UltraViT: Latency-Optimized On-device Vision Encoder for Large Vision-Language Models

Ioannis Maniadis Metaxas*[1], Adrian Bulat*[1,2], Alberto Baldrati*[1], Anestis Zaganidis[1], Yassine Ouali[1], Hyeonuk Kim[1], and Georgios Tzimiropoulos[1,3]

[1] Samsung AI Cambridge
[2] Technical University of Iasi
[3] Queen Mary University of London

Abstract. Large Vision-Language Models (LVLMs) remain bottlenecked by massive computational footprints, precluding their deployment on resource-constrained edge devices. While efforts to compress LVLMs focus heavily on vision token reduction or smaller language models, the vision encoder is largely overlooked, typically deployed as a monolithic, computationally heavy feature extractor. Moreover, there is no previous effort that designs a vision encoder for LVLMs directly optimized for on-device latency. In this paper, we present UltraViT, a vision encoder for LVLMs, explicitly designed and optimized for on-device performance. Specifically, by taking into account real on-device latencies, we systematically design a pyramidal architecture that strategically integrates and adapts heterogeneous spatial mixers at the macro-block level. Furthermore, to pre-train UltraViT, we propose a novel two-stage generative pre-training strategy: cultivating rich spatial features via dense distillation, followed by direct generative supervision from a capacity-mixed frozen LLM. Compared to standard contrastive and SSL, we show that our pre-training is much more effective for achieving high-level semantic grounding for UltraViT needed for the subsequent generative multimodal alignment of LVLM training. Extensive experiments demonstrate that our on-device latency-informed design combined with our tailored training strategy establishes a new state-of-the-art for efficient LVLM encoding, significantly outperforming existing encoder-centric baselines while operating on-device at nearly 1.7× the speed.

1 Introduction

The recent proliferation of Large Vision-Language Models (LVLMs) [24, 56, 70] has driven unprecedented advancements across a broad spectrum of multimodal tasks [23,27,28,34,37,69]. However, these models are characterized by massive parameter counts and immense computational footprints, fundamentally hindering their deployment on resource-constrained mobile and edge devices. To mitigate

* Equal contribution

these high inference costs, prior efforts targeting efficient LVLMs have predominantly focused on two strategies: aggressively reducing the length of the visual context via token sparsification or compression [8, 9, 59, 68], and/or coupling lighter language models with more computationally efficient cross-modal interaction mechanisms [13,32]. However, with a few exceptions (i.e., FastVLM [52]), existing literature largely overlooks the vision encoder architecture itself, despite the fact that relying on heavyweight models like CLIP-ViT-L [41] or SigLIP-400M [64] as fixed feature extractors creates a severe computational bottleneck, especially for the processing of higher-resolution images and video.

Concurrently, while significant progress has been made in designing efficient, mobile-friendly Vision Transformers (ViTs) [39, 53, 63], these architectures have been heavily optimized for the task of ImageNet classification, rather than the dense image understanding and reasoning demanded by LVLMs. Crucially, prior efficient ViT paradigms typically introduce a specific spatial mixer (e.g., sparse attention [39] or depthwise convolutional modules [43]), and replicate it uniformly across the entire network or combined with vanilla attention. We argue that this homogenous block formulation is suboptimal for *on-device* LVLM image encoding when latency considerations must be taken into account.

To bridge this gap, we present UltraViT, a highly efficient, pyramidal vision architecture explicitly optimized for on-device LVLM deployment. Rather than relying on a monolithic block design, we systematically engineer a heterogeneous architecture by strategically selecting distinct spatial mixers across different network stages, and further adapting their underlying operations specifically for NPU/mobile execution. UltraViT is designed to be latency-efficient for on-device deployment (measured on a real edge device) and to possess dense representation capabilities necessary for multimodal reasoning. Moreover, by designing the vision encoder from the ground-up to be inherently compact and efficient, we entirely circumvent the need for post-hoc vision token compressors.

Beyond architectural innovation, we propose a highly effective pre-training strategy for UltraViT. Our motivation is as follows: First, standard contrastive training (e.g., CLIP [41]) represents too coarse a signal for LVLMs, and is misaligned with generative token generation [4]. Furthermore, although Masked Image Modeling (MIM) and DINO [38] objectives yield strong dense localization, their reconstructive nature produces low-level visual features that lack the high-level semantics required by an LLM. To alleviate this, we introduce a novel, two-stage pre-training paradigm. First, we perform dense feature distillation from a robust, high-resolution teacher model to cultivate rich spatial representations. Subsequently, we attach the encoder to a frozen vision-enabled LLM to provide a direct generative training signal. To strictly maintain computational efficiency during this second phase, we devise a dynamic LLM mixing strategy that samples and mixes LLMs of varying capacities.

[4] This misalignment is consistently evidenced by the empirical necessity of discarding the final transformer layer before LLM integration [6].

In summary, our combined architectural optimization and generative pre-training enables UltraViT to establish a new state-of-the-art for on-device LVLMs. Our main contributions are summarized as follows:

- We introduce UltraViT, a latency-efficient pyramidal vision encoder explicitly tailored for on-device deployment of LVLMs. Informed by real latency measurements, and by systematically exploring heterogeneous spatial mixers, we overcome the limitations of previously proposed homogeneous ViT architectures for dense semantic reasoning.
- We propose a novel two-stage generative pre-training paradigm tailored to UltraViT. Our approach circumvents fundamental misalignments of contrastive pre-training by leveraging dense spatial distillation followed by direct generative supervision from a frozen, capacity-mixed LLM.
- Extensive experiments demonstrate that UltraViT achieves state-of-the-art on-device performance, outperforming prior encoder-centric computational baselines, such as FastVLM [52], while being almost 1.7× faster.

2 Related Work

Efficient Vision Encoders: Constructing efficient vision architectures is a long-standing challenge in computer vision. Based on their underlying operator primitives, existing models can be broadly classified into three categories: convolutional, transformer-based, and hybrid architectures.

Historically, early architectures were predominantly driven by Convolutional Neural Networks (CNNs). Research in this direction yielded seminal efficiency techniques such as depth-wise convolutions and inverted residuals in MobileNets [19, 43], channel shuffling in ShuffleNets [31, 67], cheap linear transformations in GhostNet [18], compound scaling laws in EfficientNet [47], and structural re-parameterization [16,53]. With the advent of the Vision Transformer [17], which demonstrated superior scaling properties compared to CNNs, numerous works have attempted to alleviate the quadratic complexity of global self-attention. Prevalent strategies include adopting pyramidal structural topologies [57], restricting attention to local windows [29], and utilizing efficient primitives such as separable self-attention [36], sparse attention [39], or single-head partial attention [63]. Current state-of-the-art vision encoders commonly combine these two paradigms, utilizing convolutional blocks in the earlier, high-resolution stages to capture local features, and attentive blocks in the later stages to model global dependencies [39, 53, 63]. However, except for FastVLM [52] none of these architectures are designed with LVLMs in mind. In contrast to the aforementioned works, we design our approach from the ground up with LVLMs in mind, carefully adapting and then selecting in a principled manner appropriate mixers depending on the location within the model, constructing a novel heterogeneous architecture.

Efficient Vision-Language Models: Despite their unrivaled accuracy and reasoning capabilities, LVLMs are often computationally prohibitive for on-device

deployment. Current research aimed at mitigating this computational bottleneck generally follows two main trajectories: (1) reducing the sequence length of visual tokens, either via training-aware compression modules [10, 13, 21, 59] or through post-hoc, zero-shot token pruning and attention sparsification techniques [2,58,66,68], and (2) coupling the vision encoder with smaller, highly optimized language models [13,15,32]. While these methods yield favorable speedups across various accuracy-latency trade-offs, the vast majority continue to rely on heavy, computationally rigid vision encoders, typically the 400M parameter SigLIP backbone, not designed with LVLMs in mind. As the LLM backbones become increasingly compact (e.g., scaling down to 0.5B or 1.5B parameters for edge deployment), the relative latency burden of the vision encoder becomes disproportionately large. For instance, in an edge-oriented LLaVA-OV model comprising a SigLIP-400M vision encoder and a 1.5B Qwen2.5 language model, for a 1024x1024px image the visual feature extraction phase alone can account for nearly 50% (!) of the total end-to-end inference time on mobile devices. Consequently, in such resource-constrained environments, optimizing the vision encoder itself, rather than exclusively attempting to shrink the LLM sequence length, represents a critical, yet under-explored avenue for reducing latency. Only very recently has FastVLM [52] introduced a custom architecture specifically aimed at efficient LVLM encoding. In contrast to FastVLM, we introduce a novel heterogeneous architecture coupled with an efficient two-stage training paradigm. By combining dense distillation with LLM-in-the-loop generative pre-training, we deviate from traditional contrastive methods, establishing a new state-of-the-art that surpasses FastVLM in accuracy while operating significantly faster.

