LOCAL-WINDOW CONTRASTIVE FORWARD-FORWARD LEARNING WITH MASKED RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Training Vision Transformers (ViTs) with end-to-end backpropagation requires storing intermediate activations across the network, making training challenging on memory-constrained hardware. Forward–Forward learning offers an alternative by optimizing local objectives without propagating gradients through the full network. Its contrastive variant (CFF) encourages discriminative visual representations, but relies on pooled features that do not directly constrain patch-level visual structure. To address this limitation, we propose a local-window training framework that introduces masked reconstruction into contrastive Forward–Forward learning, providing patch-level supervision while preserving the memory advantages of local optimization. Our framework progressively trains windows of one or two consecutive encoder blocks, allowing contrastive discrimination and masked reconstruction to jointly shape the representations learned at each depth. We further introduce a depth-dependent reconstruction strategy that decouples encoder input masking from reconstruction-loss masking, enabling the reconstruction signal to be adjusted across local windows without changing the visible-token distribution. Decoder checkpoint transfer and sequential execution of the contrastive and reconstruction branches further support memory-efficient progressive training. Experiments on CIFAR-10, ImageNet-1K, and the STB hand-pose benchmark demonstrate consistent improvements over CFF-only training, with pair-wise training achieving 64.75% top-1 accuracy on ImageNet-1K using a compact six-block ViT. Profiling on the NVIDIA Jetson Orin Nano shows up to a 4.3$\times$ reduction in peak GPU memory relative to full backpropagation, with lower measured per-batch latency and energy than activation-checkpointed backpropagation at comparable memory footprints.

1 INTRODUCTION

Vision Transformers (ViTs) have become a widely adopted backbone for visual representation learning, achieving strong performance across image classification and structured visual prediction tasks (Dosovitskiy et al., 2021). Their ability to model interactions among image patches makes them attractive for learning transferable visual representations, including in edge-perception applications that require adaptation to new tasks or environments. However, the same architecture presents substantial challenges for on-device training. Unlike hierarchical convolutional networks, which typically reduce spatial resolution as features progress through the network, standard ViTs maintain a largely fixed number of patch tokens and embedding dimensions across encoder blocks. Each block also introduces intermediate tensors associated with query, key, and value projections, multi-head self-attention, and feed-forward layers. Consequently, end-to-end backpropagation requires retaining substantial intermediate activations for gradient computation, in addition to parameter gradients and optimizer states. These costs grow with network depth and the number of image patches, making even compact ViTs difficult to train on memory-constrained edge devices. Enabling efficient on-device ViT training is therefore important for bringing representation learning and task adaptation closer to where visual data are acquired.

Existing training-efficiency techniques address different parts of this problem but do not fully resolve the memory–computation trade-off. Activation checkpointing (Chen et al., 2016) and reversible architectures (Gomez et al., 2017) reduce activation storage through recomputation or architectural changes, while preserving dependencies between the final loss and earlier network blocks. Parameter-efficient fine-tuning methods, including LoRA (Hu et al., 2021), have also been applied to pretrained ViTs (Bafghi et al., 2024). Although these methods reduce the number of trainable parameters and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

associated optimizer states, gradients may still need to propagate through multiple frozen blocks to update the selected parameters. Thus, reducing trainable parameter count does not necessarily eliminate the activation-memory burden of full-depth gradient propagation.

Local learning offers a different approach by restricting optimization to a small portion of the network at a time. Greedy layer-wise training (Belilovsky et al., 2019) and Forward–Forward learning (Hinton, 2022) replace a single end-to-end training objective with objectives applied to individual layers or blocks. Other approaches, including feedback alignment, direct feedback alignment, direct random target projection, and Hebbian learning, explore alternative feedback or parameter-update mechanisms (Lillicrap et al., 2016; Nøkland, 2016; Frenkel et al., 2021; Journé et al., 2023). These methods motivate memory-efficient training, but restricting gradient propagation also limits the information available to guide the representations learned at each depth. The challenge is therefore not only to localize training, but also to ensure that local objectives produce representations suitable for demanding visual tasks. Contrastive Forward–Forward (CFF) extends local learning to ViTs by using a contrastive objective to encourage discriminative representations (Aghagolzadeh & Ezoji, 2025). However, its supervision is applied to pooled features, without directly constraining the patch-level information retained by the encoder. This limitation is particularly relevant to visual recognition and structured prediction, where local appearance and spatial relationships can be important alongside global discrimination. Masked autoencoding (MAE) (He et al., 2022) offers a complementary learning signal: reconstructing hidden image patches encourages the encoder to retain visual information that supports prediction of missing content. Yet conventional MAE optimizes the encoder end-to-end, bringing back the full-depth training dependencies that local learning seeks to avoid.

This motivates the central question of our work: *How can local ViT training learn discriminative and structurally informative representations without sacrificing its peak-memory advantage?*

To address this challenge, we propose a **local-window contrastive learning framework with masked reconstruction** that improves visual representations without sacrificing the memory advantages of local training. By providing patch-level reconstruction targets alongside contrastive supervision, our framework encourages each locally trained encoder window to learn both discriminative and structurally informative representations. We perform training through progressive local pretraining followed by supervised head training or window-wise fine-tuning, avoiding simultaneous optimization of the full encoder.

Figure 1 demonstrates the memory benefits on NVIDIA Jetson Orin Nano hardware. Layer-wise and pair-wise training reduce peak GPU memory by up to 4.3× and 2.6×, compared with full backpropagation. At batch size 64, full backpropagation encounters out-of-memory (OOM), whereas both configurations remain feasible. Masked reconstruction introduces only a small additional memory overhead over CFF-only training, showing that stronger local supervision can be achieved within a constrained memory budget.

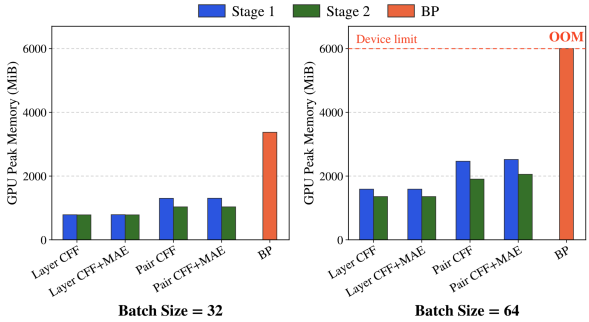


Figure 1: Peak GPU training memory on NVIDIA Jetson Orin Nano at batch sizes 32 and 64. Full backpropagation (BP) encounters out-of-memory (OOM) at batch size 64, whereas our local training remain feasible, with only a small memory overhead for masked reconstruction over CFF-only training.

Our Contributions. We introduce a progressive ViT training framework that confines pretraining gradients to local windows of one or two consecutive encoder blocks, using supervised contrastive learning and self-supervised masked reconstruction to improve representation quality. To support efficient local reconstruction, we develop depth-dependent reconstruction-loss masking, decoder checkpoint transfer across windows, and sequential backward execution to limit peak activation memory. The framework further supports supervised head training and reverse-order window-wise fine-tuning with checkpointed gradient propagation through frozen blocks. Experiments on CIFAR-10, ImageNet-1K, and STB hand-pose estimation demonstrate consistent improvements over CFF-only training, including 64.75% top-1 accuracy on ImageNet-1K with a compact six-block ViT. Our NVIDIA Jetson profiling further demonstrates favorable peak-memory, latency, and energy trade-offs against full and activation-checkpointed backpropagation.

2 Related Works

Local Learning Methods. Local learning restricts optimization to individual layers or blocks, avoiding full-depth gradient propagation. Greedy layer-wise training demonstrated the scalability of block-local optimization to ImageNet (Belilovsky et al., 2019), followed by decoupled greedy learning (Belilovsky et al., 2020), local error signals (Nøkland & Eidnes, 2019), and representation-learning objectives such as Greedy InfoMax (Löwe et al., 2019), InfoPro (Wang et al., 2021), LoCo (Xiong et al., 2020), and blockwise self-supervised learning (Siddiqui et al., 2024). Forward–Forward (FF) (Hinton, 2022) instead trains layers using local goodness objectives, with extensions exploring deeper networks (Sun et al., 2025), layer collaboration (Lorberbom et al., 2024), and self-contrastive learning (Chen et al., 2025). Contrastive Forward–Forward (CFF) adapts local contrastive learning to ViTs (Aghagolzadeh & Ezoji, 2025), providing the discriminative objective underlying our framework. Alternative approaches replace or approximate conventional backpropagation through feedback alignment, direct feedback alignment, random target projection, and Hebbian learning (Lillicrap et al., 2016; Nøkland, 2016; Frenkel et al., 2021; Journé et al., 2023). Closely related to our work, BIM (Luo et al., 2023) investigates block-wise masked image modeling, Pathak et al. (2022) incorporate reconstruction of module-input features into local contrastive learning, and LocalMIM (Wang et al., 2023a) introduces multi-scale local reconstruction, including a gradient-isolated variant. Our work specifically targets progressive CFF training of ViTs, using masked-patch reconstruction with independent input and loss masks, decoder transfer, and memory-efficient sequential branch execution.

Masked Autoencoding and Contrastive Masked Modeling. MAE (He et al., 2022) learns visual representations by reconstructing masked image patches, while SimMIM (Xie et al., 2022) explores a simpler prediction head. Subsequent methods introduce contrastive objectives alongside masked modeling: CAN (Mishra et al., 2022) incorporates noise prediction, CMAE (Huang et al., 2022) uses a momentum encoder, and ViC-MAE (Hernandez et al., 2024) extends contrastive masked autoencoding to images and videos. Unlike our framework, these methods optimize the encoder end-to-end rather than through progressive local windows. MAE-Lite (Wang et al., 2023b) also explores attention distillation from a frozen MAE teacher; we employ a related signal only at the final active window for ImageNet-scale training, while retaining local optimization throughout pretraining.

3 Method

End-to-end pretraining of vision encoders couples all blocks into a single backward graph, causing activation and optimizer-state memory to scale with network depth. Our goal is to retain the complementary benefits of contrastive learning and masked reconstruction while removing this global coupling. We therefore propose *CFF+MAE*, a local-window training framework that decomposes pretraining into a sequence of optimization problems, each restricted to only a few consecutive encoder blocks.

3.1 Architecture Overview

Figure 2 summarizes the CFF+MAE training pipeline; Figure 3 and 4 in appendix show the layer-wise and pair-wise Stage-1 units. The method first pretrains a vision encoder with local CFF+MAE objectives: each local window updates a small set of consecutive encoder blocks and a lightweight decoder, while earlier blocks provide a no-gradient boundary representation for the active window (Stage 1). After local pretraining, Stage 2 attaches task-specific heads for classification or structured prediction and uses either head-only training or reverse-order local supervised fine-tuning.

Let the encoder contain L blocks. For an input image x, let $z_0 = E(x)$ denote its patch-token embedding, including positional embeddings and a class token when used. The intermediate states are $z_{\ell+1} = f_\ell(z_\ell), \qquad \ell = 0, \dots, L-1$. CFF+MAE trains the encoder through a sequence of local windows $\{G_k\}_{k=0}^{K-1}$, where

$$G_k = [a_k, b_k], \qquad 0 \le a_k \le b_k < L, \qquad w_k = b_k - a_k + 1.$$

The window contains blocks $\{f_{a_k}, \dots, f_{b_k}\}$. Layer-wise training uses $w_k = 1$, while pair-wise training uses $w_k = 2$ with $G_k = [k, k+1]$, $k = 0, \dots, L-2$. The shared block retains its learned parameters across successive windows.

