Accelerating Vision Foundation Models with Drop-in Depthwise Convolution

Carmelo Scribano[1], Mohammad Mahdi[2], Nedyalko Prisadnikov[2], Yuqian Fu[2], Giorgia Franchini[1], Danda Pani Paudel[2], Marko Bertogna[1], Luc Van Gool[2]

[1]**UniMoRe**, University of Modena and Reggio Emilia, Modena, Italy.

[2]**INSAIT**, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria.

28th International Conference on Pattern Recognition
Lyon, July 17–22, 2026

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

dAIEDGE

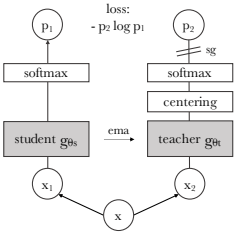
INSAIT

Overview

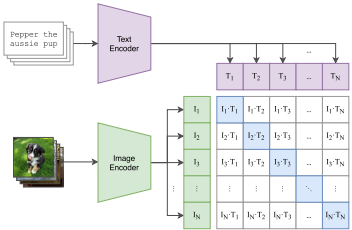
Vision Foundation Models

Large-scale pretrained ViTs are powerful, reusable visual backbones.

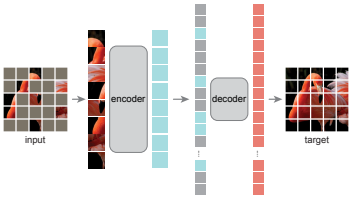
- Strong transfer with limited fine-tuning
- Classification and dense prediction from the same features
- Different pretraining objectives, shared transformer backbone



(a) DINO/V2/V3



(b) CLIP



(c) MAE

Goal: VFMs Inference at the Edge

Bringing the capabilities of fine-tuned VFMs to **embedded inference platforms** is impactful in several domains; (i.e, smart cities, automotive systems, and industrial automation).

Server Grade GPU (H200)

- 30,000 USD (per chip)
- TDP 200 W
- 141 GB VRAM
- 2,958 TOPS INT8

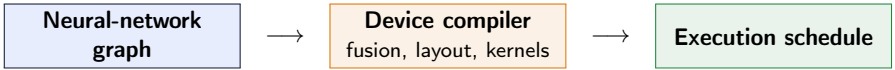
Edge AI (ORIN NANO)

- 250 USD
- TDP 7–25 W
- 8 GB VRAM
- 67 TOPS INT8

Tiny ML (STM32)

- 10 USD
- 165 mW
- 1 MB SRAM
- < 1 GOPS

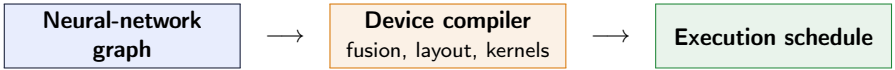
Edge Deployment: Challenges



1. Operator support

- Unsupported operations can block conversion.
- Plugins, rewrites, and fallbacks add complexity.
- Dynamic control flow is difficult to deploy.

Edge Deployment: Challenges



1. Operator support

- Unsupported operations can block conversion.
- Plugins, rewrites, and fallbacks add complexity.
- Dynamic control flow is difficult to deploy.

2. Compiler control

- The backend chooses fusions, layouts, kernels, and precision.
- Choices depend on the target and surrounding graph.
- Execution is difficult to predict analytically.

Edge Deployment: Challenges

Neural-network graph → **Device compiler** fusion, layout, kernels → **Execution schedule**

1. Operator support

- Unsupported operations can block conversion.
- Plugins, rewrites, and fallbacks add complexity.
- Dynamic control flow is difficult to deploy.

2. Compiler control

- The backend chooses fusions, layouts, kernels, and precision.
- Choices depend on the target and surrounding graph.
- Execution is difficult to predict analytically.

3. FLOPs $\neq$ latency

- Kernel launches and synchronization add overhead.
- Reshaping and memory traffic may dominate.
- Measure the compiled model on the target device.

Existing Approaches

- Efficient transformer architectures and attention mechanisms (e.g., MobileViT (**mehta2022mobilevit**), EfficientViT (**cai2023efficientvit**), Efficient Attention (**shen2021efficient**), and Mobile Attention (**yao2024mobile**))

Existing Approaches

- Efficient transformer architectures and attention mechanisms (e.g., MobileViT (mehta2022mobilevit), EfficientViT (cai2023efficientvit), Efficient Attention (shen2021efficient), and Mobile Attention (yao2024mobile))

Limitation

- Pretrained weights are discarded.

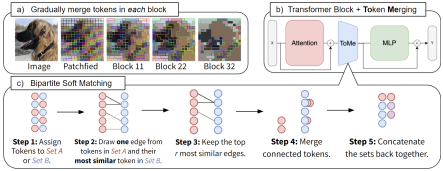
Existing Approaches

- Efficient transformer architectures and attention mechanisms (e.g., MobileViT (mehta2022mobilevit), EfficientViT (cai2023efficientvit), Efficient Attention (shen2021efficient), and Mobile Attention (yao2024mobile))

Limitation

- Pretrained weights are discarded.

- Token-count reduction (e.g., Token Merging (bolya2022tome) and Token Pooling (marin2021token))



Token Merging

Existing Approaches

- Efficient transformer architectures and attention mechanisms (e.g., MobileViT (mehta2022mobilevit), EfficientViT (cai2023efficientvit), Efficient Attention (shen2021efficient), and Mobile Attention (yao2024mobile))

Limitation

- Pretrained weights are discarded.

- Token-count reduction (e.g., Token Merging (bolya2022tome) and Token Pooling (marin2021token))

Limitation

- Introduces control flow, which is poorly supported in inference frameworks.
- Loss of spatial structure affects dense tasks (i.e., segmentation).