Vision Encoder pre-training for LVLMs: The dominant paradigm for pre-training vision encoders heavily relies on contrastive learning objectives applied to massive image-text datasets, epitomized by CLIP [41] and SigLIP [64]. While dense, representation-focused objectives such as Masked Image Modeling (MIM) or DINO have been explored, they generally underperform their contrastive counterparts in LVLM settings [14, 26]. Similarly, directly applying captioning losses (e.g., SigLIP 2 [50]) during vision encoder pre-training yields downstream performance comparable to variants trained without such losses (e.g., SigLIP [64]), suggesting limited marginal utility in the presented form [14]. Hence, current state-of-the-art is represented by contrastively trained model [50], often combined with score distillation objectives [52].

3 Architecture optimization

At its core, an LVLM is composed of a vision encoder, a cross-modal alignment projector, and an LLM. Let I denote a high-resolution input image. The vision encoder acts as the frontline feature extractor, digesting I into a sequence of visual tokens $\mathbf{F} \in \mathbb{R}^{N \times C}$. A lightweight MLP projector then aligns these visual features with the language modality, allowing the LLM to auto-regressively

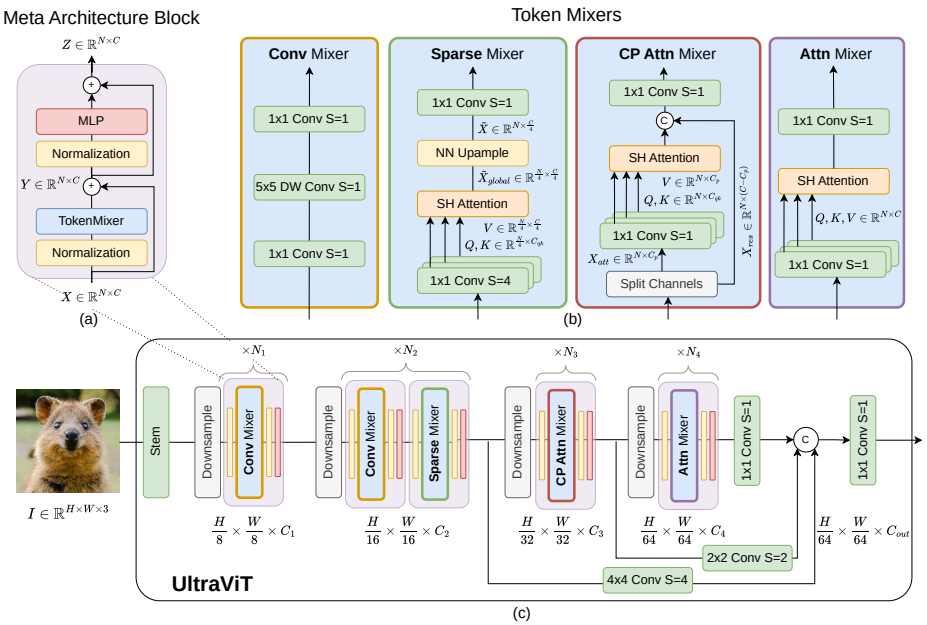

Fig. 1: (a) General Structure of the Meta Architecture Block. (b): Architecture of the four different token mixers employed. (c) Overview of the full UltraViT architecture.

condition on both the image and the query text. In this work, we adopt the widely-used LLaVA-OV [24] design.

Unlike the bulk of prior work on efficient LVLMs, we focus on the design of the vision encoder specifically for efficient on-device deployment. Recognizing the limitations of rigid, homogenous ViT backbones, we pivot to a heterogeneous macro-search space. The proposed UltraViT adopts a classic four-stage pyramidal [57] topology to progressively aggregate spatial information and reduce the token sequence length. However, unlike prior efficient models (e.g., EdgeViT [39], SH-ViT [63]) that apply a single (or two) type of spatial mixers across all stages, we hypothesize that different depths of the network necessitate fundamentally different inductive biases and computational primitives. Consequently, we systematically design and adapt distinct token mixers, specifically targeting NPU and mobile execution constraints to achieve an optimal balance of latency and representational density.

3.1 Heterogeneous Spatial Mixers

Starting from the foundational transformer block structure of [54], recent studies [61] have convincingly demonstrated that the general macro-structure of these blocks is the true catalyst for their success. Specifically, the arrangement of residual connections flanking a primary token mixer and a subsequent channel mixer (MLP) is paramount to the representation quality. Formally, let $\mathbf{X} \in \mathbb{R}^{N \times C}$

denote the input sequence of visual tokens, where $N = H \times W$ is the spatial resolution and C is the embedding dimension. The general Meta Architecture Block is defined as:

$$\mathbf{Y} = \text{TokenMixer}(\text{Norm}(\mathbf{X})) + \mathbf{X} \tag{1}$$

$$\mathbf{Z} = \text{MLP}(\text{Norm}(\mathbf{Y})) + \mathbf{Y} \tag{2}$$

Building upon this insight, UltraViT strictly preserves this proven macro-architecture across all network depths: we standardize the block topology (Eqs. 1, 2) and modularly interchange the internal TokenMixer$(\cdot)$ module.

In the following, without loss of generality, we treat $\mathbf{X}$ interchangeably as $\mathbf{X} \in \mathbb{R}^{N \times C}$ and $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ $(N = H \times W)$. The reshaping required to transition between grid-based operations (e.g., convolutions on $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$) and sequence-based operations (e.g., attention on $\mathbf{X} \in \mathbb{R}^{N \times C}$) is left implicit.

To optimize for both on-device latency and dense representational capacity demanded by LVLMs, we consider, adapt, and evaluate four distinct classes of spatial mixers:

Convolutional Mixer: Convolutional layers are a natural choice for spatial mixing due to their inductive biases for locality and translation equivariance. They are a prime candidate for the initial, high-resolution stages of the network, where the sequence length N is excessively large, rendering global routing computationally prohibitive. Moreover, they are well-supported by mobile NPUs. Following prior work [49, 53], we use DepthWise (DWConv) and Pointwise Convolutions in the spatial mixer. The convolutional mixer is defined as:

$$\mathbf{X}_{proj} = \text{Conv}_{1\times1}(\mathbf{X}) \tag{3}$$

$$\mathbf{Y} = \text{Conv}_{1\times1}(\text{DWConv}_{5\times5}(\mathbf{X}_{proj})) \tag{4}$$

Sparse Mixers: While convolutions are efficient, they struggle to capture long-range dependencies that may be desirable in LVLMs. Conversely, standard self-attention is computationally expensive. As a middle ground, sparse attention mechanisms have been proposed [5, 12, 39]. We use EdgeViT's [39] sparse attention as a starting point:

$$\mathbf{X}_{sparse} = \mathcal{S}(\mathbf{X}) \tag{5}$$

$$\tilde{\mathbf{X}}_{global} = \text{Attention}(\mathbf{X}_{sparse}W^Q, \mathbf{X}_{sparse}W^K, \mathbf{X}_{sparse}W^V) \tag{6}$$

$$\mathbf{Y} = \text{ConvTrans}_{\text{DW}}(\tilde{\mathbf{X}}_{global}), \tag{7}$$

where $\mathcal{S}$ denotes the sparse sampling operator.

