NuWa: Deriving Lightweight Class-Specific Vision Transformers for Edge Devices

Supplementary Material

Table 5. Comparison in ViT throughput (images/s) on Jetson Orin NX before and after INT8 quantization and magnitude pruning.

Methods	Base ViT (GPU)	INT8 (CPU)	Magnitude (GPU)
DeiT-Base	22.00	0.29$_{\downarrow 98.68\%}$	21.99$_{\downarrow 0.05\%}$
DeiT-Small	64.16	0.67$_{\downarrow 98.96\%}$	63.28$_{\downarrow 1.37\%}$
DeiT-Tiny	68.87	1.50$_{\downarrow 98.48\%}$	67.21$_{\downarrow 2.41\%}$

6. Calculation of FLOPs

We adopt GFLOPs, which is widely used to measure the computational cost of model inference, to reflect the resource constraints of edge devices [1]. FLOPs represent the total number of floating-point operations required for a model to perform inference on a single input. For a Vision Transformer (ViT), let the embedding dimension be d; the query–key dimension, value–output dimension, and number of heads in the MHA module of the l-th block be q_l, v_l, and H_l, respectively; and let the intermediate dimension of the MLP module in the same block be e_l. Given an input $\mathbf{X} \in \mathbb{R}^{N\times d}$ consisting of N patch tokens, the total FLOPs of a ViT with L blocks can be formulated as:

$$\text{FLOPs} = (2Nd+N^2)\sum_{l=1}^{L} H_l(q_l+v_l)+2Nd\sum_{l=1}^{L} e_l \quad (13)$$

7. Limitations of Compression Methods

Low-bit Quantization and Unstructured Pruning. As discussed in Sec. 2, quantization and unstructured pruning rely heavily on specific hardware and software architectures to achieve real acceleration. To illustrate this, we apply INT8 quantization [51] and Magnitude Pruning (with a pruning ratio of 0.50) [16], two representative techniques of quantization and unstructured pruning, to compress DeiT-Base, DeiT-Small, and DeiT-Tiny. Then, we measure the inference throughput of compressed ViTs on a widely used edge device, Jetson Orin NX, with a batch size of 1. As shown in Tab. 5, due to the lack of TensorRT support, the INT8-quantized model can only perform inference on the CPU, resulting in a 98.71% decrease in average throughput. Similarly, the model obtained by unstructured pruning shows almost no acceleration, as current frameworks lack libraries optimized for sparse matrix operations. These results indicate that both methods suffer from poor adaptability when deployed on edge devices with highly heterogeneous frameworks and drivers.

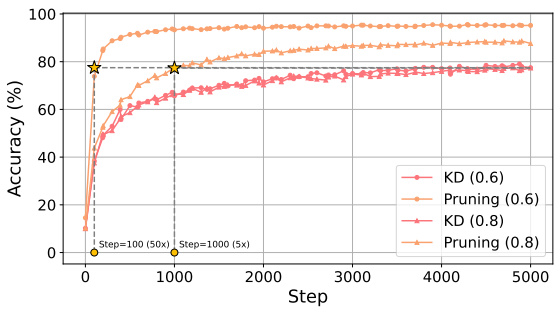

Figure 13. Comparison of convergence speed between logit-based knowledge distillation (KD) and random pruning (Pruning) across different pruning rates with DeiT-Base on CIFAR-10.

Knowledge Distillation. Knowledge distillation (KD) focuses on transferring the teacher model's knowledge to the student model in the feature space and does not provide a good initialization in the parameter space for arbitrary student architectures. As a result, the student model often needs to be trained from scratch, leading to slow convergence and high computational cost. To illustrate this, we compare the convergence of logit-based KD [20] and random structured pruning [14] with DeiT-Base on CIFAR-10 [**?**]. Specifically, we use student models that share the same architecture as the pruned ones but are randomly initialized. The teacher model is a DeiT-Base pretrained and fine-tuned on CIFAR-10. As shown in Fig. 13, the pruned models converge significantly faster than their KD counterparts, highlighting the importance of parameter-space knowledge transfer.

Although the above methods have clear limitations, they are orthogonal to NuWa and can be used together when supported by edge devices to achieve further compression.