Other Acceleration Techniques

- Model pruning and quantization

Application

- Our framework is related to structured model pruning.
- Quantization options depend on the data types supported by the hardware.

- Better hardware utilization (e.g., FlashAttention (dao2023flashattention2))

Application

- Platform-specific solutions.
- Can work in synergy with model acceleration strategies.

Why Existing Acceleration Is Not Enough

Desiderata

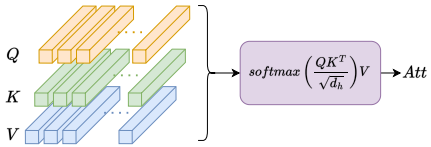
- Work across a wide range of platforms without specialized hardware or software constructs.
- Avoid inefficient control flow.
- Preserve as many large-scale pretrained weights as possible.
- Preserve spatial structure for dense vision tasks.

Can we accelerate a pretrained ViT while preserving its token grid and most of its weights?

Overview

Attention and Convolution

For one query location (i, j), an attention head aggregates values using a global, input-dependent kernel:

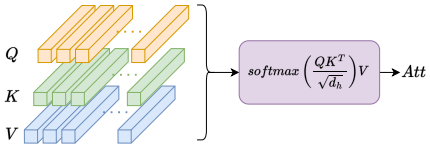


$$E(X)^h = \text{softmax}\left(\frac{Q^h {K^h}^\top}{\sqrt{d_h}}\right)$$

$$Att(X)^h_{i,j} = \sum_{r,s \in \Delta_m} E^h(X)_{(i,j),(i+r,j+s)} V^h_{i+r,j+s}$$

Attention and Convolution

For one query location (i, j), an attention head aggregates values using a global, input-dependent kernel:



$$E(X)^h = \text{softmax}\left(\frac{Q^h {K^h}^\top}{\sqrt{d_h}}\right)$$

$$Att(X)^h_{i,j} = \sum_{r,s \in \Delta_m} E^h(X)_{(i,j),(i+r,j+s)} V^h_{i+r,j+s}$$

Self-attention kernel

Global, dependent on the input, and different at every query location.

Attention and Convolution

In contrast, convolution produces a weighted local aggregation of the $k \times k$ neighborhood centered at (i, j).

$$Conv(X, W^C)_{i,j} = \sum_{r,s \in \Delta_k} {W^C_{r,s}}^\top X_{i+r,j+s}$$

Depthwise convolution performs local aggregation by applying a distinct spatial filter to each channel independently.

$$Conv_{DW}(X, W^D)_{i,j} = \sum_{(r,s) \in \Delta_k} W^D_{r,s} \odot X_{i+r,\, j+s}$$

Attention and Convolution

In contrast, convolution produces a weighted local aggregation of the $k \times k$ neighborhood centered at (i, j).

$$Conv(X, W^C)_{i,j} = \sum_{r,s \in \Delta_k} {W^C_{r,s}}^\top X_{i+r,j+s}$$

Depthwise convolution performs local aggregation by applying a distinct spatial filter to each channel independently.

$$Conv_{DW}(X, W^D)_{i,j} = \sum_{(r,s) \in \Delta_k} W^D_{r,s} \odot X_{i+r,\, j+s}$$

Convolution kernel

Local, input-independent, and translation-invariant.

Can Attention Behave Like Convolution?

Local: $\Delta_m \to \Delta_k$

Translation-invariant: $E^h(X, i, j) \to K^h$ shared across (i, j)

Input-invariant: $E^h(X, i, j) \to K^h$ shared across X

Can Attention Behave Like Convolution?

Local: $\Delta_m \to \Delta_k$

Translation-invariant: $E^h(X, i, j) \to K^h$ shared across (i, j)

Input-invariant: $E^h(X, i, j) \to K^h$ shared across X

The connection

Under these assumptions, the dynamic attention weights can be replaced by a fixed local kernel:

$$\mathrm{Att}^h(X)_{i,j} \approx \sum_{(r,s)\in\Delta_k} K^h_{r,s} V^h_{i+r,j+s}$$

Can Attention Behave Like Convolution?

Local: $\Delta_m \to \Delta_k$
Translation-invariant: $E^h(X, i, j) \to K^h$ shared across (i, j)
Input-invariant: $E^h(X, i, j) \to K^h$ shared across X

The connection

Under these assumptions, the dynamic attention weights can be replaced by a fixed local kernel:

$$\mathrm{Att}^h(X)_{i,j} \approx \sum_{(r,s)\in\Delta_k} K^h_{r,s} V^h_{i+r,j+s}$$

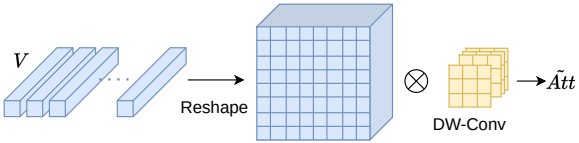
Direct implementation: full convolution

Fold the spatial weights into the pretrained value projection:

$$W^{Vh}_{r,s} = K^h_{r,s}\, W^V_h, \qquad W^{Vh} \in \mathbb{R}^{k\times k\times d_i\times d_h},$$
$$\widetilde{\mathrm{Att}}^h_C(X) = \mathrm{Conv}(X, W^{Vh}).$$

This is a standard convolution from d_i input channels to d_h head channels.

From Attention Approximation to an Efficient Layer



Pipeline: $X \to W_h^V \to V^h \to$ per-channel 3×3 filters $\to \widetilde{\mathrm{Att}}^h$.

