BLOCK-CAUSAL ENCODING: CONVERTING LARGE MLLM ENCODERS FOR EFFICIENT ON-DEVICE INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large pretrained vision encoders provide the detailed representations needed by Multimodal Large Language Models (MLLMs), but high-resolution encoding is costly on mobile Neural Processing Units (NPUs). Beyond its quadratic computation, their full attention couples image regions across layers, making it difficult to complete encoding in small blocks suited to limited on-chip memory. We introduce BLoCE (Block-Causal Encoding), which preserves the original ViT layer structure and standard operators while restructuring attention for blockwise execution. It features block-causal attention allowing each image block to complete all encoder layers without waiting for future blocks and a bounded historical key-value cache supplying context for subsequent blocks. This organizes encoding around smaller intermediate features and attention inputs while retaining every output token. We further design Progressive Conversion to adapt the pretrained encoder to the new attention pattern through post-training, avoiding encoder pretraining from scratch. Based on AIMv2-300M, BLoCE achieves a comparable performance across 12 MLLM benchmarks. On a MediaTek Dimensity 9500 NPU, it reduces encoder latency at 1344×1344 from 2220 ms to 699 ms, a $3.2\times$ speedup. These results demonstrate the value of restructuring attention for efficient high-resolution inference while reusing large pretrained vision encoders.

1 INTRODUCTION

High-resolution images provide the fine details needed for document reading, chart understanding, and other multimodal tasks. These capabilities are increasingly useful in on-device Multimodal Large Language Models (MLLMs), which support local processing and offline access (Yao et al., 2025; Yu et al., 2025). Large pretrained vision encoders such as AIMv2 (Fini et al., 2025) and InternViT (Chen et al., 2024b) supply strong visual representations, but their encoding cost rises rapidly with image resolution. Reducing this cost while retaining the pretrained encoder is therefore an important step toward high-resolution on-device MLLMs.

Efficient execution on a mobile Neural Processing Unit (NPU) depends on more than operator support or arithmetic count. Intermediate tensors must be staged and reused within limited on-chip memory, and transfers to external memory can add substantial cost. Existing practice of transformer deployment on NPUs identifies intermediate-tensor size, cache residency, and memory copies as factors affecting efficiency (Apple Machine Learning Research, 2022). At high resolution, full-attention vision transformers face both a quadratic number of patch interactions and dependencies spanning the entire image at every layer. Even when their standard operators are supported, the resulting computation can be poorly suited to execution with small intermediate buffers.

Existing approaches address parts of this problem. Compact and hybrid encoders reduce model cost by changing the backbone (Mehta & Rastegari, 2022; Vasu et al., 2025) into hierarchical architectures; distillation trains a smaller student (Wu et al., 2022; Vasu et al., 2024); and token merging reduces the representations propagated through the network (Bolya et al., 2023). These works mostly focus on traditional small encoders, and do not provide a conversion of an existing large encoder that retains the layer structure and output size. Fused kernels such as FlashAttention reduce temporary storage and memory transfers for full attention (Dao et al., 2022) from the as-

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.

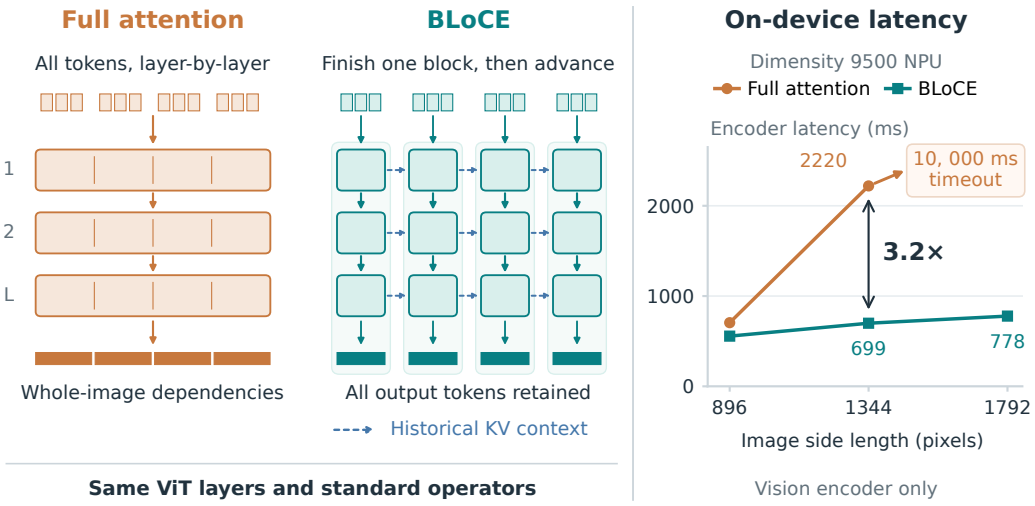

Figure 1: **Blockwise encoding and on-device latency. Left:** Full attention processes all tokens layer by layer; BLoCE completes all layers for one block before advancing. ViT layers, standard operators, and all output tokens are retained. **Right:** Vision-encoder latency on a Dimensity 9500 NPU. At 1344×1344, latency falls from 2220 to 699 ms ($3.2\times$). At 1792×1792, BLoCE takes 778 ms while the full attention encoder timed out during the 10,000 ms observation window.

pect of hardware-specific implementation, not model architectures. It does not change the global dependencies between patches spanning the entire image.

In this work, we start from a different aspect: *can we preserve a pretrained large encoder's layer structure and standard operators, while restructuring attention so that encoding can finish one block at a time?* Our starting point is block-causal attention: tokens interact bidirectionally within each block and attend to preceding blocks, but not future ones. This removes the dependency that prevents a block from passing through the full encoder on its own. The current block's intermediate features can then advance through all layers, with historical keys and values providing context for later blocks. Bounding this history keeps each attention input small as image resolution grows and further reduces the number of patch interactions.

We develop this approach as **Block-Causal Encoding (BLoCE)**. As is shown in Figure 1 (left), it reuses the original encoder's structure, while changing the attention pattern and inference execution schedule. For this target, we need to adapt representations learned with full attention to restricted cross-block context. We design *Progressive Conversion* as a post-training process that features attention-only tuning, a gradual transition to smaller blocks, and block-size sampling. This recovery procedure supports multiple inference block sizes without replacing the backbone or repeating large-scale encoder pretraining.

On 12 MLLM benchmarks, the converted AIMv2-300M encoder retains a mean score comparable to the source model and its full-attention continuation. The on-device speedup is shown in Figure 1 (right). At 1344×1344, BLoCE reduces encoder latency from 2220 to 699 ms. It completes 1792×1792 encoding in 778 ms, while the baseline does not finish within a 10,000 ms observation window.

