
ADAPTIVE LOW-RANK MULTILINEAR TRANSFORMATIONS FOR COMPACT CONVOLUTIONAL NEURAL NETWORKS

A PREPRINT

Mayur Dhanaraj^{*†}

Amazon
Arlington, VA, USA
mayur.dhanaraj1994@gmail.com

Van Tien Nguyen^{*}

Department of Electrical and Computer Engineering
The University of Texas at San Antonio
San Antonio, TX, USA
tien.nguyen@utsa.edu

Panagiotis Markopoulos[†]

Departments of Computer Engineering and Computer Science
The University of Texas at San Antonio
San Antonio, TX, USA
panagiotis.markopoulos@utsa.edu

ABSTRACT

Deep convolutional networks remain expensive to store because their later stages contain wide channel-mixing operators. This work studies a shape-preserving alternative to selected deep convolutions. The proposed *Multilinear Transformation Layer (MTL)* maps an activation tensor through one learned matrix per spatial mode and one learned channel matrix, yielding a global mode-separable activation operator rather than a factorized convolutional kernel. The *Low-Rank MTL (LR-MTL)* further factorizes the channel map and couples a normalized group penalty with scheduled structural clipping, so that the active channel rank and the physical model size can decrease during end-to-end training. We evaluate ResNet-50 on ImageNet-1K, ResNet-101 on CIFAR-10, and YOLO11x on DOTA-v1.0 against low-rank and multilinear convolutional baselines, Network Slimming, Dynamic Rank Pruning, direct truncation, and 1×1 replacements. On ImageNet-1K, LR-MTL removes 15.3% of the parameters with a 0.57-point Top-1 decrease. On CIFAR-10, LR-MTL removes 58.1% of the parameters while improving full-data test accuracy by 1.31 points, and improves the two-fifths-data result by as much as 2.91 points. On DOTA-v1.0, LR-MTL removes 22.7% of the parameters while matching the full-data baseline and improving two-thirds-data mAP@50 by 2.64 points. Measured GMACs and FP32 latency also show that parameter compaction does not automatically yield runtime acceleration: the present materialized tensor implementation is often slower than optimized convolution kernels. The results establish MTL and LR-MTL as parameter-efficient architectural alternatives whose clearest empirical advantage appears under limited-data training and in parameter-matched settings where direct removal degrades predictive performance.

IMPACT STATEMENT

Memory capacity is a practical bottleneck for vision models deployed on drones, robots, mobile devices, and remote-sensing platforms. MTL replaces selected deep convolutions with structured global activation maps, while LR-MTL additionally removes low-importance channel-rank components during training and stores the resulting physically smaller architecture. Across classification and aerial-object detection, these reductions can preserve or improve predictive performance, especially when training data are limited. The present implementation does not consistently reduce latency,

^{*}Equal contributions. [†]Corresponding author. [‡]Work was done prior to joining Amazon.

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

so the demonstrated near-term benefit is model-size reduction and adaptive capacity control rather than guaranteed speed or energy savings. Hardware-aware contraction kernels are needed before systematic runtime advantages can be claimed.

Keywords Convolutional neural networks, multilinear transformation, low-rank approximation, adaptive regularization, parameter efficiency, model compaction

1 Introduction

Convolutional neural networks (CNNs) underpin modern image classification and object detection systems [1–4]. Their predictive strength is often accompanied by large parameter counts and substantial memory demand, motivating pruning, quantization, distillation, efficient block design, and low-rank parameterizations [5–7]. Most such approaches either retain the local convolutional operator or compact a model through a separate pruning or decomposition stage.

This work investigates a complementary architectural question: *can selected deep convolutions be replaced by a different tensor operator while preserving the surrounding network interface?* Early CNN stages process high-resolution local structure, whereas later stages operate on lower-resolution, semantically richer activations [8, 9]. We therefore retain standard convolutions in the early network and replace only the deepest T eligible convolutional operators. The resulting Multilinear Transformation Layer (MTL) maps the complete activation tensor through one learned matrix per spatial mode and one learned channel matrix. MTL is consequently an activation-to-activation operator with global support and Kronecker-separable structure; it is not a decomposition of a convolutional kernel.

For an input of size $D \times D \times C$ and an output of size $d \times d \times O$, MTL contains $2dD + CO$ parameters, compared with k^2CO for a $k \times k$ convolution. It is therefore smaller exactly when $2dD < (k^2 - 1)CO$. This condition identifies the intended operating regime: late, wide layers for which the channel term CO dominates the spatial factors. A dense fully connected replacement would instead require a $d^2O \times D^2C$ matrix and would discard the modal structure that keeps MTL compact.

Our principal adaptive formulation, LR-MTL, factorizes the channel map as $\mathbf{U}_3 = \mathbf{A}\mathbf{B}$ and combines a layer-normalized group penalty with scheduled structural clipping. The active intermediate dimension is thus determined during training rather than fixed as a single manually selected rank. At a clipping event, columns of $\mathbf{A}$ and the corresponding rows of $\mathbf{B}$ are physically removed, and subsequent optimization proceeds on the reduced architecture.

The scope of this replacement is deliberate. Because MTL uses global spatial matrices, it does not inherit the translation equivariance of convolution and is tied to the spatial dimensions for which its factors are defined. These properties motivate its selective use in later, lower-resolution stages rather than as a universal substitute for convolution.

The principal contributions are:

1. We formulate a shape-preserving, global, mode-separable activation operator as a selective replacement for deep convolutions, derive its parameter-saving condition, and distinguish it from dense fully connected mappings, terminal tensor contraction layers, and local kernel factorizations.
2. We develop LR-MTL, an adaptive channel-rank formulation that combines a normalized group penalty with scheduled physical clipping, allowing the learned architecture to shrink during end-to-end optimization.
3. We evaluate the methods on ImageNet-1K, CIFAR-10, and DOTA-v1.0 against direct truncation, 1×1 replacement, low-rank and multilinear convolutions, Network Slimming, and Dynamic Rank Pruning. The study reports predictive performance, physical parameter counts, executed-operation estimates, measured FP32 latency, and training/rank dynamics.
4. We delineate the empirical limits of the method. MTL and LR-MTL provide favorable parameter–accuracy trade-offs and strong limited-data behavior, but they do not dominate every baseline or every resource metric, and the present implementation does not consistently translate parameter savings into lower latency.

2 Related Work

CNN efficiency methods span three overlapping categories: compression of an existing model, compact parameterizations used during training, and architectures designed to be efficient from the outset. MTL belongs to the third category because selected operators are replaced before training. LR-MTL combines this compact-by-design view with training-time structural adaptation.

2.1 CNN Compression and Parameter Reduction

Pruning, quantization, and distillation. Pruning removes low-importance weights, filters, channels, or other structures. Taylor criteria provide one principled importance estimate [10], while structured pruning targets hardware-usable groups such as complete channels [11]. Network Slimming [5], used as a baseline in this work, learns channel sparsity through BatchNorm scaling coefficients and then removes channels. The Lottery Ticket Hypothesis further shows that sparse trainable subnetworks can exist within dense models [6]. Quantization replaces high-precision values with lower-precision representations and can be applied after training or incorporated through quantization-aware training [7, 12]. Knowledge distillation trains a compact student to reproduce the outputs or intermediate representations of a larger teacher [13, 14].

Compact-by-design networks. Efficient architectures reduce cost through constrained operators and balanced scaling. MobileNet uses depthwise separable convolution [15], whereas EfficientNet jointly scales depth, width, and resolution [3]. Neural architecture search automates the selection of such structures under accuracy or hardware objectives [16]. Tensor parameterizations provide another compact-by-design route by replacing dense weight tensors with structured factors.

2.2 Compaction by Tensor Factorization

Low-rank tensor methods exploit multidimensional structure to reduce parameters and, when implementations are favorable, arithmetic cost.

CP and low-rank convolution. CP decomposition expresses a tensor as a sum of rank-one terms and has been used to construct separable convolutional filters [17, 18]. MLconv similarly constrains convolutional kernels through multilinear low-rank factors while retaining local sliding-window support [19].

Tucker-style mappings and tensor contractions. Tucker decompositions compact fully connected and convolutional layers through a core and mode factors [20]. Tensor Contraction Layers (TCLs) apply learned mode-wise contractions to high-order activations, typically as dimensionality-reduction or terminal prediction modules [21]; a quantum-based formulation replaces part of this contraction with parameterized quantum processing [22]. MTL uses the same algebraic primitive of mode products but serves a different architectural role: it replaces selected intermediate convolutions while reproducing the spatial and channel dimensions expected by residual classification and detection networks.

Tensor Train and Tensor Ring. Tensor Train (TT) represents high-order weights as a chain of low-dimensional cores and has compacted both fully connected and convolutional layers [23, 24]. TT ranks can be selected to balance capacity and resource use [25]. Tensor Ring closes the TT chain into a cycle and has enabled compact neural parameterizations such as TR-Nets [26].

2.3 Adaptive Training Methods

Structured sparsity regularization drives complete filters, channels, or neurons toward zero and permits their removal during training [5, 27]. Dynamic Rank Pruning [28], also evaluated here, adapts the ranks of convolutional parameterizations during optimization. Slimmable networks and channel-gating methods vary width or execution paths according to resource or input conditions [29, 30], whereas sparse evolutionary training adapts sparse connectivity directly [31]. LR-MTL differs in the object being adapted: it learns and physically clips the intermediate channel rank of a global mode-separable activation operator. The rank is adaptive conditional on the chosen replacement depth, regularization strength, and clipping schedule; these hyperparameters are not eliminated by the method.

3 Preliminaries

We denote scalars by lowercase letters (e.g., x, i, T), vectors by boldface lowercase letters (e.g., $\mathbf{x}, \mathbf{z}$), matrices by boldface uppercase letters (e.g., $\mathbf{X}, \mathbf{U}$), and tensors by bold Euler script letters (e.g., $\mathcal{X}, \mathcal{Y}$). For a 3-way tensor $\mathcal{X}$, we use $\mathcal{X}(i, j, p)$ to indicate the scalar element at position (i, j, p) .