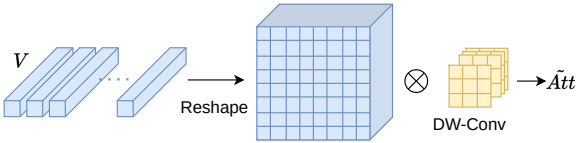
The pretrained value projection mixes channels:

$$V^h = XW_h^V$$

Depthwise convolution replaces only spatial aggregation:

$$\boxed{\widetilde{\mathrm{Att}}^h_{DW}(X) = \mathrm{Conv}_{DW}(V^h, \vec{K}^h)}$$

From Attention Approximation to an Efficient Layer



Pipeline: $X \to W_h^V \to V^h \to$ per-channel 3×3 filters $\to \widetilde{\mathrm{Att}}^h$.

The pretrained value projection mixes channels:

$$V^h = XW_h^V$$

Depthwise convolution replaces only spatial aggregation:

$$\boxed{\widetilde{\mathrm{Att}}^h_{DW}(X) = \mathrm{Conv}_{DW}(V^h, \vec{K}^h)}$$

Complexity

Full convolution: $\mathcal{O}(k^2 d_i d_h)$

Value projection + DWConv: $\mathcal{O}(d_i d_h + k^2 d_h)$

Only the aggregation operator changes; the pretrained value features and token grid are preserved.

Optional Head Ensembling

Ensembled formulation

Learn softmax weights $\sigma(\gamma_h)$ and combine the pretrained projections:

$$W^{Ve} = \sum_h \sigma(\gamma_h) W_h^V \qquad W^{Oe} = \sum_h \sigma(\gamma_h) W_h^O$$

One effective head then replaces the whole selected block:

$$\widetilde{\mathrm{MhSA}}^{e}_{DW}(X) = \mathrm{Conv}_{DW}(XW^{Ve}, \vec{K}^e)W^{Oe}$$

Optional Head Ensembling

Ensembled formulation

Learn softmax weights $\sigma(\gamma_h)$ and combine the pretrained projections:

$$W^{Ve} = \sum_h \sigma(\gamma_h) W_h^V \qquad W^{Oe} = \sum_h \sigma(\gamma_h) W_h^O$$

One effective head then replaces the whole selected block:

$$\widetilde{\mathrm{MhSA}}^{e}_{DW}(X) = \mathrm{Conv}_{DW}(XW^{Ve}, \vec{K}^e)W^{Oe}$$

Trade-off evaluated in the experiments

Head ensembling removes more computation and is faster, but collapsing multiple heads into one generally causes a larger accuracy drop.

Identifying Convolution-Like Attention

A good replacement candidate should be **local**, **translation-invariant**, and **input-invariant**.

Proposed Heuristic

Use the sum of the pointwise standard deviation of the attention weights as an empirical proxy for identifying convolution-like heads.

Identifying Convolution-Like Attention

A good replacement candidate should be **local**, **translation-invariant**, and **input-invariant**.

Proposed Heuristic

Use the sum of the pointwise standard deviation of the attention weights as an empirical proxy for identifying convolution-like heads.

Head score

Compute pointwise variation over inputs, then sum all entries:

$$\sigma_{E^h} = \mathrm{Std}_X[E^h(X)], \qquad \boxed{\Sigma_h = \sum \sigma_{E^h}}$$

Identifying Convolution-Like Attention

A good replacement candidate should be **local**, **translation-invariant**, and **input-invariant**.

Proposed Heuristic

Use the sum of the pointwise standard deviation of the attention weights as an empirical proxy for identifying convolution-like heads.

Head score

Compute pointwise variation over inputs, then sum all entries:

$$\sigma_{E^h} = \mathrm{Std}_X[E^h(X)], \qquad \boxed{\Sigma_h = \sum \sigma_{E^h}}$$

Block score

Average the scores of the n_h heads in block b:

$$\boxed{\Sigma_b = \frac{1}{n_h} \sum_{h \in \mathcal{H}_b} \Sigma_h}$$

Applying the Selection Criterion

Fix the replacement budget first

Choose $p_h = |\mathcal{S}_h|$ individual heads or $p_b = |\mathcal{S}_b|$ complete blocks to replace.

Applying the Selection Criterion

Fix the replacement budget first

Choose $p_h = |\mathcal{S}_h|$ individual heads or $p_b = |\mathcal{S}_b|$ complete blocks to replace.

Scattered: select heads

Candidates: every head in the network.
Ranking score: Σ_h.
Budget: $|\mathcal{S}_h| = p_h$ heads.

$$\boxed{\mathcal{S}_h^{[p_h]} = \underset{|\mathcal{S}|=p_h}{\arg\min} \sum_{h \in \mathcal{S}} \Sigma_h}$$

Replace only the heads in $\mathcal{S}_h^{[p_h]}$; a block may contain both MhSA and DW heads.

Applying the Selection Criterion

Fix the replacement budget first

Choose $p_h = |\mathcal{S}_h|$ individual heads or $p_b = |\mathcal{S}_b|$ complete blocks to replace.

Scattered: select heads

Candidates: every head in the network.
Ranking score: Σ_h.
Budget: $|\mathcal{S}_h| = p_h$ heads.

$$\mathcal{S}_h^{[p_h]} = \arg\min_{|\mathcal{S}|=p_h} \sum_{h \in \mathcal{S}} \Sigma_h$$

Replace only the heads in $\mathcal{S}_h^{[p_h]}$; a block may contain both MhSA and DW heads.