Our contributions are:

- An encoder design that uses block-causal attention to enable blockwise execution with a bounded historical KV cache, while preserving standard ViT structure and operators.
- Progressive Conversion, which adapts pretrained full-attention encoders to this design through efficient tuning.
- Experiments showing comparable mean performance across 12 MLLM benchmarks and a $3.2\times$ encoder speedup on a mobile NPU at 1344×1344.

2 Related Work

Vision encoders for MLLMs. CLIP (Radford et al., 2021) and SigLIP (Zhai et al., 2023; Tschannen et al., 2025) provide widely used pretrained visual representations. InternViT (Chen et al., 2024b) scales encoder capacity, while NaViT (Dehghani et al., 2023) and Qwen-VL series (Wang et al., 2024b; Bai et al., 2025) support variable-resolution inputs. AIM and AIMv2 (El-Nouby et al., 2024; Fini et al., 2025) use autoregressive pretraining, and OpenVision (Li et al., 2025; Liu et al., 2025) studies caption-based training. Efficient alternatives such as ConvLLaVA (Ge et al., 2024), ViTamin (Chen et al., 2024a), and FastVLM (Vasu et al., 2025) change the encoder design. Despite their strong performance, these encoders primarily target cloud inference with abundant GPU memory and flexible kernels. Their requirements for large memory or flexible compilation support are usually poorly matched to mobile NPUs. BLoCE instead preserves encoder capacity while restructuring attention and execution around a bounded memory budget.

Efficient vision encoders. Hybrid architectures combine convolution and attention (Mehta & Rastegari, 2022; Li et al., 2022; Vasu et al., 2023; Wang et al., 2024a), while efficient attention designs like linear attention or additive attention reduce the cost of token interactions (Liu et al., 2023a; Cai et al., 2023; Shaker et al., 2023; Zhang et al., 2021). Distillation transfers representations to smaller new models (Wu et al., 2022; Vasu et al., 2024). Token pruning and merging methods remove or compress representations in the encoder (Kong et al., 2022; Xu et al., 2022; Chen et al., 2023; Bolya et al., 2023). These approaches either require substantial retraining or sacrifice the model capacity. Compared with these works, BLoCE preserves model width, depth, and output size and allows transferring from existing pretrained large encoders.

Attention kernels and blockwise execution. FlashAttention (Dao et al., 2022) optimizes data movement for a specific chosen attention patterns through tiling, based on a specific hardware. BLoCE has a different target, which is to change the attention pattern to enable cross-layer blockwise execution and bound historical context. Chunked LLM prefill in SARATHI (Agrawal et al., 2023) changes the serving schedule of an already causal model. Fast-dLLM v2 (Wu et al., 2025) uses bidirectional attention within blocks and causal dependencies across blocks for block-diffusion language generation. XTRA (Amrani et al., 2025) adopts a block-by-block image reconstruction target to reduce the training cost and model size of token-by-token autoregressive pretraining. BLoCE studies conversion of full-attention vision encoders, where restricting access to future image patches changes representations used for visual perception.

3 Method

BLoCE preserves the standard ViT layer structure and operators, while restructuring attention for blockwise execution. Block-causal attention makes it possible to complete all layers for one block before processing the next. A bounded historical KV cache controls the context carried across blocks, and Progressive Conversion adapts the pretrained representations to the new pattern.

3.1 Preliminaries and Design Objective

A ViT embeds N image patches as $Z_0 \in \mathbb{R}^{N \times D}$ and processes them through L Transformer layers. With H heads and head width $d = D/H$, each attention head computes

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V. \tag{1}$$

Full attention evaluates N^2 query–key pairs per head and layer. A fused implementation need not store the full $N \times N$ score matrix, but each query still depends on keys and values from across the image. Consequently, tiling a layer does not remove the dependence of a block's later layers on other blocks' intermediate features.

Our objective is to let a block complete the encoder using its own features and retained historical keys and values. This exposes smaller intermediate features and attention inputs to the runtime, providing an opportunity for reuse within limited on-chip memory. It also reduces attention com-

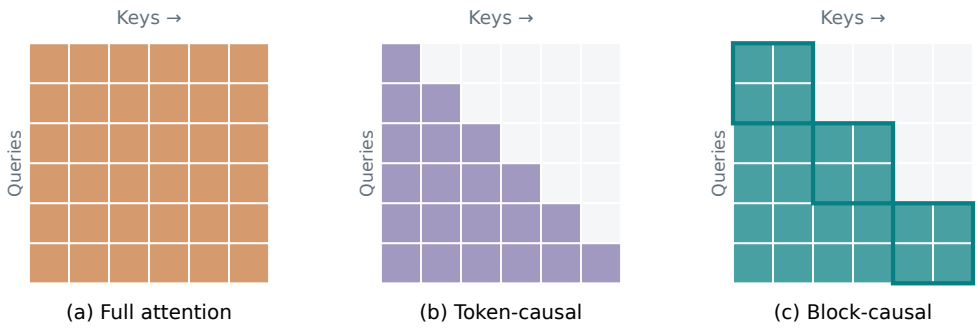

Figure 2: **Attention patterns for image tokens.** Rows are queries and columns are keys; filled cells indicate allowed interactions. Block-causal attention retains bidirectional interactions within each block while excluding future blocks.

putation. Actual latency depends on the resulting matrix sizes, repeated layer invocations, and KV management, which we evaluate on device.

3.2 Block-Causal Attention

BLoCE retains the encoder's layer composition, depth, width, parameter shapes, and output token count. Each layer continues to use the original projections, softmax attention, feed-forward network, normalization, and residual connections. It partitions the input sequence into $K = \lceil N/B \rceil$ consecutive blocks of at most B tokens, retaining the base preprocessing order and positional coordinates. A block is a segment of this sequence and need not be a square image region.

Let $b(i) = \lceil i/B \rceil$ denote the block containing token i. The allowed attention pairs are defined by

$$M_{ij}^{\mathrm{bc}} = \begin{cases} 1, & b(j) \le b(i), \\ 0, & b(j) > b(i). \end{cases} \tag{2}$$

As is shown in Figure 2, tokens interact bidirectionally within their block and attend to earlier blocks. Thus, a block's representation at every layer is independent of future blocks. Once earlier keys and values are available, the current block can pass through all L layers without being revisited.

Block causality alone does not make attention linear in N. With equal-sized blocks and full history, the number of pairs is $\sum_{k=1}^{K} B(kB) = (N^2 + NB)/2$. The history budget introduced next limits both the retained context and the work in subsequent blocks.

3.3 Blockwise Execution with Historical KV

During inference on the NPU, BLoCE adopts a block-by-block process illustrated in Figure 1. The more specific execution process is shown in Figure 3.