Tensor Operations. The mode- n matrix unfolding (or flattening) of a tensor $\mathcal{X} \in \mathbb{R}^{D_1 \times \dots \times D_N}$, denoted $\mathbf{X}_{(n)}$, is a matrix of size $D_n \times \prod_{k \neq n} D_k$ formed by arranging the mode- n fibers of $\mathcal{X}$ as its columns [32]. The mode- n product of $\mathcal{X}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times D_n}$ is written as $\mathcal{Y} = \mathcal{X} \times_n \mathbf{U}$. The resulting tensor $\mathcal{Y}$ has dimensions $D_1 \times \dots \times D_{n-1} \times J \times D_{n+1} \times \dots \times D_N$. In unfolded form, this is equivalent to a standard matrix product:

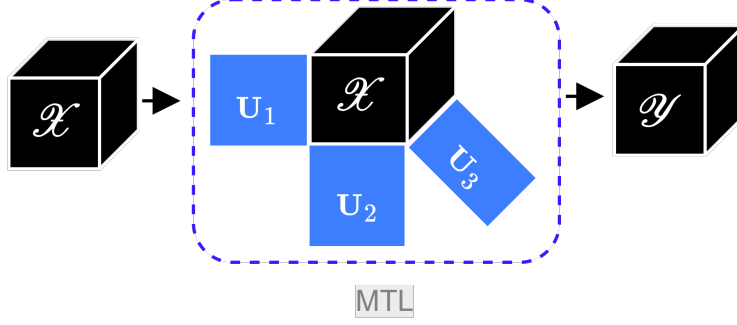


Figure 1: Multilinear Transformation Layer (MTL). The activation tensor is transformed along its two spatial modes and its channel mode by the trainable factors $\mathbf{U}_1$, $\mathbf{U}_2$, and $\mathbf{U}_3$, respectively.

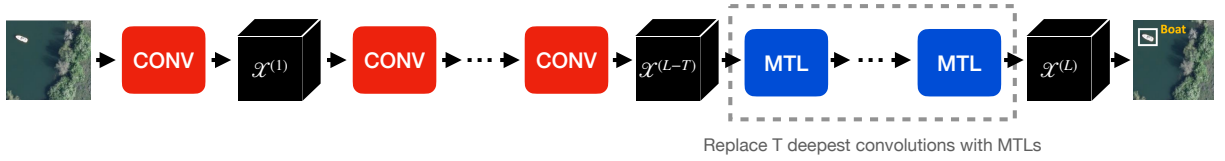


Figure 2: Selective replacement strategy. From a fixed architecture-specific ordered set of eligible convolutions, the deepest T eligible operators—implemented as the final T entries in the deterministic candidate order—are replaced by shape-preserving MTL or LR-MTL operators. All remaining convolutional, normalization, activation, residual, and task-head structures are retained.

$\mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)}$. A sequence of such products can transform a tensor across all modes:

$$\tilde{\mathcal{X}} = \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \cdots \times_N \mathbf{U}_N. \quad (1)$$

where $\mathbf{U}_n \in \mathbb{R}^{d_n \times D_n}$ and $\tilde{\mathcal{X}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$. If $d_n < D_n$, the transformation contracts mode n ; if this holds for every mode, the operation is a tensor contraction. Mode products along distinct modes commute, and the transformation can be written through unfoldings and Kronecker products [32]:

$$\tilde{\mathbf{X}}_{(n)} = \mathbf{U}_n \mathbf{X}_{(n)} (\mathbf{U}_N \otimes \cdots \otimes \mathbf{U}_{n+1} \otimes \mathbf{U}_{n-1} \otimes \cdots \otimes \mathbf{U}_1)^\top. \quad (2)$$

Convolution in CNNs. In a standard convolutional layer, the input tensor $\mathcal{X} \in \mathbb{R}^{D \times D \times C}$ is convolved with a filter tensor $\mathcal{F} \in \mathbb{R}^{k \times k \times C \times O}$ to yield the output tensor $\mathcal{Y} \in \mathbb{R}^{d \times d \times O}$. The operation is defined as:

$$\mathcal{Y}(i, j, p) = \langle \mathcal{X}_{i,j}, \mathcal{F}(:, :, :, p) \rangle, \quad (3)$$

for $1 \leq i, j \leq d$ and $1 \leq p \leq O$. Here, $\mathcal{X}_{i,j} = \mathcal{X}(i : i+k-1, j : j+k-1, :)$ $\in \mathbb{R}^{k \times k \times C}$ denotes the corresponding input patch, and $\langle \cdot, \cdot \rangle$ is the tensor inner product. For $\mathcal{H}, \mathcal{K} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$,

$$\langle \mathcal{H}, \mathcal{K} \rangle = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} \mathcal{H}(i_1, i_2, i_3) \cdot \mathcal{K}(i_1, i_2, i_3). \quad (4)$$

This formulation highlights the multidimensional nature of feature maps, which can be manipulated not only through localized sliding-window convolutions but also via global multilinear transformations.

4 Proposed Method

4.1 Replacing Deep Convolutions with Multilinear Transformations

Let the input activation be $\mathcal{X} \in \mathbb{R}^{D \times D \times C}$, and let the surrounding network require an output $\mathcal{Y} \in \mathbb{R}^{d \times d \times O}$. The proposed MTL applies one trainable matrix to each tensor mode:

$$\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3, \quad (5)$$

where $\mathbf{U}_1, \mathbf{U}_2 \in \mathbb{R}^{d \times D}$ and $\mathbf{U}_3 \in \mathbb{R}^{O \times C}$. Equivalently,

$$\mathcal{Y}(j_1, j_2, j_3) = \sum_{i_1=1}^D \sum_{i_2=1}^D \sum_{i_3=1}^C \mathcal{X}(i_1, i_2, i_3) U_1(j_1, i_1) U_2(j_2, i_2) U_3(j_3, i_3), \quad (6)$$

for $1 \leq j_1, j_2 \leq d$ and $1 \leq j_3 \leq O$. The spatial factors may preserve resolution ($d = D$) or implement a learned change of spatial size ($d \neq D$), while $\mathbf{U}_3$ maps the C input channels to the required O output channels.

The MTL itself is linear. Biases, normalization layers, nonlinearities, and residual connections already surrounding the replaced convolution remain in the backbone. This design makes the operator-level comparison with convolution explicit while preserving the nonlinear network in which the layer is embedded.

Selective replacement and interface preservation. For each backbone, the implementation constructs one fixed architecture-specific ordered set of eligible convolutions. This candidate set is held fixed across MTL, LR-MTL, direct-truncation, and 1×1 -replacement experiments. The candidates are enumerated in deterministic module-registration order, and the final T candidates in this ordered list are replaced. Each replacement reproduces the spatial and channel dimensions expected at the output of the corresponding convolution. Consequently, downstream residual additions, normalization layers, nonlinearities, and classification or detection heads receive tensors of the same shape as in the original network. The deterministic candidate-construction rules, tested replacement depths, clipping settings, and reduced-data construction are documented in Section 5.

Distinction from a dense fully connected replacement. After vectorization, an unrestricted linear map from $\text{vec}(\mathcal{X}) \in \mathbb{R}^{D^2 C}$ to $\text{vec}(\mathcal{Y}) \in \mathbb{R}^{d^2 O}$ would require $d^2 O D^2 C$ parameters. MTL instead contains

$$P_{\text{MTL}} = 2dD + CO \quad (7)$$

and imposes the Kronecker-separable structure $\mathbf{U}_3 \otimes \mathbf{U}_2 \otimes \mathbf{U}_1$. Thus, MTL is a structured global activation map rather than an unstructured fully connected layer.

Parameter-saving regime. A dense $k \times k$ convolution with C input and O output channels contains $k^2 CO$ trainable weights. Therefore, MTL is smaller exactly when

$$2dD < (k^2 - 1)CO. \quad (8)$$

This condition explains the selective use of MTL in later, wider stages, where C and O are large and spatial dimensions are relatively small. It can fail in early high-resolution stages with few channels.

Architectural scope. MTL differs deliberately from convolution. Its spatial factors have global support, so MTL does not inherit convolutional translation equivariance, and the factors are defined for particular input and output spatial dimensions. These properties motivate retaining ordinary convolutions in early feature extraction and using MTL only for selected deeper operators. Resolution-independent factors, factor interpolation, and hardware-specialized tensor kernels are outside the present formulation.

4.2 Differentiability and End-to-End Optimization

The transformation in (5) is differentiable with respect to its input and all factor matrices. For mode $n \in \{1, 2, 3\}$, define

$$\mathbf{Z}_{(n)} = \mathbf{X}_{(n)} (\mathbf{U}_3 \otimes \cdots \otimes \mathbf{U}_{n+1} \otimes \mathbf{U}_{n-1} \otimes \cdots \otimes \mathbf{U}_1)^\top, \quad (9)$$

where the factor $\mathbf{U}_n$ is omitted from the Kronecker product. The mode- n unfolding of the output is then

$$\mathbf{Y}_{(n)} = \mathbf{U}_n \mathbf{Z}_{(n)}. \quad (10)$$

For a scalar objective $\mathcal{L}$, the differential satisfies

$$d\mathcal{L} = \text{tr} \left[\left(\frac{\partial \mathcal{L}}{\partial \mathbf{Y}_{(n)}} \right)^\top d\mathbf{U}_n \mathbf{Z}_{(n)} \right]. \quad (11)$$

Using cyclic invariance of the trace gives

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}_n} = \frac{\partial \mathcal{L}}{\partial \mathbf{Y}_{(n)}} \mathbf{Z}_{(n)}^\top. \quad (12)$$

Thus, the MTL factors are optimized jointly with all unreplaced CNN parameters by ordinary end-to-end backpropagation. Modern automatic-differentiation libraries compute these derivatives through successive tensor contractions and do not require explicit construction of the Kronecker product in (9).

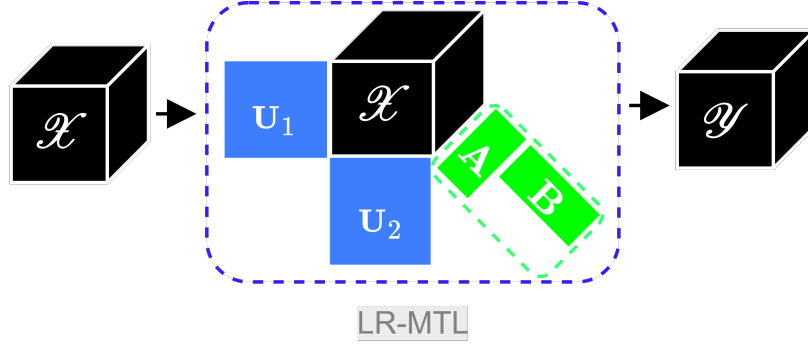


Figure 3: Low-Rank Multilinear Transformation Layer (LR-MTL). The channel factor $\mathbf{U}_3$ is represented as $\mathbf{AB}$. During training, complete columns of $\mathbf{A}$ and the matching rows of $\mathbf{B}$ are physically removed, reducing the active channel rank and stored model size.