Blockwise: select blocks

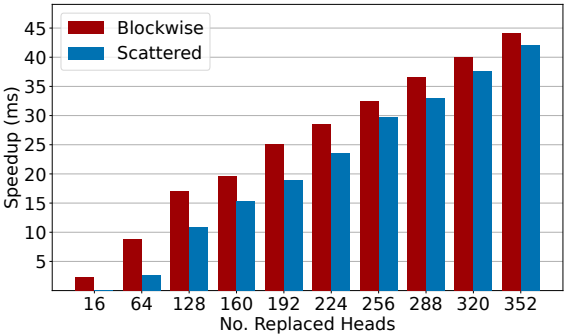
Candidates: every transformer block.
Ranking score: Σ_b.
Budget: $|\mathcal{S}_b| = p_b$ blocks.

$$\mathcal{S}_b^{[p_b]} = \arg\min_{|\mathcal{S}|=p_b} \sum_{b \in \mathcal{S}} \Sigma_b$$

Replace all heads in blocks $\mathcal{S}_b^{[p_b]}$: p_b blocks correspond to $p_b n_h$ heads.

Blockwise Selection Matters on Hardware

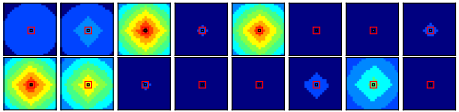
- **Scattered:** arbitrary heads across blocks
- **Blockwise:** all heads in selected blocks



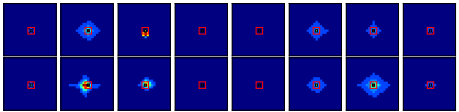
Result

For the same number of replaced heads, blockwise replacement is consistently faster. Scattered execution incurs extra kernel launches and memory overhead, so its speedup scales non-linearly.

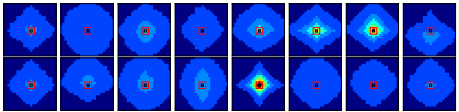
What Does the Standard-Deviation Score Capture?



(a) **Block 0** Mixed low- and high-σ heads High Σ_b



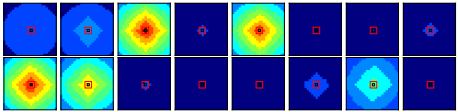
(b) **Block 4** Mostly low-σ heads Low Σ_b



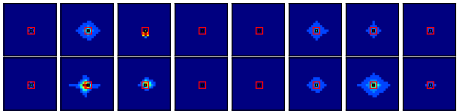
(c) **Block 22** Mostly high-σ heads High Σ_b

Each map aligns the query at the center and averages σ_h over spatial locations; the red square marks the 3×3 convolutional receptive field.

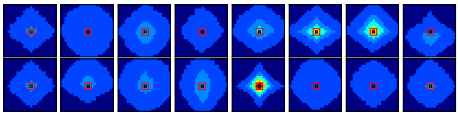
What Does the Standard-Deviation Score Capture?



(a) **Block 0** Mixed low- and high-σ heads High Σ_b



(b) **Block 4** Mostly low-σ heads Low Σ_b



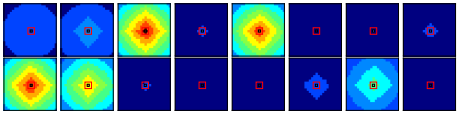
(c) **Block 22** Mostly high-σ heads High Σ_b

Each map aligns the query at the center and averages σ_h over spatial locations; the red square marks the 3×3 convolutional receptive field.

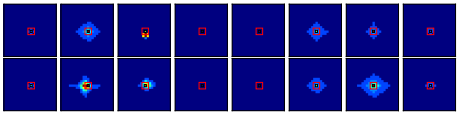
Interpretation

- If $\Sigma_h = 0$, then $E^h(X) = E^h(Y)$ for every X, Y: the head is exactly input-invariant.
- Positional encoding induces the spatial structure and locality bias needed for a convolution-like approximation.

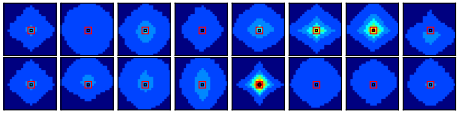
What Does the Standard-Deviation Score Capture?



(a) **Block 0** Mixed low- and high-σ heads High Σ_b



(b) **Block 4** Mostly low-σ heads Low Σ_b



(c) **Block 22** Mostly high-σ heads High Σ_b

Each map aligns the query at the center and averages σ_h over spatial locations; the red square marks the 3×3 convolutional receptive field.

Interpretation

- If $\Sigma_h = 0$, then $E^h(X) = E^h(Y)$ for every X, Y: the head is exactly input-invariant.
- Positional encoding induces the spatial structure and locality bias needed for a convolution-like approximation.

Limitation

- This spatial behavior is difficult to guarantee formally because it emerges during pretraining rather than from the architecture alone.

Alternative Selection: DSP

As a learned alternative to the Σ heuristic, we adapt Differentiable Subset Pruning (DSP) to choose which candidates are converted to convolution. (li2021differentiable)

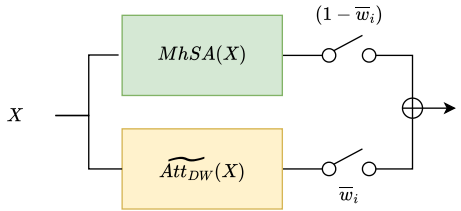
Stochastic gating with a fixed budget

Trainable scores w_i control the operator used at candidate i. For a fixed budget $p = |\mathcal{S}|$, hard top-k selection gives

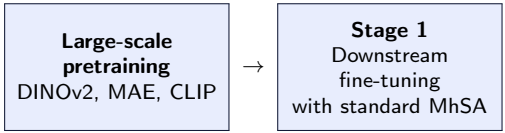
$$\bar{w}_i = \mathrm{topk}(w, p)_i \in \{0, 1\}$$

During training, a differentiable Gumbel top-k relaxation $\widetilde{w}_i \in [0, 1]$ switches between the two paths:

$$\bar{F}_i(X) = (1 - \widetilde{w}_i)\,\mathrm{MhSA}_i(X) + \widetilde{w}_i\,\widetilde{\mathrm{MhSA}}_i(X)$$



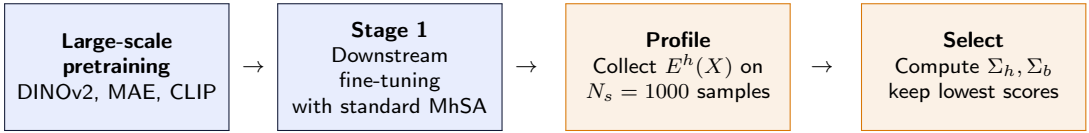
Complete Training and Conversion Pipeline



What is preserved?