For active window G_k, define $F_{<a_k} = f_{a_k-1} \circ \cdots \circ f_0$, $F_{a_k:b_k} = f_{b_k} \circ \cdots \circ f_{a_k}$, with $F_{<a_k}$ defined as the identity when $a_k = 0$. We write $F^{\mathrm{ng}}_{<a_k}$ for the prefix evaluated with autograd disabled. The boundary and active-window representations are

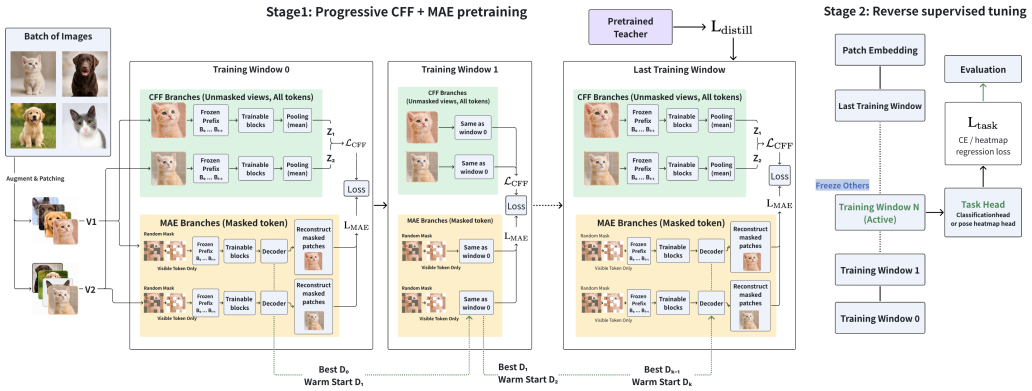


Figure 2: Overview of CFF+MAE. Stage 1 trains the encoder from shallow to deep local windows. For each window, the prefix is evaluated with autograd disabled, and only the active window and its local decoder are updated. The CFF branch uses unmasked augmented views, while the MAE branch uses masked views and reconstructs selected masked patches. Stage 2 attaches task heads and uses either head-only training or reverse-order local supervised fine-tuning.

$$h_k = F^{\mathrm{ng}}_{<a_k}(z_0), \qquad y_k = F_{a_k:b_k}(h_k).$$

During Stage 1, the blocks in G_k and the local decoder g_{b_k} are trainable. The patch embedding E is trained with the first window and frozen thereafter. For $a_k = 0$, the window input is $h_k = E(x)$ with autograd enabled. For later windows, E and the frozen prefix are evaluated without autograd.

3.2 Stage 1: Local-window CFF+MAE pretraining

Stage 1 trains the encoder from shallow to deep local windows. For each window $G_k = [a_k, b_k]$, CFF+MAE updates $F_{a_k:b_k}$ and the local decoder g_{b_k}, together with the patch embedding E when $a_k = 0$. The local loss has two branches: a supervised CFF branch on unmasked augmented views, and a masked-reconstruction branch on masked views. Both branches use the same active encoder window. The full procedure is summarized in Algorithm 1.

Algorithm 1 Stage 1: Progressive CFF+MAE pretraining

Require: Patch embedding E, Encoder blocks $\{f_\ell\}_{\ell=0}^{L-1}$, local decoders $\{g_\ell\}_{\ell=0}^{L-1}$, active windows $\{G_k = (a_k, b_k)\}_{k=0}^{K-1}$

for $k = 0$ to $K-1$ **do**
  Train $\{f_\ell : \ell \in [a_k, b_k]\}$ and g_{b_k}; also train E if $a_k = 0$. Freeze all remaining parameters.
  if $k > 0$ **then**
    Initialize g_{b_k} from the selected decoder checkpoint of the previous active window
  end if
  for each epoch and minibatch **do**
    Zero gradients of the active-window optimizer
    Draw two augmented views $x^{(1)}, x^{(2)}$ and patch targets $P^{(1)}, P^{(2)}$
    CFF branch: run unmasked views through $F^{\mathrm{ng}}_{<a_k}$ and $F_{a_k:b_k}$; Compute $\mathcal{L}^{(k)}_{\mathrm{CFF}}$, backpropagate, and release the CFF graph
    MAE branch: sample $M^{(k)}_{\mathrm{in}}$ and $M^{(k)}_{\mathrm{loss}} \subseteq M^{(k)}_{\mathrm{in}}$; Run visible tokens through $F^{\mathrm{ng}}_{<a_k}$ and $F_{a_k:b_k}$
    Decode with g_{b_k}, compute $\lambda_{k,t}\mathcal{L}^{(k)}_{\mathrm{MAE}}$, backpropagate, and release the MAE graph
    Take one optimizer step on the accumulated gradients
  end for
  Save the selected local checkpoint and transfer $g^\star_{b_k}$ to initialize the decoder of the next active window
end for

Unmasked-view CFF objective. Following the CFF formulation, we use its local contrastive objective on pooled representations as the discriminative branch of CFF+MAE. For each minibatch $\{(x_i, c_i)\}_{i=1}^B$, we sample two augmented views $x_i^{(1)}, x_i^{(2)} \sim \mathcal{A}(x_i)$, For classification, c_i is the ground-truth class label, and positive pairs are defined by shared labels. The CFF branch is therefore supervised, whereas the MAE reconstruction targets are derived from the input images. Each unmasked view is passed through the no-gradient prefix and the active window:

$$z^{(v)}_{i,k} = F_{a_k:b_k}\left(\overline{F}_{<a_k}(E(x_i^{(v)}))\right), \qquad u_i^{(v)} = \mathrm{norm}\left(\mathrm{pool}(z^{(v)}_{i,k})\right), \quad v \in \{1,2\}.$$

Let $\mathcal{I} = \{(i,v) : i = 1,\dots,B,\ v \in \{1,2\}\}$ index the $2B$ features. For anchor $q = (i,v)$, positives are $\mathcal{P}(q) = \{(j,v') \in \mathcal{I} : (j,v') \neq q,\ c_j = c_i\}$. All non-self features are included in the denominator.

We use the marginal contrastive loss employed by CFF. A positive margin is applied to positive pairs: $\tilde{s}^{(k)}_{q,a} = \min(\mathrm{sim}(u_q, u_a) + m_{b_k}, 1)$ if $a \in \mathcal{P}(q)$, and $\tilde{s}^{(k)}_{q,a} = \mathrm{sim}(u_q, u_a)$ otherwise. The CFF loss is

$$\mathcal{L}^{(k)}_{\mathrm{CFF}} = -\frac{1}{|\mathcal{I}|} \sum_{q \in \mathcal{I}} \frac{1}{|\mathcal{P}(q)|} \sum_{p \in \mathcal{P}(q)} \log \frac{\exp(\tilde{s}^{(k)}_{q,p}/\tau)}{\sum_{a \in \mathcal{I},\ a \neq q} \exp(\tilde{s}^{(k)}_{q,a}/\tau)}.$$

Here sim is cosine similarity and τ is the temperature. The positive margin is scheduled by the deepest block in the active window: $m_{b_k} = m_{\mathrm{shallow}} + \frac{b_k}{L-1}(m_{\mathrm{deep}} - m_{\mathrm{shallow}})$.

Masked reconstruction branch. The MAE branch adds patch-level pixel-reconstruction targets to the active window used by CFF. For each minibatch and each augmented view $x^{(v)}_i$, let $P_j(x^{(v)}_i) \in \mathbb{R}^D$ be the target patch vector at patch index j, and let N_p be the number of patch tokens. We sample a masked set $M^{(k)}_{i,\mathrm{in}} \subset \{1, \ldots, N_p\}$. The visible set is $V^{(k)}_{i,\mathrm{in}} = \{1, \ldots, N_p\} \setminus M^{(k)}_{i,\mathrm{in}}$. The encoder processes only tokens in $V^{(k)}_{i,\mathrm{in}}$.

The visible tokens are passed through the no-gradient prefix and the active window defined in Section 3.1. The local decoder g_{b_k} then restores the full patch order using the restore indices and predicts patch targets:

$$\hat{P}^{(v)}_i = g_{b_k}\left(y^{(v)}_{i,k,\mathrm{vis}}, \mathrm{ids}^{(k)}_{i,\mathrm{restore}}\right), \qquad v \in \{1, 2\},$$

where $y^{(v)}_{i,k,\mathrm{vis}}$ denotes the visible-token output of the active window. The decoder output has one prediction for each patch position.

Input-mask and loss-mask separation. We separate the mask that determines the encoder input from the subset of masked tokens used in the reconstruction loss. The encoder mask ratio ρ_{in} is fixed across active windows. The reconstruction loss is applied only to a scheduled subset $M^{(k)}_{i,\mathrm{loss}} \subseteq M^{(k)}_{i,\mathrm{in}}$, so every patch used in the loss has been hidden from the encoder. We set

$$\rho^{(k)}_{\mathrm{loss}} = \min(\rho_{\mathrm{in}},\ \rho_{\mathrm{in}} s_{b_k}), \qquad s_{b_k} = s_{\mathrm{shallow}} + \frac{b_k}{L-1}(s_{\mathrm{deep}} - s_{\mathrm{shallow}}).$$

Fixing ρ_{in} keeps the encoder input distribution fixed, while the scheduled loss subset controls how many masked patches contribute to the reconstruction loss at each depth.

Let $\mathcal{I}_{\mathrm{mae}} = \{(i, v) : i = 1, \ldots, B,\ v \in \{1, 2\}\}$ denote the two-view MAE batch. The MAE reconstruction loss is the mean squared error on the selected masked patches:

$$\mathcal{L}^{(k)}_{\mathrm{MAE}} = \frac{1}{|\mathcal{I}_{\mathrm{mae}}|} \sum_{(i,v) \in \mathcal{I}_{\mathrm{mae}}} \frac{1}{|M^{(k)}_{i,\mathrm{loss}}| D} \sum_{j \in M^{(k)}_{i,\mathrm{loss}}} \left\| \hat{P}^{(v)}_{i,j} - P_j(x^{(v)}_i) \right\|_2^2.$$

Here D is the patch-vector dimension. $P_j(x^{(v)}_i)$ is the target patch at position j, while $\hat{P}^{(v)}_{i,j}$ is the corresponding decoder prediction. The set $M^{(k)}_{i,\mathrm{loss}}$ contains the masked patch positions selected for the reconstruction loss for sample i at active window G_k; by construction, $M^{(k)}_{i,\mathrm{loss}} \subseteq M^{(k)}_{i,\mathrm{in}}$, so every reconstructed patch has been hidden from the encoder.