4.3 Low-Rank Multilinear Transformation Layer

In deep and wide MTLs, the channel factor $\mathbf{U}_3 \in \mathbb{R}^{O \times C}$ usually dominates the parameter count. LR-MTL retains the same spatial transformations and factorizes the channel map as

$$\mathbf{U}_3 = \mathbf{AB}, \quad \mathbf{A} \in \mathbb{R}^{O \times H}, \quad \mathbf{B} \in \mathbb{R}^{H \times C}, \quad (13)$$

where $H \leq \min(C, O)$ is the active intermediate channel dimension. The LR-MTL mapping is

$$\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 (\mathbf{AB}), \quad (14)$$

with elementwise form

$$\mathcal{Y}(j_1, j_2, j_3) = \sum_{i_1=1}^D \sum_{i_2=1}^D \sum_{i_3=1}^C \mathcal{X}(i_1, i_2, i_3) U_1(j_1, i_1) U_2(j_2, i_2) \sum_{h=1}^H A(j_3, h) B(h, i_3). \quad (15)$$

Its trainable parameter count is

$$P_{\text{LR-MTL}} = 2dD + H(C + O). \quad (16)$$

We initialize $H = \min(C, O)$. At this width, $\mathbf{AB}$ can represent any matrix in $\mathbb{R}^{O \times C}$, so the initial LR-MTL channel map is not constrained to a manually selected low rank. The architecture subsequently becomes compact through the adaptive procedure below.

4.4 Adaptive Rank Learning

Consider T LR-MTL layers. For layer t , let $\mathbf{A}^{(t)} \in \mathbb{R}^{O_t \times H_t}$ and denote its i th column by $\mathbf{a}_i^{(t)}$. We encourage complete channel-rank components to become small through the layer-normalized group penalty

$$\mathcal{L}_{\text{reg}} = \frac{1}{T} \sum_{t=1}^T \frac{1}{\sqrt{O_t}} \left(\frac{1}{H_t} \sum_{i=1}^{H_t} \|\mathbf{a}_i^{(t)}\|_2 \right). \quad (17)$$

The inner average reduces direct dependence on the current active width H_t , the factor $1/\sqrt{O_t}$ compensates for the dimension of each column group, and $1/T$ averages across the replaced layers. The implemented training objective is

$$\mathcal{L}_{\text{train}} = (1 - \lambda) \mathcal{L}_{\text{app}} + \lambda \mathcal{L}_{\text{reg}}, \quad 0 \leq \lambda < 1, \quad (18)$$

where $\mathcal{L}_{\text{app}}$ is the task loss. Multiplication by the positive constant $(1 - \lambda)^{-1}$ shows that (18) has the same minimizers as

$$\mathcal{L}_{\text{app}} + \beta \mathcal{L}_{\text{reg}}, \quad \beta = \frac{\lambda}{1 - \lambda}. \quad (19)$$

This equivalence clarifies that λ values near one correspond to large conventional regularization weights. Their numerical values should therefore be interpreted together with the scale of the task loss and should not be compared across datasets in isolation.

Algorithm 1 Adaptive training of LR-MTL

Require: Dataset $\mathcal{D}$; CNN f_θ ; number of replaced layers T ; task loss $\mathcal{L}_{\text{app}}$; regularization weight λ ; clipping threshold ϵ ; clipping epochs $\mathcal{E}_{\text{clip}}$; total epochs E

- 1: Construct the fixed ordered candidate set and replace its final T eligible convolutions by shape-preserving LR-MTL layers.
- 2: For each replaced layer t , initialize $H_t = \min(C_t, O_t)$ and the factors $\mathbf{U}_1^{(t)}, \mathbf{U}_2^{(t)}, \mathbf{A}^{(t)}, \mathbf{B}^{(t)}$.
- 3: **for** epoch $e = 1, \dots, E$ **do**
- 4: **for** mini-batch $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ **do**
- 5: Compute the model output and task loss $\mathcal{L}_{\text{app}}$.
- 6: Compute $\mathcal{L}_{\text{reg}}$ from (17).
- 7: Update all surviving network parameters using (18).
- 8: **end for**
- 9: **if** $e \in \mathcal{E}_{\text{clip}}$ **then**
- 10: **for** LR-MTL layer $t = 1, \dots, T$ **do**
- 11: Form $\mathcal{I}_t$ according to (20).
- 12: Set $\mathbf{A}^{(t)} \leftarrow \mathbf{A}_{:, \mathcal{I}_t}^{(t)}$ and $\mathbf{B}^{(t)} \leftarrow \mathbf{B}_{\mathcal{I}_t, :}^{(t)}$.
- 13: Set $H_t \leftarrow |\mathcal{I}_t|$.
- 14: **end for**
- 15: Rebuild optimizer parameter groups using the same optimizer hyperparameters.
- 16: **end if**
- 17: **end for**
- 18: **return** The checkpoint selected according to the dataset-specific evaluation protocol in Section 5, together with its physically compact architecture.

At a clipping epoch $e \in \mathcal{E}_{\text{clip}}$, layer t retains the component indices

$$\mathcal{I}_t = \left\{ i : \left\| \mathbf{a}_i^{(t)} \right\|_2 > \epsilon \max_j \left\| \mathbf{a}_j^{(t)} \right\|_2 \right\}, \quad 0 \leq \epsilon < 1. \quad (20)$$

The columns outside $\mathcal{I}_t$ are deleted from $\mathbf{A}^{(t)}$, and the corresponding rows are deleted from $\mathbf{B}^{(t)}$. This is a physical structural update, not a binary mask: the parameter tensors become smaller, and subsequent optimization proceeds on the reduced architecture. Because parameter objects change, the optimizer is rebuilt after each clipping event while preserving its stated hyperparameters. The dataset-specific clipping epochs and threshold values are reported in the experimental setup.

Interpretation of the clipping score. The factorization $\mathbf{AB}$ is non-unique: a column of $\mathbf{A}$ can be multiplied by a nonzero scalar while the matching row of $\mathbf{B}$ is divided by the same scalar without changing their product. Therefore, the score in (20) is an optimization-induced gate in the learned factor coordinates; it is not an invariant singular value of $\mathbf{U}_3$. Both factors are nevertheless trained jointly under the same optimizer and weight-decay settings, and the reported active ranks and parameter counts are read from the resulting physically clipped architecture. Replacing (20) with a scale-invariant pair score would define a different algorithm and is not assumed in the reported experiments.

Standard mini-batch optimization, automatic differentiation, and full-width initialization are not individually novel. The contribution of LR-MTL is the combination of (i) a global mode-separable replacement operator, (ii) a factorized channel map, (iii) normalized group regularization on complete rank components, and (iv) scheduled physical clipping during end-to-end training.

4.5 Operator-Level Positioning

Let $\mathbf{x} = \text{vec}(\mathcal{X})$ and $\mathbf{y} = \text{vec}(\mathcal{Y})$. The proposed MTL induces

$$\mathbf{y} = (\mathbf{U}_3 \otimes \mathbf{U}_2 \otimes \mathbf{U}_1) \mathbf{x}, \quad (21)$$

and LR-MTL induces

$$\mathbf{y} = ((\mathbf{AB}) \otimes \mathbf{U}_2 \otimes \mathbf{U}_1) \mathbf{x}. \quad (22)$$

For an output index (j_1, j_2, j_3) , the corresponding row of the MTL operator factorizes as

$$\mathbf{w}_{\text{MTL}}(j_1, j_2, j_3) = \mathbf{U}_3(j_3, :) \otimes \mathbf{U}_2(j_2, :) \otimes \mathbf{U}_1(j_1, :). \quad (23)$$

Table 1: Structural comparison of the operators considered in this work.

Method	Object parameterized	Spatial support	Adaptive physical rank
Convolution	Dense $k \times k$ convolution kernels	Local	No
LR convolution [18]	Separable convolution kernels	Local	No
MLconv [19]	Multilinearly factorized convolution kernels	Local	No
Dense FC replacement	Unstructured activation map	Global	No
MTL (ours)	Mode-separable activation map	Global	No
LR-MTL (ours)	Low-rank mode-separable activation map	Global	Yes

Table 2: Trainable parameters and arithmetic complexity for an input of size $D \times D \times C$ and an output of size $d \times d \times O$. For the local convolutional operators, MAC counts are evaluated over the $d \times d$ output grid. The MTL and LR-MTL costs are the minimum over the displayed valid contraction orders.

Method	Trainable parameters	Multiply–accumulate complexity
Conv	$k^2 CO$	$\mathcal{O}(k^2 d^2 CO)$
LR [18]	$K(k^2 C + O)$	$\mathcal{O}(k^2 d^2 CK + d^2 KO)$
MLconv [19]	$R(k^2 C + k^2 O + CO)$	$\mathcal{O}(Rk^2 d^2 (C + O))$
MTL	$2dD + CO$	$\mathcal{O}\left(\min\left\{\begin{array}{l} D^2 Cd + DCd^2 + Cd^2 O, \\ D^2 Cd + DCdO + Dd^2 O, \\ D^2 CO + D^2 dO + Dd^2 O \end{array}\right\}\right)$
LR-MTL	$2dD + H(C + O)$	$\mathcal{O}\left(\min\left\{\begin{array}{l} D^2 Cd + DCd^2 + d^2 CH + d^2 HO, \\ D^2 CH + D^2 dH + Dd^2 H + d^2 HO, \\ D^2 Cd + DCdH + Dd^2 H + d^2 HO \end{array}\right\}\right)$

Hence, every MTL output depends on the complete input tensor, while the same three-mode factors are shared across all output positions.

This structure is fundamentally different from convolutional kernel compression. A standard convolution induces a sparse block-Toeplitz linear operator whose nonzero entries correspond to a local $k \times k$ receptive field. Spatial low-rank convolution [18] factorizes local convolution kernels into separable stages, and MLconv [19] constrains the local convolutional kernel through multilinear factors. Both retain sliding-window support and convolutional weight sharing. MTL and LR-MTL instead parameterize the activation-to-activation map directly and have global spatial support. A dense fully connected replacement is also global, but it lacks the Kronecker separation that makes MTL compact. Table 1 summarizes these distinctions without claiming that one operator family contains another as a special case.

4.6 Parameters and Arithmetic Complexity

Table 2 collects the trainable parameter counts and multiply–accumulate (MAC) complexities. The expressions assume square input/output spatial dimensions and omit constant factors. For MTL and LR-MTL, the cost depends on the order of the mode products because each contraction changes the dimensions of the intermediate tensor; the table therefore reports the minimum over representative valid orders.