Completing a block through all layers. At layer ℓ, the current block k produces $Q_{\ell,k}, K_{\ell,k}, V_{\ell,k}$. Let $K_{\ell,k-1}^{\mathrm{hist}}, V_{\ell,k-1}^{\mathrm{hist}}$ be the historical KV cache after block $k-1$, initially empty. Attention uses

$$A_{\ell,k} = \mathrm{Attn}\big(Q_{\ell,k}, [K_{\ell,k-1}^{\mathrm{hist}}; K_{\ell,k}], [V_{\ell,k-1}^{\mathrm{hist}}; V_{\ell,k}]\big)\,, \tag{3}$$

where $[\,;\,]$ concatenates tokens. The other layer operations follow the base encoder. The block passes through all layers, and its output is appended to the completed patch representations. With full history, this schedule exactly reorders block-causal inference.

Bounding the historical KV cache. A fixed KV budget $C \ge B$ limits the history to C entries per layer and head between blocks. After a block completes all layers, we protect the most recent $W = B$ entries and select $C - W$ older entries if the cache exceeds its budget. Each entry contains a key and its corresponding value. During a block, each layer therefore uses at most $C + B$ entries;

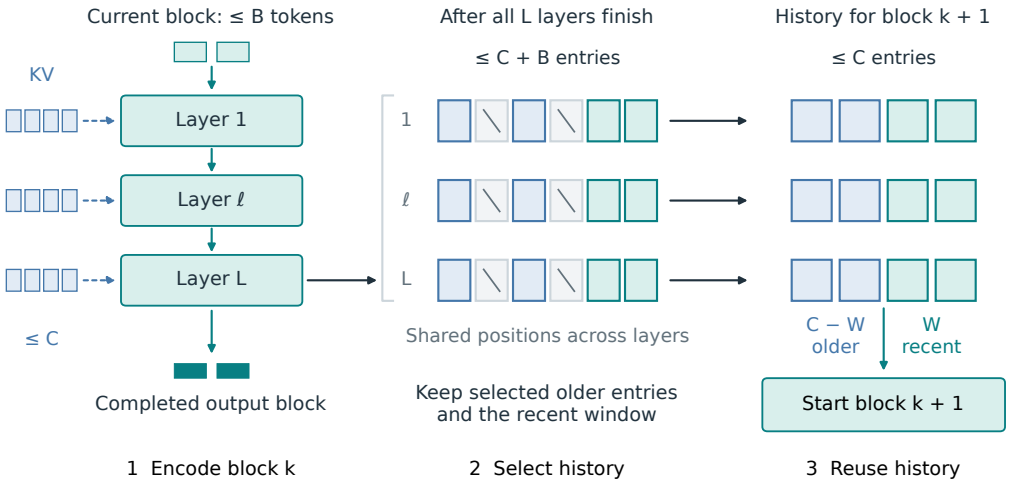

Figure 3: **Blockwise encoding and historical KV selection.** Each block completes all L layers using its own tokens and the retained history. Selection then limits the historical KV cache to C entries per layer and head for the next block. The recent window has $W = B$ entries; older selected entries occupy the remaining slots. Selected positions are shared across layers. Completed output tokens are retained separately and never evicted.

after selection, it retains at most C. This operation compresses only the KV history and preserves every output token.

We adapt the importance-redundancy principle of R-KV (Cai et al., 2025) to select older entries. For each head, attention received from current-block queries estimates importance, averaged over queries and layers and smoothed by one-dimensional max pooling. Similarity between first-layer keys estimates redundancy. We rank historical entries by

$$s_{h,i} = \lambda I_{h,i} - (1 - \lambda) R_{h,i}, \tag{4}$$

and retain the highest-scoring $C - W$ positions per head, shared across layers. Details appear in Section A.5. The cache bound holds independently of this scoring rule. When $B = C = W$, only the most recent block is retained directly, although its representations can carry context from earlier blocks.

Computation and storage. Let $n_k \leq B$ be the number of tokens in block k. The attention pair count per head and layer satisfies

$$\sum_{k=1}^{K} n_k \left(\min\left(C, \sum_{j<k} n_j \right) + n_k \right) \leq N(C + B). \tag{5}$$

Thus, attention computation is linear in N for fixed B and C, including a partial final block. The historical KV cache contains at most $2LCD$ elements between blocks, or $2L(C + B)D$ together with current-block keys and values before eviction. These bounds concern attention and KV state; completed outputs still require $O(ND)$ storage. Projection and feed-forward computation also remain linear in N, with matrix sizes determined by the block size.

Setting $C \geq N$ disables eviction and recovers full-history block-causal inference. The single-block mode $B = N$ recovers the full-attention pattern and does not apply the fixed-history rule. Section B derives the bounds and compares logical tensor sizes at the main configuration.

3.4 Progressive Conversion

Progressive Conversion initializes from the pretrained full-attention checkpoint. Training uses the standard layer-wise execution order with a block-causal mask in each attention layer. For the main

MLLM setting, the projector and language model remain frozen. We use three strategies for this adaptation:

Attention-only tuning. The motivation is that BLoCE changes the attention pattern while retaining the rest of the encoder's layer structure. We therefore update only the query, key, value, and attention output projections, concentrating adaptation on the modules whose attention context changes. Patch embeddings, feed-forward networks, normalization, and all other parameters remain frozen. Compared with updating all encoder weights, this reduces training cost and slightly improves performance in our experiments.

Progressive masking. The source encoder is pretrained with full attention, whereas BLoCE restricts access to future blocks. To ease this transition, we start with full attention ($B = N$) and reduce the nominal block size at equal intervals during training. This gives the pretrained representations time to adapt as cross-block context becomes more restricted, rather than imposing the final restriction from the first update.

Block-size sampling. For nominal block size b, we sample the actual training block size from $\{b, 2b, 4b, 8b\}$. This exposes the encoder to different amounts of within-block and historical context during conversion. The resulting checkpoint can support several inference configurations without retraining, allowing deployment choices based on measured quality and NPU latency. Training settings are given in Sections A.2 and 4.1.

4 Experiments

We evaluate whether conversion retains the source encoder's benchmark performance, how blockwise execution affects mobile NPU latency, and which training settings are suitable. Additional experiments include transfer to other models and vision tasks.

4.1 Experimental Setup

Models and conversion. The main experiments use AIMv2-300M (Fini et al., 2025) in AIMv2-Qwen3-4B (Jin et al., 2025). Conversion uses 5.7M post-training samples with the projector and language model frozen. It takes approximately 60 hours on 32 H20 GPUs. This is a post-training alternative to repeating large-scale MLLM encoder pretraining: for example, AIMv2 reports pretraining on 12 billion image-text samples (Fini et al., 2025). The training recipe is detailed in Section A.2. Unless specified otherwise, inference uses $B = 128$, a fixed KV budget $C = 256$, and a protected recent window $W = B$.