While conceptually effective, this instantiation incurs notable latency penalties when deployed on mobile NPUs. To circumvent this, we propose a new optimized Sparse mixer. Specifically, we introduce four core adaptations: (S1) we replace multi-head attention with single-head attention; (S2) we fuse the discrete sampling operator $\mathcal{S}$ directly into the $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ projections by implementing them as strided convolutions; (S3) we replace the transposed depthwise

convolution ($\text{ConvTrans}_{\text{DW}}$) with a hardware-friendly nearest-neighbor upsampling operation; and crucially, (S4) we reduce the dimensionality of the attention projections by mapping queries and keys to a lower-dimensional space and compressing the value projections by a factor of 4. Fig. 2 (left) shows the cumulative effect of applying these structural refinements, with each yielding substantial on-device speed-ups. The new mixer is defined as:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \text{Conv}_{\text{stride}}(\mathbf{X}; W^Q),\ \text{Conv}_{\text{stride}}(\mathbf{X}; W^K),\ \text{Conv}_{\text{stride}}(\mathbf{X}; W^V) \tag{8}$$

$$\tilde{\mathbf{X}}_{global} = \text{SH-Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \tag{9}$$

$$\mathbf{Y} = \text{Upsample}_{\text{NN}}(\tilde{\mathbf{X}}_{global}) W^O, \tag{10}$$

where $W^Q, W^K \in \mathbb{R}^{C \times C_{qk}}$ and $W^V \in \mathbb{R}^{C \times \frac{C}{4}}$ specify the channel-reduced projection matrices, and $W^O \in \mathbb{R}^{\frac{C}{4} \times C}$ represents the output projection layer. In all our experiments, we set $C_{qk} = 16$.

Channel-Partitioned (CP) Attention Mixer: Traditional Multi-Head Self Attention (MHSA) splits the feature dimension C across multiple independent attention heads, causing memory fragmentation and bloated memory access patterns on edge devices. Drawing from SH-ViT [63], we employ Channel-Partitioned Self-Attention (CPSA), which computes single-head attention over a subset of the embedding dimension:

$$\mathbf{X}_{att}, \mathbf{X}_{res} = \text{Split}(\mathbf{X}, [C_p, C - C_p]) \tag{11}$$

$$\tilde{\mathbf{X}}_{att} = \text{SH-Attention}(\mathbf{X}_{att} W^Q, \mathbf{X}_{att} W^K, \mathbf{X}_{att} W^V) \tag{12}$$

$$\mathbf{Y} = \text{Concat}(\tilde{\mathbf{X}}_{att}, \mathbf{X}_{res}) W^O, \tag{13}$$

where $W^Q, W^K \in \mathbb{R}^{C \times C_{qk}}$ and $W^V \in \mathbb{R}^{C \times C_p}$ are the attention projection matrices, and $W^O \in \mathbb{R}^{C \times C}$ is the output projection matrix. C_p and C_{qk} denote the number of channels used for attention and the key-query dimension, respectively. Following SH-ViT [63], we set $C_{qk} = 16$ and $C_p = \frac{C}{4}$ in all experiments.

Vanilla Attention Mixer: In the final stages where spatial downsampling reduces N significantly, attention's complexity is no longer the primary bottleneck. Hence, vanilla attention can be beneficial thanks to its stronger representational capacity. However, we opt for a single-head attention mechanism instead of the traditional multi-head one. This design choice is tailored to on-device deployment: multi-head attention introduces significant memory access overhead due to the partitioning of the embedding dimension, which leads to fragmented tensor operations and sub-optimal utilization of mobile NPUs. Single-head attention, conversely, simplifies data access patterns and maintains a contiguous memory layout, enabling faster execution and lower latency on edge devices while still harnessing the global receptive field benefits of standard attention. Note that, unlike the CP Attention Mixer, the vanilla attention mixer updates all channels. Fig. 2 (right) shows the performance comparison between vanilla attention and single-head attention.

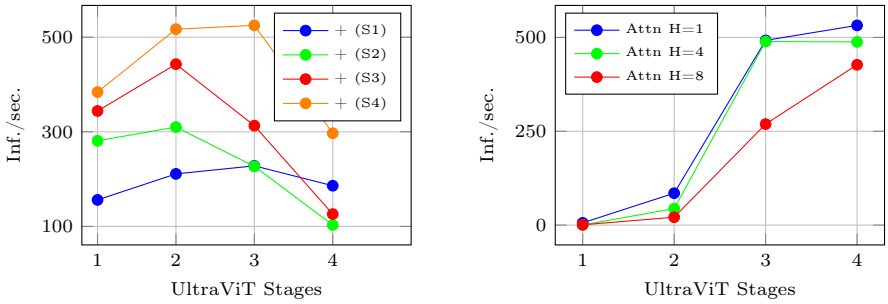

Fig. 2: Inferences per second across UltraVit stages. Left: the throughput when the sparse attention mixer is applied cumulatively up to the i-th stage. Right: stage-wise throughput with attention mixers with a varying number of attention heads.

3.2 Block Selection Process

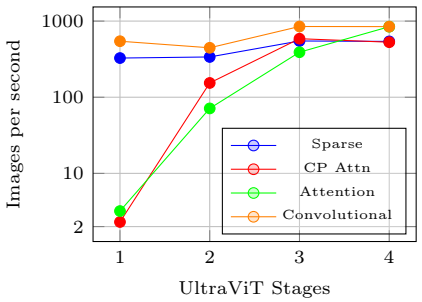

Fig. 3: Inferences per second for images with resolution 512x512 at each stage.

We postulate that the optimal mixer operation, at different stages of the network, varies as a ratio of accuracy-to-latency trade-off. Hence, to optimize the architecture for edge deployment, we employ a systematic block selection process tailored to each stage of the network. The computational dynamics of different spatial mixers: Convolutional, CP, Sparse, and Vanilla Attention, vary significantly depending on the feature resolution (N) and channel dimension (C). Therefore, instead of relying on theoretical FLOPs, complexity estimates, on measurements on proxy devices (GPUs/CPUs), we conduct rigorous on-device latency measurements for each block candidate across a comprehensive grid of configurations, varying the number of channels and tokens. To our knowledge, this is the first work to perform such a systematic block selection process for mobile vision encoders for LVLMs. The results are shown in Fig. 3. Based on these measurements, we then select a targeted subset of these configurations for each stage, train them using a standard SigLIP [64] loss with multiple captions [7], and evaluate their feature representations in a zero-shot manner for image retrieval, classification, and post LLaVA-OV finetuning, as part of the LVLM. The configurations are subsequently ranked based on their combined on-device performance and accuracy. As the results in Table 1 demonstrate, the optimal configuration is heterogeneous, with different mixers suitable at different stages. In particular, it employs highly efficient convolutional mixers during the high-resolution Stage 1, an interleaved combination of convolutional and sparse attentive mixers in Stage 2, channel-partitioned attention in Stage 3, and full single-head attention at the

lowest resolution. The resulting, final architecture, is depicted in Fig. 1, where C_{1-4}=[192, 512, 768, 1536] and N_{1-4}=[4, 7, 10, 2].

3.3 Overall architecture

Pyramidal Structure and Stem: Our network adopts a hierarchical pyramidal architecture. The processing begins with a stem block that rapidly downsamples the input high-resolution image by a factor of 4. Following the stem, the architecture is divided into four distinct stages, progressively downsampling the spatial dimensions by a factor of two while expanding the channel capacity. The downsampling block bridging these stages follows [63] and consists of a 1x1 conv layers, a 3x3 strided DW conv layer, another 1x1 conv. layer, and a SE module [20].

Multi-scale Feature Aggregator: To support diverse vision-language tasks, including dense prediction and spatial grounding, we incorporate a lightweight feature aggregator combining semantically rich features from later stages with high-resolution, detailed features from earlier stages. The features are first projected to a common spatial resolution via strided convolutions, concatenated along the channel dimension, and finally fused through a 1×1 convolution to produce the aggregated representation (see Fig. 1, bottom).