8. Importance Metrics for Pruning

Magnitude. Magnitude pruning is a simple and widely used pruning approach in model compression [16]. The basic idea is to prune the weights with the smallest magnitudes, assuming that smaller weights contribute less to the model's output and, thus, can be removed without significantly affecting performance. In the case of pruning according to the magnitude of weights, the importance score is calculated as $I(W) = |W| = \sum_i |w_i|$.

Activation. Activation pruning measures the importance of each structure by analyzing its response to input data [6, 44]. The importance score is calculated as $I(W) = \sum_i a_i$. Structures with lower average activation

Table 6. Accuracy and pruning rate (Acc./Rate) of the anchor models under different SKP application settings.

Setting	$\mathcal{S}_4$/25	$\mathcal{S}_5$/25	$\mathcal{S}_6$/25	*Avg*
MLP Only	94.80/25.38	97.92/21.52	96.72/18.82	**96.48/21.91**
MHA Only	81.20/7.38	82.80/7.13	80.80/8.87	81.60/7.79
MHA + MLP	93.52/30.05	95.52/28.01	94.08/23.54	94.37/27.20

values are considered less important for the current task. Since it relies on input data, activation pruning can capture features specific to certain classes, making it well-suited for our class-specific model derivation.

Gradient. During backward propagation, gradients indicate how sensitive the loss function is to changes in each weight [27, 60]. Weights with smaller gradients are considered less important for the model's predictions and can be pruned with minimal impact on performance. The importance score is calculated as $I(W) = |\frac{\mathcal{L}(x)}{\partial W}| = \sum_i |\frac{\mathcal{L}(x)}{\partial w_i}|$.

Taylor Expansion Approximation. Taylor pruning assumes that a smaller change in the loss value after removing a weight indicates lower importance of that weight for prediction [37, 53]. However, accurately evaluating the importance of all N weights would require N forward propagations to compute the corresponding loss changes, which is computationally expensive. To reduce this overhead, these methods adopt the following approximation based on Taylor expansion:

$$\begin{aligned} I(w) &= |\mathcal{L}(x) - \mathcal{L}_{w=0}(x)| \\ &= |\mathcal{L}(x) - \big(\mathcal{L}(x) - \frac{\partial \mathcal{L}(x)}{\partial w} w + R(w)\big)| \\ &\overset{R(w)\approx 0}{\approx} \left| \frac{\partial \mathcal{L}(x)}{\partial w} w \right| \end{aligned} \tag{14}$$

This approximation reduces the complexity of evaluating the importance of all weights from $O(N)$ to $O(1)$.

9. Ablation and Design Justification

Application of SKP. As discussed in Sec. 3.2, NuWa does not apply SKP to the MHA modules. We conduct experiments to justify this design. Specifically, we apply SKP to different modules of DeiT-Base for the three sub-tasks $\mathcal{S}_4$/25-$\mathcal{S}_6$/25 in Tab. 1 under three settings, i.e., applying SKP only to MLPs (MLP only), only to MHAs (MHA only), and to both (MHA+MLP). As shown in Tab. 6, when SKP is applied to the MHA modules to prune attention heads, the accuracy of the resulting anchor model decreases, regardless of whether SKP is also applied to the MLP modules. This is because SKP fails to control the pruning rate of the base ViT, leading to the loss of class-relevant knowledge. These results indicate that class-detrimental knowledge mainly resides in the MLP modules, while applying SKP to the MHA interferes with the model's ability to locate and filter such knowledge. This observation is consistent with our SVD-based pruning of the MHA module, since SVD is data-free and suggests that the MHA stores class-agnostic general knowledge.

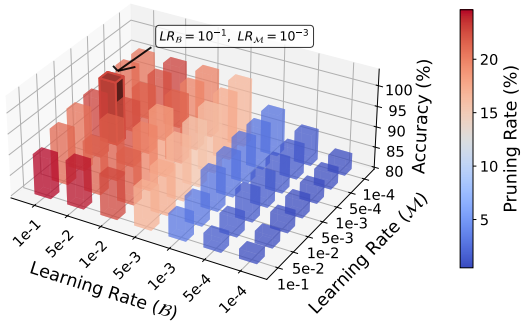

Figure 14. Effect of learning rates for $\mathcal{M}$ and $\mathcal{B}$ on the accuracy and pruning rate of the anchor model.