Pretrained value and output projections remain; only selected spatial aggregation operators are replaced and adapted.

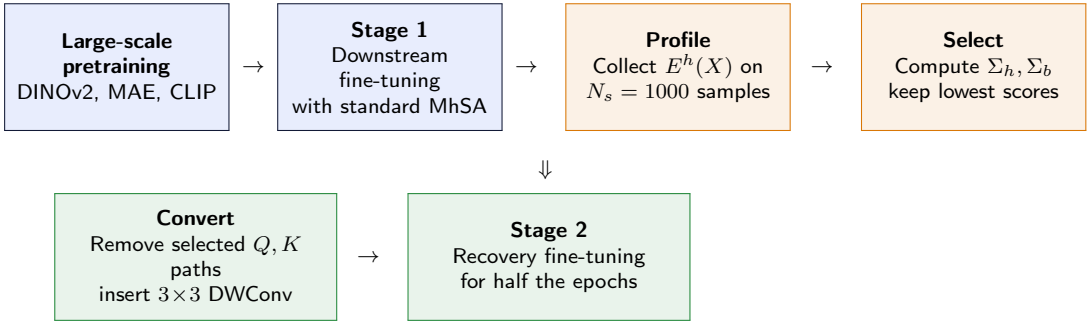
Complete Training and Conversion Pipeline



What is preserved?

Pretrained value and output projections remain; only selected spatial aggregation operators are replaced and adapted.

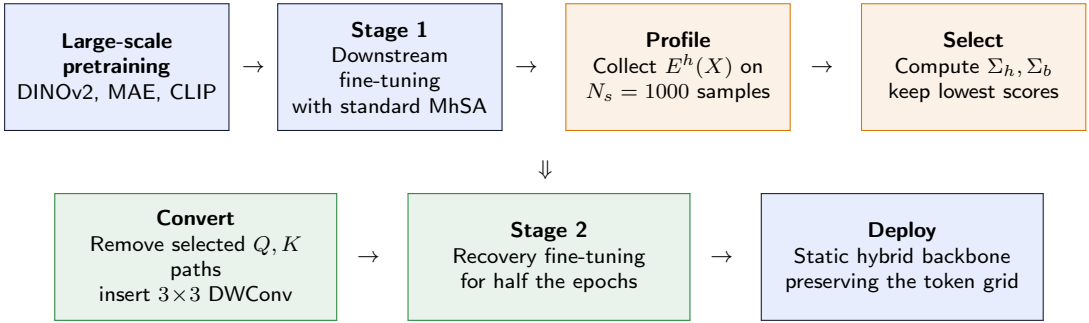
Complete Training and Conversion Pipeline



What is preserved?

Pretrained value and output projections remain; only selected spatial aggregation operators are replaced and adapted.

Complete Training and Conversion Pipeline



What is preserved?

Pretrained value and output projections remain; only selected spatial aggregation operators are replaced and adapted.

Overview

How Do We Evaluate the Drop-in Replacement?

Foundation models

DINOv2 ViT-L
Primary analysis

MAE **CLIP**
Generalization across pretraining objectives

How Do We Evaluate the Drop-in Replacement?

Foundation models	Downstream tasks
DINOv2 ViT-L Primary analysis	**COCO segmentation** Dense prediction · mIoU
MAE **CLIP** Generalization across pretraining objectives	**ImageNet-1K** Classification · Top-1

How Do We Evaluate the Drop-in Replacement?

Foundation models

DINOv2 ViT-L
Primary analysis

MAE **CLIP**
Generalization across pretraining objectives

Downstream tasks

COCO segmentation
Dense prediction · mIoU

ImageNet-1K
Classification · Top-1

Deployment benchmark

PyTorch → ONNX → TensorRT

NVIDIA Jetson Orin Nano

Input:	336×336
Batch size:	1
Latency:	backbone only
DW kernels:	fixed 3×3

How Do We Evaluate the Drop-in Replacement?

Foundation models

DINOv2 ViT-L
Primary analysis

MAE **CLIP**
Generalization across pretraining objectives

Downstream tasks

COCO segmentation
Dense prediction · mIoU

ImageNet-1K
Classification · Top-1

Deployment benchmark

PyTorch → ONNX → TensorRT
NVIDIA Jetson Orin Nano

Input:	336×336
Batch size:	1
Latency:	backbone only
DW kernels:	fixed 3×3

Evaluation target

Downstream accuracy *and* compiled edge-device latency—not FLOPs alone.

Control Experiment: Profile a Single Attention Block

Profiling setup

We profile the latency and memory footprint of a **single DINOv2 ViT-L self-attention block** with 16 heads and 576 input tokens.

Single ViT-L block	**GFLOPs**	**Params (M)**	**Latency (ms)**	**Memory (MB)**
MhSA	6.19	4.20	3.20	47.2
Full convolution	12.08	2.10	3.71	4.5
Depthwise (ours)	2.43	2.11	1.26	6.75
Full convolution + ensemble	0.75	2.10	0.641	4.5
Depthwise + ensemble	0.15	2.10	0.215	0.288

Control Experiment: Profile a Single Attention Block

Profiling setup

We profile the latency and memory footprint of a **single DINOv2 ViT-L self-attention block** with 16 heads and 576 input tokens.