Local objective. The local objective for active window G_k combines the unmasked-view CFF loss and the MAE reconstruction loss:

$$\mathcal{L}^{(k)}_{\mathrm{local}} = \mathcal{L}^{(k)}_{\mathrm{CFF}} + \lambda_{k,t}\mathcal{L}^{(k)}_{\mathrm{MAE}} + \alpha_t \mathbf{1}[b_k = L-1]\mathcal{L}^{(k)}_{\mathrm{distill}}.$$

Here $\lambda_{k,t}$ is the MAE weight for window G_k, optionally warmed up during local training. The last term is used only when final-window distillation is enabled; otherwise $\alpha_t = 0$.

The MAE weight is warmed up within each local window: $\lambda_{k,t} = \lambda_k \min\left(1, \frac{t}{T_{\mathrm{warm}}}\right)$, where t is the local optimization step and λ_k is the scheduled MAE weight for window G_k. This warmup prevents the MAE loss from dominating early updates before the local decoder is calibrated to the current active representation.

Although the local objective is written as a weighted sum, we implement CFF and MAE backward passes sequentially. We first backpropagate the CFF loss and release its graph, then backpropagate the weighted MAE loss, and finally update the active window with the accumulated gradients. This preserves the same summed objective while reducing temporary branch-graph memory. For

ImageNet-scale experiments, we optionally add teacher supervision only at the final active window. A frozen MAE teacher produces attention maps A_i^T, and the student produces attention maps A_i^S at the final active window. We use $\mathcal{L}_{\text{distill}}^{(k)} = \frac{1}{B}\sum_{i=1}^{B} \left\|A_i^S - A_i^T\right\|_F^2$. This term is active only when $b_k = L-1$. Earlier windows are trained without teacher supervision, so distillation does not change the locality of the preceding windows. When distillation is not used, its coefficient is set to zero.

Each active window uses a local decoder attached to its deepest block. After training G_k, we select the decoder checkpoint with the best validation objective and use it to initialize the decoder for the next active window: $g_{b_{k+1}} \leftarrow g_{b_k}^{\star}$. Because neighboring windows face similar patch-reconstruction problems, this checkpoint transfer reduces repeated decoder cold starts across depth. The transfer applies only to local decoders; encoder blocks are trained according to the active-window schedule.

3.3 Stage 2: Downstream supervised training

After local pretraining, Stage 2 evaluates or adapts the learned representation with two supervised protocols. Algorithm 2 summarizes reverse-order supervised fine-tuning.

Algorithm 2 Stage 2: Reverse-order window-wise supervised fine-tuning

Require: Pretrained encoder F, task head h, and Stage-1 windows $\{G_k = [a_k, b_k]\}_{k=0}^{K-1}$
for $r = 0, \ldots, K-1$ **do**
  Set $k = K-1-r$ and $H_r = G_k$ ▷ Deep to shallow
  Train H_r, h, and final normalization where used; also train E if $a_k = 0$. Freeze all remaining parameters.
  Configure the optimizer for the trainable parameters.
  for each scheduled epoch and supervised minibatch (x, y) **do**
    Zero optimizer gradients.
    Window input: $z_{a_k} \leftarrow F_{<a_k}(E(x))$, with autograd if $a_k = 0$, and without autograd otherwise.
    Active window: $z_{b_k+1} \leftarrow F_{a_k:b_k}(z_{a_k})$ with autograd enabled.
    Frozen suffix: $z_L \leftarrow F_{b_k+1:L-1}(z_{b_k+1})$, preserving input gradients and using checkpointing where specified.
    Supervised loss: $\mathcal{L}_{\text{adapt}}^{(r)} \leftarrow \ell_{\text{task}}(h(R(z_L)), y)$.
    Backpropagate $\mathcal{L}_{\text{adapt}}^{(r)}$ and take one optimizer step on the trainable parameters.
  end for
  Save the adapted checkpoint.
end for
R applies final normalization and pooling where required; ℓ_{task} is the supervised task loss. Empty prefixes and suffixes are identity mappings.

Head-only training. Following the CFF evaluation protocol (Aghagolzadeh & Ezoji, 2025), we freeze the pretrained encoder and train only a lightweight task head on the final pooled representation:

$$\bar{z}_L = \texttt{pool}(F(x)), \qquad \mathcal{L}_{\text{sup}} = \texttt{CE}(h(\bar{z}_L), y).$$

Here h is the classifier head. This protocol measures the quality of the representation learned in Stage 1 without updating the pretrained encoder blocks.

Reverse-order local fine-tuning. We also conduct a local supervised fine-tuning that reuses the same window sequence defined in Section 3.1. Let $\{G_k\}_{k=0}^{K-1}$ be the Stage-1 windows ordered from shallow to deep. Reverse-order fine-tuning visits the same windows from deep to shallow:

$$H_r = G_{K-1-r} = [a_{K-1-r}, b_{K-1-r}], \qquad r = 0, \ldots, K-1.$$

At reverse window H_r, only the blocks in H_r, the task head, and the final normalization where used are trainable; all other encoder blocks are frozen. The supervised loss is evaluated at the final output as $\mathcal{L}_{\text{adapt}}^{(r)} = \texttt{CE}(h(\texttt{pool}(F(x))), y)$. The prefix is evaluated without autograd, while gradients from the final loss pass through the frozen suffix to reach the active window. For structured prediction tasks such as hand-pose estimation, the classifier head is replaced by a task-specific prediction head, and cross-entropy is replaced by the corresponding task loss. Reverse-order adaptation is used since the head operates on the deepest representation: later windows are updated first to stabilize features for the head, followed by refinement of earlier windows.

3.4 Training Memory Analysis

CFF+MAE reduces Stage 1 training memory by localizing the backward graph. The expressions below are memory-accounting approximations and omit common input buffers and small task heads unless stated otherwise. In end-to-end pretraining, all encoder blocks are trainable and their backward activations and optimizer states must be retained:

$$M_{\text{e2e}}^{\text{peak}} \approx M_{\theta}(\Theta_{\text{enc}}) + M_{\text{opt}}(\Theta_{\text{enc}}) + \sum_{\ell=0}^{L-1} M_{\text{act}}(f_\ell).$$

For Stage 1 window $G_k = [a_k, b_k]$, sequential branch execution gives the approximate peak memory:

$$M^{(k)}_{\text{local}} \approx M^{(k)}_{\text{persistent}} + \max\{A^{(k)}_{\text{CFF}}, A^{(k)}_{\text{MAE}}, A^{(k)}_{\text{step}}\}.$$

Here, $M^{(k)}_{\text{persistent}}$ accounts for all resident parameters, retained optimizer states, and accumulated parameter gradients. The A terms denote peak additional memory during the CFF branch, MAE branch, and optimizer step, excluding persistent memory. They include activations and temporary buffers; $A^{(k)}_{\text{MAE}}$ includes activations of both the active encoder window and the decoder. When distillation is enabled, teacher parameters and temporary tensors are included in the corresponding terms. The CFF graph is released before the MAE forward pass; both branch graphs are released before the optimizer step. The Stage 1 peak is $M^{\text{peak}}_{\text{Stage1}} = \max_{0 \le k < K} M^{(k)}_{\text{local}}$. Resident encoder parameters still contribute memory that grows with network depth. Figure 5 illustrates the execution schedule. Throughout this section, M_θ, M_{grad}, M_{opt}, and M_{act} denote parameter, parameter-gradient, optimizer-state, and retained backward-activation memory; Θ_{enc} denotes all encoder parameters.

Stage 2 has a different memory profile. In head-only training, the encoder is frozen and only the task head stores optimizer states. In reverse-order local fine-tuning, only the current window H_r, the task head, and the optional final normalization are trainable. However, the supervised loss is evaluated at the final output, so gradients must pass through the frozen suffix blocks to reach the active window. The approximate memory for reverse-order fine-tuning is

$$M^{(r)}_{\text{stage2}} \approx M^{(r)}_{\text{persistent}} + M_{\text{act}}(H_r) + M^{\text{impl}}_{\text{suf}}(b_r + 1 : L - 1).$$

where b_r is the deepest block in H_r, and $M^{\text{impl}}_{\text{suf}}$ denotes the implementation-dependent activation cost of the frozen suffix. $M^{(r)}_{\text{persistent}}$ includes all resident parameters, retained optimizer states, and parameter gradients. In the checkpointed implementation, the frozen suffix preserves input gradients while recomputing activations during backward. When the earliest window is active, the backward dependency spans the full encoder.

4 Experiments

Experimental setup: We evaluate CFF+MAE on CIFAR-10, ImageNet-1K, and STB hand-pose estimation (Krizhevsky & Hinton, 2009; Russakovsky et al., 2015; Zhang et al., 2017). We denote ViT$[d, h, L]$ as a Vision Transformer with embedding dimension d, h attention heads, and L encoder blocks. CIFAR-10 experiments use ViT[128,4,5] and ResNet[128,5] (Dosovitskiy et al., 2021; He et al., 2016). For ImageNet-1K, we use ViT[192,12,6], a 6-block MAE-ViT-Tiny-style encoder aligned with the DeiT-6 and D-MAE-6 configurations of Zhou et al. (2025). For STB, we use a ViT[192,6,6] heatmap-based model. Our CFF-only (Aghagolzadeh & Ezoji, 2025) and CFF+MAE are compared using the same backbone, window configuration, and downstream protocol. Published DeiT-6 and D-MAE-6 results from Zhou et al. (2025) are reported separately as external references. Classification performance is measured by top-1 accuracy, and STB performance by AUC.

Table 1: Main task performance. CIFAR-10 reports top-1 accuracy; STB reports AUC. BP is reported as a full end-to-end supervised reference, while CFF-only and CFF+MAE use local training protocols.

(a) Full BP reference

Dataset	BP
CIFAR-10 ViT	88.93%
CIFAR-10 ResNet	91.46%
STB ViT	85.87%

(b) Local training methods

Dataset	Model	Schedule / Protocol	CFF-only	CFF+MAE
CIFAR-10	ViT	Layer-wise / head-only	83.65%	85.88%
CIFAR-10	ResNet	Layer-wise / head-only	86.69%	87.72%
STB	ViT	Layer-wise / local FT	83.03%	84.20%

Table 2: ImageNet-1K performance with ViT-MAE-Tiny. External DeiT-6 and D-MAE-6 references are taken from (Zhou et al., 2025); they correspond to 6-layer depth-scaled ViT references under supervised DeiT training and MAE-based pretraining. Our local pair-wise CFF+MAE exceeds the DeiT-6 reference while remaining below the stronger D-MAE-6 reference.

(a) Full-training references

Method	Top-1
DeiT-6 (Zhou et al., 2025)	63.4%
D-MAE-6 (Zhou et al., 2025)	68.6%

(b) Local training methods

Schedule	CFF-only	CFF+MAE (**Ours**)	CFF+MAE + Distill (**Ours**)
Layer-wise ($w = 1$)	45.90%	49.17%	50.20%
Pair-wise ($w = 2$)	63.30%	64.75%	64.98%

Task Performance: Tables 1 and 2 summarize the main results. Across CIFAR-10, ImageNet-1K, and STB, CFF+MAE consistently improves over CFF-only within each local training configuration, indicating that masked reconstruction provides a useful patch-level signal for local contrastive

Table 3: Learning locality and reported accuracy under each method's experimental setting. For CFF-based methods, locality refers to Stage 1; our IN-1K results include Stage 2 supervised fine-tuning.