MTL is smaller than the replaced convolution under (8). LR-MTL further replaces the channel term CO by $H(C + O)$ and is smaller than MTL exactly when

$$H < \frac{CO}{C + O}. \quad (24)$$

The active H is learned through the regularization-and-clipping procedure rather than selected as a single fixed final rank before training.

The LR-MTL complexity expressions assume that $\mathbf{B}$ is applied before $\mathbf{A}$, so that the intermediate channel dimension is reduced from C to H before it is expanded to O . The current reference implementation instead materializes $\mathbf{AB}$ and then applies the resulting channel matrix. This does not change the physical parameter count in (16), but it can increase executed operations and latency. Accordingly, Section 5 reports physical parameters, measured operations, and latency separately rather than treating parameter reduction as a guarantee of runtime acceleration.

5 Numerical Studies

5.1 Evaluation Protocol

We evaluate ResNet-50 on ImageNet-1K, ResNet-101 on CIFAR-10, and YOLO11x on DOTA-v1.0. The comparison set includes low-rank convolution [18], MLconv [19], Network Slimming [5], Dynamic Rank Pruning [28], direct truncation, and 1×1 replacement. The simple baselines use the same ordered candidate-layer set as MTL/LR-MTL and preserve the tensor interfaces required by the surrounding network. Low-rank and multilinear convolution ranks are chosen to span parameter budgets that overlap the proposed methods. Complete measured grids are retained in the tables so that method comparison is not reduced to a single favorable point. Table 6 is a retrospective, descriptive resource summary: its representative rows are selected by a stated score-only rule and are not presented as an independent model-selection experiment.

Candidate construction and replacement ordering. For the reported torchvision ResNet-50 and ResNet-101 experiments, the candidate set comprises all `nn.Conv2d` modules, including 1×1 channel projections and short-cut/downsampling projections. The modules are enumerated in the deterministic registration order returned by `named_modules()`, and the final T entries in this ordered list are selected for replacement.

For the reported YOLO11x-OBb experiments, the candidate set comprises all Ultralytics Conv modules outside the OBb prediction head, identified in the implementation by the prefix `model.23`. These modules are likewise enumerated in deterministic module-registration order, and the final T eligible entries are selected.

No candidate-exclusion options or run-specific replacement lists were used in the reported experiments. The same ordered candidate set is used for MTL, LR-MTL, direct truncation, and the 1×1 replacement baseline, so the compared methods modify the same architectural positions for a given T .

Clipping and reduced-data construction. In the reported LR-MTL experiments, the clipping criterion in (20) was evaluated after every training epoch, i.e., $\mathcal{E}_{\text{clip}} = \{1, \dots, E\}$. The relative threshold was $\epsilon = 0.05$ for ImageNet-1K, CIFAR-10, and the standard DOTA-v1.0 configurations; the aggressive DOTA-v1.0 $T = 60$ sweep used $\epsilon = 0.1$. The fixed CIFAR-10 two-fifths regime contains 20,000 training images sampled uniformly without replacement from the 50,000-image training set using NumPy’s `default_rng` with seed 2505. The standard 10,000-image test set is unchanged. For DOTA-v1.0, the reduced-data runs use the Ultralytics trainer with `fraction=2/3`, seed 2505, and deterministic execution enabled. All methods within a reduced-data regime use the same subset-generation settings. The DOTA experiments use Ultralytics version 8.3.146.

Trainable parameter counts are obtained from the physical model after clipping or pruning. For each run, the reported predictive score and parameter count correspond to the checkpoint with the highest monitored evaluation score under the dataset-specific protocol. ImageNet-1K and DOTA-v1.0 use their validation sets for this selection. For CIFAR-10, the archived experiments use the standard 10,000-image test loader because no separate validation split was created; consequently, the CIFAR-10 results are interpreted as single-run operating-point observations rather than strictly held-out test estimates. Fig. 10 marks the selected checkpoint and the final epoch separately so that they are not conflated. Predictive scores are never synthesized from architecture-only graphs. For resource measurements whose original trained checkpoint was unavailable, an architecture-equivalent physical graph was instantiated solely for GMAC and latency profiling; this does not affect the reported accuracy or mAP.

One MAC denotes one multiply–accumulate. ImageNet latency is synchronized validation-batch time per image over 10 batches of 128 images. CIFAR-10 and DOTA latency use FP32 batch-1 CUDA events over 3,000 timed forwards after 200 warm-up forwards. All latency measurements use an NVIDIA RTX A6000. LR-MTL GMACs include materialization of **AB** in the reference implementation. The DOTA profiler follows the archived hook convention and can omit unsupported functional operations; DOTA GMAC values are therefore used only for within-block comparisons. Parameter count, executed operations, and measured latency are reported separately because none is treated as a proxy for the others.

5.2 Large-Scale Classification on ImageNet-1K

Setup. ImageNet-1K [33] contains approximately 1.28 million training images and 50,000 validation images from 1,000 classes. ResNet-50 [34] is trained for 90 epochs using SGD with momentum 0.9, initial learning rate 0.1 with cosine decay, weight decay 10^{-4} , and batch size 256. MTL evaluates $T \in \{5, 15, 45\}$, and LR-MTL evaluates the (T, λ) pairs listed in Table 3. The common candidate construction and epoch-wise clipping protocol are specified in Section 5. Top-1 and Top-5 accuracy are reported on the standard validation set.

Table 3: ImageNet-1K results for ResNet-50. All evaluated configurations are retained to separate method comparison from hyperparameter sensitivity. Parameter deltas are relative to the 25.56M-parameter baseline; positive values denote reductions.

Configuration	Top-1 / Top-5 (%)	Δ accuracy (pp)	# Params (M)	Δ Params (%)
<i>Baseline Network</i>				
ResNet-50	76.70 / 93.22	–	25.56	–
<i>Simple Baselines</i>				
Truncated ($T = 5$)	76.44 / 92.85	−0.26/ − 0.37	17.69	+30.8
Truncated ($T = 15$)	74.02 / 91.78	−2.68/ − 1.44	8.65	+66.2
Truncated ($T = 45$)	48.50 / 72.44	−28.20/ − 20.78	2.25	+91.2
1×1 Bottleneck ($T = 5$)	76.07 / 92.95	−0.63/ − 0.27	21.36	+16.4
1×1 Bottleneck ($T = 15$)	73.81 / 91.81	−2.89/ − 1.41	18.20	+28.8
1×1 Bottleneck ($T = 45$)	44.88 / 69.07	−31.82/ − 24.15	15.54	+39.2
<i>Replacement Baselines</i>				
LR [18] ($K = 298$)	71.38 / 90.40	−5.32/ − 2.82	21.35	+16.5
LR [18] ($K = 250$)	68.87 / 88.83	−7.83/ − 4.39	18.25	+28.6
LR [18] ($K = 209$)	71.58 / 90.38	−5.12/ − 2.84	15.60	+39.0
MLconv [19] ($R = 6$)	75.70 / 92.58	−1.00/ − 0.64	21.92	+14.2
MLconv [19] ($R = 3$)	74.69 / 92.09	−2.01/ − 1.13	18.07	+29.3
MLconv [19] ($R = 1$)	75.49 / 92.55	−1.21/ − 0.67	15.51	+39.3
Net Slim [5] ($p = 0.08$)	76.26 / 93.01	−0.44/ − 0.21	21.55	+15.7
Net Slim [5] ($p = 0.16$)	75.45 / 92.47	−1.25/ − 0.75	18.32	+28.3
Net Slim [5] ($p = 0.23$)	65.85 / 86.75	−10.85/ − 6.47	16.37	+36.0
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.35$)	71.22 / 90.35	−5.48/ − 2.87	24.50	+4.1
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.55$)	68.11 / 88.61	−8.59/ − 4.61	20.99	+17.9
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.75$)	68.36 / 88.14	−8.34/ − 5.08	19.54	+23.6
<i>Proposed Methods</i>				
MTL ($T = 5$)	75.82 / 92.83	−0.88/ − 0.39	21.36	+16.4
MTL ($T = 15$)	75.17 / 92.33	−1.53/ − 0.89	18.22	+28.7
MTL ($T = 45$)	70.34 / 89.03	−6.36/ − 4.19	15.62	+38.9
LR-MTL ($T = 5, \lambda = 0.600$)	76.13 / 93.01	−0.57/ − 0.21	21.64	+15.3
LR-MTL ($T = 5, \lambda = 0.500$)	76.02 / 92.99	−0.68/ − 0.23	22.30	+12.8
LR-MTL ($T = 15, \lambda = 0.150$)	75.92 / 92.78	−0.78/ − 0.44	21.95	+14.1
LR-MTL ($T = 15, \lambda = 0.145$)	75.98 / 92.65	−0.72/ − 0.57	21.90	+14.3
LR-MTL ($T = 45, \lambda = 0.050$)	70.24 / 89.33	−6.46/ − 3.89	20.73	+18.9
LR-MTL ($T = 45, \lambda = 0.045$)	71.64 / 90.20	−5.06/ − 3.02	20.69	+19.1

Results. Table 3 and Fig. 4 show that the proposed operators scale to ImageNet-1K but do not support universal superiority. MTL with $T = 5$ removes 16.4% of the parameters and reaches 75.82% Top-1, 0.88 points below the baseline. LR-MTL with $T = 5$ and $\lambda = 0.600$ removes 15.3% of the parameters with a smaller 0.57-point decrease. At similar parameter counts, the evaluated low-rank convolution and Dynamic Rank configurations lose more than five Top-1 points, whereas MLconv ($R = 6$) is one point below the baseline. Direct truncation ($T = 5$) and Network Slimming ($p = 0.08$), however, are stronger ImageNet competitors: they retain 76.44% and 76.26% Top-1 and also reduce measured computation and latency. ImageNet therefore establishes large-scale viability and competitiveness against tensor and dynamic-rank baselines, not dominance over all structured-pruning or removal strategies. The corresponding optimization and validation trajectories are summarized in Fig. 5.