Table 1: Accuracy-performance tradeoffs for different model configurations. Zero-shot results for classification (Imagenet, top-1) and image retrieval (a suite of 5 OCR datasets, @R1). LLaVA finetuning results reported at patch resolution of 512x512px, as part of a 1.5B LVLM.

Mixer per Stage	Params (M)	Flops (B)	Inf.	ImageNet	OCR	TextVQA	DocVQA	InfoVQA	RWQA	MME	Avg.
C, C+S, CP, A	139.4	143.9	148.9	67.8	33.4	59.1	64.8	39.8	**57.9**	71.1	**58.5**
C, C, CP, A	142.4	144.7	147.8	68.0	33.2	58.2	**65.4**	39.5	57.7	68.5	57.9
C, C+S, A, A	144.8	154.7	126.7	**68.3**	**33.6**	**59.4**	63.9	**40.7**	**57.9**	70.6	**58.5**
C, C, C, C	143.4	141.8	171.0	67.2	31.4	57.1	62.4	39.4	56.1	67.2	56.4
C, C, C, A	143.4	147.2	160.7	67.8	32.6	58.1	63.7	38.5	56.5	67.6	56.9

4 Efficient Vision Encoder pre-training for LVLMs

Going beyond standard contrastive (CLIP or SigLIP) training, in this section, we present a new sample-efficient pre-training strategy for vision encoders tailored for LVLMs. The training strategy is based on two ideas: dense distillation and generative pre-training. During the first stage, we distil the knowledge of a large teacher model into our student model using a modified dense distillation. During the second stage, we use generative pre-training against a pretrained, frozen LLM to further improve the performance of our student model.

4.1 Dense distillation

To effectively train the model, we propose to transfer the rich spatial and semantic knowledge of a large-scale pretrained teacher model into our compact architecture using a tailored dense distillation strategy. A primary challenge in cross-architecture distillation arises from the mismatch in resolution and feature dimensionality between the teacher and student.

Rather than naively downsampling the teacher's high-resolution feature maps to match the student, a process that incurs a loss of fine-grained spatial information crucial for dense prediction tasks, we primarily preserve the spatial detail of the features via structural re-alignment. Specifically, denote the teacher features as $F_T \in \mathbb{R}^{H_T \times W_T \times C_T}$ and the student features as $F_S \in \mathbb{R}^{H_S \times W_S \times C_S}$. When the spatial resolutions are compatible ($H_T/H_S = r$ is an integer), we apply Pixel Shuffle [44] to rearrange the student's tokens by reshaping local spatial blocks into the channel dimension. For an $r \times r$ spatial block, the reshaped student feature becomes $\hat{F}_S \in \mathbb{R}^{H_S \cdot r \times W_S \cdot r \times \frac{C_S}{r^2}}$. In cases where exact spatial alignment cannot be achieved, e.g., due to non-integer scaling factors or channel constraints, we apply a resolution-aligned downsampling of the teacher features to the closest compatible resolution. This preserves the overall alignment structure while avoiding excessive distortion of the teacher's representations.

Following this spatial-to-channel rearrangement, we apply a dual-objective loss to guide the student. First, we compute a dense cosine similarity loss applied patch-by-patch between the student features and the re-aligned teacher features, forcing the student to mimic the teacher's localized representational distribution:

$$\mathcal{L}_{dense} = 1 - \frac{1}{r^2 H_S W_S} \sum_{i=1}^{H_T} \sum_{j=1}^{W_T} \frac{\hat{F}_S(i,j) \cdot F_T(i,j)}{\|\hat{F}_S(i,j)\| \|F_T(i,j)\|}. \tag{14}$$

Second, to ensure that the student's global semantic understanding is preserved, we pass both the student's and teacher's features through the teacher's frozen attention pooling block, denoted as $\mathcal{P}_T(\cdot)$. We then apply a secondary cosine similarity loss on these aggregated, global representation tokens:

$$\mathcal{L}_{global} = 1 - \frac{\mathcal{P}_T(\hat{F}_S) \cdot \mathcal{P}_T(F_T)}{\|\mathcal{P}_T(\hat{F}_S)\| \|\mathcal{P}_T(F_T)\|} \tag{15}$$

The final dense distillation objective is simply a weighted combination of these two alignment losses: $\mathcal{L}_{distill} = 0.5\mathcal{L}_{dense} + 0.5\mathcal{L}_{global}$.

4.2 Generative pre-training

Generative pre-training of the vision encoder utilizing an autoregressive next-token prediction task has historically received limited attention compared to contrastive methods. This limited adoption stems from two critical drawbacks: the substantial computational cost inherent to training autoregressive decoders, and their fundamental unsuitability for straightforward zero-shot deployment, the

very capability that popularized models like CLIP. Consequently, while recent works such as GIT [55], CapPa [51], and the captioning extension in SigLIP 2 [50] have demonstrated the conceptual viability of generative representation learning, these methods have not yet shown notable gains over contrastive methods for LVLMs and exhibit fundamental limitations.

First, while SigLIP 2 incorporates a generative captioning objective alongside contrastive learning, empirical evidence shows that post-LLaVA fine-tuning, its performance is highly similar to the original SigLIP model [14]. This suggests that their specific generative formulation does not yield representations that are uniquely advantageous for downstream LVLM tasks. Second, prior methods typically employ extremely small textual decoders during the pre-training phase. This design choice inadvertently shifts the linguistic and syntactic modeling burden onto the vision encoder, forcing it to allocate its limited representational capacity to non-visual tasks. For mobile-centric, parameter-constrained vision architectures, this waste of representational capacity is highly detrimental to dense spatial grounding.

To address these shortcomings, we propose a tailored generative pre-training strategy. Crucially, the target Language Model (LLM) acting as the decoder must be explicitly pre-aligned to vision. Once alignment is achieved, we freeze the LLM. By utilizing a strong, pre-aligned LLM as the frozen decoder, we relieve the vision encoder of the linguistic modeling burden, ensuring its capacity is dedicated entirely to extracting rich visual features. Furthermore, to optimize training efficiency and accelerate convergence without compromising semantic alignment, we dynamically alternate the frozen decoder between a 0.5B and a 1.5B parameter model during the generative pre-training phase. We note that fully utilizing smaller models shifts the burden of language modeling again to the vision encoder. This model-switching strategy allows us to achieve deep alignment with high-capacity LLMs, while significantly reducing the overall computational cost of the pre-training pipeline.

5 Experiments

5.1 Experimental Setup

Our training pipeline consists of three distinct phases: dense pre-training, generative pre-training, and supervised fine-tuning.

1. Dense Distilation Pre-training: The model is trained on a randomly sampled subset of 150M examples from DataComp-1B [52]. We optimize the dense objective of Sec. 4.1 for 25 epochs using a cosine learning rate schedule, a peak learning rate of 1×10^{-4}, a weight decay of 0.1, and a batch size of 32k.

2. Generative Pre-training: Initialized from the dense pre-trained weights, the model is trained for 1 epoch on the 85M data mix from [1]. We employ a cosine schedule with a learning rate of 1×10^{-5}, no weight decay, and a batch size of 192. The generative signal is provided through a frozen Qwen2 LLM, which was previously vision-aligned using 4M LLaVA-OV samples. The resulting

model represents our final vision encoder, referred to throughout this paper as **UltraViT**.

3. Supervised Fine-tuning: The final phase evaluates UltraViT within an LVLM. We substitute the original vision encoder in the LLaVA-OV architecture with UltraViT, resulting in an LVLM coined **UltraVLM**. To guarantee direct and fair comparisons, UltraVLM strictly follows the 3-stage training recipe of LLaVA-OV [24], utilizing a total of 7.1M samples: 4M OCR and captioning examples for Step 2, and 3.1M instructionally-guided examples for Step 3.

All models were trained on 32 H100 using PyTorch [40] and deepspeed [42].