Family	Method	Locality	Dataset	Local objective	Backbone / Acc. (%)
Bio-motivated	FA (Lillicrap et al., 2016)	layer (w=1)	CIFAR-10	random feedback	FC1-500 /50.62
	DFA (Nøkland, 2016)	layer (w=1)	CIFAR-10	Projected output error	FC1-500 /50.38
	DRTP (Frenkel et al., 2021)	layer (w=1)	CIFAR-10	Random target projection	FC1-500 / 46.08
	SoftHebb (Journé et al., 2023)	layer (w=1)	IN-1K	Soft-WTA Hebbian rule	SoftHebb / 27.3
Block-local optimization	Greedy layer-wise (Belilovsky et al., 2019)	block (w=1)	IN-1K	Local CE	SimCNN (k = 1) / 58.1
	LoCo (Xiong et al., 2020)	stage (w=7-10)	IN-1K	Instance contrastive	ResNet-50 / 69.5
	InfoPro (Wang et al., 2021)	module (K=2)	IN-1K	Reconstruction + CE	ResNet-101 / 78.15
	Blockwise SSL (Siddiqui et al., 2024)	stage (w=4)	IN-1K	Barlow Twins	ResNet-50 / 70.48
FF-based learning	CFF+margin (Aghagolzadeh & Ezoji, 2025)	block (w=1)	CIFAR-10	SupCon with margin	ViT[128,4,5] / 80.42
	CFF+margin (RandAugment)	block (w=1)	CIFAR-10	SupCon with margin	ViT[240,6,7] / 85.37
	HFF (Sarode et al., 2026)	layer (w=1)	IN-1K	Prototype-based	VGG-16 / 25.70
Ours	**CFF+MAE**	**block (w=1)**	CIFAR-10	**SupCon + MAE**	**ResNet[128,5] / 87.72**
Ours	**CFF+MAE**	**block (w=1)**	CIFAR-10	**SupCon + MAE**	**ViT[128,4,5] / 85.88**
Ours	**CFF+MAE**	**block (w=1)**	IN-1K	**SupCon + MAE**	**ViT[192,12,6] / 49.17**
Ours	**CFF+MAE**	**block (w=2)**	IN-1K	**SupCon + MAE**	**ViT[192,12,6] / 64.75**
Ours	**CFF+MAE+Distillation**	**block (w=2)**	IN-1K	**SupCon + MAE + KD**	**ViT[192,12,6] / 64.98**

learning. The gains are $+2.23$ and $+1.03$ percentage points on CIFAR-10 ViT and ResNet, $+3.27$ and $+1.45$ percentage points on ImageNet-1K layer-wise and pair-wise schedules, and $+1.17$ AUC points on STB layer-wise local fine-tuning. The layer-wise and pair-wise settings are treated as distinct local-training configurations rather than as a compute-matched ablation. On ImageNet-1K, pair-wise CFF+MAE reaches 64.75%, exceeding the external DeiT-6 reference (63.4%) under the same 6-block depth setting; final-window distillation further improves it to 64.98%. The stronger D-MAE-6 reference remains higher at 68.6%, indicating that CFF+MAE improves local training but does not yet match MAE-based full-training.

Comparison with prior works. Table 3 reveals an important gap in prior local-learning results. Methods reporting strong ImageNet-1K accuracy are predominantly demonstrated on CNN backbones and often use substantially coarser notions of locality. For example, InfoPro's (Wang et al., 2021) 78.15% result uses only two gradient-isolated modules (K=2) in ResNet-101, retaining a deep backward graph within each module. LoCo (Xiong et al., 2020) overlaps local blocks to enable cross-block feedback, increasing the active activation scope, while Blockwise SSL (Siddiqui et al., 2024) operates on only the four major ResNet-50 stages and relies on simultaneous block training. Thus, these methods use coarser or more coupled local units than our one- or two-block windows, and fine-grained local ViT training at ImageNet scale remains largely unexplored. This gap is particularly pronounced for FF-based learning. Prior CFF results demonstrate ViTs only on smaller-scale datasets, while the reported ImageNet-1K result for HFF (Sarode et al., 2026) reaches 25.70% using VGG-16. In contrast, **CFF+MAE is, to our knowledge, the first CFF-based approach to demonstrate competitive ImageNet-1K learning** with a ViT, reaching 49.17% with strictly layer-wise (w=1) training and 64.98% with two-block (w=2) local windows. Under the same backbone and training protocol, masked reconstruction improves CFF-only by 3.27 and 1.45 percentage points for layer-wise and pair-wise training, respectively. These results show that patch-level reconstruction substantially improves the scalability of CFF while retaining a tightly bounded one- or two-block backward graph, including at ImageNet scale.

Table 4: Ablation study on CIFAR-10 using ViT[128,4,5]. Uniform mask means the same mask reconstruction rate (1.0) across layers, while Layer-wise mask means layer-dependent rates from 0.8 to 1.0.

	Full BP (1-view)	**CFF only (2-view)**	**MAE only (2-view)**		**CFF + MAE (2-view)**		
Setting	No MAE	Layer-wise	Uniform loss mask $s = 1$	Depth-scheduled loss mask	Uniform loss mask $s = 1$	Depth-scheduled loss mask without decoder transfer	Depth-scheduled loss mask
Accuracy (%)	88.93	83.65	36.82	37.70	85.28	85.37	**85.88**

Ablation Study: Table 4 evaluates masked reconstruction, proposed input-mask/loss-mask separation, and decoder transfer on CIFAR-10. Under our local-pretraining and head-only evaluation protocol, MAE-only achieves 36.82–37.70%, below CFF-only at 83.65%. CFF uses class-based contrastive supervision on pooled features, whereas MAE reconstructs pixels without class labels during pretraining. Adding reconstruction over all input-masked patches raises CFF accuracy to 85.28%, supporting masked reconstruction as an auxiliary objective for local contrastive learning. Keeping the input-mask ratio fixed, applying reconstruction loss to a depth-scheduled subset $M_{\text{loss}}^{(k)} \subseteq M_{\text{in}}^{(k)}$ further improves accuracy to 85.88%, a gain of 0.60 percentage points over reconstruction on the entire masked set. With the same loss-mask schedule, removing decoder transfer reduces accuracy to 85.37%, showing a further 0.51-point benefit from transferring the decoder between windows.

Edge Efficiency Profiling: A central motivation of CFF+MAE is edge-side trainability. We therefore profile its training efficiency on Jetson-class hardware. For each method and batch

size, we report: latency per batch and per image, peak allocated and reserved GPU memory, peak CPU RSS memory, and dynamic energy per batch and per image. Dynamic power is computed as $P_{\mathrm{dyn}} = P_{\mathrm{run}} - P_{\mathrm{idle}}$, and dynamic energy per image is $E_{\mathrm{dyn/img}} = P_{\mathrm{dyn}} T_{\mathrm{run}} / N_{\mathrm{images}}$. Detailed latency, memory, throughput, and energy measurements are provided in Appendix C.

Table 5 shows the cost of adding reconstruction to CFF. In Stage 1, CFF+MAE increases peak allocated memory by about 0.3% for layer-wise training and 0.2% for pair-wise training. Step latency rises by 25.8% and 22.2%, and dynamic board energy per batch by 23.5% and 19.0%. Relative to two-view Full BP, its peak allocations are 76.6% and 61.4% lower. These are costs per local-window update; Full BP updates the entire encoder per batch. **Sequential Prefix Rematerialization (SPR)**, an **activation checkpointing scheme** detailed in Appendix C.4.1, provides a rematerialization-based alternative that retains full-network gradient propagation: smaller backward windows reduce activation memory but incur additional prefix recomputation, increasing latency and dynamic energy. At the update level, layer-wise CFF+MAE has lower memory, latency, and dynamic energy than SPR-W1, with the same pattern for pair-wise CFF+MAE relative to SPR-W2. These update-level comparisons do not imply equal complete-training cost because the local and SPR updates perform different optimization work. Separating the CFF and MAE backward passes further reduces the memory peak relative to joint backward, with little measured change in dynamic energy; two-view Full BP also benefits from sequential view-wise backward (Appendix C).

Table 5: Jetson training efficiency on ImageNet-1K at batch size 32. For CFF-based methods, memory is the maximum per-window peak, while other metrics are averaged across windows. SPR-Wk denotes a k-block backward window.

	Layer-wise		Pair-wise	
Metric	**CFF-only**	**CFF+MAE**	**CFF-only**	**CFF+MAE**
Peak memory (MiB)	785.9	788.3	1301.4	1303.8
Latency (ms/batch)	335.2	421.7	458.6	560.6
Dynamic power (W)	4.46	5.01	5.40	5.84
Dynamic energy (mJ/batch)	3966	4897	5413	6441

	Full backpropagation (2-view)			
Metric	**Full BP**	**SPR-W1**	**SPR-W2**	**SPR-W3**
Peak memory (MiB)	3373.9	891.5	1396.8	1893.5
Latency (ms/batch)	962.1	1933.4	1342.3	1151.1
Dynamic power (W)	7.06	8.84	8.03	7.61
Dynamic energy (mJ/batch)	10937	22199	15464	13193

Cumulative training cost. A local-window update and a full-network update perform different work, so update-level measurements do not establish a complete-training speedup. We therefore project the cumulative cost of the ImageNet-1K training schedules from H200 profiling (Appendix D). The layer-wise and pair-wise CFF+MAE schedules require 81.40 and 86.84 hours of projected training-step time, compared with 62.26 hours for the 800-epoch Full BP reference.

Quantization Robustness: To further reduce the peak training memory of CFF+MAE, we consider lowering the precision of both model weights and intermediate activations. Weight quantization reduces resident parameter storage, while activation quantization reduces active-window representations and temporary tensors; W8A8 therefore targets both major memory components relative to BF16. Table 6 evaluates the corresponding accuracy robustness. W8A16, W16A8, and W8A8 use PTQ, while W4A8 uses QAT. Relative to BF16, W8A8 reduces top-1 accuracy by only 0.32 points for layer-wise training and 0.10 points for pair-wise training, with similarly small degradation for weight-only and activation-only INT8. In contrast, W4A8 QAT reduces accuracy by 2.98 and 2.77 points. These results show that CFF+MAE is robust to INT8 quantization, supporting lower-memory implementations, while 4-bit weight quantization incurs a larger accuracy cost.

Table 6: ImageNet-1K top-1 accuracy under PTQ and QAT for ViT[192,12,6]. All models use final-window distillation during Stage 1.

	Float	INT8 PTQ			INT4 QAT
Training	**BF16**	**W8A16**	**W16A8**	**W8A8**	**W4A8**
Layer-wise	50.20%	49.86%	49.96%	49.88%	47.22%
Pair-wise	64.98%	64.79%	64.98%	64.88%	62.21%

5 Conclusion

We presented CFF+MAE, which adds local masked reconstruction to contrastive Forward–Forward learning. Across CIFAR-10, ImageNet-1K, and STB hand-pose estimation, it improves task performance over CFF-only, with pair-wise training reaching 64.75% ImageNet top-1 accuracy without distillation. These results support masked reconstruction as an auxiliary objective for improving local CFF training. Sequential branch execution keeps the additional Stage 1 peak memory over CFF-only small. On Jetson Orin Nano, peak allocated GPU memory is reduced by up to a factor of 4.3 relative to joint two-view Full BP. Local-window updates also have lower measured latency and energy than the tested SPR full-network updates. H200 profiling shows a memory–computation trade-off: lower peak memory than the Full BP reference, but higher projected cumulative training-step time. CFF+MAE thus improves CFF task performance while retaining a small peak-memory footprint, making it useful for memory-constrained training where additional computation is affordable.