Comparison settings. **AIMv2** is the source model; **FA-tuned** continues training with full attention on the same data. **BC-direct** applies block-causal attention mask without conversion training. **BC-conv** uses Progressive Conversion and retains full history at inference; **BC-FT** tunes all encoder weights with a fixed block size; **BLoCE** uses the same conversion with bounded history. **BC-LP** trains only a classification head. The full definitions appear in Table 14.

Benchmarks and device measurements. We report the results over 12 MLLM benchmarks covering document and chart understanding, OCR, visual reasoning, and general multimodal tasks; benchmark names and references are given in Section A.3. Encoder latency is measured on a MediaTek Dimensity 9500 NPU. Both encoders use dynamic quantization and the vendor-optimized attention operator through the official SDK (MediaTek Inc., 2026). Timings include encoder execution and exclude LLM prefill and decoding. Section C states the measurement scope.

4.2 Retaining MLLM Benchmark Performance

In Table 1, directly applying block-causal attention reduces the mean from 73.9 to 70.1, showing that restricting attention at inference can harm a pretrained full-attention encoder. Progressive Conversion recovers it to 74.1 with full history, and BLoCE retains 74.0 after bounding the KV history. To separate conversion from the effect of additional training data, we also continue full-attention

Table 1: **Results on 12 MLLM benchmarks.** Bold marks the best score per column; gray highlights BLoCE. Both panels share the overall mean. Benchmark abbreviations are defined in Section A.3.

Setting	CC	CQA	C-OCR	DVQA	Hallu.	IVQA	Mean
AIMv2	**61.4**	87.2	86.0	94.7	60.7	78.5	73.9
FA-tuned	61.2	**87.3**	87.1	**94.9**	60.3	**79.1**	**74.2**
BC-direct	58.1	74.3	82.7	91.2	58.7	72.4	70.1
BC-conv	61.0	86.9	**87.2**	93.9	60.4	78.1	74.1
BLoCE	60.6	86.5	86.7	94.2	**61.2**	78.9	74.0

Setting	MMB	MMMU	MMT	OCRB	R-QA	SEED2+	Mean
AIMv2	80.2	52.0	64.8	82.7	70.0	68.6	73.9
FA-tuned	**81.0**	52.2	**65.4**	84.0	70.3	68.1	**74.2**
BC-direct	78.5	52.1	62.8	78.9	64.8	67.1	70.1
BC-conv	80.8	52.0	64.7	**84.1**	70.8	**68.7**	74.1
BLoCE	79.6	**52.7**	63.9	83.7	**71.0**	68.3	74.0

Table 2: **BLoCE encoder latency (ms) for different block sizes B on the Dimensity 9500 NPU.** $B = N$ uses full attention without KV eviction. Bold marks the fastest completed run per row; T.O. denotes a 10,000 ms timeout.

Resolution	Full attn. ($B = N$)	$B = 64$	$B = 128$	$B = 256$	Selected B
448×448	**111**	527	403	341	N
672×672	**342**	581	475	447	N
896×896	705	625	556	**539**	256
1344×1344	2220	724	**699**	708	128
1792×1792	T.O.	801	**778**	791	128

training on the same data (FA-tuned), reaching 74.2. Thus, conversion largely recovers the source encoder's benchmark performance under restricted attention, while the full-attention continuation remains a strong control.

Recovery varies across tasks. Relative to AIMv2, BLoCE improves OCRBench from 82.7 to 83.7 and RealWorldQA from 70.0 to 71.0, while ChartQA decreases from 87.2 to 86.5 and DocVQA from 94.7 to 94.2. The comparable aggregate reflects a balance of benchmark-level gains and losses.

4.3 On-Device Latency

Latency advantage at high resolutions. As is shown in Figure 1 (right), BLoCE's latency advantage grows at higher resolutions. At 1344×1344, the default setting takes 699 ms versus 2220 ms for full attention, a $3.2\times$ speedup. At 1792×1792, the same configuration completes encoding in 778 ms, while full attention does not finish within the 10,000 ms observation window. These gains accompany both fewer attention interactions and smaller per-block tensor ranges. Section B gives the detailed theoretical analysis. We report and discuss BLoCE under smaller resolutions next.

The inference block size. Table 2 shows that the fastest execution mode depends on input resolution. Full attention is fastest at 448×448 and 672×672, whereas blockwise execution with $B = 256$ is fastest at 896×896, and $B = 128$ is fastest at the two larger resolutions. At small resolutions, smaller matrix operations, repeated layer invocations, and KV management can outweigh the reduction in attention computation.

BLoCE supports full-attention inference by setting $B = N$, placing all tokens in a single block without KV eviction. The same converted checkpoint can therefore use full attention at small resolutions and blockwise execution at larger resolutions, selecting the fastest execution mode at each tested resolution without retraining. Block-size sampling during Progressive Conversion supports this flexibility; benchmark performance across inference block sizes is evaluated in Table 3.

Table 3: **Conversion methods and inference block size.** Means over 12 benchmarks; column maxima are bold. The final three rows ablate Progressive Conversion. Training uses 32 H20 GPUs.

Setting	$B = 64$	$B = 128$	$B = N$	Time (h)
BC-direct	68.7	70.1	73.9	–
BC-FT	72.5	72.0	72.4	74.9
BC-conv	**74.2**	**74.1**	**74.2**	60.4
BLoCE	73.8	74.0	**74.2**	60.4
No progression	73.1	73.2	73.2	60.4
All weights	73.3	73.1	73.3	74.9
Fixed B in training	73.7	73.5	73.6	60.4

Table 4: **Token- vs. block-causal attention.** $B = 1$ is token-causal. BC-FT tunes all weights; † tunes attention only. Bold marks column maxima.

B	BC-direct	BC-FT	BC-FT†
1	62.7	67.3	67.6
128	**70.1**	**72.0**	**72.1**

Table 5: **Conversion data scale.** Mean over 12 benchmarks (best in bold). AIMv2: 73.9; BC-direct: 70.1.

Data fraction	Samples	Mean
20%	1.14M	72.7
50%	2.85M	73.4
100%	5.70M	**74.0**

4.4 Conversion Ablations

Conversion methods and inference block size. Table 3 compares directly applying block-causal attention, fixed-block finetuning, and Progressive Conversion across three inference block sizes. At $B = 128$, BC-direct gives 70.1 and all-weight finetuning with a fixed training block size (BC-FT) gives 72.0. Progressive Conversion reaches 74.1 with full history and 74.0 with the bounded history used by BLoCE. Across $B \in \{64, 128, N\}$, BC-conv reaches 74.1 to 74.2 and BLoCE reaches 73.8 to 74.2, supporting deployment choices across the reported block sizes.