On-device Benchmarking Protocol: All mobile NPU inference latencies and throughput measurements reported in this work, both during our block-level latency search (Sec. 3.1) and full-model evaluation, were fully measured on physical hardware. Specifically, models were compiled using the Qualcomm Neural Processing SDK (QNN), fully quantized to INT8 precision, and benchmarked natively on a Samsung Galaxy S25 Ultra smartphone.

5.2 Comparison with state-of-the-art

We primarily compare UltraVLM with the current state-of-the-art approach for efficient vision encoders in LVLMs, FastVLM [52], evaluating it on a large suite of benchmarks including GQA [22], SQA [30], TextVQA [46], POPE [25], DocVQA [35], InfoVQA [34], RealWorldQA, MME [69], MMMU [62], ChartQA [33], and MMSTAR [11]. To ensure a fair comparison, we retrain FastVLM using the same LLaVA-OV training recipe as UltraVLM, starting from their vision encoder, prior to their LVLMs training. For completeness, we also include the results reported in the original FastVLM paper under the data setting most comparable to ours. Additionally, to place our results in the context of the broader LVLM landscape, we include the performance of LLaVA-OV [24], QwenVL-2, QwenVL-2.5, and QwenVL-3. These are evaluated under two settings: native full resolution (an unconstrained number of vision tokens) and, following [52], restricted to a matching number of tokens per image.

As Table 2 shows, UltraVLM outperforms FastVLM, across most benchmarks, achieving particularly large gains on challenging tasks such as TextVQA (+6.0%), DocVQA (+5.1%), and ChartQA (+6.2%), all while employing a much faster vision encoder (see also Sec. 5.4). It also outperforms the original FastVLM despite FastVLM being trained with more than twice the amount of supervision during the LVLM fine-tuning phase (+4.1% on DocVQA, +2.9% on TextVQA, etc.). Finally, when the QwenVL-2/2.5/3 series models are restricted to the same token budget (256 tokens) as ours, our UltraVLM surpasses them on the majority of benchmarks. This highlights UltraVLM's efficiency, especially considering that the QwenVL models utilize much larger and slower vision encoders, more modern LLM backbones, and are trained on significantly larger pools of proprietary private data. Though we report the unconstrained full-resolution results for these models for completeness, operating them without token constraints yields LVLMs that are more than an order of magnitude slower than UltraVLM.

Table 2: Comparison with state-of-the-art LVLMs. Grayedout entries denote models evaluated at full resolution, using an unrestricted number of tokens. All other entries are evaluated under the same number of tokens per image. * denotes results taken from [52]. Re-trained models are trained under the same LLaVA-OV training recipe. Vis. inf. - denotes the on-device inference speed of the corresponding vision encoder at a reference resolution of 512x512px. Note that our approach (1) outperforms FastVLM on most benchmarks, (2) while having a much faster vision encoder (see also Sec. 5.4).

Model	Vis Arch.	Vis inf./sec	Data amount	GQA	SQA	TextVQA	POPE	DocVQA	InfoVQA	RealWQA	MME	MMMU	ChartQA	MMSTAR
Pre-trained baselines														
LLaVA-OV [24]	SigLIP-400M	7.6	7.1M	58.6	75.6	69.9	88.1	76.6	46.7	57.3	63.7	38.6	64.2	40.5
QwenVL-2 [56]	QwenVL-2	3.7	-	60.1	77.8	79.4	87.7	89.5	63.9	61.2	75.1	38.1	75.0	43.5
QwenVL-2.5 [4]	QwenVL-2.5	6.9	-	60.0	80.7	78.6	87.3	92.4	75.0	59.1	75.8	39.6	83.7	56.2
QwenVL-3 [3]	SigLIP2-400M	7.6	-	59.4	86.3	79.2	89.4	92.7	72.4	63.5	74.6	46.9	79.8	43.1
QwenVL-2 [56]	QwenVL-2	3.7	-	59.3	77.3	66.8	86.6	64.5	30.7	51.2	72.1	38.1	49.4	43.0
QwenVL-2.5 [4]	QwenVL-2.5	6.9	-	60.5	80.3	66.5	86.8	61.8	35.0	58.8	77.5	39.8	71.5	55.0
QwenVL-3 [3]	SigLIP2-400M	7.6	-	59.5	86.1	68.8	89.5	67.8	34.6	61.3	73.7	47.1	73.9	43.1
FastVLM [52]*	FastViTHD	91.4	16.1M	64.2	74.8	66.0	88.0	67.7	-	-	-	33.1	-	-
Re-trained models														
FastVLM [52]	FastViTHD	91.4	7.1M	59.9	80.0	62.9	**87.3**	66.7	43.5	**60.1**	69.9	**41.1**	66.0	44.1
UltraVLM	UltraViT	148.9	7.1M	**60.5**	**82.9**	**68.9**	87.1	**71.8**	**47.9**	59.5	**70.0**	38.1	**72.2**	**49.8**

Table 3: Comparison with token reduction methods. Unlike post-hoc pruning approaches that hurt accuracy and leave vision encoder latency unchanged, UltraVLM's pyramidal design natively achieves a massive 16.0× token reduction.

Method	GQA	SQA	TextVQA	POPE	DocVQA	InfoVQA	RealWQA	MME	ChartQA	MMSTAR	Avg. num. tokens reduction	Vis Enc Speedup
LLaVA-OV [24]	58.6	75.6	**69.9**	**88.1**	**76.6**	46.7	57.3	63.7	64.2	40.5	1.0×	1.0×
VisionZip [59]	57.2	76.2	62.0	87.0	55.1	30.8	54.6	63.9	50.0	38.1	5.7×	1.0×
PyramidDrop [58]	55.9	76.0	60.3	87.2	54.5	34.2	56.7	63.7	47.9	37.5	4.6×	1.0×
VisPruner [65]	53.4	76.0	54.8	83.3	51.2	28.4	52.9	62.3	40.0	38.4	5.7×	1.0×
HiRED [2]	56.3	76.0	59.8	85.9	52.3	27.6	54.8	63.2	44.1	39.3	5.0×	1.0×
UltraVLM	**60.5**	**82.9**	68.9	87.1	71.8	**47.9**	**59.5**	**70.0**	**72.2**	**49.8**	16.0×	19.1×

5.3 Comparison with token reduction methods

In addition to utilizing a highly efficient vision encoder, UltraVLM organically achieves a significant 16.0× reduction in the number of tokens without any external token reduction mechanisms. Herein, we compare UltraVLM with previously proposed state-of-the-art token reduction methods. As shown in Table 3, UltraVLM comfortably outperforms all existing reduction strategies by significant

margins across every benchmark. Crucially, because our token reduction is an intrinsic property of the underlying architecture, UltraVLM is the only method that additionally delivers a significant inference speed-up for the vision encoder itself. Note that all methods use the same LLM and training data.

5.4 Inference and train-time efficiency

Inference efficiency: *Compared to the commonly adopted SigLIP or SigLIP-2 400M:* First, the newly proposed UltraViT vision encoder executes 19.1× faster during inference. Secondly, our pyramidal architecture intrinsically outputs 16× fewer visual tokens, directly shrinking the input sequence length and slashing the computational burden of the LLM proportionally. *Compared to FastVLM:* UltraViT operates nearly 1.7× faster than FastViT at a standard 512 × 512px resolution. Furthermore, as illustrated in Fig. 4, this relative speed-up remains highly consistent across a wide range of input resolutions. As noted before, all models are benchmarked natively on a Samsung Galaxy S25 Ultra.

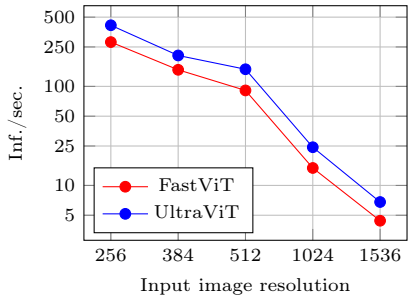

Fig. 4: Inferences per second for UltraViT vs FastViT for varying input image resolutions.

Table 4: Pre-training efficiency. UltraViT achieves state-of-the-art results while processing 3× fewer total samples and 10× fewer unique images than FastVLM.

