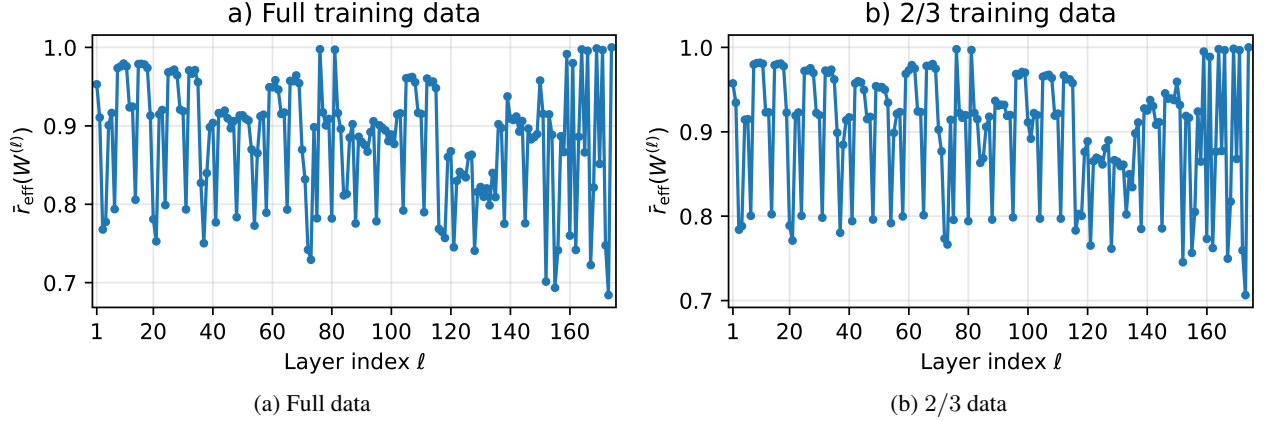


Figure 7: Exp. 3: Per-layer $\bar{r}_{\text{eff}}(\mathbf{W}^{(\ell)})$ for YOLOv11x baseline. All layers sustain high rank ($\bar{r}_{\text{eff}} \gtrsim 0.7$) with oscillatory profile. Under reduced data, *fewer* layers fall below τ (opposite of ResNet-101).

Table 4: Exp. 3: mAP@50 (%) and model size (M) for YOLOv11x on DOTA-v1. For Rk-MTL, N is reported as Full/2/3.

Config	mAP@50 (Full / 2/3)	# Params	N
<i>YOLOv11x (baseline)</i>			
–	53.15 / 50.34	56.89	–
<i>LR [18]</i>			
$K = 180$	51.70 / 48.30	35.13	–
$K = 220$	51.95 / 48.49	39.93	–
$K = 265$	50.94 / 48.17	45.33	–
$K = 300$	51.71 / 48.67	49.53	–
<i>MLconv [19]</i>			
$R = 1$	52.06 / 48.81	35.74	–
$R = 2$	53.11 / 47.54	39.44	–
$R = 4$	51.90 / 49.54	45.36	–
$R = 5$	53.06 / 48.02	50.54	–
<i>D-MTL (Ours)</i>			
$T = 61$	43.83 / 39.23	35.07	61
$T = 60$	48.92 / 38.86	39.78	60
$T = 24$	53.21 / 51.69	45.72	24
$T = 17$	53.47 / 50.33	49.83	17
<i>Rk-MTL (Ours)</i>			
$\tau = 0.90$	53.32 / 48.20	36.02 / 43.95	23 / 11
$\tau = 0.87$	51.88 / 47.90	38.95 / 48.67	13 / 10
$\tau = 0.83$	51.67 / 50.10	44.80 / 50.99	8 / 2
$\tau = 0.81$	53.43 / 49.19	50.71 / 51.00	3 / 2

D-MTL. D-MTL matches or exceeds YOLOv11x accuracy while reducing parameters by 12–20%. With $T=17$, it achieves 53.47% mAP@50 (vs. baseline 53.15%) with 49.83M parameters; with $T=24$, 53.21% mAP@50 with 45.72M parameters. All LR and MLconv configurations fall below baseline accuracy at comparable budgets. On 2/3 data, D-MTL ($T=24$) maintains 51.69% mAP@50 vs. baseline 50.34%, confirming improved data efficiency.