Token-causal versus block-causal attention. Setting $B = 1$ gives standard token-causal attention, where each token attends only to itself and preceding tokens. In Table 4, directly applying this mask yields a mean of 62.7, compared with 70.1 for $B = 128$. All-weight and attention-only tuning narrow this gap, but token-causal attention still trails $B = 128$ by 4.7 and 4.5 points, respectively. This persistent gap supports retaining bidirectional interactions within blocks when converting pretrained vision encoders for blockwise execution.

Contributions of the training components. The final three rows of Table 3 remove one component at a time. At $B = 128$, removing progressive masking lowers the mean by 0.8 points, tuning all weights by 0.9, and fixing the training block size by 0.5. Without progression, the drop is 0.7–0.8 points across finite block sizes and 1.0 in single-block mode. Attention-only tuning also reduces training time from 74.9 to 60.4 hours. Together, these results support gradual context restriction, attention-only adaptation, and block-size sampling.

Conversion data requirements. Table 5 varies the fraction of the 5.7M conversion set. Using 20% (1.14M samples) reaches 72.7, recovering much of the drop from AIMv2's 73.9 to BC-direct's 70.1. With 50% (2.85M samples), the mean reaches 73.4; the full set yields 74.0. Smaller subsets therefore recover much of the source model's performance, while the full conversion set reaches a comparable aggregate score. This remains far fewer samples than the 12B used for AIMv2 pretraining.

4.5 Transfer Beyond the Main MLLM

Standard vision tasks. Table 6 evaluates classification with DeiT-Base, and detection and segmentation with ViTDet and Mask R-CNN. Direct block-causal conversion degrades all three tasks. BLoCE recovers ImageNet top-1 accuracy from 76.8% to the full-attention baseline's 82.9%. Dense

Table 6: **Vision-task transfer.** ImageNet top-1 (%) and COCO AP. Bold marks column maxima.

Setting	IN-1K	Box AP	Mask AP
Full attention	**82.9**	**51.6**	**45.9**
BC-direct	76.8	40.9	37.8
BLoCE	**82.9**	50.5	44.6

Table 7: **InternViT-6B transfer.** ImageNet top-1 (%).

Setting	Top-1 (%)
InternViT-6B	**88.2**
BC-direct	85.8
BC-LP	87.4

Table 8: **Vision encoders in LLaVA-1.5 at** 336×336**.** Top: published results with different training recipes. Bottom: AIMv2-300M and BLoCE with Llama-3-8B. Bold marks the best reported score per column. VQA abbreviations follow Section A.3; TVQA is TextVQA. Dashes denote unreported results.

Encoder	CQA	DVQA	IVQA	MME	TVQA
CLIP	18.5	25.6	21.8	1481	47.5
SigLIP	14.4	16.9	20.7	1416	44.1
OpenVision	15.7	–	–	**1525**	61.2
OpenVision2	14.5	–	–	1486	**63.0**
AIMv2	**19.2**	26.6	22.8	1472	53.6
BLoCE	18.8	**27.2**	**23.5**	1433	56.2

prediction retains a gap: detection reaches 50.5 box AP versus 51.6, and segmentation reaches 44.6 mask AP versus 45.9. These smaller encoders test generalization beyond BLoCE's large-MLLM-encoder setting. Task-specific training details appear in Section E.

A huge vision encoder. Applying block-causal attention to InternViT-6B (Chen et al., 2024b) gives 85.8% ImageNet top-1 accuracy, compared with 88.2% for the source model (Table 7). Training only the classification head (BC-LP) raises it to 87.4%. This tests transfer of the attention pattern to a huge encoder. The linear-probing recipe is given in Table 16.

Another MLLM setup. We evaluate BLoCE in a LLaVA-1.5 (Liu et al., 2024a) setup with Llama-3-8B (Grattafiori et al., 2024), using AIMv2-300M at 336×336. Table 8 includes published results for CLIP (Radford et al., 2021), SigLIP (Zhai et al., 2023), OpenVision (Li et al., 2025), and OpenVision2 (Liu et al., 2025). Relative to AIMv2, BLoCE scores higher on DocVQA, InfoVQA, and TextVQA, and lower on ChartQA and MME (Fu et al., 2023). BLoCE also exceeds the reported CLIP and SigLIP scores on all four VQA benchmarks, while the OpenVision variants score higher on MME and TextVQA. These results show that the converted BLoCE encoder retains competitive quality as an encoder for other MLLMs.

5 Conclusion

BLoCE converts large pretrained vision encoders for blockwise inference while retaining their ViT layer structure, standard operators, and output size. Block-causal attention allows each block to complete all layers, historical KV selection controls the context carried forward, and Progressive Conversion adapts pretrained representations to these changes. On the evaluated AIMv2-based MLLM, BLoCE retains a comparable mean across 12 benchmarks and reduces encoder latency by $3.2\times$ at 1344×1344 on a Dimensity 9500 NPU. The results support restructuring attention and execution as a practical way to reuse large vision encoders under mobile inference constraints.

Limitations. Due to the significant engineering cost and effort in deploying customized models to commercial chips, the current hardware evaluation covers one mobile NPU. Block causality makes representations depend on the chosen token order, so tasks like detection and segmentation may retain accuracy gaps. Conversion requires post-training a large vision encoder, which is expensive for academic laboratories to carry out.

Ethics statement

More efficient vision encoding can support local multimodal applications with lower latency and less dependence on remote processing. Privacy and safe use also depend on data handling and the application in which the model is deployed.

AI use statement

Generative AI tools were used to assist with language editing, manuscript revision, and figure preparation. The authors are responsible for the scientific content and the final manuscript.

Reproducibility statement

The model and conversion settings are described in Sections 3.4 and 4.1, and conversion ablations are reported in Section 4.4. Additional training details, complexity analysis, and device measurement scope are provided in Sections A to C.

References

Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, and Ramachandran Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369*, 2023.

Elad Amrani, Leonid Karlinsky, and Alex Bronstein. Sample- and parameter-efficient autoregressive image models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30127–30136, 2025.

Apple Machine Learning Research. Deploying transformers on the Apple Neural Engine, 2022. URL https://machinelearning.apple.com/research/neural-engine-transformers. Accessed: 2026-09-22.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *International Conference on Learning Representations*, 2023.

Han Cai, Junyan Li, Muyang Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17256–17267, 2023.

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-kv: Redundancy-aware kv cache compression for reasoning models. In *Advances in Neural Information Processing Systems*, 2025.

Jieneng Chen, Qihang Yu, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Vitamin: Designing scalable vision models in the vision-language era. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12954–12966, 2024a.

Xuanyao Chen, Zhijian Song, Yujun Liu, Han Cai, Haotian Yan, Ji Lin, and Song Han. Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2061–2070, 2023.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24185–24198, 2024b.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. Patch n'pack: Navit, a vision transformer for any aspect ratio and resolution. *Advances in Neural Information Processing Systems*, 36:2252–2274, 2023.

Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Ángel Bautista, Vaishaal Shankar, Alexander T Toshev, Joshua M. Susskind, and Armand Joulin. Scalable pre-training of large autoregressive image models. In *International Conference on Machine Learning*, pp. 12371–12384, 2024.

Enrico Fini, Mustafa Shukor, Xiujun Li, Philipp Dufter, Michal Klein, David Haldimann, Sai Aitharaju, Victor G Turrisi da Costa, Louis Béthune, Zhe Gan, et al. Multimodal autoregressive pre-training of large vision encoders. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 9641–9654, 2025.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

Chunjiang Ge, Sijie Cheng, Ziming Wang, Jiale Yuan, Yuan Gao, Jun Song, Shiji Song, Gao Huang, and Bo Zheng. Convllava: Hierarchical backbones as visual encoder for large multimodal models. *arXiv preprint arXiv:2405.15738*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14375–14385, 2024.

Zhiwei Jin, Xiaohui Song, Nan Wang, Yafei Liu, Chao Li, Xin Li, Ruichen Wang, Zhihao Li, Qi Qi, Long Cheng, et al. Andesvl technical report: An efficient mobile-side multimodal large language model. *arXiv preprint arXiv:2510.11496*, 2025.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, Minghai Zhan, and Yanzhi Wang. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *European Conference on Computer Vision*, pp. 620–640, 2022.

Bohao Li, Yuying Ge, Yi Chen, Yixiao Ge, Ruimao Zhang, and Ying Shan. Seed-bench-2-plus: Benchmarking multimodal large language models with text-rich visual comprehension. *arXiv preprint arXiv:2404.16790*, 2024.

Xianhang Li, Yanqing Liu, Haoqin Tu, and Cihang Xie. Openvision: A fully-open, cost-effective family of advanced vision encoders for multimodal learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3977–3987, 2025.

Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. In *Advances in Neural Information Processing Systems*, volume 35, pp. 12934–12949, 2022.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 26296–26306, 2024a.

Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14420–14430, 2023a.

Yanqing Liu, Xianhang Li, Letian Zhang, Zirui Wang, Zeyu Zheng, Yuyin Zhou, and Cihang Xie. Openvision 2: A family of generative pretrained visual encoders for multimodal learning. *arXiv preprint arXiv:2509.01644*, 2025.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, pp. 216–233, 2024b.

Yuliang Liu, Zhang Li, Hongliang Li, Wenwen Yu, Mingxin Huang, Dezhi Peng, Mingyu Liu, Mingrui Chen, Chunyuan Li, Lianwen Jin, et al. On the hidden mystery of ocr in large multimodal models. *arXiv preprint arXiv:2305.07895*, 2023b.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics*, pp. 2263–2279, 2022.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *IEEE Winter Conference on Applications of Computer Vision*, pp. 2200–2209, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1697–1706, 2022.

MediaTek Inc. MediaTek's Ecosystem for AI Development, 2026. URL https://neuropilot.mediatek.com/. Accessed: 2026-01-27.

Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763, 2021.

Abdelrahman Shaker, Muhammad Maaz, Hanoona Rasheed, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Swiftformer: Efficient additive attention for transformer-based real-time mobile vision applications. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17425–17436, 2023.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5785–5795, 2023.

Pavan Kumar Anasosalu Vasu, Hadi Pouransari, Fartash Faghri, and Oncel Tuzel. Mobileclip: Fast image-text models through multi-modal reinforced training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15963–15974, 2024.

Pavan Kumar Anasosalu Vasu, Fartash Faghri, Chun-Liang Li, Cem Koc, Nate True, Albert Antony, Gokula Santhanam, James Gabriel, Peter Grasch, Oncel Tuzel, et al. Fastvlm: Efficient vision encoding for vision language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19769–19780, 2025.

Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15909–15920, 2024a.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm. *arXiv preprint arXiv:2509.26328*, 2025.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European Conference on Computer Vision*, pp. 68–85, 2022.

x.ai. Grok-1.5 Vision Preview, 2026. URL https://x.ai/blog/grok-1.5v. Accessed: 2026-01-27.

Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 2964–2972, 2022.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Chi Chen, Haoyu Li, Weilin Zhao, et al. Efficient gpt-4v level multimodal large language model for deployment on edge devices. *Nature Communications*, 16(1):5509, 2025.

Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, et al. Mmt-bench: a comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 57116–57198, 2024.

Tianyu Yu, Zefan Wang, Chongyi Wang, Fuwei Huang, Wenshuo Ma, Zhihui He, Tianchi Cai, Weize Chen, Yuxiang Huang, Yuanqian Zhao, et al. Minicpm-v 4.5: Cooking efficient mllms via architecture, data, and training recipe. *arXiv preprint arXiv:2509.18154*, 2025.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Ming Yin, Botao Yu, Ge Zhang, et al. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*, 2024.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986, 2023.

Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2998–3008, 2021.

Table 9: **Configuration shared by AIMv2 and BLoCE.**

Configuration	Value
Parameters	0.3B
Hidden dimension	1024
Encoder layers	24
Intermediate dimension	2816
Positional interpolation	2D
Attention heads	8
Patch size	14
Learnable position embedding	Yes
Linear bias	No

Table 10: **Main MLLM conversion settings.**

Configuration	Value
Precision	bf16
Batch size	32
Learning rate	10^{-5}
Training steps	179,200
Maximum sequence length	16,384
Learning-rate schedule	Cosine decay
Training samples	5.7M
Hardware	32 H20 GPUs

A Additional Implementation Details

A.1 AIMv2-Based BLoCE Configuration

Table 9 gives the configuration shared by the source AIMv2 encoder and its BLoCE conversion. Conversion changes the attention pattern and parameter values while preserving the layer structure and parameter shapes. KV eviction only limits context for later blocks during inference; every completed output token is retained.

A.2 Conversion Training

The main conversion initializes the vision encoder from AIMv2-Qwen3-4B and uses 5.7M samples from its post-training data. The projector and language model remain frozen. Attention-only tuning updates the Q/K/V and attention output projections; the objective is the MLLM cross-entropy loss. Training uses PyTorch and DeepSpeed on 32 H20 GPUs and takes approximately 60 hours (1920 GPU-hours).

Starting from full attention, we divide the training steps equally into stages with the nominal block sizes decrease through 2048, 512, 128, 64, and 32. Within a stage with nominal size b, training samples block sizes from $\{b, 2b, 4b, 8b\}$. Cache eviction is applied only at inference. Table 10 lists the remaining training settings.