Model	Stage	Data	Seen
FastVLM	Contrastive	2B	13B
UltraViT	Dense Distill.	150M	4B
	Generative	85M	85M
	Total	**235M**	**4.1B**

Pre-training efficiency: While prior contrastive strategies demand massive data regimes to converge, our two-stage pre-training framework (dense distillation followed by generative supervision) achieves superior performance with a fraction of the samples. As detailed in Table 4, compared with FastVLM, UltraViT achieves state-of-the-art results while seeing approximately 3× fewer total training samples (4.1B vs. 13B) and processing nearly 10× fewer unique images (235M vs. 2B) compared to the contrastive distillation employed by FastVLM.

6 Ablation Studies

Impact of the dense distillation: Table 5(a) compares our dense distillation with standard contrastive (SigLIP) training. With UltraViT, dense distillation

brings consistent gains, with larger improvements on fine-grained tasks such as DocVQA and InfoVQA. Further training with a larger teacher improves performance, indicating that our approach scales well with stronger teachers and longer training. Similarly, for FastViT, dense distillation consistently outperforms contrastive training across all benchmarks. Under the same training setup, UltraViT achieves comparable performance to FastViT while running significantly faster on-device.

Table 5: Ablation studies on the proposed Dense distillation and Generative pre-training techniques.

(a) Dense Distillation

Model	Variant	DocVQA	InfoVQA	MMSTAR	SQA
UltraViT	Contrastive	58.9	36.4	42.3	75.8
	Dense Distil.	63.9	39.5	44.6	76.9
	+ Large Teacher	67.5	42.8	44.4	78.5
FastViT	Contrastive	60.7	39.1	44.4	74.0
	Dense Distil.	62.1	39.4	45.9	76.3

(b) Gen. Pre-training (UltraViT)

Variant	DocVQA	InfoVQA	MMSTAR	SQA
Dense Distil. (+ LT).	67.5	42.8	44.4	78.5
+ Gen (0.5B)	70.0	44.7	47.3	81.5
+ Gen (1.5B)	72.0	46.8	49.1	83.1
+ Gen (Dyn.)	71.8	47.9	49.8	82.9

Impact of generative pre-training: Herein, we ablate the impact of the generative pre-training stage for the UltraViT model and the choice of the LLM decoder. As shown in Table 5(b), further fine-tuning the vision encoder using a frozen LLM results in notable accuracy gains. When utilizing a small 0.5B LLM, performance improves, but to a smaller degree; the limited linguistic capacity of the small decoder forces the vision encoder to compensate for language modeling, bottlenecking visual representation quality. Utilizing a 1.5B LLM yields strong semantic alignment but incurs a larger computational pre-training overhead. Our proposed dynamic switching strategy, which alternates between the 0.5B and 1.5B decoders, matches or exceeds the performance of the pure 1.5B model while notably reducing the total training cost.

7 Conclusion

In this paper, we presented UltraViT, a latency-optimized vision encoder explicitly designed for deploying Large Vision-Language Models (LVLMs) on edge devices. Our contributions are two-fold: First, we introduce a novel heterogeneous architecture guided by real on-device latency measurements. Then, we propose a novel two-stage pre-training strategy that combines dense feature distillation with generative LLM supervision. Extensive empirical evaluations demonstrate that our holistic co-design establishes a new state-of-the-art for efficient LVLM encoding. UltraViT significantly outperforms existing on-device baselines, such as FastVLM, across diverse multimodal benchmarks while operating at nearly 2x the speed on mobile hardware.

References

1. An, X., Xie, Y., Yang, K., Zhang, W., Zhao, X., Cheng, Z., Wang, Y., Xu, S., Chen, C., Zhu, D., et al.: Llava-onevision-1.5: Fully open framework for democratized multimodal training. arXiv preprint arXiv:2509.23661 (2025)
2. Arif, K.H.I., Yoon, J., Nikolopoulos, D.S., Vandierendonck, H., John, D., Ji, B.: Hired: Attention-guided token dropping for efficient inference of high-resolution vision-language models. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 1773–1781 (2025)
3. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
4. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., Lin, J.: Qwen2.5-vl technical report (2025), https://arxiv.org/abs/2502.13923
5. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
6. Bolya, D., Huang, P.Y., Sun, P., Cho, J.H., Madotto, A., Wei, C., Ma, T., Zhi, J., Rajasegaran, J., Rasheed, H., et al.: Perception encoder: The best visual embeddings are not at the output of the network. arXiv preprint arXiv:2504.13181 (2025)
7. Bulat, A., Ouali, Y., Tzimiropoulos, G.: Fff: Fixing flawed foundations in contrastive pre-training results in very strong vision-language models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14172–14182 (2024)
8. Bulat, A., Ouali, Y., Tzimiropoulos, G.: Compress & cache: Vision token compression for efficient generation and retrieval. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025)
9. Cai, M., Yang, J., Gao, J., Lee, Y.J.: Matryoshka multimodal models. arXiv preprint arXiv:2405.17430 (2024)
10. Cai, M., Yang, J., Gao, J., Lee, Y.J.: Matryoshka multimodal models. In: ICLR (2025)
11. Chen, L., Li, J., Dong, X., Zhang, P., Zang, Y., Chen, Z., Duan, H., Wang, J., Qiao, Y., Lin, D., et al.: Are we on the right way for evaluating large vision-language models? Advances in Neural Information Processing Systems **37**, 27056–27087 (2024)
12. Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509 (2019)
13. Chu, X., Qiao, L., Zhang, X., Xu, S., Wei, F., Yang, Y., Sun, X., Hu, Y., Lin, X., Zhang, B., et al.: Mobilevlm v2: Faster and stronger baseline for vision language model. arXiv preprint arXiv:2402.03766 (2024)
14. Cocchi, F., Moratelli, N., Caffagni, D., Sarto, S., Baraldi, L., Cornia, M., Cucchiara, R.: Llava-more: A comparative study of llms and visual backbones for enhanced visual instruction tuning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4278–4288 (2025)
15. Deitke, M., Clark, C., Lee, S., Tripathi, R., Yang, Y., Park, J.S., Salehi, M., Muennighoff, N., Lo, K., Soldaini, L., et al.: Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 91–104 (2025)

16. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13733–13742 (2021)
17. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
18. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: Ghostnet: More features from cheap operations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1580–1589 (2020)
19. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1314–1324 (2019)
20. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
21. Hu, W., Dou, Z.Y., Li, L.H., Kamath, A., Peng, N., Chang, K.W.: Matryoshka query transformer for large vision-language models. arXiv preprint arXiv:2405.19315 (2024)
22. Hudson, D.A., Manning, C.D.: Gqa: A new dataset for real-world visual reasoning and compositional question answering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 6700–6709 (2019)
23. Kembhavi, A., Salvato, M., Kolve, E., Seo, M., Hajishirzi, H., Farhadi, A.: A diagram is worth a dozen images. In: European conference on computer vision. pp. 235–251. Springer (2016)
24. Li, B., Zhang, Y., Guo, D., Zhang, R., Li, F., Zhang, H., Zhang, K., Zhang, P., Li, Y., Liu, Z., et al.: Llava-onevision: Easy visual task transfer. arXiv preprint arXiv:2408.03326 (2024)
25. Li, Y., Du, Y., Zhou, K., Wang, J., Zhao, W.X., Wen, J.R.: Evaluating object hallucination in large vision-language models. In: Proceedings of the 2023 conference on empirical methods in natural language processing. pp. 292–305 (2023)
26. Liu, Y., Zhang, Y., Ghosh, D., Schmidt, L., Yeung-Levy, S.: Data or language supervision: What makes clip better than dino? arXiv preprint arXiv:2510.11835 (2025)
27. Liu, Y., Duan, H., Zhang, Y., Li, B., Zhang, S., Zhao, W., Yuan, Y., Wang, J., He, C., Liu, Z., et al.: Mmbench: Is your multi-modal model an all-around player? In: European conference on computer vision. pp. 216–233. Springer (2024)
28. Liu, Y., Li, Z., Huang, M., Yang, B., Yu, W., Li, C., Yin, X.C., Liu, C.L., Jin, L., Bai, X.: Ocrbench: on the hidden mystery of ocr in large multimodal models. Science China Information Sciences **67**(12), 220102 (2024)
29. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)
30. Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.W., Zhu, S.C., Tafjord, O., Clark, P., Kalyan, A.: Learn to explain: Multimodal reasoning via thought chains for science question answering. Advances in neural information processing systems **35**, 2507–2521 (2022)
31. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV). pp. 116–131 (2018)