AI USE STATEMENT

Generative AI tools were used solely to assist with language editing, including improving grammar, clarity, and presentation of author-written text. They were not used to generate the research ideas, formulate the proposed method, design the experiments, produce or analyze experimental results, or make scientific claims. All AI-assisted content was reviewed and revised by the authors to ensure technical correctness, consistency with the conducted experiments, and appropriate attribution of prior work. The authors take full responsibility for the final content of the manuscript.

ETHICS STATEMENT

This work uses established benchmark datasets and does not involve new human subject recruitment or collection of personally identifiable information. Experiments are conducted for methodological evaluation of memory-efficient vision learning. As with other models trained on existing visual datasets, the resulting models may inherit biases or limitations present in those datasets. We do not identify additional ethical risks specific to the proposed training method.

REPRODUCIBILITY STATEMENT

We provide the model architecture, local-window training procedure, objectives, and optimization schedule in the main paper. Dataset settings, hyperparameters, training schedules, and implementation details are documented in the appendix and supplementary materials. The submitted supplementary code contains the implementations used for the reported experiments, including layer-wise and pair-wise CFF+MAE, ablation studies, quantization, and Jetson and H200 efficiency profiling to support reproduction of the reported experiments.

REFERENCES

Hossein Aghagolzadeh and Mehdi Ezoji. Contrastive forward-forward: A training algorithm of vision transformer. *Neural Networks*, 192:107867, 2025.

Reza Akbarian Bafghi, Nidhin Harilal, Claire Monteleoni, and Maziar Raissi. Parameter efficient fine-tuning of self-supervised ViTs without catastrophic forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 3679–3684, 2024.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pp. 583–593. PMLR, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pp. 736–745. PMLR, 2020.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Xing Chen, Dongshu Liu, Jérémie Laydevant, and Julie Grollier. Self-contrastive Forward-Forward algorithm. *Nature Communications*, 16:5978, 2025. arXiv:2409.11593.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Charlotte Frenkel, Martin Lefebvre, and David Bol. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. *Frontiers in neuroscience*, 15:629892, 2021.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. volume 30, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022. doi: 10.1109/CVPR52688.2022.01553.

Jefferson Hernandez, Ruben Villegas, and Vicente Ordonez. ViC-MAE: Self-supervised representation learning from images and video with contrastive masked autoencoders. In *European Conference on Computer Vision*, 2024.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Zhicheng Huang, Xiaojie Jin, Chengze Lu, et al. Contrastive masked autoencoders are stronger vision learners. *arXiv:2207.13532*, 2022.

Adrien Journé, Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Hebbian deep learning without feedback. In *International Conference on Learning Representations*, 2023.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1): 13276, 2016.

Guy Lorberbom, Itai Gat, Yossi Adi, Alexander Schwing, and Tamir Hazan. Layer collaboration in the Forward-Forward algorithm. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 14141–14148, 2024.

Sindy Löwe, Peter O'Connor, and Bastiaan S. Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *Advances in Neural Information Processing Systems*, 2019.

Yixuan Luo, Mengye Ren, and Sai Qian Zhang. Bim: Block-wise self-supervised learning with masked image modeling. *arXiv preprint arXiv:2311.17218*, 2023.

Shlok Mishra, Joshua Robinson, Huiwen Chang, David Jacobs, Aaron Sarna, Aaron Maschinot, and Dilip Krishnan. A simple, efficient and scalable contrastive masked autoencoder for learning visual representations. *arXiv preprint arXiv:2210.16870*, 2022.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *International conference on machine learning*, pp. 4839–4850. PMLR, 2019.

Priyank Pathak, Jingwei Zhang, and Dimitris Samaras. Local learning on transformers via feature reconstruction. *arXiv preprint arXiv:2212.14215*, 2022. URL https://arxiv.org/abs/2212.14215.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Shalini Sarode, Brian Moser, Joachim Folz, Federico Raue, Tobias Nauen, Stanislav Frolov, and Andreas Dengel. Hyperspherical forward-forward with prototypical representations. *arXiv preprint arXiv:2605.00082*, 2026. URL https://arxiv.org/abs/2605.00082.

Shoaib Ahmed Siddiqui, David Krueger, Yann LeCun, and Stéphane Deny. Blockwise self-supervised learning at scale. *Transactions on Machine Learning Research*, 2024.

Liang Sun, Yang Zhang, Weizhao He, Jiajun Wen, Linlin Shen, and Weicheng Xie. Deeperforward: Enhanced forward-forward training for deeper and better performance. In *The Thirteenth International Conference on Learning Representations*, 2025.

Haoqing Wang, Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhi-Hong Deng, and Kai Han. Masked image modeling with local multi-scale reconstruction. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2122–2131. IEEE, 2023a.

Shaoru Wang, Jin Gao, Zeming Li, Xiaoqin Sun, and Weiming Hu. A closer look at self-supervised lightweight vision transformers. In *International conference on machine learning*, pp. 35624–35641. PMLR, 2023b.

Yulin Wang, Zanlin Ni, Shiji Song, Le Yang, and Gao Huang. Revisiting locally supervised learning: An alternative to end-to-end training. In *International Conference on Learning Representations*, 2021.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9653–9663, 2022.

Yuwen Xiong, Mengye Ren, and Raquel Urtasun. LoCo: Local contrastive representation learning. In *Advances in Neural Information Processing Systems*, 2020.

Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. A hand pose tracking benchmark from stereo matching. In *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 982–986, 2017. doi: 10.1109/ICIP.2017.8296428.

Chengwei Zhou, Vipin Chaudhary, and Gourav Datta. Rethinking vision transformer depth via structural reparameterization. *arXiv preprint arXiv:2511.19718*, 2025.

A Experiment Settings

A.1 Implementation Details and Hyperparameters

A.1.1 CIFAR-10

CIFAR-10 ViT. For CIFAR-10 ViT experiments, we use an input resolution of 96×96 with patch size 8×8, resulting in 144 patch tokens. The ViT-style encoder has embedding dimension $E = 128$, $H = 4$ attention heads, and $L = 5$ Transformer blocks. All methods use batch size 512, RandAugment, and report top-1 accuracy. The full BP baseline is trained end-to-end with supervised cross-entropy for 600 epochs using AdamW with learning rate 5.0×10^{-4}. For CFF-only and CFF+MAE, Stage 1 uses layer-wise local pretraining from shallow to deep, where each active window contains one encoder block and the preceding prefix is evaluated with no-gradient forward. Each block is trained for 600 epochs with AdamW and Stage-1 learning rate 2.5×10^{-3}. The CFF objective uses temperature $\tau = 0.20$ and a linear margin schedule from $m_0 = 0.40$ to $m_L = 0.10$. CFF-only disables the MAE branch by setting the MAE mask ratio and MAE loss weights to zero. CFF+MAE uses MAE mask ratio $\rho_{\text{in}} = 0.6$, linear mask scheduling, 50 MAE warmup epochs, MAE scale from 0.8 to 1.0, MAE loss weight from $\lambda_0 = 0.05$ to $\lambda_L = 0.50$, MAE decoder weight decay 0.02, and decoder checkpoint transfer across layers. In Stage 2, local methods freeze the encoder and train only the classifier head for 100 epochs using AdamW with learning rate 5.0×10^{-4} and cosine decay.

CIFAR-10 ResNet. For CIFAR-10 ResNet experiments, we use the same CIFAR-10 data pipeline, On-RAM loading, input resolution 96×96, patch size 8×8, batch size 512, RandAugment, and top-1 accuracy evaluation. The ResNet-style encoder uses embedding dimension $E = 128$ and $L = 5$ blocks. The full BP baseline is trained end-to-end with supervised cross-entropy for 600 epochs using AdamW with learning rate 7.0×10^{-4}. For CFF-only and CFF+MAE, Stage 1 uses layer-wise

local pretraining with one active block at a time, trained from shallow to deep with no-gradient prefix forwarding. CFF-only trains each block for 700 epochs using AdamW with learning rate 2.5×10^{-3}, while CFF+MAE uses learning rate 2.7×10^{-3}. Both local methods use CFF temperature $\tau = 0.20$ and a linear margin schedule from $m_0 = 0.40$ to $m_L = 0.10$. CFF-only disables the MAE branch by setting $\rho_{\text{in}} = 0$, MAE scales to zero, and MAE weights to zero. CFF+MAE uses MAE mask ratio $\rho_{\text{in}} = 0.4$, linear mask scheduling, 50 MAE warmup epochs, MAE scale from 1.0 to 0.20, MAE loss weight $\lambda_0 = \lambda_L = 0.05$, MAE weight-decay argument 0.02, and decoder checkpoint transfer. In Stage 2, the encoder is frozen and only the classifier head is trained for 200 epochs using AdamW with learning rate 1.0×10^{-3}, cross-entropy loss, and cosine decay to zero.

A.1.2 STB HAND-POSE ESTIMATION.

For STB, we evaluate 3D hand-pose estimation using a ViT-style encoder with input resolution 96×96, patch size 8×8, embedding dimension $E = 192$, $H = 6$ attention heads, and $L = 6$ Transformer blocks. The batch size is 512, the train/test split ratio is 0.8333, hand cropping and 3D keypoint prediction are enabled, and performance is reported using AUC/PCK. The pose prediction loss uses 18×18 heatmaps, pose loss weight $\lambda_{\text{pose}} = 4.0$, and depth loss coefficient 0.05. For CFF-only and CFF+MAE, Stage 1 performs layer-wise local pretraining from shallow to deep with one active encoder block, no-gradient prefix forwarding, and batch size 512. Each block is trained for 500 epochs using Adam with learning rate 1.0×10^{-3} and weight decay 1.0×10^{-4}. The CFF objective uses temperature $\tau = 0.20$ and a linear margin schedule from $m_0 = 0.35$ to $m_L = 0.10$. CFF-only disables the MAE branch with zero MAE mask ratio and zero MAE loss weights. CFF+MAE uses MAE mask ratio $\rho_{\text{in}} = 0.4$, linear mask scheduling, 10 MAE warmup epochs, MAE scale from 0.9 to 1.0, and MAE loss weight from $\lambda_0 = 0.05$ to $\lambda_L = 0.4$, with decoder checkpoint transfer. In Stage 2, both local methods visit the encoder blocks once from deep to shallow, updating the current block together with the XY heatmap and depth heads. Both CFF-only and CFF+MAE use 150 epochs per block with a Stage-2 base learning rate of 1.0×10^{-3}. In both cases, the backbone learning rate is $0.1\eta_2$, the head learning rate is η_2, backbone weight decay is 1.0×10^{-4}, and head weight decay is zero. The full BP baseline directly trains all encoder blocks, the patching layer, and the pose heads end-to-end for 2000 epochs using AdamW, base learning rate 1.0×10^{-3}, backbone learning rate $0.1\eta_2$, head learning rate η_2, AdamW betas $(0.9, 0.999)$, epsilon 1.0×10^{-8}, and linear warmup followed by cosine decay.