A.3 MLLM Benchmarks

We evaluate CCBench (Liu et al., 2024b), ChartQA (Masry et al., 2022), ChineseOCRBench, DocVQA (Mathew et al., 2021), HallusionBench (Guan et al., 2024), InfoVQA (Mathew et al., 2022), MMBench (Liu et al., 2024b), MMMU (Yue et al., 2024), MMT-Bench (Ying et al., 2024), OCRBench (Liu et al., 2023b), RealWorldQA (x.ai, 2026), and SEEDBench2+ (Li et al., 2024). Table abbreviations are CC (CCBench), CQA (ChartQA), C-OCR (ChineseOCRBench), DVQA (DocVQA), Hallu. (HallusionBench), IVQA (InfoVQA), MMB (MMBench), MMT (MMT-Bench), OCRB (OCRBench), R-QA (RealWorldQA), and SEED2+ (SEEDBench2+). The reported mean

aggregates these 12 scores; it is not an accuracy metric for a single task. Benchmark-specific values and means retain the reported rounding.

A.4 BLOCKWISE INFERENCE

The pseudocode describes a single image with block size B, fixed KV budget $C \geq B$, and recent window $W = B$. The input follows the base encoder's patch ordering and positional coordinates. Each block completes all layers before history selection. The layer operation includes the original projections, normalization, residual connections, and feed-forward network. Output tokens are collected separately from the historical KV cache.

Listing 1: BLoCE inference with historical KV selection.

```
cache = [empty_cache() for _ in layers]
outputs = []
for block in image_blocks:  # At most B tokens per block.
    x = patch_embed_with_original_positions(block)
    attention_maps = []
    for ell, layer in enumerate(layers):
        x, cache[ell], attn = layer.forward_block(
            x, cache[ell]
        )
        attention_maps.append(attn)
    outputs.append(final_norm(x))

    if cache_length(cache[0]) > C:
        keep = select_history_per_head(
            attention_maps, cache[0].keys,
            count=C - W, exclude_recent=W
        )
        for ell in range(len(layers)):
            cache[ell] = gather_history_and_recent(
                cache[ell], keep, recent_count=W
            )

return concatenate(outputs)
```

The cache grows to at most $C + B$ entries before selection and returns to at most C afterwards. A partial final block has fewer than B tokens, so its protected recent window may include entries from the preceding block. When $C = W = B$, the selected older-history set is empty. The $B = N$ full-attention mode is a separate path without this eviction rule.

A.5 HISTORICAL KV SELECTION

For each attention head, importance is the attention received by historical entries from current-block queries, averaged over queries and encoder layers, followed by one-dimensional max pooling with kernel size 7.

For redundancy, let $k_{h,i}$ be the first-layer key of older candidate i in head h, excluding the W protected recent entries. We index the M candidates by their current cache positions, $i \in \{0, \ldots, M-1\}$. We normalize the keys and compute pairwise cosine similarities:

$$\widehat{k}_{h,i} = \frac{k_{h,i}}{\|k_{h,i}\|_2 + 10^{-8}}, \qquad S_{h,i,j} = \begin{cases} \widehat{k}_{h,i}^{\top} \widehat{k}_{h,j}, & i \neq j, \\ 0, & i = j. \end{cases} \tag{6}$$

The implementation uses the `first_percent` index rule with threshold $\tau = 0.5$ and ratio $\rho = 0.1$. For each row i, form $u_{h,i,j} = j$ if $S_{h,i,j} > \tau$, and $u_{h,i,j} = 0$ otherwise. Let $r = \max(1, \lfloor \rho M \rfloor)$ and let $p_{h,i}$ be the r-th smallest value in $\{u_{h,i,j}\}_{j=0}^{M-1}$, counting repeated values. We obtain $\widetilde{S}$ by setting $S_{h,i,p_{h,i}}$ to zero in each row and leaving the other entries unchanged. This rule suppresses one similarity entry per row; ρ controls the index selection rather than the fraction of KV entries

Table 11: **Historical KV selection settings.** The main configuration uses $B = 128$ and $C = 256$. Selection positions are shared across layers and may differ across heads.

Configuration	Default
Fixed KV budget C	256
Recent window W	B
Importance pooling	1D max pooling
Pooling kernel	7
Importance weight λ	0.1
Similarity threshold	0.5
Neighbor-index ratio ρ	0.1
Neighbor-index rule	`first_percent`
Layer policy	Shared selection
Eviction stage	Inference only

retained. The redundancy score averages the modified similarities over the row index and applies softmax over candidate positions:

$$\overline{S}_{h,i} = \frac{1}{M} \sum_{j=0}^{M-1} \widetilde{S}_{h,j,i}, \qquad R_{h,i} = \frac{\exp(\overline{S}_{h,i})}{\sum_{t=0}^{M-1} \exp(\overline{S}_{h,t})}. \tag{7}$$

The combined score is $\lambda I_{h,i} - (1 - \lambda) R_{h,i}$, with $\lambda = 0.1$. The highest-scoring $C - W$ older positions are gathered at every layer alongside the W protected recent entries. Although the number of retained positions is bounded, their identities are data-dependent.

The importance–redundancy principle follows R-KV (Cai et al., 2025). BLoCE applies it to image blocks, aggregates attention evidence across vision-encoder layers, and shares selected positions across those layers. Other fixed-budget selection rules would satisfy the same cache bound. The selection rule affects which historical context is retained, whereas the budget determines how much can be retained.

B Complexity and KV Cache Analysis

B.1 Attention Pairs

Let block k contain $n_k \le B$ tokens, $K = \lceil N/B \rceil$, and $S_k = \sum_{j \le k} n_j$. Full attention evaluates N^2 query–key pairs per head and layer. Full-history block-causal attention evaluates

$$P_{\text{history}} = \sum_{k=1}^{K} n_k S_k. \tag{8}$$

For equal-sized blocks, this is $(N^2 + NB)/2$. The logical attention shape for block k is $n_k \times S_k$, at most $B \times N$.

With a fixed historical KV budget C, the pair count is

$$P_{\text{bounded}} = \sum_{k=1}^{K} n_k \big(\min(C, S_{k-1}) + n_k\big) \le N(C + B), \tag{9}$$

where $S_0 = 0$. This includes a partial final block and assumes no padding work; padding to deployment shapes may add masked computation. These counts describe attention interactions rather than hardware memory transfers.

B.2 Logical Tensor Sizes at the Main Configuration

For 1344×1344 images with 14-pixel patches, $N = 9216$. We use $L = 24$, $D = 1024$, $H = 8$, $B = 128$, and $C = 256$, matching the main latency configuration. Table 12 compares the feature

Table 12: **Logical feature and attention-input ranges.** Full attention can process these tensors in tiles; the table does not require simultaneous on-chip storage. BLoCE changes the dependency pattern to limit the K/V set referenced by a current block.

Quantity	Full attention	BLoCE current block
Token features	ND: 18 MiB	BD: 0.25 MiB
K/V input set	$2ND$: 36 MiB	$2(C+B)D$: 1.5 MiB

Table 13: **Analytical attention and history counts at** 1344×1344**.** BC-full denotes block-causal execution with full history. Both blockwise settings use $B = 128$; BLoCE uses $C = 256$. Score counts sum over the eight heads in one layer. MiB equivalents assume two bytes per element. A dash denotes no cross-block history across all layers.