32. Marafioti, A., Zohar, O., Farré, M., Noyan, M., Bakouch, E., Cuenca, P., Zakka, C., Allal, L.B., Lozhkov, A., Tazi, N., et al.: Smolvlm: Redefining small and efficient multimodal models. arXiv preprint arXiv:2504.05299 (2025)
33. Masry, A., Do, X.L., Tan, J.Q., Joty, S., Hoque, E.: Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In: Findings of the association for computational linguistics: ACL 2022. pp. 2263–2279 (2022)
34. Mathew, M., Bagal, V., Tito, R., Karatzas, D., Valveny, E., Jawahar, C.: Infographicvqa. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1697–1706 (2022)
35. Mathew, M., Karatzas, D., Jawahar, C.: Docvqa: A dataset for vqa on document images. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 2200–2209 (2021)
36. Mehta, S., Rastegari, M.: Separable self-attention for mobile vision transformers. arXiv preprint arXiv:2206.02680 (2022)
37. Methani, N., Ganguly, P., Khapra, M.M., Kumar, P.: Plotqa: Reasoning over scientific plots. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1527–1536 (2020)
38. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
39. Pan, J., Bulat, A., Tan, F., Zhu, X., Dudziak, L., Li, H., Tzimiropoulos, G., Martinez, B.: Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In: European conference on computer vision. pp. 294–311. Springer (2022)
40. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
41. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PmLR (2021)
42. Rasley, J., Rajbhandari, S., Ruwase, O., He, Y.: Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 3505–3506 (2020)
43. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
44. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1874–1883 (2016)
45. Sidorov, O., Hu, R., Rohrbach, M., Singh, A.: Textcaps: a dataset for image captioning with reading comprehension. In: European conference on computer vision. pp. 742–758. Springer (2020)
46. Singh, A., Natarajan, V., Shah, M., Jiang, Y., Chen, X., Batra, D., Parikh, D., Rohrbach, M.: Towards vqa models that can read. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8317–8326 (2019)

47. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
48. Tang, Z., Lian, L., Eisape, S., Wang, X., Herzig, R., Yala, A., Suhr, A., Darrell, T., Chan, D.M.: Tulip: Towards unified language-image pretraining. arXiv preprint arXiv:2503.15485 (2025)
49. Trockman, A., Kolter, J.Z.: Patches are all you need? arXiv preprint arXiv:2201.09792 (2022)
50. Tschannen, M., Gritsenko, A., Wang, X., Naeem, M.F., Alabdulmohsin, I., Parthasarathy, N., Evans, T., Beyer, L., Xia, Y., Mustafa, B., et al.: Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. arXiv preprint arXiv:2502.14786 (2025)
51. Tschannen, M., Kumar, M., Steiner, A., Zhai, X., Houlsby, N., Beyer, L.: Image captioners are scalable vision learners too. Advances in Neural Information Processing Systems **36**, 46830–46855 (2023)
52. Vasu, P.K.A., Faghri, F., Li, C.L., Koc, C., True, N., Antony, A., Santhanam, G., Gabriel, J., Grasch, P., Tuzel, O., et al.: Fastvlm: Efficient vision encoding for vision language models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 19769–19780 (2025)
53. Vasu, P.K.A., Gabriel, J., Zhu, J., Tuzel, O., Ranjan, A.: Fastvit: A fast hybrid vision transformer using structural reparameterization. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 5785–5795 (2023)
54. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
55. Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C., Wang, L.: Git: A generative image-to-text transformer for vision and language. arXiv preprint arXiv:2205.14100 (2022)
56. Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al.: Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191 (2024)
57. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 568–578 (2021)
58. Xing, L., Huang, Q., Dong, X., Lu, J., Zhang, P., Zang, Y., Cao, Y., He, C., Wang, J., Wu, F., et al.: Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. CVPR (2025)
59. Yang, S., Chen, Y., Tian, Z., Wang, C., Li, J., Yu, B., Jia, J.: Visionzip: Longer is better but not necessary in vision language models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19792–19802 (2025)
60. Yu, S., Tang, C., Xu, B., Cui, J., Ran, J., Yan, Y., Liu, Z., Wang, S., Han, X., Liu, Z., et al.: Visrag: Vision-based retrieval-augmented generation on multi-modality documents. In: The Thirteenth International Conference on Learning Representations
61. Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., Yan, S.: Metaformer is actually what you need for vision. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10819–10829 (2022)

62. Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., et al.: Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9556–9567 (2024)
63. Yun, S., Ro, Y.: Shvit: Single-head vision transformer with memory efficient macro design. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5756–5767 (2024)
64. Zhai, X., Mustafa, B., Kolesnikov, A., Beyer, L.: Sigmoid loss for language image pre-training. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 11975–11986 (2023)
65. Zhang, Q., Cheng, A., Lu, M., Zhang, R., Zhuo, Z., Cao, J., Guo, S., She, Q., Zhang, S.: Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 20857–20867 (2025)
66. Zhang, Q., Cheng, A., Lu, M., Zhuo, Z., Wang, M., Cao, J., Guo, S., She, Q., Zhang, S.: [cls] attention is all you need for training-free visual token pruning: Make vlm inference faster. arXiv e-prints pp. arXiv–2412 (2024)
67. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6848–6856 (2018)
68. Zhang, Y., Fan, C.K., Ma, J., Zheng, W., Huang, T., Cheng, K., Gudovskiy, D., Okuno, T., Nakata, Y., Keutzer, K., et al.: Sparsevlm: Visual token sparsification for efficient vision-language model inference. arXiv preprint arXiv:2410.04417 (2024)
69. Zhang, Y.S.Y.Q.M., Zheng, X.L.J.Y.X., Wu, K.L.X.S.Y., Fu, R.J.C., Chen, P.: Mme: A comprehensive evaluation benchmark for multimodal large language models. arXiv preprint arXiv:2306.13394 (2021)
70. Zhu, J., Wang, W., Chen, Z., Liu, Z., Ye, S., Gu, L., Tian, H., Duan, Y., Su, W., Shao, J., et al.: Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. arXiv preprint arXiv:2504.10479 (2025)

A Additional Implementation Details

For the majority of our dense distillation experiments, we use SigLIP2-B16 NaFlex[5] as the teacher model. For the `+ Large Teacher` setting reported in Table 5(a), we resume training from the dense-distilled checkpoint and replace the teacher with a larger SigLIP2-400M model[6].

Fig. 5 illustrates the overall pipeline of the proposed dense distillation strategy.

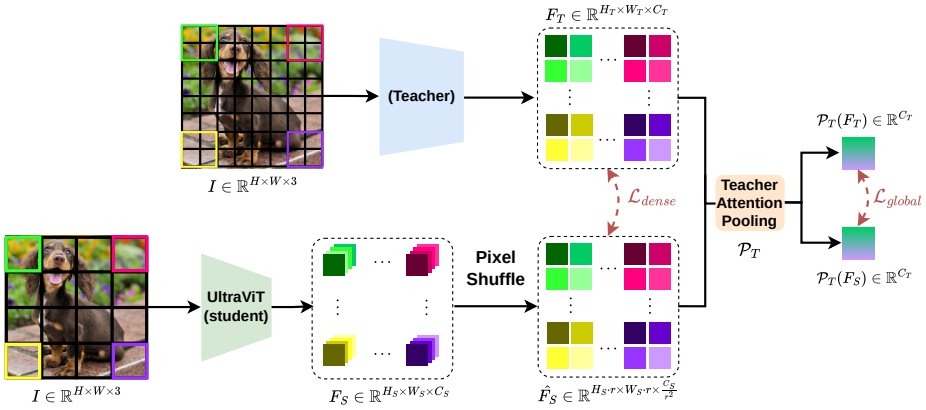

Fig. 5: Overview of the proposed dense distillation strategy. The teacher and UltraViT (student) extract multi-scale features from the same input. To address resolution mismatch, student features are re-aligned via Pixel Shuffle before applying a dense patch-wise alignment loss ($\mathcal{L}_{dense}$). Both re-aligned student and teacher features are further processed through the teacher's attention pooling module $\mathcal{P}_T$ to enforce global semantic consistency ($\mathcal{L}_{global}$).