Rk-MTL. Fig. 7 shows all YOLOv11x layers maintain $\bar{r}_{\text{eff}} \gtrsim 0.7$ with an oscillatory profile—in sharp contrast to ResNet-101—so τ must be set near the global rank level ($\tau \in [0.81, 0.90]$). At $\tau=0.81$ (3 layers), Rk-MTL matches D-MTL ($T=17$) at 53.43% mAP@50. Aggressive $\tau=0.90$ achieves 53.32% with only 36.02M parameters. Under 2/3 data, $\tau=0.81$ yields 49.19% mAP@50, marginally below baseline, so D-MTL ($T=24$) is the stronger data-limited choice.

Table 5: Whole-model computational efficiency. GMACs via thop on $3 \times 224 \times 224$ (Exp. 1), $3 \times 32 \times 32$ (Exp. 2), $3 \times 640 \times 640$ (Exp. 3). Latency: mean single-image inference time (ms) over 1,000 GPU runs. Δ relative to respective baseline.

Config	Params (M)	Δ Params	GMACs	Δ MACs	Latency (ms)	Δ Latency
<i>Exp. 1 — ResNet-50 / ImageNet-1K (224×224)</i>						
Baseline	25.56	—	4.130	—	0.617	—
D-MTL ($T = 45$)	15.62	$\downarrow 38.9\%$	0.610	$\downarrow 85.2\%$	1.432	$\uparrow 132.1\%$
D-MTL ($T = 15$)	18.22	$\downarrow 28.7\%$	2.940	$\downarrow 28.8\%$	0.736	$\uparrow 19.3\%$
D-MTL ($T = 5$)	21.36	$\downarrow 16.4\%$	3.750	$\downarrow 9.2\%$	0.638	$\uparrow 3.4\%$
Rk-MTL ($\tau = 0.92$)	15.53	$\downarrow 39.3\%$	2.280	$\downarrow 44.8\%$	0.698	$\uparrow 13.1\%$
Rk-MTL ($\tau = 0.91$)	17.75	$\downarrow 30.6\%$	2.520	$\downarrow 39.0\%$	0.395	$\downarrow 36.0\%$
Rk-MTL ($\tau = 0.88$)	21.68	$\downarrow 15.2\%$	3.210	$\downarrow 22.3\%$	0.360	$\downarrow 41.7\%$
<i>Exp. 2 — ResNet-101 / CIFAR-10 (32×32)</i>						
Baseline	42.52	—	0.159	—	5.930	—
D-MTL ($T = 42$)	30.46	$\downarrow 28.4\%$	0.137	$\downarrow 13.8\%$	9.280	$\uparrow 56.5\%$
D-MTL ($T = 15$)	35.18	$\downarrow 17.3\%$	0.156	$\downarrow 2.1\%$	7.075	$\uparrow 19.3\%$
D-MTL ($T = 3$)	40.42	$\downarrow 4.9\%$	0.157	$\downarrow 1.3\%$	6.130	$\uparrow 3.4\%$
Rk-MTL ($\tau = 0.9$)	27.18	$\downarrow 36.1\%$	0.136	$\downarrow 14.5\%$	7.018	$\uparrow 18.3\%$
Rk-MTL ($\tau = 0.3$)	27.84	$\downarrow 34.5\%$	0.120	$\downarrow 24.5\%$	7.529	$\uparrow 27.0\%$
Rk-MTL ($\tau = 0.01$)	33.61	$\downarrow 20.9\%$	0.116	$\downarrow 27.0\%$	7.947	$\uparrow 34.0\%$
<i>Exp. 3 — YOLOv11x / DOTA-v1 (640×640)</i>						
Baseline	56.89	—	97.242	—	16.201	—
D-MTL ($T = 61$)	35.07	$\downarrow 38.3\%$	80.173	$\downarrow 17.6\%$	21.248	$\uparrow 31.1\%$
D-MTL ($T = 60$)	39.78	$\downarrow 30.1\%$	81.316	$\downarrow 16.4\%$	21.113	$\uparrow 30.3\%$
D-MTL ($T = 24$)	45.72	$\downarrow 19.6\%$	89.823	$\downarrow 7.6\%$	16.113	$\downarrow 0.5\%$
D-MTL ($T = 17$)	49.83	$\downarrow 12.4\%$	95.229	$\downarrow 2.1\%$	14.400	$\downarrow 11.1\%$
Rk-MTL ($\tau = 0.90$)	36.02	$\downarrow 36.7\%$	82.332	$\downarrow 15.3\%$	16.440	$\uparrow 1.5\%$
Rk-MTL ($\tau = 0.87$)	38.95	$\downarrow 31.5\%$	85.475	$\downarrow 12.1\%$	14.203	$\downarrow 12.3\%$
Rk-MTL ($\tau = 0.83$)	44.80	$\downarrow 21.2\%$	88.407	$\downarrow 9.1\%$	12.307	$\downarrow 24.0\%$
Rk-MTL ($\tau = 0.81$)	50.71	$\downarrow 10.9\%$	94.620	$\downarrow 2.7\%$	11.638	$\downarrow 28.2\%$