Table 7. Comparison of different strategies for determining e_l during Optimized-based Fast Pruning (OFP).

Setting	$\mathcal{S}_4$/25	$\mathcal{S}_5$/25	$\mathcal{S}_6$/25	*Avg*
Pruning Rate = 0.40				
Uniform	94.40	97.04	96.72	**96.05**
Proportion	95.57	96.85	95.57	96.00
Adaptive	95.49	96.53	95.81	95.94
Pruning Rate = 0.60				
Uniform	89.84	92.00	92.48	**91.44**
Proportion	88.72	92.28	91.65	90.88
Adaptive	88.88	92.20	91.57	90.88

Learning Rate of $\mathcal{M}$ and $\mathcal{B}$. During SKP, the learning rates of the mask vectors $\mathcal{M}$ and control factors $\mathcal{B}$ significantly influence how effectively class-detrimental knowledge is filtered. We evaluate different learning rate combinations on DeiT-Base for sub-tasks $\mathcal{S}_4$–$\mathcal{S}_6$, and report the average accuracy and pruning rate of the resulting anchor models $\mathcal{V}_A$. As shown in Fig. 14, a larger learning rate for $\mathcal{B}$ leads to more thorough filtering of class-detrimental knowledge, while a slightly smaller learning rate for $\mathcal{M}$ yields more precise localization of such detrimental weights. We therefore set $\text{LR}_{\mathcal{M}} = 0.001$ and $\text{LR}_{\mathcal{B}} = 0.1$, which achieve the highest $\mathcal{V}_A$ with a balanced pruning rate.

Target Architecture. Before applying OFP, NuWa needs to determine the target model architecture, i.e., the sizes of q_l, v_l, and e_l. For the MHA modules, For the MHA modules, NuWa controls q_l and v_l through the retained energy rate ρ. For the MLP modules, three strategies are considered for determining e_l:

- *Uniform*: pruning e_l across all blocks to similar sizes to avoid excessive compression in specific MLP modules.
- *Proportion*: allocating the total number of pruned neurons e_{prune} to each block proportionally based on the original e_l of the anchor model.
- *Adaptive*: globally pruning the e_{prune} neurons with the

Table 8. Comparison in model accuracy under different features sampling strategies during OFP.

Methods	$\mathcal{S}_4$/25		$\mathcal{S}_5$/25		$\mathcal{S}_6$/25		*Avg*	
	0.40	0.60	0.40	0.60	0.40	0.60	0.40	0.60
Random	94.40	89.84	97.04	92.00	96.72	92.48	96.05	91.44
Max-L2	94.16	85.36	95.68	90.48	95.92	90.24	95.25	88.69

smallest normalized activation values across all blocks. As shown in Tab. 7, when deriving models for $\mathcal{S}_4$–$\mathcal{S}_6$ from DeiT-Base at pruning rates of 0.60 and 0.40, the *Uniform* strategy consistently achieves the best performance. Considering its superior accuracy and hardware friendliness, we adopt the uniform strategy to determine e_l.

Sampling Strategy. During OFP, NuWa requires intermediate features $\mathcal{H}^{(l)} \in \mathbb{R}^{(KN)\times d}$ from K sampled images to compute the optimal $W_2^{(l)\prime}$ according to Eq. (12). We compare two sampling strategies, i.e., random sampling (Random) and selecting samples with the largest patch-token L2 norms (Max-L2). As shown in Tab. 8, random sampling yields higher model accuracy. Moreover, since Random does not involve sorting operations, its $\mathcal{P}_2^{(2)}$ in Fig. 9 is smaller than that of Max-L2. Therefore, NuWa adopts random sampling to get calibration features.

10. Proof

In this section, we prove that Eq. (8) and Eq. (12) are the closed-form solutions to Eq. (7) and Eq. (11), respectively.