5.3 Classification on CIFAR-10

Setup. CIFAR-10 [35] contains 50,000 training and 10,000 test images. We report accuracy on this standard test set; the fixed two-fifths regime subsamples only the 50,000-image training set and evaluates on the same 10,000-image test set. The archived training logs and convergence plots label this evaluation loader “validation” (retained verbatim in Fig. 7’s axis), but no distinct CIFAR-10 validation split was created. The same 10,000-image test loader was monitored during training and used to identify the reported checkpoint for each run. ResNet-101 [34] is trained for 200 epochs with Adam, batch size 128, initial learning rate 10^{-3} , and weight decay 10^{-5} . We evaluate both the full training set and the fixed 20,000-image two-fifths subset described in Section 5.1, shared by all methods. MTL uses $T \in \{3, 4, 15, 16, 42, 43\}$, and LR-MTL uses the (T, λ) pairs listed in Table 4. The table retains every

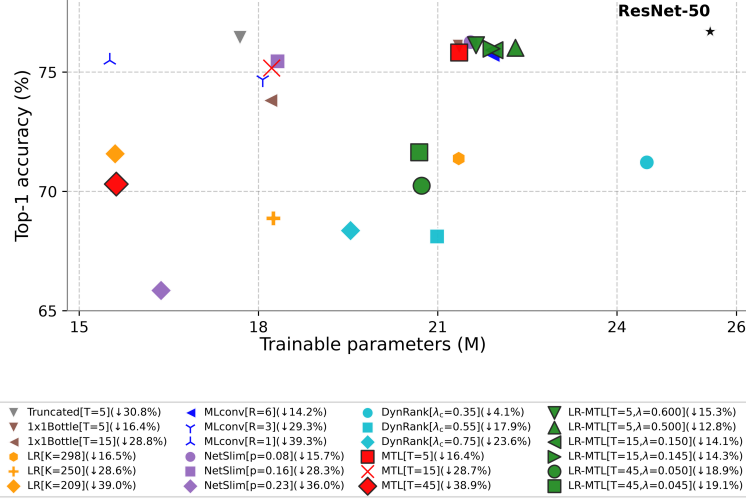


Figure 4: ImageNet-1K Top-1 validation accuracy versus physical parameter count for ResNet-50. All configurations in Table 3 are shown as individual operating points. Direct truncation and Network Slimming provide strong competing trade-offs, whereas moderate MTL and LR-MTL settings demonstrate scalable parameter reduction with limited accuracy degradation.

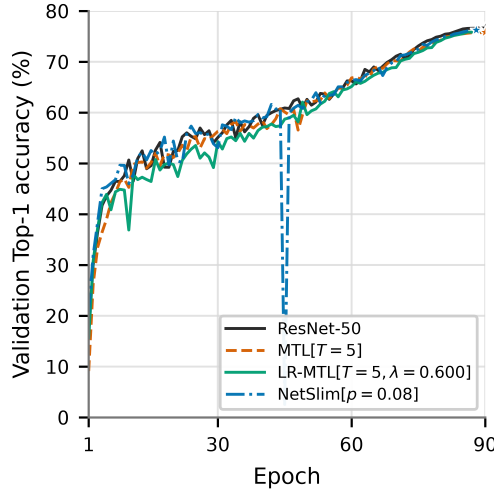


Figure 5: Validation Top-1 accuracy convergence on ImageNet-1K for the original ResNet-50, MTL with $T=5$, LR-MTL with $T=5$ and $\lambda=0.600$, and Network Slimming with $p=0.08$. The transient Network Slimming drop corresponds to its pruning and model-reconstruction transition, after which training continues on the compact architecture.

completed operating point to expose sensitivity to depth and regularization rather than reporting only the most favorable configuration.

Full-data accuracy and compaction. Table 4 and Fig. 6(a) show that MTL improves the 86.07% baseline across every evaluated depth, reaching 87.45% with $T=43$ while removing 29.6% of the parameters. LR-MTL provides the strongest proposed compaction result: $T=42$, $\lambda=0.2$ reaches 87.38% with 17.80M parameters, corresponding to a 58.1% reduction and a 1.31-point accuracy gain. The direct truncation result is an important counterexample to a universal MTL claim: Truncated $T=42$ reaches 87.50% with only 15.59M parameters and is also faster. We therefore interpret CIFAR-10 as evidence that the baseline network is heavily overparameterized and that multiple structural reductions can improve generalization. The distinctive advantage of LR-MTL is adaptive rank learning and its reduced-data behavior, rather than sole ownership of the full-data Pareto frontier.

Table 4: CIFAR-10 test accuracy and trainable parameters for ResNet-101 using the full and two-fifths training sets. A single parameter count indicates the same physical architecture in both regimes; paired counts indicate data-dependent LR-MTL, Network Slimming, or Dynamic Rank architectures. Positive deltas denote improvements or parameter reductions.

Configuration	Accuracy: Full / 2/5 (%)	Δ accuracy (pp)	# Params: Full / 2/5 (M)	Δ Params (%)
<i>Baseline Network</i>				
ResNet-101	86.07 / 79.86	–	42.52	–
<i>Simple Baselines</i>				
Truncated ($T = 42$)	87.50 / 80.83	+1.43/ + 0.97	15.59	+63.3
Truncated ($T = 15$)	86.22 / 80.35	+0.15/ + 0.49	25.61	+39.8
Truncated ($T = 3$)	87.45 / 80.11	+1.38/ + 0.25	38.06	+10.5
1×1 Bottleneck ($T = 42$)	87.02 / 80.18	+0.95/ + 0.32	30.43	+28.4
1×1 Bottleneck ($T = 15$)	87.08 / 80.35	+1.01/ + 0.49	35.17	+17.3
1×1 Bottleneck ($T = 3$)	86.94 / 81.54	+0.87/ + 1.68	40.42	+4.9
<i>Replacement Baselines</i>				
LR [18] ($K = 229$)	86.28 / 80.25	+0.21/ + 0.39	30.86	+27.4
LR [18] ($K = 260$)	87.22 / 79.73	+1.15/ – 0.13	35.03	+17.6
LR [18] ($K = 300$)	86.76 / 79.90	+0.69/ + 0.04	40.39	+5.0
MLconv [19] ($R = 4$)	86.76 / 79.44	+0.69/ – 0.42	30.85	+27.4
MLconv [19] ($R = 6$)	86.37 / 79.09	+0.30/ – 0.77	35.70	+16.0
MLconv [19] ($R = 8$)	86.94 / 80.29	+0.87/ + 0.43	40.82	+4.0
Net Slim [5] ($p = 0.159$)	87.15 / 81.61	+1.08/ + 1.75	29.43 / 30.33	+30.8/ + 28.7
Net Slim [5] ($p = 0.019$)	87.23 / 81.13	+1.16/ + 1.27	39.88 / 40.29	+6.2/ + 5.2
Net Slim [5] ($p = 0.091$)	87.43 / 81.36	+1.36/ + 1.50	34.27 / 34.71	+19.4/ + 18.4
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.5$)	87.15 / 82.53	+1.08/ + 2.67	24.31 / 30.82	+42.8/ + 27.5
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.3$)	87.42 / 82.03	+1.35/ + 2.17	24.43 / 29.73	+42.5/ + 30.1
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.1$)	86.77 / 82.53	+0.70/ + 2.67	25.67 / 30.13	+39.6/ + 29.1
<i>Proposed Methods</i>				
MTL ($T = 43$)	87.45 / 80.84	+1.38/ + 0.98	29.94	+29.6
MTL ($T = 42$)	87.36 / 81.19	+1.29/ + 1.33	30.46	+28.4
MTL ($T = 16$)	87.35 / 81.25	+1.28/ + 1.39	34.66	+18.5
MTL ($T = 15$)	87.43 / 81.33	+1.36/ + 1.47	35.18	+17.3
MTL ($T = 4$)	87.32 / 82.06	+1.25/ + 2.20	38.33	+9.9
MTL ($T = 3$)	87.39 / 81.48	+1.32/ + 1.62	40.42	+4.9
LR-MTL ($T = 42, \lambda = 0.2$)	87.38 / 82.13	+1.31/ + 2.27	17.80 / 25.89	+58.1/ + 39.1
LR-MTL ($T = 42, \lambda = 0.05$)	86.42 / 82.77	+0.35/ + 2.91	23.27 / 26.54	+45.3/ + 37.6
LR-MTL ($T = 15, \lambda = 0.15$)	87.12 / 81.98	+1.05/ + 2.12	28.45 / 30.68	+33.1/ + 27.8
LR-MTL ($T = 15, \lambda = 0.1$)	87.31 / 82.47	+1.24/ + 2.61	27.50 / 31.39	+35.3/ + 26.2
LR-MTL ($T = 3, \lambda = 0.05$)	86.96 / 82.55	+0.89/ + 2.69	37.20 / 39.48	+12.5/ + 7.1
LR-MTL ($T = 3, \lambda = 0.03$)	87.01 / 82.54	+0.94/ + 2.68	37.12 / 38.75	+12.7/ + 8.9

Limited-data generalization. Under two-fifths training data, Fig. 6(b) shows that the baseline reaches 79.86%. MTL improves this to 82.06% at $T = 4$, a 2.20-point gain. LR-MTL reaches 82.77% at $T = 42, \lambda = 0.05$ and 82.47% at $T = 15, \lambda = 0.1$, gains of 2.91 and 2.61 points. Dynamic Rank is the closest adaptive competitor at 82.53%, but LR-MTL attains the best reported test accuracy and, for the $T = 42$ settings, smaller physical models. These results support the hypothesis that the multilinear constraint and adaptive rank removal act as useful capacity controls when training data are limited.

5.4 Object Detection on DOTA-v1.0

Setup. DOTA-v1.0 [36] contains 2,806 aerial images across 15 categories. YOLO11x [37, 38] is trained for 300 epochs with Adam, batch size 16, initial learning rate 10^{-3} , weight decay 5×10^{-4} , and nonmaximum-suppression IoU threshold 0.2. We report mAP@50 for the full training set and the deterministic two-thirds regime described in Section 5.1, using identical subset-generation settings across methods. The standard LR-MTL configurations use $\epsilon = 0.05$, while the aggressive $T = 60$ sweep uses $\epsilon = 0.1$. All completed configurations are retained in Table 5.