4.4 GMACs and Latency Analysis

Several patterns emerge from Table 5.

MAC reductions scale with input resolution. On Exp. 1, D-MTL ($T=15$) achieves a 28.8% MAC reduction alongside a 28.7% parameter reduction. On Exp. 2, replacing 42 layers yields only 13.8% fewer whole-model MACs despite a 28.4% parameter reduction, because early high-resolution layers dominate computation and are not replaced. On Exp. 3, $T=60$ achieves a 16.4% MAC reduction with 30.1% parameter reduction at 640×640 input.

Latency does not always track MACs. `torch.nn.Conv2d` benefits from vendor-optimized cuDNN kernels (Winograd, FFT-based, or direct GEMM), whereas MTL mode products execute as general BLAS calls without the same hardware specialization. This kernel-dispatch overhead dominates at small input resolutions, causing latency increases for D-MTL on Exps. 1 and 2. For Rk-MTL on ImageNet-1K (224×224), rank-guided selection targets computationally dominant layers, yielding up to 41.7% latency reduction. At large input (Exp. 3), D-MTL ($T=17$: -11.1%) and Rk-MTL ($\tau=0.81$: -28.2%) confirm that latency gains are achievable when replaced layers are both computationally significant and spatially large. Hardware-aware kernel fusion for MTL mode products is a natural direction for future work.

5 Conclusion

We presented two complementary MTL variants as efficient, drop-in replacements for convolutional layers. **D-MTL** replaces the deepest T layers via a single hyperparameter; **Rk-MTL** targets layers whose normalized effective rank falls below τ , guided by a one-time SVD pass. Both are architecture-agnostic, differentiable, and require no specialized optimization. On ResNet-50/ImageNet-1K, D-MTL reduces parameters by up to 29% within 1.5 pp of baseline accuracy,

outperforming LR by 4–6 pp; Rk-MTL achieves up to 41.7% latency reduction at $\tau=0.88$. On ResNet-101/CIFAR-10, D-MTL achieves up to 30% fewer parameters while matching or improving accuracy. On YOLOv11x/DOTA-v1, D-MTL cuts parameters by up to 20%; Rk-MTL ($\tau=0.81$) achieves 28.2% latency reduction. D-MTL is recommended for depth-redundant architectures (ResNet-like); Rk-MTL is preferable for architectures with irregular rank profiles (YOLO-like) or when a data-driven criterion is desired. Future work will investigate hardware-aware kernel fusion and a unified D-MTL + Rk-MTL two-criterion strategy.

Acknowledgment

This work is based upon work supported by the National Science Foundation under Grant No. 2332744.