Single ViT-L block	GFLOPs	Params (M)	Latency (ms)	Memory (MB)
MhSA	**6.19**	**4.20**	**3.20**	**47.2**
Full convolution	12.08	2.10	3.71	4.5
Depthwise (ours)	**2.43**	**2.11**	**1.26**	**6.75**
Full convolution + ensemble	0.75	2.10	0.641	4.5
Depthwise + ensemble	0.15	2.10	0.215	0.288

Key results

- The unensembled depthwise layer is about $2.5\times$ faster than MhSA.

Control Experiment: Profile a Single Attention Block

Profiling setup

We profile the latency and memory footprint of a **single DINOv2 ViT-L self-attention block** with 16 heads and 576 input tokens.

Single ViT-L block	GFLOPs	Params (M)	Latency (ms)	Memory (MB)
MhSA	**6.19**	**4.20**	**3.20**	**47.2**
Full convolution	**12.08**	**2.10**	**3.71**	**4.5**
Depthwise (ours)	2.43	2.11	1.26	6.75
Full convolution + ensemble	0.75	2.10	0.641	4.5
Depthwise + ensemble	0.15	2.10	0.215	0.288

Key results

- The unensembled depthwise layer is about $2.5\times$ faster than MhSA.
- The full-convolution implementation is slower than the MhSA baseline.

Control Experiment: Profile a Single Attention Block

Profiling setup

We profile the latency and memory footprint of a **single DINOv2 ViT-L self-attention block** with 16 heads and 576 input tokens.

Single ViT-L block	GFLOPs	Params (M)	Latency (ms)	Memory (MB)
MhSA	**6.19**	**4.20**	**3.20**	**47.2**
Full convolution	12.08	2.10	3.71	4.5
Depthwise (ours)	2.43	2.11	1.26	6.75
Full convolution + ensemble	0.75	2.10	0.641	4.5
Depthwise + ensemble	**0.15**	**2.10**	**0.215**	**0.288**

Key results

- The unensembled depthwise layer is about $2.5\times$ faster than MhSA.
- The full-convolution implementation is slower than the MhSA baseline.
- Head ensembling combined with the depthwise decomposition achieves $\approx 15\times$ speedup over MhSA.

Accuracy–Speed Trade-off: COCO Segmentation

Experiment.

Starting from the fine-tuned DINOv2 ViT-L, we replace the blocks with the lowest Σ_b and vary both the *budget* $|\mathcal{S}|$ and the use of *head ensembling*.

Attention	$\lvert\mathcal{S}\rvert$	**mIoU**	**Δ**	**ms**	**Speedup**
MhSA baseline	–	66.03	–	161.4	–
Depthwise	12/24	65.95	-0.08	133.6	17.21%
Ens. + Depthwise	10/24	64.81	-1.22	131.1	18.79%
Depthwise	16/24	64.64	-1.39	126.1	21.85%
Ens. + Depthwise	12/24	63.92	-2.11	124.5	22.87%

Accuracy–Speed Trade-off: COCO Segmentation

Experiment.

Starting from the fine-tuned DINOv2 ViT-L, we replace the blocks with the lowest Σ_b and vary both the *budget* $|\mathcal{S}|$ and the use of *head ensembling*.

Attention	$\lvert\mathcal{S}\rvert$	**mIoU**	**Δ**	**ms**	**Speedup**
MhSA baseline	–	**66.03**	–	**161.4**	–
Depthwise	**12/24**	**65.95**	-0.08	**133.6**	17.21%
Ens. + Depthwise	10/24	64.81	-1.22	131.1	18.79%
Depthwise	16/24	64.64	-1.39	126.1	21.85%
Ens. + Depthwise	12/24	63.92	-2.11	124.5	22.87%

Key findings

- Replacing half of the blocks saves **27.8 ms** with only a **0.08 mIoU** loss.

Accuracy–Speed Trade-off: COCO Segmentation

Experiment.

Starting from the fine-tuned DINOv2 ViT-L, we replace the blocks with the lowest Σ_b and vary both the *budget* $|\mathcal{S}|$ and the use of *head ensembling*.

Attention	$\lvert\mathcal{S}\rvert$	**mIoU**	**Δ**	**ms**	**Speedup**
MhSA baseline	–	**66.03**	–	**161.4**	–
Depthwise	12/24	65.95	−0.08	133.6	17.21%
Ens. + Depthwise	10/24	64.81	−1.22	131.1	18.79%
Depthwise	**16/24**	**64.64**	−1.39	**126.1**	21.85%
Ens. + Depthwise	12/24	63.92	−2.11	124.5	22.87%

Key findings

- Replacing half of the blocks saves **27.8 ms** with only a **0.08 mIoU** loss.
- At $|\mathcal{S}| = 16$, speedup exceeds **20%**, at the cost of a larger 1.39 mIoU drop.

Accuracy–Speed Trade-off: COCO Segmentation

Experiment.

Starting from the fine-tuned DINOv2 ViT-L, we replace the blocks with the lowest Σ_b and vary both the *budget* $|\mathcal{S}|$ and the use of *head ensembling*.

Attention	$\lvert\mathcal{S}\rvert$	mIoU	Δ	ms	Speedup
MhSA baseline	–	**66.03**	–	**161.4**	–
Depthwise	12/24	65.95	−0.08	133.6	17.21%
Ens. + Depthwise	**10/24**	**64.81**	−1.22	**131.1**	18.79%
Depthwise	16/24	64.64	−1.39	126.1	21.85%
Ens. + Depthwise	**12/24**	**63.92**	−2.11	**124.5**	22.87%

Key findings

- Replacing half of the blocks saves **27.8 ms** with only a **0.08 mIoU** loss.
- At $|\mathcal{S}| = 16$, speedup exceeds **20%**, at the cost of a larger 1.39 mIoU drop.
- Head ensembling is slightly faster but consistently sacrifices more segmentation accuracy.