A.1.3 IMAGENET-1K.

For ImageNet-1K, we use a MAE-ViT-Tiny-style encoder with input resolution 224×224, patch size 16×16, 196 patch tokens, embedding dimension $E = 192$, $H = 12$ attention heads, and $L = 6$ Transformer blocks. Features are obtained by mean pooling over patch tokens, and the local MAE decoder is attached to the deepest block of the active window. The reconstruction target is the pixel-space patch vector with normalized pixel loss. Stage 1 uses two augmented views with RandAugment, while Stage 2 disables RandAugment. For both layer-wise and pair-wise CFF+MAE, Stage 1 trains from shallow to deep with no-gradient prefix forwarding, 350 epochs per window, AdamW, batch size 128 per GPU on 8 GPUs, effective global batch size 1024, and autocast. Because adjacent pair-wise windows overlap, the layer-wise and pair-wise settings are not compute-matched and are treated as distinct local-training configurations rather than as a compute-matched ablation. The layer-wise setting uses one active block and learning rate 8.0×10^{-4} with cosine decay to 1.0×10^{-4}; the pair-wise setting uses two adjacent active blocks and learning rate 8.0×10^{-4} with linear warmup plus cosine decay to 1.0×10^{-4}. Both settings use 10 warmup epochs, warmup start factor 0.5, CFF temperature $\tau = 0.1$, and a linear margin schedule from $m_0 = 0.1$ to $m_L = 0.02$. The MAE branch uses mask ratio $\rho_{\text{in}} = 0.75$, linear loss-mask scheduling, MAE scale from 0.8 to 1.0, MAE loss weight $\lambda_0 = \lambda_L = 1.0$, 10 MAE warmup epochs, and decoder checkpoint transfer. In Stage 2, ImageNet-1K uses reverse-order local fine-tuning with no-gradient prefix forwarding and checkpointed suffix forwarding. The layer-wise setting trains one block at a time for 100 epochs per reverse window, while the pair-wise setting trains adjacent block pairs for 100 epochs per reverse window. Both use SGD with momentum 0.9, zero weight decay, cross-entropy loss, no label smoothing, head learning rate 0.1, 10 warmup epochs, batch size 128 per GPU, and effective global batch size 1024. The layer-wise block learning rate is 0.036 for the deepest layer and 0.032 for the other layers, with cosine decay to 0.01. The pair-wise block learning rate is 0.020 for the deepest pair and 0.016 for the other pairs, with cosine decay to 0.005. The CFF-only ImageNet-1K

baseline uses the same backbone, data pipeline, local-window schedule, CFF temperature, margin schedule, and downstream protocol as CFF+MAE, but disables the MAE branch and final-window distillation unless explicitly stated. Distilled and non-distilled variants use the same Stage 2 protocol; reconstruction and teacher distillation are disabled during Stage 2.

B Illustration of our CFF+MAE Pipeline

Figure 3 and Figure 4 illustrate the Stage 1 local training units. The prefix is evaluated without a backward graph, while the active window and local decoder are updated. CFF supervises pooled representations from unmasked views, and MAE reconstructs selected masked patches from masked views. Layer-wise training uses $G_k = [k, k]$; pair-wise training uses overlapping windows $G_k = [k, k+1]$.

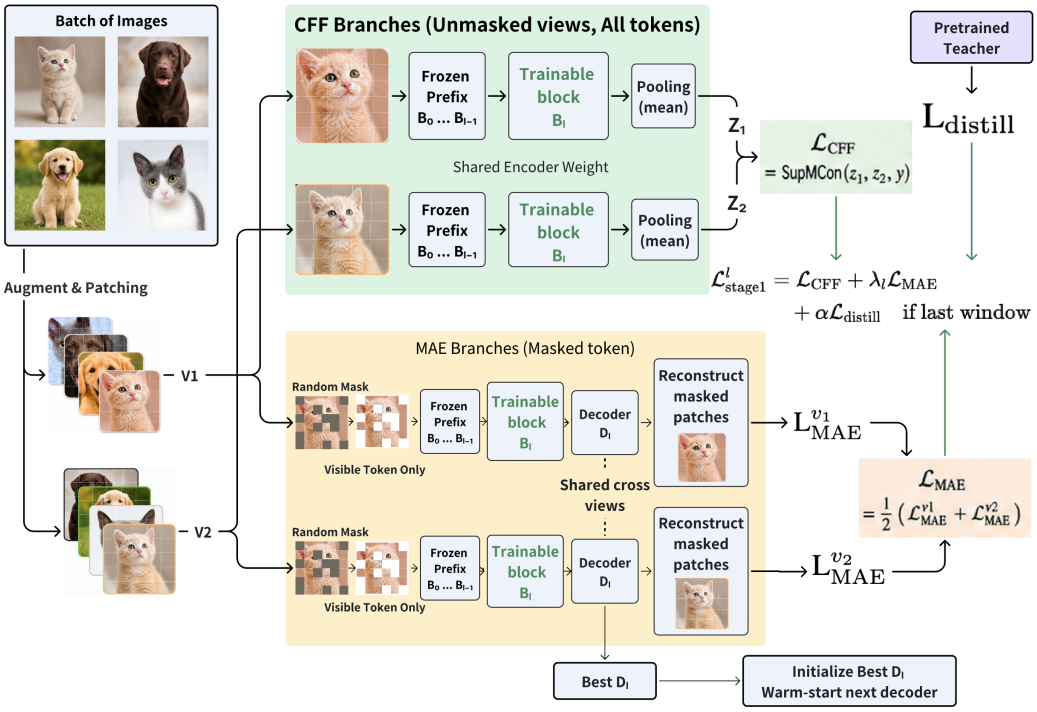


Figure 3: Layer-wise Stage 1 unit. One encoder block is active, the prefix is evaluated without a backward graph, and the local MAE decoder is attached to the active block.

B.1 Memory Efficiency

Figure 5 summarizes the memory logic used in Section 3.4. For a comparable two-view full-BP baseline, both augmented views are backpropagated jointly through the full encoder, so intermediate activations from all encoder blocks must be retained until the backward pass. In our Jetson profiling, this two-view full-BP setting reaches a peak allocated GPU memory of 3373.9 MiB. During Stage 1, CFF+MAE evaluates the prefix without autograd and retains backward activations only for the active window and local decoder. The CFF and MAE branches are backpropagated sequentially, so their temporary branch graphs do not need to coexist at peak memory.

C Full Jetson Efficiency Results

C.1 Measurement scope

GPU memory is measured using PyTorch's peak allocated and reserved counters, reset before each profiled training interval and read before validation. For each stage, the main table reports the maximum per-window peak allocated memory, $M^{\text{peak}}_{\text{stage}} = \max_k M_{\text{peak},k}$. Memory is expressed in

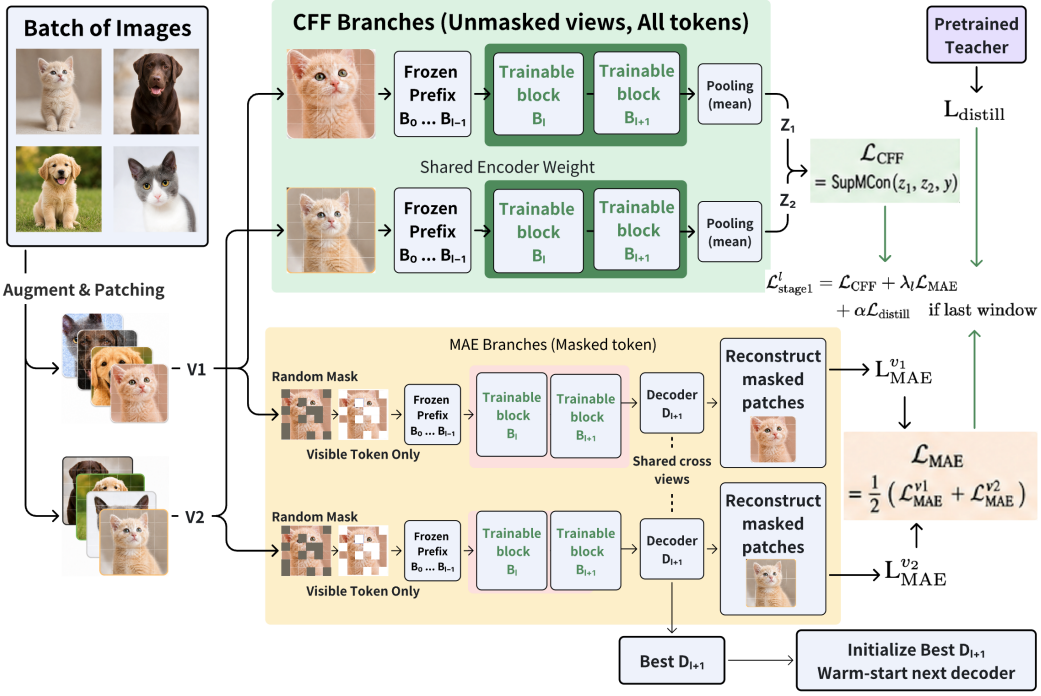


Figure 4: Pair-wise Stage 1 unit. Two adjacent encoder blocks are active, and the local MAE decoder is attached to the deeper block.

MiB (2^{20} bytes). The measurements cover the profiled training intervals and exclude setup, validation, and window transitions. CPU RSS is sampled once per batch, and its reported peak is the maximum sampled value.

The step timer starts after the data loader yields a batch and ends after synchronized training computation. It includes device transfer, forward, backward, and the optimizer update, but excludes loader waiting and subsequent logging or RSS sampling. Loop throughput is $N_{\text{images}}/T_{\text{loop}}$, with T_{loop} spanning the training function and its data-loading and loop overhead. Validation and the idle-baseline interval are excluded. Consequently, throughput is not the reciprocal of the reported step latency.

The main table uses the total-board power channel. The profiler computes arithmetic sample means $\bar{P}_{\text{run}}$ and $\bar{P}_{\text{idle}}$ and estimates

$$P_{\text{dyn}} = \bar{P}_{\text{run}} - \bar{P}_{\text{idle}}, \qquad \widehat{E}_{\text{dyn}} = P_{\text{dyn}} T_{\text{sample}}, \qquad \widehat{E}_{\text{dyn/batch}} = \frac{\widehat{E}_{\text{dyn}}}{N_{\text{batches}}}.$$

Here T_{sample} is the span between the first and last power samples inside the training interval; it can be slightly shorter than T_{loop}. Per-image energy divides by N_{images}. These estimates retain the original logging convention; they are not computed as dynamic power times step latency. Run energy, which includes idle power, is separately computed by trapezoidal integration of the sampled power trace. The arithmetic-mean dynamic estimate and the integrated run energy need not obey an exact subtraction identity for irregularly spaced samples. Window-level energies are computed before stage averaging; a product of stage-average power and stage-average time is not used.