References

- [1] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” in *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, ser. NIPS’89. Cambridge, MA, USA: MIT Press, 1989, p. 396–404.
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017. [Online]. Available: <https://doi.org/10.1145/3065386>
- [3] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114. [Online]. Available: <https://proceedings.mlr.press/v97/tan19a.html>
- [4] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 4700–4708.
- [5] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [7] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 778–10 787.
- [8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *ArXiv*, vol. abs/1704.04861, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:12670695>
- [9] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” in *International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: <https://openreview.net/forum?id=Sy8gdb9xx>
- [10] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, “Predicting parameters in deep learning,” in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS’13. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 2148–2156.
- [11] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, “Towards understanding the role of over-parametrization in generalization of neural networks,” in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: <https://openreview.net/forum?id=BygfgHAcYX>
- [12] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 2820–2828.
- [13] M. Sharma, P. P. Markopoulos, E. Saber, M. S. Asif, and A. Prater-Bennette, “Convolutional auto-encoder with tensor-train factorization,” in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 198–206.
- [14] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: <https://doi.org/10.1038/nature14539>
- [15] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.
- [16] H. Sedghi, V. Gupta, and P. M. Long, “The singular values of convolutional layers,” in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: <https://openreview.net/forum?id=rJevYoA9Fm>
- [17] O. Roy and M. Vetterli, “The effective rank: A measure of effective dimensionality,” in *2007 15th European Signal Processing Conference (EUSIPCO)*. IEEE, 2007, pp. 606–610.
- [18] C. Tai, T. Xiao, Y. Zhang, X. Wang, and W. E, “Convolutional neural networks with low-rank regularization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. [Online]. Available: <https://arxiv.org/abs/1511.06067>

- [19] D. T. Tran, A. Iosifidis, and M. Gabbouj, “Improving efficiency in convolutional neural networks with multilinear filters,” *Neural Networks*, vol. 105, pp. 328–339, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608018301795>
- [20] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: <https://arxiv.org/abs/1611.06440>
- [21] Y. He and S. Han, “Structured pruning for deep convolutional neural networks: A survey,” *arXiv preprint arXiv:2303.00566*, 2023.
- [22] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: <https://openreview.net/forum?id=rJI-b3RcF7>
- [23] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A survey of quantization methods for efficient neural network inference,” 2021. [Online]. Available: <https://arxiv.org/abs/2103.13630>
- [24] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015. [Online]. Available: <https://arxiv.org/abs/1503.02531>
- [25] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019. [Online]. Available: <http://jmlr.org/papers/v20/18-598.html>
- [26] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, “Compression of deep convolutional neural networks for fast and low power mobile applications,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016, pp. 3738–3747. [Online]. Available: <https://arxiv.org/abs/1511.06530>
- [27] J. Kossaifi, A. Khanna, Z. Lipton, T. Furlanello, and A. Anandkumar, “Tensor contraction layers for parsimonious deep nets,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 26–32.
- [28] A. Novikov, D. Podoprikin, A. Osokin, and D. P. Vetrov, “Tensorizing neural networks,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2015, pp. 442–450.
- [29] T. Garipov, D. Podoprikin, A. Novikov, and D. Vetrov, “Ultimate tensorization: compressing convolutional and FC layers alike,” *ArXiv preprint*, Nov. 2016. [Online]. Available: <https://arxiv.org/pdf/1611.03214.pdf>
- [30] V. Aggarwal, W. Wang, B. Eriksson, Y. Sun, and W. Wang, “Wide compression: Tensor ring nets,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9329–9338.
- [31] Y. Idelbayev and M. Á. Carreira-Perpiñán, “Low-rank compression of neural nets: Learning the rank of each layer,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8049–8059.
- [32] S. Schotthofer, E. Zangrando, J. Kusch, G. Ceruti, and F. Tudisco, “Low-rank lottery tickets: Finding efficient low-rank neural networks via matrix differential equations,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/5a8c04bb2f5e8b5c6c4a3b2b4e9b8a2e-Abstract-Conference.html
- [33] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, Jun. 2016, pp. 770–778.
- [34] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Toronto, Ontario, 2009. [Online]. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- [35] G. Jocher and J. Qiu, “Ultralytics yolo11,” 2024. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [36] N. Jegham, C. Y. Koh, M. Abdelatti, and A. Hendawi, “Yolo evolution: A comprehensive benchmark and architectural review of yolov12, yolo11, and their previous versions,” 2025. [Online]. Available: <https://arxiv.org/abs/2411.00201>
- [37] J. Ding, N. Xue, G.-S. Xia, X. Bai, W. Yang, M. Y. Yang, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, “Object detection in aerial images: A large-scale benchmark and challenges,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7778–7796, 2022.