Accuracy–Speed Trade-off: ImageNet Classification

Cross-task test

We repeat the conversion protocol with DINOv2 ViT-L on ImageNet-1K. Because conversion requires removing the class token, the classifier uses the mean of the output patch tokens.

Attention	$\lvert\mathcal{S}\rvert$	**Top-1**	**Δ**	**ms**	**Speedup**
MhSA baseline	–	86.22	–	161.4	–
Depthwise	12/24	85.45	-0.77	133.6	17.21%
Ens. + Depthwise	10/24	84.96	-1.26	131.1	18.79%
Depthwise	16/24	84.88	-1.34	126.1	21.85%
Ens. + Depthwise	12/24	84.65	-1.57	124.5	22.87%

Accuracy–Speed Trade-off: ImageNet Classification

Cross-task test

We repeat the conversion protocol with DINOv2 ViT-L on ImageNet-1K. Because conversion requires removing the class token, the classifier uses the mean of the output patch tokens.

Attention	$\lvert\mathcal{S}\rvert$	Top-1	Δ	ms	Speedup
MhSA baseline	–	**86.22**	–	**161.4**	–
Depthwise	**12/24**	**85.45**	−0.77	**133.6**	17.21%
Ens. + Depthwise	10/24	84.96	−1.26	131.1	18.79%
Depthwise	16/24	84.88	−1.34	126.1	21.85%
Ens. + Depthwise	12/24	84.65	−1.57	124.5	22.87%

Key findings

- Half-block replacement retains **85.45% top-1** while keeping the same **17.21% backbone speedup**.

Accuracy–Speed Trade-off: ImageNet Classification

Cross-task test

We repeat the conversion protocol with DINOv2 ViT-L on ImageNet-1K. Because conversion requires removing the class token, the classifier uses the mean of the output patch tokens.

Attention	$\lvert\mathcal{S}\rvert$	Top-1	Δ	ms	Speedup
MhSA baseline	–	**86.22**	–	**161.4**	–
Depthwise	12/24	85.45	−0.77	133.6	17.21%
Ens. + Depthwise	10/24	84.96	−1.26	131.1	18.79%
Depthwise	**16/24**	**84.88**	−1.34	**126.1**	21.85%
Ens. + Depthwise	12/24	84.65	−1.57	124.5	22.87%

Key findings

- Half-block replacement retains **85.45% top-1** while keeping the same **17.21% backbone speedup**.
- Replacing 16/24 blocks raises speedup to **21.85%**, with a 1.34-point accuracy loss.

Accuracy–Speed Trade-off: ImageNet Classification

Cross-task test

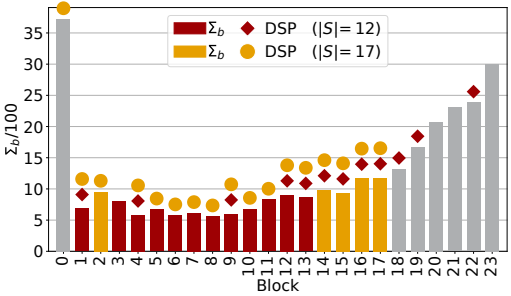
We repeat the conversion protocol with DINOv2 ViT-L on ImageNet-1K. Because conversion requires removing the class token, the classifier uses the mean of the output patch tokens.

Attention	$\lvert\mathcal{S}\rvert$	Top-1	Δ	ms	Speedup
MhSA baseline	–	**86.22**	–	**161.4**	–
Depthwise	12/24	85.45	−0.77	133.6	17.21%
Ens. + Depthwise	**10/24**	**84.96**	−1.26	**131.1**	18.79%
Depthwise	16/24	84.88	−1.34	126.1	21.85%
Ens. + Depthwise	**12/24**	**84.65**	−1.57	**124.5**	22.87%

Key findings

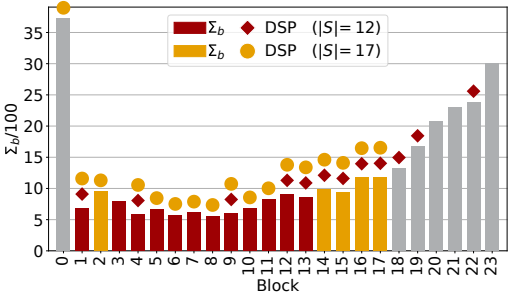
- Half-block replacement retains **85.45% top-1** while keeping the same **17.21% backbone speedup**.
- Replacing 16/24 blocks raises speedup to **21.85%**, with a 1.34-point accuracy loss.
- As on COCO, head ensembling produces the fastest configurations, but the unensembled formulation preserves accuracy better.

Selection Heuristics: Validation and Comparisons



Criterion	$\lvert\mathcal{S}\rvert$	**mIoU**
Lowest Σ_b	12/24	65.95
DSP end-to-end	12/24	64.15
DSP two-stage	12/24	64.00
Highest Σ_b	12/24	61.46
One-stage, low Σ_b	12/24	65.63

Selection Heuristics: Validation and Comparisons

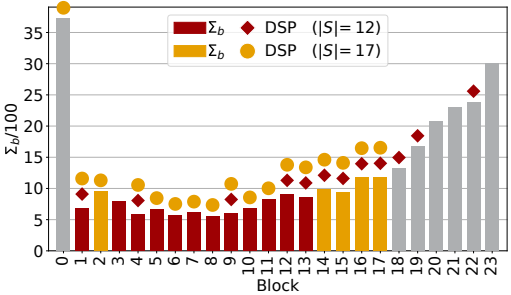


Criterion	$\lvert\mathcal{S}\rvert$	mIoU
Lowest Σ_b	**12/24**	**65.95**
DSP end-to-end	**12/24**	**64.15**
DSP two-stage	**12/24**	**64.00**
Highest Σ_b	12/24	61.46
One-stage, low Σ_b	12/24	65.63

Takeaway

- Low-variance selection beats learned DSP gates at the 12-block operating point.