Proof1: Optimal MHA Pruning. Take $W_Q \in \mathbb{R}^{q\times d}$ and $W_K \in \mathbb{R}^{q\times d}$ ($q < d$) as an example, the optimization objective is to find two matrices $W_Q' \in \mathbb{R}^{q'\times d}$ and $W_K' \in \mathbb{R}^{q'\times d}$ ($q' < q$), such that the Frobenius norm of the difference between $W_{QK} = W_Q^\top W_K \in \mathbb{R}^{d\times d}$ and $W_{QK}' = W_Q'^\top W_K' \in \mathbb{R}^{d\times d}$ is minimized, i.e.,

$$\min_{W_Q', W_K'} \|W_{QK} - W_Q'^\top W_K'\|_F^2 \tag{15}$$

We first perform a singular value decomposition (SVD) on W_{QK}, obtaining:

$$\begin{aligned} &W_{QK} = U_{QK}\Sigma_{QK}V_{QK}^\top, \quad U_{QK}, V_{QK} \in \mathbb{R}^{d\times q} \\ &\Sigma_{QK} = \text{diag}(\sigma_1, \cdots, \sigma_q) \in \mathbb{R}^{q\times q} \\ &\sigma_1 \geq \cdots \geq \sigma_q \geq 0, \quad \text{rank}(W_{QK}) \leq q \end{aligned} \tag{16}$$

For any $W \in \mathbb{R}^{d\times d}$ with $\text{rank}(W) \leq q'$, let $Y = U_{QK}^\top W V_{QK} \in \mathbb{R}^{q\times q}$. Since the Frobenius norm $\|\cdot\|_F$ is invariant under left and right multiplication by orthogonal matrices, we have:

$$\begin{aligned} &\|W_{QK} - W\|_F = \|\Sigma_{QK} - Y\|_F \\ &\text{rank}(Y) = \text{rank}(W) \leq q' \end{aligned} \tag{17}$$

Therefore, it suffices to minimize $\|\Sigma_{QK} - Y\|_F^2$ over all matrices Y with $\text{rank}(Y) \leq q'$. Expanding the above expression, we obtain:

$$\begin{aligned} \|\Sigma_{QK} - Y\|_F^2 &= \|\Sigma_{QK}\|_F^2 + \|Y\|_F^2 - 2\langle\Sigma_{QK}, Y\rangle \\ &= \sum_{i=1}^{q}\sigma_i^2 + \sum_i s_i(Y)^2 - 2\text{tr}(\Sigma_{QK}^\top Y) \end{aligned} \tag{18}$$

where $s_i(Y)$ denotes the i-th singular value of Y. Since the von Neumann trace inequality satisfies:

$$\text{tr}(\Sigma_{QK}^\top Y) \leq \sum_i \sigma_i s_i(Y) \tag{19}$$

and equality holds when Y and Σ_{QK} share the same left and right singular vectors with their singular values aligned in the same order, we have:

$$\begin{aligned} \|\Sigma_{QK} - Y\|_F^2 &\geq \sum_{i=1}^{q}\sigma_i^2 + \sum_i s_i(Y)^2 - 2\sum_i \sigma_i s_i(Y) \\ &= \sum_{i=1}^{q}(\sigma_i - s_i(Y))^2 \end{aligned} \tag{20}$$

Since $\text{rank}(Y) \leq q'$ implies $s_i(Y) = 0$ for all $i > q'$, it follows that:

$$\begin{aligned} \|\Sigma_{QK} - Y\|_F^2 &\geq \sum_{i=1}^{q'}(\sigma_i - s_i(Y))^2 + \sum_{i=q'+1}^{q}\sigma_i^2 \\ &\geq \sum_{i=q'+1}^{q}\sigma_i^2 \end{aligned} \tag{21}$$

The lower bound is achieved when $s_i(Y) = \sigma_i$ for $i \leq q'$ and Y shares the same singular vectors as Σ_{QK}. That is:

$$\begin{aligned} &Y^\star = \begin{bmatrix} \Sigma_{QK,q'} & 0 \\ 0 & 0 \end{bmatrix}, \quad W^\star = U_{QK}Y^\star V_{QK}^\top \\ &W_{QK}'^\star = W^\star = U_{QK}[:, :q']\Sigma_{QK}[:q', :q']V_{QK}[:, :q']^\top \end{aligned} \tag{22}$$

where $\Sigma_{QK,q'} = \text{diag}(\sigma_1, \cdots, \sigma_{q'})$. Therefore, when W_Q' and W_K' are given by:

$$\begin{aligned} W_Q' &= (U_{QK}[:, :q']\Sigma_{QK}[:q', :q'])^\top \\ W_K' &= V_{QK}[:, :q']^\top \end{aligned} \tag{23}$$

the quantity $\|W_{QK} - W_Q'^\top W_K'\|_F^2$ attains its minimum value $\sum_{i=q'+1}^{q}\sigma_i^2$.