Quantity	Full attention	BC-full	BLoCE
Pairs per head/layer	84,934,656	43,057,152	3,489,792
Max. score elements	679,477,248	9,437,184	393,216
Score equivalent (MiB)	1296	18	0.75
Historical KV (MiB)	–	≤ 864	≤ 24

and K/V ranges involved in a layer; byte equivalents use two bytes per element for illustration, without specifying the deployed precision.

Full-history block-causal execution retains up to $2LND$ KV elements across layers. BLoCE retains at most $2LCD$ between blocks and $2L(C+B)D$ before eviction, corresponding to 24 and 36 MiB in this example. A conventional layer-wise full-attention encoder does not need to retain all layers' KV tensors simultaneously; its storage should not be identified with $2LND$.

BLoCE evaluates approximately $24.3\times$ fewer attention pairs than full attention in this example. Its history bound is $36\times$ smaller than for full-history block-causal execution. Neither ratio predicts an equal reduction in overall FLOPs or latency. Completed outputs require $O(ND)$ storage, and model weights, projection/FFN activations, attention statistics, and runtime buffers also contribute to memory use.

B.3 Selection Overhead

Storing attention maps from all layers requires up to $LHB(C+B)$ elements per block. At the main configuration, this is 9,437,184 elements, or 18 MiB at two bytes per element. Aggregation may use a different precision. Reducing statistics as layers are processed can reduce storage, but is not assumed in this count.

Let $M \leq C + B - W$ be the number of older candidates before selection. Pairwise key similarity requires $O(M^2D)$ arithmetic and $O(HM^2)$ temporary elements. With $W = B$, $M \leq C$. Top-k selection depends on M, and gathering KV requires $O(LCD)$ element accesses per block. These dimensions are bounded at fixed B, C, L, D, H, but the cost repeats for every block. Materializing or recomputing attention statistics can also add work; they need not be exposed without cost by a fused attention operator.

B.4 Relationship to the NPU Memory Hierarchy

Smaller intermediate tensors can improve opportunities for on-chip reuse, as discussed in official ANE deployment guidance (Apple Machine Learning Research, 2022). BLoCE exposes this opportunity at the encoder level: block causality allows a block to advance through successive layers, while the historical KV budget limits its attention context. Actual placement, reuse, and transfers depend on the compiler and runtime. Blockwise execution may also increase weight reads and invocation overhead, which can offset gains at small image resolutions.

Table 14: **Comparison settings.** FA and BC denote full and block-causal attention. Full history disables KV eviction; BLoCE limits history to C entries per layer and head. All BC settings retain the base encoder's layer structure.

Setting	Training after initialization	Inference attention/history
AIMv2	None	Full attention
FA-tuned	Continued training, same data	Full attention
BC-direct	None	Block-causal, full history
BC-FT	All encoder weights; fixed block size	Block-causal, full history
BC-LP	Classification head only	Block-causal, full history
BC-conv	Progressive Conversion	Block-causal, full history
BLoCE	Progressive Conversion	Block-causal, budget C

FlashAttention-style kernels reduce transfers for a fixed attention pattern (Dao et al., 2022). They can tile full attention without storing all scores, but do not remove its global dependencies or pair count. BLoCE changes the pattern and the context supplied to each call. The logical analysis and measured latency therefore support the execution design without attributing a particular fraction of its speedup to reduced DRAM traffic or improved cache-hit rate.

C Device Measurement Scope

Both encoders are compiled with dynamic quantization through the official MediaTek SDK and use its available optimized attention operator. Reported latency covers the vision encoder, including BLoCE's block execution and historical KV management, and excludes LLM prefill and decoding. In Table 15, benchmark quality and encoder latency are evaluated using the same quantized model at the same input resolution of 1344×1344 for each setting.

The right part of Figure 1 shows the high-resolution latency comparison; Table 2 reports all measured resolutions and block sizes. Full attention does not finish the 1792×1792 input within a 10,000 ms observation window; this is a timeout rather than a measured 10,000 ms latency. The analytical tables report logical tensor ranges and KV counts, not device peak memory or memory-traffic measurements.

D Additional Comparison Details

Table 14 defines the shortened setting names used throughout the paper. BC-conv and BLoCE use the same conversion procedure; their inference history differs. BC-LP is used only for the InternViT classification experiment.

Table 15 reports benchmark means and encoder latency for the quantized deployment models at 1344×1344. For each row, both measurements use the same quantized model and input resolution. BLoCE obtains a mean of 71.0, compared with 71.4 for AIMv2 and 70.8 for FA-tuned, while reducing encoder latency from 2220 to 699 ms. Inference with these quantized models is difficult to parallelize in our evaluation setup, making benchmark evaluation slow. We therefore limit this joint quality and latency evaluation to the three models at this resolution and do not conduct a broader deployment evaluation across configurations.

E Vision-Task Training Details

E.1 Image Classification

The DeiT-based conversion uses task-specific supervised training instead of the frozen-MLLM setting. Table 16 gives the classification recipe.

Table 15: **Benchmark means and encoder latency of quantized models at** 1344×1344**.** Both measurements in each row use the same quantized model and input resolution. Latency is measured on the Dimensity 9500 NPU. BLoCE uses $B = 128$, $C = 256$. Bold marks the highest mean and lowest latency.

Setting	Benchmark mean	Encoder latency (ms)
AIMv2	**71.4**	2220
FA-tuned	70.8	2220
BLoCE	71.0	**699**

Table 16: **Vision-task training settings.** The task-specific recipes used for classification, detection/segmentation, and linear probing.

Task	Configuration	Value
ImageNet classification	Input size / patch size	384 / 16
	Backbone	DeiT-Base
	Batch size / epochs	128 / 30
	Learning rate / weight decay	10^{-6} / 0.05
COCO detection and segmentation	Backbone / detection head	ViTDet / Mask R-CNN
	Learning rate	10^{-6}
	Learning-rate schedule	Multi-step
	Batch size / epochs / GPUs	8 / 50 / 8
InternViT-6B probe	Frozen ViT / patch size	Yes / 14
	Pretraining input size	224
	Epochs / warm-up epochs	10 / 1
	Base learning rate / optimizer	0.1 / SGD

E.2 Object Detection and Instance Segmentation

Detection and segmentation use ViTDet with Mask R-CNN and the Detectron2 recipe. Table 16 lists the main training settings.

E.3 InternViT-6B Linear Probe

BC-LP freezes InternViT-6B and trains the classification head following the InternVL recipe. Both InternViT block-causal variants retain full history. Table 16 gives the linear-probing settings.