B Additional Experiments

B.1 Extended OCR retrieval results

During the block selection process (Section 3.2), we compared different token mixer configurations in a zero-shot setting on multiple benchmarks, including image classification, OCR-based retrieval, and post LLaVA-OV fine-tuning.

Due to space constraints, the main paper only reports the averaged OCR retrieval score. Here, we detail the five OCR datasets used. These include TextCaps [45]

[5] `https://huggingface.co/google/siglip2-base-patch16-naflex`
[6] `https://huggingface.co/google/siglip2-so400m-patch16-512`

and four VisRAG-Ret [60] datasets: VisRAG-Ret-SlideVQA[7], VisRAG-Ret-InfoVQA[8], VisRAG-Ret-ChartQA[9], and VisRAG-Ret-ArxivQA[10].

Table 6 extends Table 1 from the main paper and reports the recall@1 performance on each individual OCR retrieval dataset, together with their average (AVG). While the configuration (C, C+S, A, A) achieves the highest average performance across the five OCR benchmarks, the selected configuration (C, C+S, CP, A) remains competitive, with only a marginal drop in accuracy, while offering better efficiency.

Table 6: Zero-shot OCR retrieval performance (R@1) for different mixer configurations across five datasets: TCAP (TextCaps), SVQA (SlideVQA), IVQA (InfoVQA), CHQA (ChartQA), and ARXQ (ArxivQA). AVG denotes the mean recall across all five datasets.

Mixer per Stage	Params (M)	Flops (B)	Inf.	TCAP	SVQA	IVQA	CHQA	ARXQ	**AVG**
C, C+S, CP, A	139.4	143.9	148.9	**66.6**	40.3	<u>25.6</u>	**22.2**	12.1	<u>33.4</u>
C, C, CP, A	142.4	144.7	147.8	<u>66.3</u>	41.0	24.1	<u>20.6</u>	**13.7**	33.1
C, C+S, A, A	144.8	154.7	126.7	**66.6**	<u>41.5</u>	**27.7**	19.0	13.1	**33.6**
C, C, C, C	143.4	141.8	171.0	66.0	37.8	24.7	17.5	11.2	31.4
C, C, C, A	143.4	147.2	160.7	65.8	**43.2**	25.1	15.9	<u>13.2</u>	32.6

B.2 Effect of different teachers for dense pretraining

In this section, we investigate the impact of utilizing different teacher models during the dense pretraining phase for UltraViT. We evaluate four variants using different teacher targets: SigLIP2-NF (naflex) [50], Perception Encoder (PE) [6], SigLIP2 [50], and TULIP [48]. The detailed results across eleven distinct vision-language benchmarks are presented in Table 7.

As shown in the table, employing strong, feature-rich vision models as teachers, such as SigLIP2 and TULIP, yields the best overall performance. Both SigLIP2 and TULIP consistently outperform the other variants across the majority of tasks. However, overall, our approach is robust to the choice of teacher.

B.3 Efficient VLM comparisons

We expand on our experiments presented in Table 4 of the paper comparisons with VLMs designed with the goal of being efficient. We present these results in Table 8. There, in addition to performance measurements on two popular benchmarks, we report the inference times for those VLMs' vision encoders for

[7] https://huggingface.co/datasets/openbmb/VisRAG-Ret-Test-SlideVQA
[8] https://huggingface.co/datasets/openbmb/VisRAG-Ret-Test-InfoVQA
[9] https://huggingface.co/datasets/openbmb/VisRAG-Ret-Test-ChartQA
[10] https://huggingface.co/datasets/openbmb/VisRAG-Ret-Test-ArxivQA

Table 7: UltraVLM performance under different teachers during the dense pretraining phase.

Model	Teacher	GQA	SQA	TextVQA	POPE	DocVQA	InfoVQA	RealWQA	MME	MMMU	ChartQA	MMSTAR
UltraVLM	SigLIP2-NF	58.5	76.9	57.1	86.3	63.9	39.5	53.1	68.9	33.1	62.8	44.6
UltraVLM	PE	58.8	77.3	57.3	86.3	63.6	38.1	55.3	65.8	32.0	63.8	43.5
UltraVLM	SigLIP2	59.5	78.8	59.4	86.7	65.3	40.3	57.3	69.4	33.9	64.5	43.6
UltraVLM	TULIP	59.5	78.6	59.6	86.7	65.1	40.9	56.3	69.1	33.6	64.1	43.6

Table 8: Vision-only and end-to-end TTFT; peak vision-only RAM.

	Vis. (ms) $512^2/1024^2$	TTFT (ms) $512^2/1024^2$	Peak RAM(MB)	Text VQA	ChartQA
LLaVA-OV(SigLIP)	127.8/2,200.0	483/2555	300/349	69.9	64.2
SmolVLM	36.6/690.1	100.6/755.1	77/92	60.5	62.8
Florence-VL	305.2/11,837.8	569/12101	422/584	69.1	70.7
FastVLM† (re-tr.)	10.9/65.9	98/154	66/91	62.9	66.0
UltraVLM (ours)	6.6/41.0	94/129	60/75	68.9	72.2

a fixed input resolution, the Time To First Token (TTFT) for those VLMs following their inference recipe (i.e. using the native resolution and patch strategy recommended by each model), and the peak memory consumption of each vision encoder during inference on-device. These resoluts demonstrate that, even compared to other highly efficient VLMs, our approach is significantly faster, showcasing the advantages of designing the vision encoder architecture specifically for on-device deployment.

B.4 On-device measurements methodology

As mentioned in the paper, on-device inference latency measurements were performed on a Samsung Galaxy S25 Ultra smartphone, with models compiled using the Qualcomm Neural Processing SDK. More specifically, to mitigate randomness in our measurements, for each model we ran 7 rounds of 100 inference passes. We subsequently ignore the fastest and slowest rounds, and report the average inference speed of the 5 remaining rounds. To decrease the impact of the device overheating and interference from other on-device factors, we include wait times of 10s between rounds, and reboot the device between model measurements.

For block-level measurements (Figure 2) we stack 10 blocks of a given type and perform measurements for various stages of our model. For each stage, the block's width follows UltraViT's configuration as described in Section 3, and

the input tensor corresponds to what those blocks would be processing in the corresponding stage of the model for an input image of resolution 512×512. E.g. for Stage 1 measurements, the blocks have width $C = 192$ and the input tensor to the block stack has shape $(192, 64, 64)$. For Figure 3, we vary the number of blocks per block type so that, across types, we have approximately the same number of parameters.

B.5 GPU and CPU performance

In this section we contrast UltraViT's performance with FastViT on CPUs and GPUs. These measurements are made for input images of resolution 512x512 on a Nvidia RTX4090 GPU and a Inter i7-14700K CPU. For GPU measurements we set the batch size to 96 (close to the limit of our GPU's capacity). For CPU measurements, we convert our model to onnx format and run single-image inferences. In both cases, we report the inference time per sample processed. Similarly to the on-device measurements, we conduct 7 rounds of measurements with 20 inferences per round, and average the latencies excluding the fastest and slowest rounds for robustness. As seen in 9, UltraViT, despite being designed specifically for optimal on-device speed, outperforms FastViT in both CPUs and GPUs, highlighting the efficiency of our proposed architecture.

Table 9: Inference times on CPU and GPU for UltraViT and FastViT.

Model	CPU (ms/sample)	GPU (ms/sample)
FastViT	118.0	2.5
UltraViT	**72.7**	**2.2**