C.2 Sequential versus Joint Backward

We examine whether sequentially releasing computation graphs reduces peak memory without changing the training objective or parameter-update frequency. We evaluate this scheduling choice for both CFF+MAE and two-view Full BP.

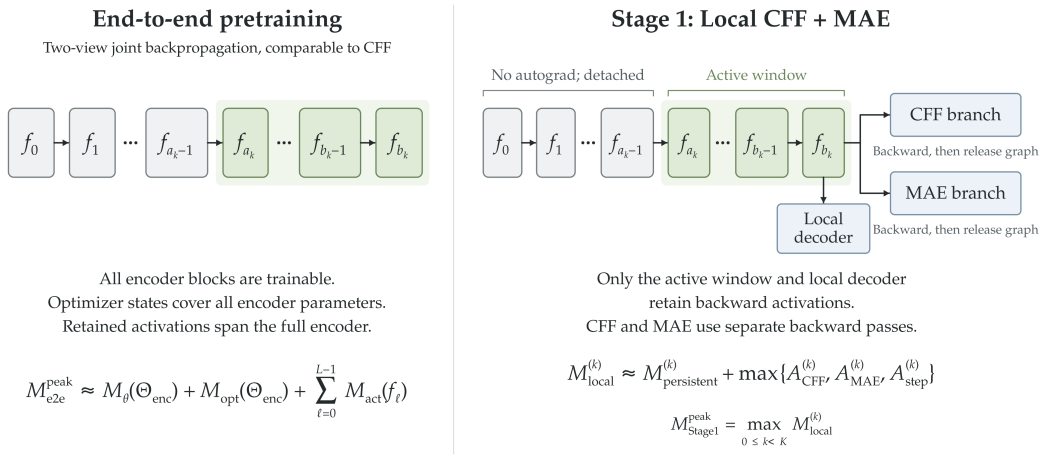


Figure 5: Memory logic for two-view full BP and Stage 1 CFF+MAE. Full BP retains intermediate activations for all encoder blocks, while CFF+MAE keeps the backward graph local to the active window and decoder. Sequential branch backward prevents the CFF and MAE branch graphs from coexisting at peak memory.

C.2.1 Sequential versus Joint Backward: CFF+MAE

Table 7 compares sequential and joint backward for the layer-wise and pair-wise schedules.

Table 7: Jetson efficiency comparison between sequential and joint backward for CFF+MAE with batch size 32. Memory is the maximum of the per-window peak allocated GPU memory. Dynamic power and energy are computed after subtracting the idle-power baseline. Lower is better for all metrics. Stage 2 labels identify the corresponding Stage 1 runs; both use the same Stage 2 procedure.

		Layer-wise		Pair-wise	
Stage	Metric	Sequential	Joint	Sequential	Joint
Stage 1	Peak memory (MiB)	**788.3**	950.6	**1303.8**	1532.7
	Step latency (ms/batch)	**421.7**	438.6	**560.6**	584.3
	Dynamic power (W)	5.01	**4.90**	5.84	**5.64**
	Dynamic energy (mJ/batch)	4897	**4864**	6441	**6359**
Stage 2	Peak memory (MiB)	782.9	782.9	1033.8	1033.8
	Step latency (ms/batch)	**428.5**	430.0	439.0	**438.4**
	Dynamic power (W)	7.10	**7.02**	7.20	**7.06**
	Dynamic energy (mJ/batch)	4966	**4941**	5070	**4980**

In Stage 1, sequential backward reduces peak allocated memory by 17.1% for layer-wise training and 14.9% for pair-wise training compared with joint backward. This occurs because sequential backward completes the CFF backward pass and releases its computation graph before constructing and backpropagating through the MAE graph. In contrast, joint backward retains both graphs until the combined loss is backpropagated.

Sequential execution reduces measured step latency by 3.9% and 4.1% for layer-wise and pair-wise training. Its measured dynamic energy is 0.7% and 1.3% higher. Repeated measurements are needed to determine whether these small energy differences are reproducible. The clearest benefit of sequential execution is the reduction in peak allocated memory.

C.2.2 Sequential versus Joint Backward: Two-view Full BP

Table 8 applies the same scheduling comparison to two-view Full BP. Both variants optimize the same mean two-view classification loss and perform one optimizer update per batch.

Table 8: Joint and sequential two-view Full BP efficiency on Jetson with batch size 32. Sequential backward processes the two views separately, accumulates their gradients, and performs one optimizer update per batch. Lower is better for all metrics.

Metric	Joint	Sequential
Peak memory (MiB)	3373.9	**1839.1**
Step latency (ms/batch)	962.1	**959.8**
Dynamic power (W)	7.06	**7.04**
Dynamic energy (mJ/batch)	10937	**10876**

Sequential view-wise backward reduces peak memory by 45.5%, from 3373.9 to 1839.1 MiB, because the first view's computation graph is released before the second view is processed. The reduction is not exactly 50% because model parameters, gradients, optimizer states, input tensors, and CUDA workspaces remain resident.

Latency, dynamic power, and dynamic energy differ by less than 1% between the two schedules. This is expected because both execute two forward passes and equivalent total backward computation. Sequential view-wise backward therefore provides a substantial peak-memory reduction without an evident latency or energy penalty in these measurements.

Table 9: Per-window (Layer-wise/Pair-wise) Jetson memory, latency, throughput, dynamic power, and dynamic energy for layer-wise and pair-wise training. S1 and S2 denote Stage 1 and Stage 2, respectively. Dynamic quantities subtract the immediately preceding idle baseline.

		Peak allocated memory (MiB)		Step latency (ms/batch)		Loop throughput (img/s)		Dynamic power (W)		Dynamic energy (mJ/image)	
Stage	**Window**	**CFF-only**	**CFF+MAE**	**CFF-only**	**CFF+MAE**	**CFF-only**	**CFF+MAE**	**CFF-only**	**CFF+MAE**	**CFF-only**	**CFF+MAE**
Layer-wise training											
S1	0	785.9	788.3	192.2	264.0	41.2	37.6	2.87	3.42	69.5	90.8
S1	1	783.9	786.3	234.6	310.2	42.0	38.1	3.74	4.30	88.9	112.9
S1	2	783.9	786.3	302.7	379.7	38.6	35.3	4.18	5.00	108.4	141.5
S1	3	783.9	786.3	362.7	452.5	35.9	32.6	4.92	5.43	136.6	166.4
S1	4	783.9	786.3	427.1	525.6	33.6	30.3	5.33	5.78	158.5	190.4
S1	5	783.9	786.3	491.9	598.2	31.3	28.4	5.71	6.14	181.8	216.1
S2	5	489.1	489.1	247.0	246.4	63.3	63.4	5.73	5.77	90.1	90.8
S2	4	741.0	741.0	319.6	319.1	55.3	55.4	6.36	6.47	114.7	116.4
S2	3	749.0	749.0	392.6	392.1	49.1	49.0	7.01	6.96	142.2	141.9
S2	2	757.0	757.0	465.5	464.9	44.2	44.3	7.44	7.44	167.9	167.4
S2	1	765.0	765.0	538.2	537.6	40.1	40.2	7.79	7.80	194.0	193.6
S2	0	782.9	782.9	611.7	610.8	36.7	36.9	8.10	8.17	220.3	221.0
Pair-wise training											
S1	0–1	1301.4	1303.8	347.5	438.4	34.5	31.4	4.27	4.78	123.6	151.8
S1	1–2	1300.0	1302.4	389.0	482.8	35.0	31.7	4.98	5.49	142.0	172.8
S1	2–3	1300.0	1302.4	452.8	554.7	32.7	29.7	5.51	6.01	168.2	201.8
S1	3–4	1300.0	1302.4	517.0	627.2	30.7	27.8	5.86	6.33	190.7	227.1
S1	4–5	1300.0	1302.4	586.8	699.9	28.7	26.1	6.36	6.61	221.2	252.9
S2	4–5	751.4	751.4	291.9	291.0	58.2	58.3	6.05	6.25	103.8	106.7
S2	3–4	999.9	999.9	391.7	376.1	49.2	50.6	6.30	6.72	128.0	132.5
S2	2–3	1007.9	1007.9	436.4	436.5	46.0	46.1	7.29	7.31	157.9	158.3
S2	1–2	1015.9	1015.9	508.7	509.0	41.7	41.8	7.69	7.69	183.9	183.9
S2	0–1	1033.8	1033.8	582.3	582.5	38.0	38.0	8.03	8.04	210.7	210.8

C.3 Details of Jetson efficiency results: CFF+MAE

Table 9 reports per-window Jetson efficiency results on ImageNet-1K at batch size 32, with separate columns for CFF-only and CFF+MAE. The reported metrics are peak allocated GPU memory, step latency, loop throughput, dynamic power, and dynamic energy. Dynamic power and energy are computed relative to the immediately preceding idle baseline.

C.3.1 CFF AND CFF+MAE ANALYSIS

Sequential branch backward releases the CFF graph before constructing the MAE graph. Adding MAE increases Stage 1 mean window peak memory from 784.3 to 786.6 MiB for layer-wise training and from 1300.3 to 1302.7 MiB for pair-wise training. The corresponding maximum window peaks are 785.9 versus 788.3 MiB and 1301.4 versus 1303.8 MiB. Stage 1 latency increases by 25.8% and 22.2%, and dynamic energy per batch by 23.4% and 19.0%. Pair-wise training retains activations for two adjacent blocks. In Stage 2 the MAE branch is inactive, and the measured costs of CFF-only and CFF+MAE are close. Stage 2 mean window peaks (714.0 and 961.8 MiB) are lower than the respective maxima (782.9 and 1033.8 MiB).

C.4 2-VIEW FULL BP AND SEQUENTIAL PREFIX REMATERIALIZATION

C.4.1 SEQUENTIAL PREFIX REMATERIALIZATION

Sequential Prefix Rematerialization (SPR) is an activation checkpointing scheme based on sequential prefix recomputation. It reduces activation memory by recomputing encoder prefixes for successive backward windows while preserving full-network supervised gradients. For each mini-batch, SPR processes backward windows from deep to shallow. It recomputes each window's input by running the preceding encoder prefix without autograd, then evaluates the active window with autograd and backpropagates the incoming gradient through it. The resulting input gradient is passed to the preceding window, and the current graph is released. Parameter gradients are accumulated across all windows before a single optimizer update.

SPR-Wk uses k encoder blocks per backward window. SPR-W1 and SPR-W2 match the window widths of layer-wise and pair-wise CFF, although CFF uses overlapping pairs. Both approaches evaluate prefixes without autograd and retain a small encoder backward graph at a time. Their objectives and gradient scopes differ: SPR propagates the final supervised loss across all windows, whereas Stage 1 of CFF/CFF+MAE updates one active window using local contrastive and optional reconstruction losses. With the same encoder, batch size, precision, and two-view inputs, SPR provides a baseline for comparing the memory, latency, and energy costs of local optimization with those of prefix recomputation. Per-batch costs represent one local-window update for CFF/CFF+MAE and one full-network update for SPR; complete-training costs also depend on the number of windows and training epochs.