Results. Table 5 and Fig. 8 provide the clearest comparison with naive structural removal. At approximately 20% parameter reduction, Truncated $T = 23$ loses 0.48/1.53 mAP points in the full/two-thirds regimes. MTL $T = 24$

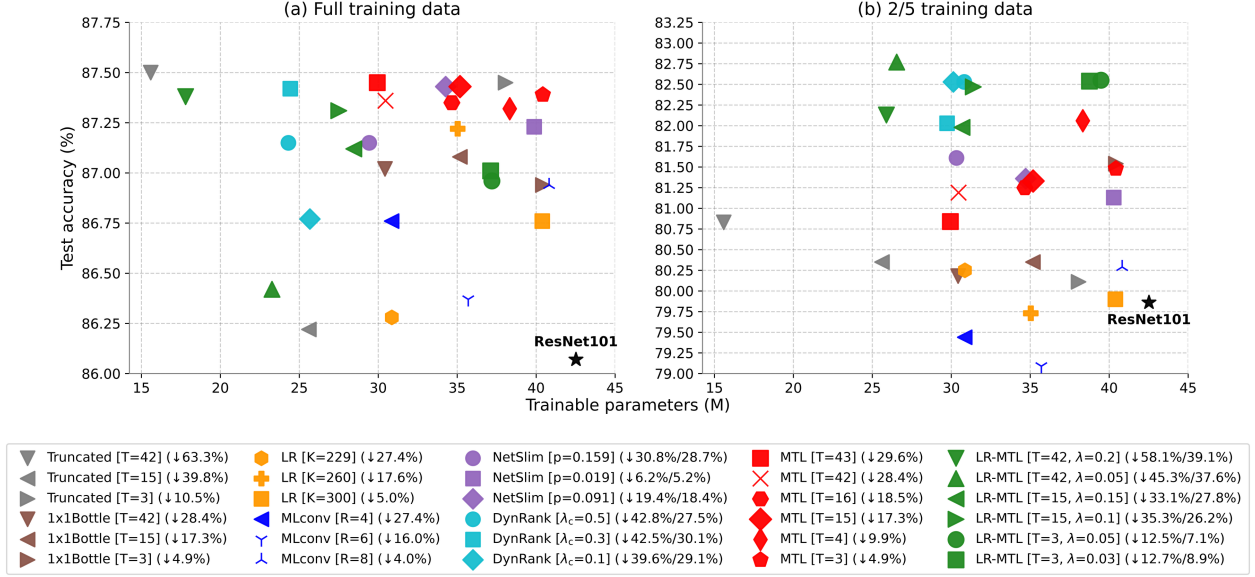


Figure 6: CIFAR-10 accuracy–parameter trade-offs for ResNet-101 using (a) the complete training set and (b) a fixed two-fifths subset. Every configuration in Table 4 is shown as an unconnected operating point. Direct truncation provides the strongest full-data resource trade-off, whereas LR-MTL achieves the highest limited-data test accuracy.

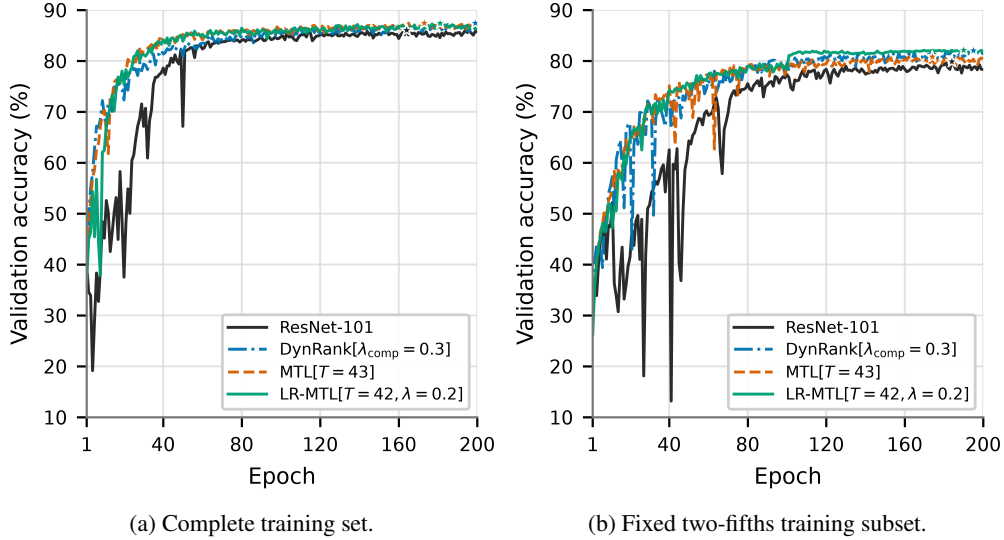


Figure 7: Test-set accuracy convergence on CIFAR-10 using (a) the complete training set and (b) a fixed two-fifths subset. The plotted axes retain the archived log label “validation accuracy”; see Section 5.3. The same representative configurations are shown in both regimes: the original ResNet-101, MTL with $T = 43$, LR-MTL with $T = 42$ and $\lambda = 0.2$, and Dynamic Rank Pruning with $\lambda_{comp} = 0.3$.

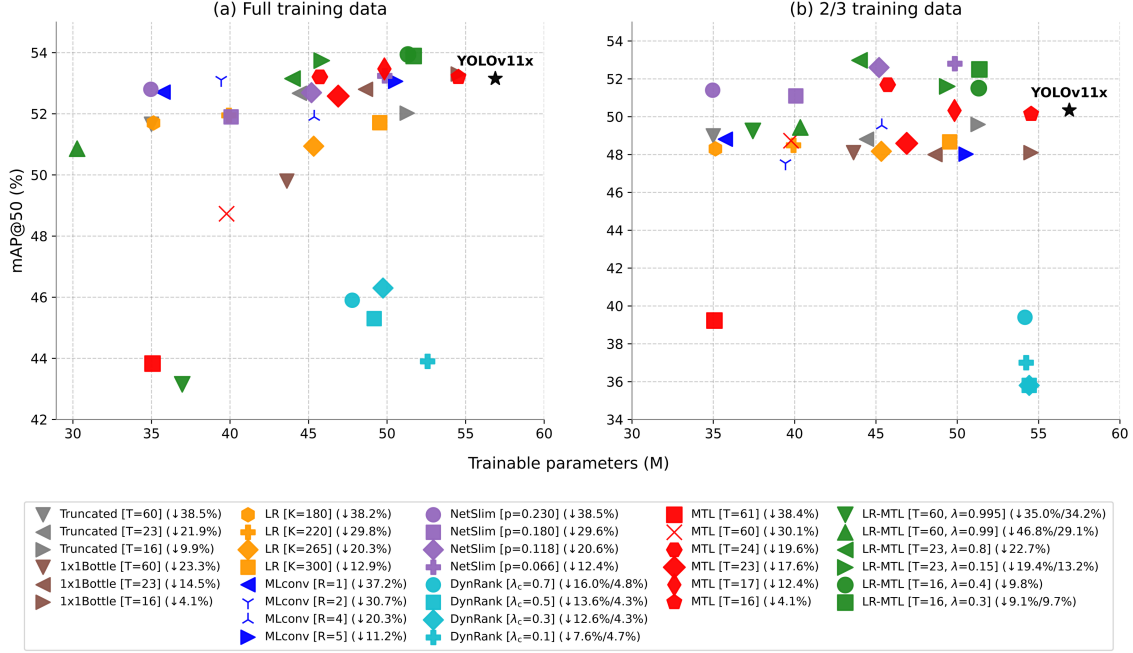


Figure 8: DOTA-v1.0 mAP@50-parameter trade-offs for YOLO11x using (a) the complete training set and (b) a fixed two-thirds subset. Every configuration in Table 5 is shown as an unconnected operating point. Selected MTL and LR-MTL settings preserve or improve detection performance where comparable direct truncation reduces mAP, while Network Slimming remains a strong structured-pruning competitor.

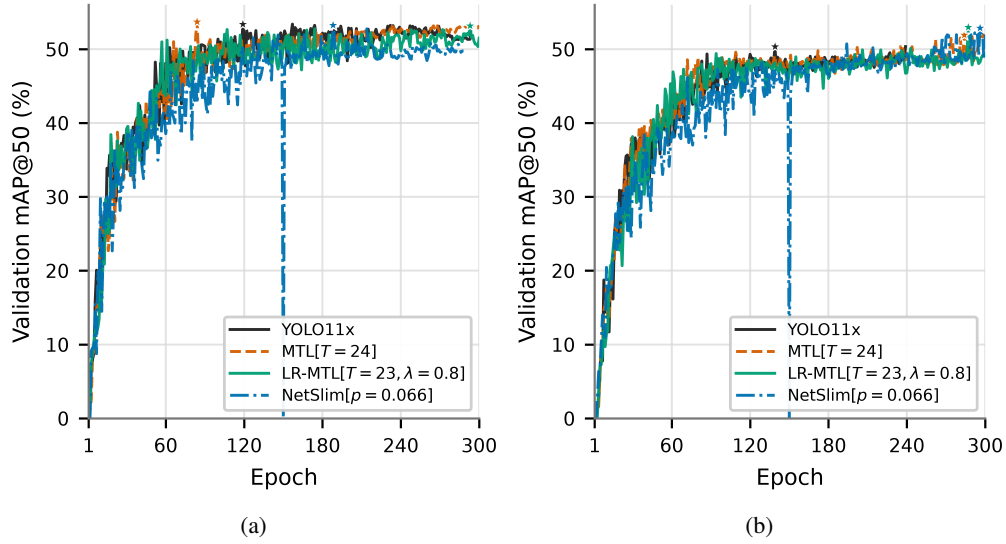


Figure 9: Validation mAP@50 convergence on DOTA-v1.0 using (a) the complete training set and (b) a fixed two-thirds training subset. The same representative configurations are shown in both regimes: the original YOLO11x, MTL with $T=24$, LR-MTL with $T=23$ and $\lambda=0.8$, and Network Slimming with $p=0.066$.

Table 5: DOTA-v1.0 mAP@50 and trainable parameters for YOLO11x using the full and two-thirds training sets. A single parameter count indicates a shared physical architecture; paired counts indicate different learned compact architectures. Positive deltas denote improvements or parameter reductions.