□

Proof2: Optimal MLP Pruning. For given $W_2 \in \mathbb{R}^{d\times e}$, $\mathcal{H} \in \mathbb{R}^{(KN)\times e}$, and $\mathcal{I}_r \in \mathbb{R}^{e'}$ ($e' < e$), consider the following problem:

$$\min_{W_2'} \|\mathcal{H}W_2^\top - \mathcal{H}[\mathcal{I}_r]W_2'^\top\|_F^2 \tag{24}$$

which essentially amounts to finding the least-squares solution of $W_2' \in \mathbb{R}^{d\times e'}$.

Write the column vectors of $B = \mathcal{H}W_2^\top \in \mathbb{R}^{(KN)\times d}$ as $B = [b_1, \ldots, b_d]$ ($b_i \in \mathbb{R}^{KN}$) and those of $W_2'^\top \in \mathbb{R}^{e'\times d}$ as $W_2'^\top = [w_1, \ldots, w_d]$ ($w_i \in \mathbb{R}^{e'}$). Then,

$$\begin{aligned}\|\mathcal{H}W_2^\top - \mathcal{H}[\mathcal{I}_r]W_2'^\top\|_F^2 &= \|\mathcal{H}_r W_2'^\top - B\|_F^2 \\ &= \sum_{i=1}^{d} \|\mathcal{H}_r w_i - b_i\|_2^2\end{aligned} \tag{25}$$

where $\mathcal{H}_r = \mathcal{H}[\mathcal{I}_r] \in \mathbb{R}^{KN\times e'}$. Hence, the problem decomposes completely column-wise. For any fixed i, we aim to minimize $\|\mathcal{H}_r w_i - b_i\|_2^2$. Let $\mathcal{C} = \mathrm{col}(\mathcal{H}_r) \subset \mathbb{R}^{KN}$. According to the orthogonal projection theorem, each b_i admits a unique decomposition:

$$b_i = \underbrace{P_{\mathcal{C}} b_i}_{\in\mathcal{C}} + \underbrace{r_i}_{\perp\mathcal{C}} \tag{26}$$

where $P_{\mathcal{C}}$ denotes the orthogonal projection onto $\mathcal{C}$ and r_i is residual vector. For any $w \in \mathbb{R}^{e'}$, we have:

$$\begin{aligned}\|\mathcal{H}_r w - b_i\|_2^2 &= \|\mathcal{H}_r w_i - P_{\mathcal{C}} b_i + r_i\|_2^2 \\ &= \|\mathcal{H}_r w - P_{\mathcal{C}} b_i\|_2^2 + \|r_i\|_2^2 \geq \|r_i\|_2^2\end{aligned} \tag{27}$$

This shows that $\min_w \|\mathcal{H}_r w - b_i\|_2^2$ achieves its minimum when b_i is projected onto $\mathcal{C}$, and the image of the optimal solution w must be $\mathcal{H}_r w = P_{\mathcal{C}} b_i$.

By combining the columns together, we obtain the overall optimality condition:

$$\mathcal{H}_r W^\star = P_{\mathcal{C}} B \tag{28}$$

Geometrically, this means that each column of B is simultaneously projected onto $\mathrm{col}(\mathcal{H}_r)$. From the necessary and sufficient condition of orthogonal projection, we obtain:

$$\mathcal{H}_r^\top (\mathcal{H}_r W^\star - B) = 0 \tag{29}$$

which is precisely the normal equation for the matrix least-squares problem. It is equivalent to saying that $\mathcal{H}_r W^\star$ is the orthogonal projection of B onto $\mathrm{col}(\mathcal{H}_r)$.