Selection Heuristics: Validation and Comparisons

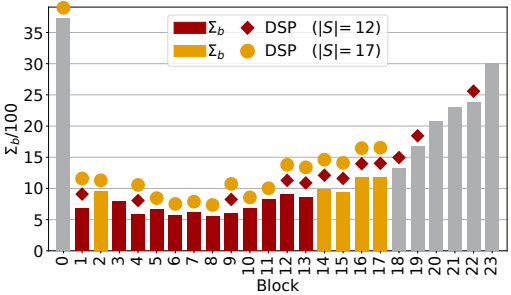


Criterion	$\lvert\mathcal{S}\rvert$	mIoU
Lowest Σ_b	12/24	65.95
DSP end-to-end	12/24	64.15
DSP two-stage	12/24	64.00
Highest Σ_b	**12/24**	**61.46**
One-stage, low Σ_b	12/24	65.63

Takeaway

- Low-variance selection beats learned DSP gates at the 12-block operating point.
- Selecting high-variance blocks severely degrades performance.

Selection Heuristics: Validation and Comparisons

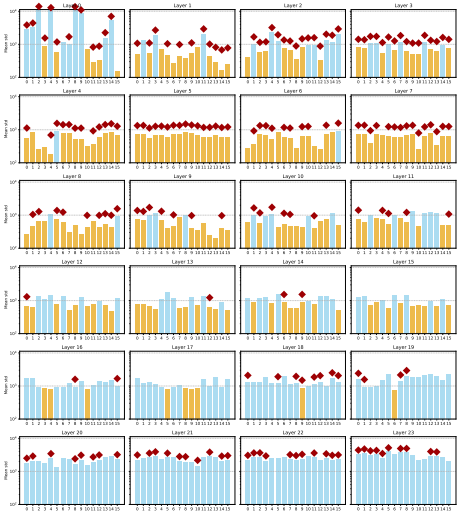


Criterion	$\lvert\mathcal{S}\rvert$	mIoU
Lowest Σ_b	12/24	65.95
DSP end-to-end	12/24	64.15
DSP two-stage	12/24	64.00
Highest Σ_b	12/24	61.46
One-stage, low Σ_b	**12/24**	**65.63**

Takeaway

- Low-variance selection beats learned DSP gates at the 12-block operating point.
- Selecting high-variance blocks severely degrades performance.
- One-stage fine-tuning reaches 65.63 mIoU; selection still uses post-fine-tuning statistics.

Extra: Distribution of Selected Heads (Scattered)



- Bars show Σ_h for every head after COCO fine-tuning.
- Orange: the 192 lowest-Σ_h heads selected by our scattered criterion.
- Red diamonds: heads selected by learned DSP gates.

Unexpected finding

The two methods achieve similar scattered-selection performance, yet choose markedly different head distributions.

Hypothesis: DSP-selected heads adapt during fine-tuning to satisfy the convolutional constraint.

Extra: Fine-tuning Only the Convolutional Heads

In the second training stage, we freeze the first-stage backbone and update only the affected heads.

- Same COCO segmentation setup
- DINOv2 ViT-L, 12/24 depthwise blocks
- Only 8.9% of the parameters are updated

Second stage	Trainable	mIoU
Full model	284.4M	**65.95**
Conv. heads only	**25.3M**	65.12

Promising result

Training only 8.9% of the parameters retains all but **0.83 mIoU** of full second-stage fine-tuning.

This preliminary result suggests that adaptation can be made substantially cheaper in memory and optimization cost.

Generalizes Beyond DINOv2

Evaluation

We validate the proposed approach on other VFMs (MAE and CLIP): the same ViT architecture, but different pretraining objectives.

Pretraining	**Backbone**	$\lvert\mathcal{S}\rvert$	**Baseline (mIoU)**	**Ours (mIoU)**	**Latency gain**
MAE	ViT-B/16	6/12	58.84	57.91	18.4%
MAE	ViT-L/16	12/24	60.22	59.81	16.5%
CLIP	ViT-B/16	6/12	61.52	59.06	18.3%
CLIP	ViT-L/14	12/24	64.65	62.17	16.6%

Key finding

Similar trends are observed for DINOv2, MAE, and CLIP, suggesting that convolution-like behavior is not specific to DINO-style pretraining.

Overview

Conclusions

- Some pretrained attention heads in Foundation ViTs behave like static local filters and can be approximated efficiently.
- Pointwise value projection plus depthwise convolution is a drop-in replacement that preserves the token grid and pretrained weights.
- Selecting low-variance blocks yields a strong operating point: **17.2% speedup with only a 0.08 mIoU loss** on COCO with DINOv2 ViT-L.
- The method transfers across classification, segmentation, model sizes, and pretraining paradigms.

Perspective

Drop-in depthwise convolution is complementary to pruning and affine methods, opening the way to combined acceleration strategies.

Reference implementation: `https://github.com/cscribano/DWConv_VFM`

Acknowledgements

This research was partially funded by the **dAIedge** project (HORIZON-CL4-2022-HUMAN-02-02, Grant Agreement Number: 101120726) and the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

Thank you for your attention!