Configuration	mAP@50: Full / 2/3	Δ mAP (pp)	# Params: Full / 2/3 (M)	Δ Params (%)
<i>Baseline Network</i>				
YOLO11x	53.15 / 50.34	–	56.89	–
<i>Simple Baselines</i>				
Truncated ($T = 60$)	51.66 / 48.99	–1.49/ – 1.35	35.00	+38.5
Truncated ($T = 23$)	52.67 / 48.81	–0.48/ – 1.53	44.42	+21.9
Truncated ($T = 16$)	52.02 / 49.59	–1.13/ – 0.75	51.28	+9.9
1×1 Bottleneck ($T = 60$)	49.80 / 48.10	–3.35/ – 2.24	43.62	+23.3
1×1 Bottleneck ($T = 23$)	52.80 / 48.00	–0.35/ – 2.34	48.63	+14.5
1×1 Bottleneck ($T = 16$)	53.30 / 48.10	+0.15/ – 2.24	54.53	+4.1
<i>Replacement Baselines</i>				
LR [18] ($K = 180$)	51.70 / 48.30	–1.45/ – 2.04	35.13	+38.2
LR [18] ($K = 220$)	51.95 / 48.49	–1.20/ – 1.85	39.93	+29.8
LR [18] ($K = 265$)	50.94 / 48.17	–2.21/ – 2.17	45.33	+20.3
LR [18] ($K = 300$)	51.71 / 48.67	–1.44/ – 1.67	49.53	+12.9
MLconv [19] ($R = 1$)	52.71 / 48.81	–0.44/ – 1.53	35.74	+37.2
MLconv [19] ($R = 2$)	53.11 / 47.54	–0.04/ – 2.80	39.44	+30.7
MLconv [19] ($R = 4$)	51.90 / 49.54	–1.25/ – 0.80	45.36	+20.3
MLconv [19] ($R = 5$)	53.06 / 48.02	–0.09/ – 2.32	50.54	+11.2
Net Slim [5] ($p = 0.230$)	52.80 / 51.40	–0.35/ + 1.06	34.96	+38.5
Net Slim [5] ($p = 0.180$)	51.90 / 51.10	–1.25/ + 0.76	40.07	+29.6
Net Slim [5] ($p = 0.118$)	52.69 / 52.60	–0.46/ + 2.26	45.19	+20.6
Net Slim [5] ($p = 0.066$)	53.24 / 52.80	+0.09/ + 2.46	49.85	+12.4
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.7$)	45.90 / 39.40	–7.25/ – 10.94	47.78 / 54.16	+16.0/ + 4.8
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.5$)	45.30 / 35.80	–7.85/ – 14.54	49.18 / 54.42	+13.6/ + 4.3
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.3$)	46.30 / 35.80	–6.85/ – 14.54	49.74 / 54.42	+12.6/ + 4.3
Dyn. Rank [28] ($\lambda_{\text{comp}} = 0.1$)	43.90 / 37.00	–9.25/ – 13.34	52.58 / 54.24	+7.6/ + 4.7
<i>Proposed Methods</i>				
MTL ($T = 61$)	43.83 / 39.23	–9.32/ – 11.11	35.07	+38.4
MTL ($T = 60$)	48.73 / 48.73	–4.42/ – 1.61	39.78	+30.1
MTL ($T = 24$)	53.21 / 51.69	+0.06/ + 1.35	45.72	+19.6
MTL ($T = 23$)	52.58 / 48.59	–0.57/ – 1.75	46.89	+17.6
MTL ($T = 17$)	53.47 / 50.33	+0.32/ – 0.01	49.83	+12.4
MTL ($T = 16$)	53.20 / 50.14	+0.05/ – 0.20	54.54	+4.1
LR-MTL ($T = 60, \lambda = 0.995$)	43.15 / 49.25	–10.00/ – 1.09	36.96 / 37.43	+35.0/ + 34.2
LR-MTL ($T = 60, \lambda = 0.99$)	50.86 / 49.43	–2.29/ – 0.91	30.27 / 40.35	+46.8/ + 29.1
LR-MTL ($T = 23, \lambda = 0.8$)	53.15 / 52.98	0.00/ + 2.64	43.98 / 43.99	+22.7/ + 22.7
LR-MTL ($T = 23, \lambda = 0.15$)	53.74 / 51.60	+0.59/ + 1.26	45.84 / 49.38	+19.4/ + 13.2
LR-MTL ($T = 16, \lambda = 0.4$)	53.94 / 51.51	+0.79/ + 1.17	51.33 / 51.32	+9.8/ + 9.8
LR-MTL ($T = 16, \lambda = 0.3$)	53.89 / 52.50	+0.74/ + 2.16	51.69 / 51.36	+9.1/ + 9.7

instead changes performance by +0.06/ + 1.35 points, and LR-MTL $T = 23, \lambda = 0.15$ by +0.59/ + 1.26 points. LR-MTL $T = 23, \lambda = 0.8$ removes 22.7% of the parameters, matches the full-data baseline exactly, and improves two-thirds-data mAP by 2.64 points. The highest full-data mAP, 53.94%, is obtained by LR-MTL $T = 16, \lambda = 0.4$, 0.79 points above the baseline. Network Slimming remains a strong deployment-oriented competitor: $p = 0.066$ reaches 53.24/52.80 with lower GMACs and near-baseline latency. Dynamic Rank performs poorly in this detector adaptation and is retained for completeness rather than used as the main comparative evidence. The full optimization trajectories are shown in Fig. 9.

5.5 Runtime Interpretation

Table 6 separates computational behavior from accuracy–parameter results. On ImageNet, MTL $T = 5$ remains close to baseline latency, whereas Network Slimming and truncation are faster. On CIFAR-10, MTL and LR-MTL are substantially slower despite fewer parameters; the many general tensor contractions are less efficient than vendor-optimized convolution kernels, and LR-MTL additionally reconstructs AB. On DOTA, MTL and LR-MTL retain parameter and accuracy advantages at selected operating points but are slower than baseline, while Network Slimming

Table 6: Representative full-data whole-model efficiency. For each dataset, rows are selected by a fixed accuracy-based rule, independent of GMACs or latency: the baseline and the highest-scoring configuration from each of the simple, conventional tensor, modern structural/adaptive, MTL, and LR-MTL families. This is a retrospective descriptive rule, not a validation protocol. Complete accuracy-parameter sweeps appear in Tables 3–5. Latency is measured in FP32 on an NVIDIA RTX A6000. ImageNet uses synchronized batches of 128 over 1,280 images; CIFAR-10 and DOTA use batch-1 CUDA-event timing after 200 warm-ups. LR-MTL GMACs include materialization of **AB**. DOTA GMACs follow the archived hook convention and are interpreted only within the DOTA block. Positive deltas denote reductions.

Dataset	Configuration	Score (%)	Params (M)	ΔP (%)	GMACs	ΔG (%)	Latency (ms/img)	ΔL (%)
<i>ImageNet-1K / ResNet-50</i>								
ImageNet	Baseline	76.70	25.56	0.0	4.089	0.0	0.611	0.0
ImageNet	Truncated ($T = 5$)	76.44	17.69	+30.8	3.704	+9.4	0.596	+2.5
ImageNet	MLconv ($R = 6$)	75.70	21.92	+14.2	4.260	-4.2	1.856	-203.7
ImageNet	Net Slim ($p = 0.08$)	76.26	21.55	+15.7	3.079	+24.7	0.489	+19.9
ImageNet	MTL ($T = 5$)	75.82	21.36	+16.4	3.888	+4.9	0.633	-3.6
ImageNet	LR-MTL ($T = 5, \lambda = 0.600$)	76.13	21.64	+15.3	5.402	-32.1	0.634	-3.8
<i>CIFAR-10 / ResNet-101</i>								
CIFAR-10	Baseline	86.07	42.52	0.0	0.159	0.0	6.08	0.0
CIFAR-10	Truncated ($T = 42$)	87.50	15.59	+63.3	0.095	+40.5	5.41	+11.0
CIFAR-10	LR ($K = 260$)	87.22	35.03	+17.6	0.241	-51.5	8.74	-43.8
CIFAR-10	Net Slim ($p = 0.091$)	87.43	34.27	+19.4	0.137	+13.9	6.34	-4.3
CIFAR-10	MTL ($T = 43$)	87.45	29.94	+29.6	0.135	+15.1	10.24	-68.4
CIFAR-10	LR-MTL ($T = 42, \lambda = 0.2$)	87.38	17.80	+58.1	0.838	-426.3	10.51	-72.8
<i>DOTA-v1.0 / YOLO11x</i>								
DOTA	Baseline	53.15	56.89	0.0	97.242	0.0	12.12	0.0
DOTA	1×1 Bottleneck ($T = 16$)	53.30	54.53	+4.1	96.299	+1.0	12.25	-1.1
DOTA	MLconv ($R = 2$)	53.11	39.44	+30.7	99.154	-2.0	15.89	-31.1
DOTA	Net Slim ($p = 0.066$)	53.24	49.85	+12.4	84.679	+12.9	12.21	-0.7
DOTA	MTL ($T = 17$)	53.47	49.83	+12.4	95.229	+2.1	15.25	-25.8
DOTA	LR-MTL ($T = 16, \lambda = 0.4$)	53.94	51.33	+9.8	96.501	+0.8	15.45	-27.5

reduces GMACs with near-baseline latency. These measurements directly address the gap between asymptotic complexity and hardware execution: the present contribution is model compaction and adaptive capacity control, not guaranteed acceleration.

5.6 Adaptive Training and Hyperparameter Sensitivity

Figs. 10 and 11 provide mechanism-level evidence, whereas Figs. 4–8 compare predictive trade-offs. The parameter trajectories show that clipping changes the physical architecture during training rather than merely masking stored weights. The rank plots further show that adaptation is layer dependent: some channel factors retain substantial width, while others collapse to only a few active components.

The experiments use task-specific grids because T and λ control different mechanisms. The depth T determines which operators are exposed to replacement; λ , equivalently β in (19), determines the pressure toward small component groups; and ϵ converts the learned norm distribution into a discrete architecture at scheduled epochs. The response is consequently not expected to be globally monotone. On CIFAR-10 at $T = 42$, $\lambda = 0.2$ produces a smaller full-data model and higher full-data accuracy than $\lambda = 0.05$, whereas $\lambda = 0.05$ gives the highest limited-data accuracy. On DOTA at $T = 60$, increasing λ from 0.99 to 0.995 does not produce a smaller full-data model and substantially lowers mAP. These cases demonstrate that excessively aggressive regularization can alter optimization without yielding a better compact architecture.

For future studies, a stricter selection protocol should choose T from a coarse parameter-budget grid, hold ϵ and the clipping schedule fixed, and compare nearby regularization values using a held-out validation split while reserving the test set for final reporting. The present archived CIFAR-10 experiments did not include such a separate validation split. The complete measured grids are reported rather than suppressing unsuccessful settings. Automatic selection of T , regularization strength, and clipping times is outside the present scope.