Let $\mathcal{H}_r^\dagger$ denote the Moore–Penrose pseudoinverse of $\mathcal{H}_r$. The orthogonal projection operator can then be written as $P_{\mathcal{C}} = \mathcal{H}_r \mathcal{H}_r^\dagger$. Hence,

$$\begin{aligned}&\mathcal{H}_r W^\star = (\mathcal{H}_r \mathcal{H}_r^\dagger) B \\ \Longrightarrow\quad &W^\star = \mathcal{H}_r^\dagger B + Z, \quad \mathcal{H}_r Z = 0\end{aligned} \tag{30}$$

Here, Z is arbitrary (it does not affect $\mathcal{H}_r W^\star$, and thus yields the same optimality). In particular, the minimum-norm optimal solution is:

$$\begin{aligned}W_2'^{\star\top} &= W^\star = \mathcal{H}_r^\dagger B = \mathcal{H}_r^\dagger \mathcal{H} W_2^\top \\ W_2'^\star &= W_2 \mathcal{H}^\top \mathcal{H}_r^{\dagger\top} = W_2 \mathcal{H}^\top ((\mathcal{H}_r^\top \mathcal{H}_r)^\dagger \mathcal{H}_r^\top)^\top \\ &= W_2 \mathcal{H}^\top \mathcal{H}_r (\mathcal{H}_r^\top \mathcal{H}_r)^\dagger\end{aligned} \tag{31}$$

Table 9. Hyperparameters for the class-specific model derivation process of NuWa. Here, $\alpha \in (0, 1)$ denotes the target pruning rate.

Hyperparameter	Value
Self-Knowledge Purification	
Steps	10000
Optimizer	AdamW [34]
Batch Size	1
Learning Rate (LR)	$\mathrm{LR}_{\mathcal{B}}$=1e-1, $\mathrm{LR}_{\mathcal{M}}$=1e-3
LR Scheuler	Constant
Weight Decay	0.05
Optimization-based Fast Pruning	
Retained Energy Ratio ρ	$-0.41\alpha^3 + 0.14\alpha^2$ $-0.03\alpha + 1.0$
#Calibration Samples	128

Algorithm 1 Block-Uniform Pruning

Input: neuron counts $\{e_l\}_{l=1}^L$, prune neuron count e_{prune}

Output: intermediate size list $\{e_l'\}_{l=1}^L$

$N_{\text{total}} \leftarrow \sum_{l=1}^L e_l$ ▷ total neurons count
$N_{\text{target}} \leftarrow \lfloor (N_{\text{total}} - e_{\text{prune}})/L \rfloor$ ▷ target neurons count
$I \leftarrow \mathrm{argsort}(e_l)$ ▷ Sort blocks in ascending order
$e_l' \leftarrow e_l$ for all l, $L_{\text{count}} \leftarrow 0$
for each i in I **do**
  $L_{\text{count}} \leftarrow L_{\text{count}} + 1$
  if $e_i \geq N_{\text{target}}$ **then**
    $e_i' \leftarrow N_{\text{target}}$ ▷ Eq. (10)
    $e_{\text{prune}} \leftarrow e_{\text{prune}} - (e_i - e_i')$
  end if
  $N_{\text{total}} \leftarrow N_{\text{total}} - e_i$
  $N_{\text{target}} \leftarrow \lfloor (N_{\text{total}} - e_{\text{prune}})/(L - L_{\text{count}}) \rfloor$
end for
return $\{e_l'\}_{l=1}^L$

The minimum objective value is determined by the projection residual:

$$\begin{aligned}\min_W \|\mathcal{H}_r W - B\|_F^2 &= \|(I - \mathcal{H}_r \mathcal{H}_r^\top) B\|_F^2 \\ &= \sum_{i=1}^{d} \|(I - \mathcal{H}_r \mathcal{H}_r^\top) b_i\|_2^2\end{aligned} \tag{32}$$

That is, it represents the sum of squared components of B lying in the orthogonal complement of $\mathrm{col}(\mathcal{H}_r)$. □

11. Implementation Details

We summarize the hyperparameter settings of NuWa in Tab. 9. In practical implementation, NuWa further optimizes Eq. (10) to strictly enforce:

$$\sum_{l=1}^{L} e_l' = \left(\sum_{l=1}^{L} e_l\right) - e_{\text{prune}} \tag{33}$$

as described in Algorithm 1.

For the Swin Transformer [33], since the number of input patch tokens varies with image size, the corresponding GFLOPs are not fixed. Therefore, NuWa adopts the number of parameters (#Param) as the metric to measure the resource constraint and pruning rate. NuWa distributes the total prunable parameter budget proportionally among the MLP modules of each stage in $\mathcal{V}_A$, computes e_{prune} for each stage, and determines e_l' for each block following Algorithm 1.