C.4.2 EFFICIENCY RESULTS AND ANALYSIS

Tables 10 and 11 report two-view Full BP and SPR-W1/W2/W3 with the same six-block encoder, batch size, BF16 autocast, and measurement definitions. SPR preserves the final supervised loss and full-network gradient propagation. It processes non-overlapping backward windows from deep to shallow, recomputes each preceding prefix without autograd, and passes the current window's input gradient to the preceding window. Parameters are updated once after both views and all windows. The measured SPR implementation retains both views' active-window graphs before backward and uses no additional checkpointing inside a window. The first loss-evaluating pass retains the final window's graph; one-, two-, and three-block windows therefore require 21, 12, and 9 encoder-block forward evaluations per view. These counts describe encoder work, not latency ratios.

SPR analysis. SPR-W1/W2/W3 use 891.5/1396.8/1893.5 MiB, with step latencies of 1933.4/1342.3/1151.1 ms/batch and dynamic energies of 22.20/15.46/13.19 J/batch. Larger windows reduce repeated prefix execution while retaining more backward activations. Thus SPR-W1 has the lowest measured memory and highest latency and dynamic energy among the three SPR configurations. Relative to Full BP, all three reduce memory but increase latency and dynamic energy. This comparison concerns the tested SPR schedule and does not establish an optimal checkpointing trade-off.

Comparison scope. Stage 1 CFF and CFF+MAE restrict encoder gradients to the active local window; SPR passes the final-loss gradient through all windows. Layer-wise CFF and CFF+MAE have lower peak allocations, step latency, and dynamic energy per window update than SPR-W1; the pair-wise schedules have lower values than SPR-W2. However, a local-window update and a full-network update perform different work. The results characterize their update-level resource

Table 10: Expanded two-view Full BP and SPR memory, step latency, and training-loop throughput on Jetson. Bold rows reproduce the main-table values. Memory is in MiB; CPU RSS is the maximum sampled value.

Metric	Full BP	SPR-W1	SPR-W2	SPR-W3
Blocks per backward window	–	1	2	3
Number of backward windows	–	6	3	2
Number of views	2	2	2	2
Encoder tokens per view	196	196	196	196
Peak allocated GPU memory (MiB)	**3373.9**	**891.5**	**1396.8**	**1893.5**
Peak reserved GPU memory (MiB)	3560.0	1098.0	1620.0	2140.0
Peak sampled CPU RSS (MiB)	4845.1	2387.7	2907.1	3430.0
Mean step latency (ms/batch)	**962.1**	**1933.4**	**1342.3**	**1151.1**
Median step latency (ms/batch)	935.5	1909.7	1324.8	1130.7
Mean step latency (ms/image)	30.07	60.42	41.95	35.97
Median step latency (ms/image)	29.24	59.68	41.40	35.33
Loop throughput (images/s)	20.63	12.73	16.61	18.45
Loop throughput (batches/s)	0.645	0.398	0.519	0.576

Table 11: Expanded two-view Full BP and SPR power and energy on Jetson. Board denotes the total-board channel; CPU+GPU+CV denotes the rail aggregate. Run energy includes idle power, whereas dynamic energy uses the baseline-subtracted estimate defined above. Bold rows reproduce the main-table values.

Metric	Full BP	SPR-W1	SPR-W2	SPR-W3
Board idle power (W)	6.05	6.02	6.00	6.01
Board run power (W)	13.10	14.85	14.04	13.61
Board dynamic power (W)	**7.06**	**8.84**	**8.03**	**7.61**
Board run energy (mJ/batch)	20312	37318	27018	23615
Board dynamic energy (mJ/batch)	**10937**	**22199**	**15464**	**13193**
Board run energy (mJ/image)	634.7	1166.2	844.3	738.0
Board dynamic energy (mJ/image)	341.8	693.7	483.3	412.3
CPU+GPU+CV idle power (W)	1.27	1.27	1.27	1.27
CPU+GPU+CV run power (W)	4.65	5.51	5.11	4.90
CPU+GPU+CV dynamic power (W)	3.38	4.24	3.84	3.63
CPU+GPU+CV run energy (mJ/batch)	7213	13853	9841	8494
CPU+GPU+CV dynamic energy (mJ/batch)	5243	10656	7391	6290
CPU+GPU+CV run energy (mJ/image)	225.4	432.9	307.5	265.4
CPU+GPU+CV dynamic energy (mJ/image)	163.9	333.0	231.0	196.6

requirements and do not imply equal computation, equal accuracy, or lower complete-training cost. Stage 2 reverse-order fine-tuning propagates gradients through the frozen suffix and uses checkpointing; its gradient locality differs from Stage 1.

D H200 Efficiency Results and Training Cost

Two-view Full BP provides an architecture-matched efficiency baseline for CFF and CFF+MAE. All three use the same six-block ViT backbone with embedding dimension 192, 12 attention heads, 224×224 inputs, and 16×16 patches. For the cumulative-cost comparison, we use an 800-epoch two-view Full BP training schedule.

We profile CFF, CFF+MAE and two-view Full BP on eight H200 GPUs, with global batch size of 1,024, BF16 autocast, and the same attention backend and data-loading configuration. Stage 1 and Full BP process two augmented views per image, whereas Stage 2 uses one view. Their objectives and update scopes differ: CFF-based training updates an active window, while Full BP updates the full encoder. The comparison therefore measures the computing costs of these training procedures under matched backbone and input settings.

Table 12 shows that sequential branch execution preserves the low-memory footprint of CFF. In Stage 1, adding MAE increases peak allocated memory by only 2.36 MiB in both layer-wise and pair-wise training, less than 0.1% over CFF-only. Sequential execution reduces peak memory relative to joint execution by 19.1% and 16.2%, consistent with releasing each branch's graph before constructing the next. Compared with two-view Full BP, sequential CFF+MAE reduces peak memory by 77.0% and 61.6%; with distillation, the reductions remain 74.6% and 59.2%. Stage 2 memory is nearly unchanged across configurations, consistent with disabling reconstruction and distillation. These results establish a peak-memory advantage.

Detailed per-batch results are reported in Table 13 for layer-wise training and pair-wise training. Table 14 provides the Full BP reference. Dynamic energy is summed over the eight GPUs after subtracting the idle-power baseline; host-system energy is excluded.

We project cumulative training costs by assigning 350 Stage 1 epochs and 100 Stage 2 epochs to each of the six layer-wise windows or five overlapping pair-wise windows. Assuming constant per-epoch costs, CFF requires 473.18 MJ for layer-wise training and 505.47 MJ for pair-wise training; CFF+MAE requires 558.34 MJ and 595.69 MJ. The projected training times are 68.98 and 73.69 hours for CFF, and 81.40 and 86.84 hours for CFF+MAE.

The projected dynamic GPU energy and training time for 800 epochs of two-view Full BP are 423.00 MJ and 62.26 hours.

Table 12: H200 peak allocated GPU memory (MiB). Values are maxima across GPUs, profiled epochs, and windows where applicable.

		Layer-wise			Pair-wise		
Backward	**Stage**	**CFF**	**CFF+MAE**	**CFF+MAE +KD**	**CFF**	**CFF+MAE**	**CFF+MAE +KD**
Sequential	Stage 1	3047.72	3050.08	3375.76	5089.98	5092.34	5419.01
	Stage 2	2988.02	2988.02	2988.02	3987.65	3987.65	3987.65
Joint	Stage 1	3047.72	3769.41	4240.71	5089.98	6076.99	6713.47
	Stage 2	2988.02	2988.52	2988.52	3987.65	3989.06	3989.06

2-view Full BP
13267.81

Table 13: H200 per-window dynamic energy and latency for layer-wise and pair-wise training. Stage 1 uses two views, and Stage 2 uses one view. Mean denotes the arithmetic mean across windows within each stage.

Layer-wise training					**Pair-wise training**				
	CFF		**CFF+MAE**			**CFF**		**CFF+MAE**	
Window	**Dynamic energy (J/batch)**	**Latency (ms/batch)**	**Dynamic energy (J/batch)**	**Latency (ms/batch)**	**Window**	**Dynamic energy (J/batch)**	**Latency (ms/batch)**	**Dynamic energy (J/batch)**	**Latency (ms/batch)**
Stage 1 (two views)					**Stage 1 (two views)**				
L0	69.21	36.32	94.61	49.66	L0–L1	123.83	64.99	163.94	86.04
L1	90.59	47.54	120.24	63.10	L1–L2	150.37	78.92	188.66	99.01
L2	114.33	60.01	145.78	76.51	L2–L3	173.50	91.05	211.67	111.09
L3	135.30	71.01	169.63	89.02	L3–L4	195.65	102.68	237.64	124.72
L4	158.72	83.30	197.31	103.55	L4–L5	218.13	114.48	263.35	138.21
L5	183.63	96.37	221.50	116.25					
Mean	**125.30**	**65.76**	**158.18**	**83.02**	**Mean**	**172.30**	**90.42**	**213.05**	**111.81**
Stage 2 (one view)					**Stage 2 (one view)**				
L0	248.13	130.22	246.28	129.26	L0–L1	244.09	128.10	245.51	128.85
L1	226.98	119.12	218.88	114.88	L1–L2	222.76	116.91	226.07	118.65
L2	197.50	103.65	197.54	103.67	L2–L3	207.24	108.76	208.39	109.37
L3	184.98	97.08	179.55	94.23	L3–L4	185.99	97.61	185.83	97.53
L4	156.85	82.32	157.79	82.81	L4–L5	165.27	86.74	167.47	87.89
L5	136.79	71.79	141.29	74.15					
Mean	**191.87**	**100.70**	**190.22**	**99.83**	**Mean**	**205.07**	**107.63**	**206.65**	**108.46**

Table 14: H200 Dynamic energy and Latency results of 2-view Full BP.

Method	Dynamic energy (J/batch)	Latency (ms/batch)
2-view Full BP	422.67	223.95

E Relationship to Local Reconstruction Methods

Local reconstruction and gradient-isolated masked modeling have been explored previously. BIM (Luo et al., 2023) decomposes masked image modeling into block-wise local learning tasks, Pathak et al. (2022) combine local contrastive learning with reconstruction of Transformer-module input features, and LocalMIM (Wang et al., 2023a) applies reconstruction at multiple depths, including a gradient-isolated variant. Thus, neither local reconstruction nor gradient isolation alone constitutes the contribution of CFF+MAE.

Our focus is instead on using masked *pixel* reconstruction to strengthen CFF while preserving its fine-grained local-training footprint. CFF+MAE progressively trains one- or two-block windows and introduces separate input and reconstruction-loss masks, decoder transfer across windows, and sequential CFF/MAE execution. These design choices improve the local CFF objective while keeping the active backward graph tightly bounded and limiting peak training memory.

Our focus is the incremental benefit of masked pixel reconstruction over the CFF objective. Stage 1 uses progressively trained one-block or overlapping two-block windows, separate input and loss masks, decoder transfer, and sequential branch execution. The CFF-only and CFF+MAE comparisons directly assess this benefit. For classification, the Stage 1 contrastive objective uses class-based positives, while the reconstruction targets require no class labels.

In our experiments, CFF+MAE improves accuracy over CFF-only while retaining window-local backpropagation in Stage 1. Sequential branch execution limits peak memory by releasing the CFF graph before constructing the MAE graph.