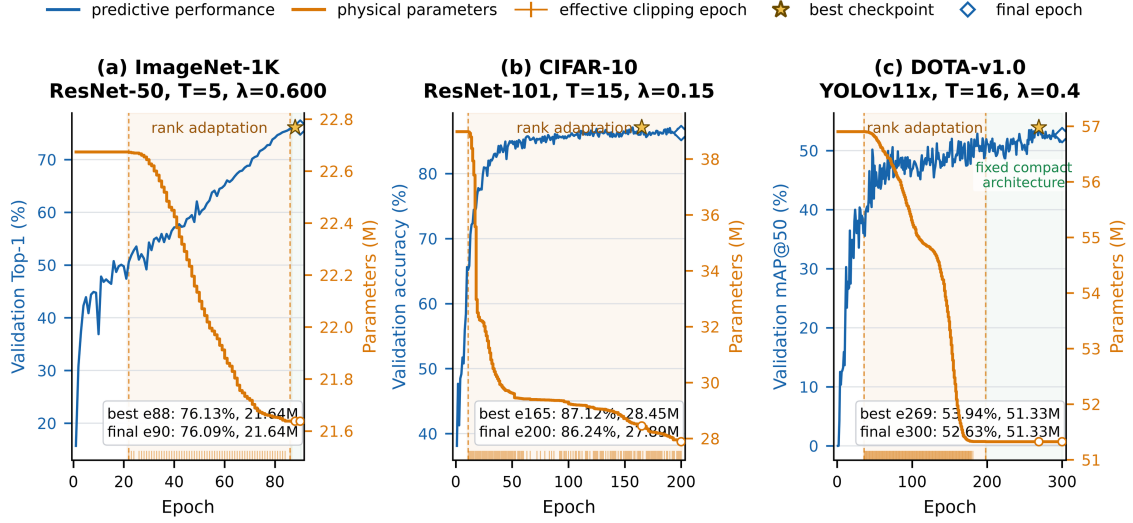


Figure 10: Representative LR-MTL training dynamics on ImageNet-1K, CIFAR-10, and DOTA-v1.0. Predictive performance and physical parameter count are plotted against epoch, with clipping epochs, the selected checkpoint, and the final epoch indicated. This separates transient post-clipping behavior from the performance of the reported compact checkpoint.

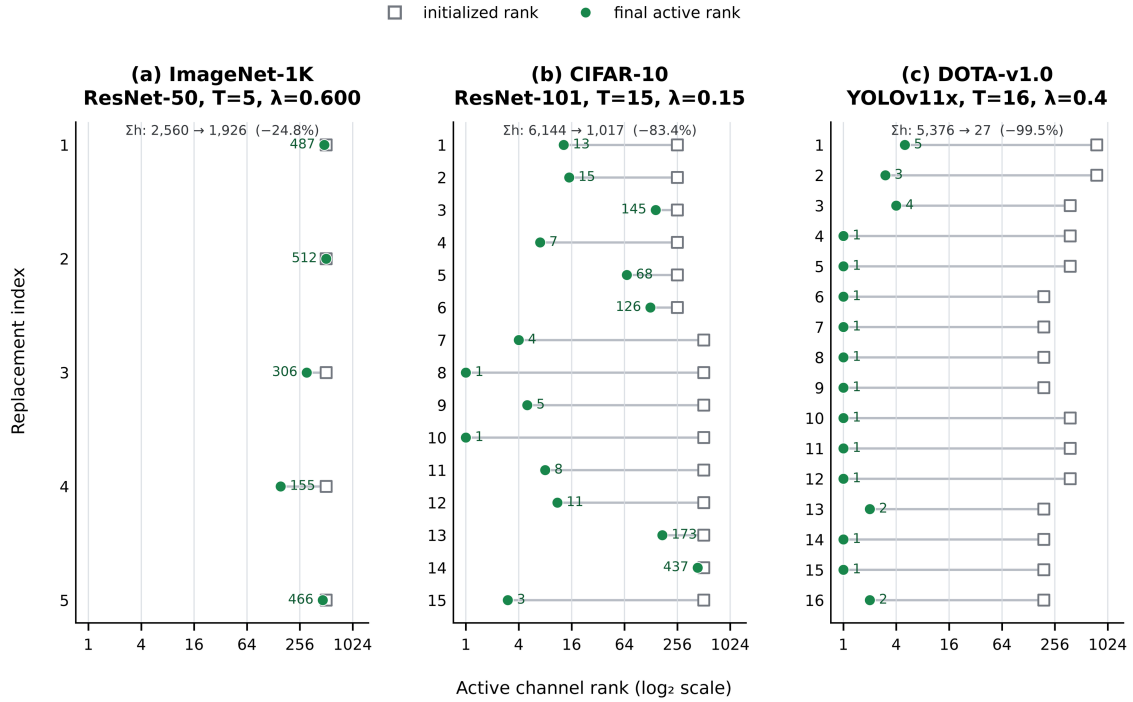


Figure 11: Layer-wise rank adaptation learned by representative LR-MTL models on ImageNet-1K, CIFAR-10, and DOTA-v1.0. Layers follow the implementation's replacement order; each panel compares the initialized rank with the active rank of the checkpoint identified in Fig. 10.

5.7 Discussion and Limitations

The evidence supports four main conclusions. First, MTL is a viable activation-level alternative to selected deep convolutions: well-calibrated configurations on CIFAR-10 and DOTA improve predictive performance while reducing parameters. Second, LR-MTL provides genuine structural adaptation because the active ranks and stored tensors change during training; its strongest results combine physical compaction with improved limited-data performance. Third, the strong CIFAR truncation and ImageNet Network Slimming results show that no method is uniformly best. The parameter-matched DOTA comparisons are therefore especially informative: MTL and LR-MTL preserve or improve mAP where direct removal loses accuracy. Fourth, resource metrics must be separated. The current implementation materializes **AB** and relies on general tensor contractions, so parameter efficiency does not imply fewer executed operations or lower latency.

Several limitations bound these conclusions. MTL is global, fixed-resolution, and not translation equivariant; its selective use in later stages mitigates but does not remove these architectural differences from convolution. The clipping score is coordinate-dependent because **AB** admits reciprocal factor rescaling, so the learned active rank is an optimization-induced architectural outcome rather than an invariant matrix-rank estimator. Hyperparameter selection remains task-dependent, and the experiments cover three CNN backbones but not transformers or variable-resolution deployment. The DOTA profiler does not capture every functional operation, and some resource-only measurements use architecture-equivalent graphs when original checkpoints are unavailable. For CIFAR-10, the standard test loader was also used for within-run checkpoint selection because the archived experiments did not include a separate validation split; the corresponding numbers should therefore be interpreted as operating-point observations rather than strictly held-out test estimates. Finally, the full sweep is reported as single trained outcomes rather than multi-seed confidence intervals. Small differences should therefore be interpreted as operating-point observations, while the larger limited-data gains and parameter-matched comparisons provide the principal empirical support.

6 Conclusion

We presented MTL, a shape-preserving global mode-separable replacement for selected deep convolutional operators, and LR-MTL, an adaptive low-rank channel formulation trained with a normalized group penalty and scheduled structural clipping. The analysis identifies the regime in which MTL reduces parameters and separates its activation-level operator from local convolutional factorizations and dense fully connected mappings. Experiments on ImageNet-1K, CIFAR-10, and DOTA-v1.0 show that these operators can substantially reduce physical model size while preserving or improving predictive performance. The clearest benefits occur under limited-data training and in DOTA settings where parameter-matched direct truncation degrades mAP. Conversely, strong truncation and Network Slimming results demonstrate that no single compaction strategy dominates every architecture or resource metric.

Measured GMACs and latency are equally important to this interpretation. The present tensor implementation does not consistently outperform optimized convolution kernels, and explicit materialization of **AB** can offset the arithmetic benefit of low rank. The contribution is therefore adaptive architectural compaction and capacity control, not a universal acceleration mechanism. Future work will apply the low-rank factors without reconstruction, develop fused hardware-aware contractions, investigate scale-invariant component criteria, automate depth and regularization selection, and extend the operator to variable-resolution and transformer-based backbones.

Acknowledgment

This material 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] 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.
- [5] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2755–2763.
- [6] 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=rJl-b3RcF7>
- [7] 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>
- [8] 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>
- [9] 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.
- [10] 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>
- [11] Y. He and L. Xiao, “Structured Pruning for Deep Convolutional Neural Networks: A Survey,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 46, no. 05, pp. 2900–2919, May 2024. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/TPAMI.2023.3334614>
- [12] B. Rokh, A. Azarpeyvand, and A. Khanteymoori, “A comprehensive survey on model quantization for deep neural networks in image classification,” *arXiv preprint arXiv:2205.07877*, 2022.
- [13] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015. [Online]. Available: <https://arxiv.org/abs/1503.02531>
- [14] G. Yang, S. Yu, Y. Sheng, and H. Yang, “Attention and feature transfer based knowledge distillation,” *Scientific Reports*, vol. 13, no. 1, p. 18369, 2023.
- [15] 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>
- [16] 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>
- [17] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, “Speeding-up convolutional neural networks using fine-tuned cp-decomposition,” *arXiv preprint arXiv:1412.6553*, 2015.
- [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] 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. [Online]. Available: <https://arxiv.org/abs/1511.06530>
- [21] 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.
- [22] V. T. Nguyen and P. P. Markopoulos, “Quantum-based tensor contraction layers,” in *Machine Learning from Challenging Data 2026*, P. P. Markopoulos, G. Sklivanitis, and B. Ouyang, Eds., vol. 14030, International Society for Optics and Photonics. SPIE, 2026, p. 140300I. [Online]. Available: <https://doi.org/10.1117/12.3098024>
- [23] 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.
- [24] 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>
- [25] M. Sharma, P. P. Markopoulos, E. Saber, M. S. Asif, and A. Prater-Bennette, “Convolutional auto-encoder with tensor-train factorization,” in *ICCV Workshops*, 2021, pp. 198–206.
- [26] W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal, “Wide compression: Tensor ring nets,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [27] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, “Learning structured sparsity in deep neural networks,” in *Advances in neural information processing systems*, ser. NIPS’16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 2082–2090.
- [28] M. Sharma, J. Heard, E. Saber, and P. Markopoulos, “Convolutional neural network compression via dynamic parameter rank pruning,” *IEEE Access*, vol. 13, pp. 18 441–18 456, 2025.

- [29] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, “Slimmable neural networks,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=H1gMCsAqY7>
- [30] W. Hua, Y. Zhou, C. M. De Sa, Z. Zhang, and G. E. Suh, “Channel gating neural networks,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/68b1fbe7f16e4ae3024973f12f3cb313-Paper.pdf
- [31] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, “Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science,” *Nature Communications*, vol. 9, no. 1, p. 2383, Jun 2018. [Online]. Available: <https://doi.org/10.1038/s41467-018-04316-3>
- [32] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Sep. 2009.
- [33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, Jun. 2009, pp. 248–255.
- [34] 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.
- [35] 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>
- [36] 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.
- [37] G. Jocher and J. Qiu, “Ultralytics yolo11,” 2024. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [38] 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>