Moreover, NuWa ensures that the pruned dimensions q_l', v_l', and e_l' are multiples of 8, in order to achieve accelerated inference across a wide range of edge devices.

12. Baselines

NuWa is compared with seven baselines implemented based on the open-source code from GitHub.

- **Magnitude Structured Pruning** [16]. Given the overall pruning rate α, uniform pruning is applied to the q_l, v_l, and e_l dimensions in each block. Specifically, dimensions with the smallest L2 norms are pruned by removing the corresponding rows or columns in the weight matrices without post-pruning retraining.
- **Wanda-sp** [44]. Wanda-SP is the structured-pruning variant of Wanda. Given the overall pruning rate α, it uniformly prunes the q_l, v_l, and e_l dimensions in each block. Considering the massive activations in Transformer forward propagation, Wanda computes importance scores as the product of activation magnitudes and corresponding weight magnitudes. Weights with the lowest scores are pruned by removing the associated rows or columns in the weight matrices. No retraining is performed after pruning.
- **Numerical Pruning** [43]. Numerical Pruning estimates the importance of attention heads in MHA and neuron groups in MLP using Newton method. According to the computed importance scores and the overall pruning rate α, it adaptively prunes H_l and e_l across blocks. After pruning, a compensation weights is calculated for each pruned weight to restore accuracy, without performing any retraining.
- **X-Pruner** [59]. X-Pruner applies learnable masks to H_l and e_l in each block, which are optimized via gradient descent with sparsity-inducing regularization added to the loss function. During training, dimensions with smaller mask values are progressively pruned until convergence. After pruning based on the learned sparse masks, X-Pruner performs retraining to recover accuracy.
- **DC-ViT** [61]. DC-ViT determines pruning candidates based on each block's recoverability and the overall pruning rate α. It then removes entire MHA modules from the selected blocks and randomly prunes the expansion dimensions e_l of the MLP modules to achieve the target sparsity. After pruning, the model is retrained to restore accuracy.
- **RECAP** [22]. RECAP alternates between pruning, fine-tuning, and updating to reduce memory usage while preserving accuracy. It estimates weight importance via a Taylor-based criterion and prunes H_l and e_l across blocks accordingly. Only important weights are updated with a Fisher-based mask, achieving substantial memory savings without full-model retraining.
- **MDP** [45].MDP jointly prunes multiple dimensions of ViTs, including the embedding dimension (d), the number of attention heads (H_l), the query–key and value–output dimensions (q_l, v_l), and the MLP intermediate dimension (e_l). It formulates pruning as a mixed-integer nonlinear program (MINLP) problem under latency budgets, solved with Hessian-based importance scores and a precomputed latency lookup table (LUT). After pruning, the model is retrained to restore accuracy.

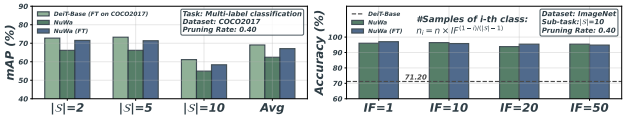

Figure 15. Performance of NuWa under class co-occurrence and long-tailed settings.

13. Performance under Complex Settings

To account for the complexity of real-world deployment scenarios, we further evaluate NuWa under two challenging settings: class co-occurrence and long-tailed distributions.

For the class co-occurrence setting, where multiple categories may appear within a single image, we sample multi-label classification sub-tasks of varying scales from COCO2017 and derive edge ViTs from a DeiT-Base model fine-tuned on COCO2017. As shown in the left panel of Fig. 15, at a pruning rate of 0.40, the models derived by NuWa achieve an average precision of 97.13% relative to the base ViT. This result demonstrates the strong generality of NuWa in complex multi-label scenarios.

For the long-tailed setting, we construct long-tailed variants of each ImageNet sub-task dataset using the following transformation:

$$n_i = n \times \text{IF}^{(1-i)/(|\mathcal{S}|-1)} \tag{34}$$

where n is the per-class count in the balanced set, n_i is the count for the i-th class after imbalance, $|\mathcal{S}|$ is the number of classes in the sub-task, and IF is the imbalance factor (IF=1 means balanced). As shown in the right panel of Fig. 15, for sub-tasks containing 10 classes, the models derived by NuWa maintain stable accuracy across different imbalance levels at a pruning rate of 0.40, demonstrating its robustness under long-tailed data distributions.

14. Pseudocode

Algorithm 2 NuWa – Class-Specific Model Derivation

Input: pretrained all-class base ViT $\mathcal{V}_B$, sub-task $\mathcal{S}$, class-specific data $\mathcal{D}_\mathcal{S}$, overall pruning rate α

Output: lightweight class-specific edge ViT $\mathcal{V}_E$

 # Self-Knowledge Purification (SKP)
$\mathcal{M} = \{M^{(l)}\}_{l=1}^{L} \leftarrow \{\mathbf{1} \in \mathbb{R}^{e_l}\}_{l=1}^{L}$
$\mathcal{B} = \{\beta^{(l)}\}_{l=1}^{L} \leftarrow \{5.0\}_{l=1}^{L}$
Freeze parameters of $\mathcal{V}_B$
Embed $\mathcal{M}$ and $\mathcal{B}$ into the MLP modules of $\mathcal{V}_B$
Train $\mathcal{M}$ and $\mathcal{B}$ on $\mathcal{D}_\mathcal{S}$ under supervision of $\mathcal{L}_T$ ▷ $\mathcal{M}$ and $\mathcal{B}$ are involved in the forward propagation according to Eq. (3) and Eq. (4)
Prune $\mathcal{V}_B$ according to $\mathcal{M}$ and $\mathcal{B}$ to obtain the anchor model $\mathcal{V}_A$
 # Optimization-based Fast Pruning
Compute ρ based on α ▷ equation in Tab. 9
Compute $\{q_l'\}_{l=1}^{L}$ and $\{v_l'\}_{l=1}^{L}$ based on ρ ▷ Eq. (9)
$\mathcal{V}_A \leftarrow$ Prune MHA modules of $\mathcal{V}_A$ using SVD based on $\{q_l'\}_{l=1}^{L}$ and $\{v_l'\}_{l=1}^{L}$ ▷ Eq. (8)
$e_{\text{prune}} = \left((1-\alpha)\mathcal{F}(\mathcal{V}_B) - \mathcal{F}(\mathcal{V}_A)\right)/2Nd$ ▷ $\mathcal{F}$ denotes the function that computes GFLOPs or #Params
Compute $\{e_l'\}_{l=1}^{L}$ based on e_{prune} ▷ Eq. (10) and Algorithm 1
$\{a^{(l)}\}_{l=1}^{L}$,$\{\mathcal{H}^{(l)}\}_{l=1}^{L} \leftarrow$ Perform one epoch of forward propagation of $\mathcal{V}_A$ on $\mathcal{D}_\mathcal{S}$
$\mathcal{V}_A \leftarrow$ Prune MLP modules of $\mathcal{V}_A$ based on $\{e_l'\}_{l=1}^{L}$, $\{a^{(l)}\}_{l=1}^{L}$, and $\{H^{(l)}\}_{l=1}^{L}$

15. Limitations and Future Works

A common limitation of NuWa, as well as other pruning methods that do not rely on post-pruning retraining, is that excessive pruning inevitably leads to significant accuracy degradation. As shown in Fig. 6, pruning-only methods, i.e., Random Pruning, Magnitude Pruning, and Wanda-sp suffer sharp accuracy drops when the pruning rate exceeds 20%. Numerical Pruning, which introduces compensation matrices to recover accuracy, maintains stable performance until around 50% pruning. In contrast, NuWa effectively removes class-detrimental weights through SKP, enabling the derived models to outperform the base ViT on target classes even at a pruning rate of 60%. However, when the pruning rate exceeds 70%, noticeable accuracy degradation still occurs. Moreover, as the sub-task size increases (e.g., $|\mathcal{S}| > 50$), the performance of derived models also declines.

To address the performance drop under extremely high pruning rates or large-scale sub-tasks, we envision two promising directions. First, lightweight retraining can be introduced to recover accuracy, as NuWa-derived models retain more class-specific knowledge and therefore require less retraining overhead. Second, an offline-trained hypernetwork could be developed to predict weight updates, enabling the online derivation process to efficiently map task specifications, such as model architecture, sub-task, and pruning rate, to corresponding pruned weights.