AdaVFM: Adaptive Vision Foundation Models for Edge Intelligence via LLM-Guided Runtime Execution

Yiwei Zhao[1] Yi Zheng[2] Huapeng Su[2] Jieyu Lin[2] Stefano Ambrogio[2]
Cijo Jose[2] Michaël Ramamonjisoa[2] Patrick Labatut[2] Barbara De Salvo[2]
Chiao Liu[2] Phillip B. Gibbons[1] Ziyun Li[2]

[1]Carnegie Mellon University [2]Meta

Abstract

Language-aligned vision foundation models (VFMs) enable versatile visual understanding for always-on contextual AI, but their deployment on edge devices is hindered by strict latency and power constraints. We present AdaVFM, an adaptive framework for efficient on-device inference of language-aligned VFMs that dynamically adjusts computation based on scene context and task complexity. Our key insight is that the effect of model size reduction on performance is task-dependent in vision applications, motivating a runtime-adaptive execution strategy. AdaVFM integrates neural architecture search (NAS) into the language-aligned VFM backbone to enable lightweight subnet execution during runtime. A multimodal large language model (LLM) deployed on the cloud enables runtime control with a context-aware agent. This synergy allows efficient model adaptation under diverse conditions while maintaining strong accuracy. Extensive experiments on zero-shot classification and open-vocabulary segmentation demonstrate that AdaVFM achieves state-of-the-art accuracy-efficiency trade-offs, surpassing prior baselines by up to 7.9% in acc@1 on IN1K and 5.2% mIoU on ADE20K over the best models of comparable VFM sizes. For models with similar accuracy, AdaVFM further reduces average FLOPs by up to 77.9%.

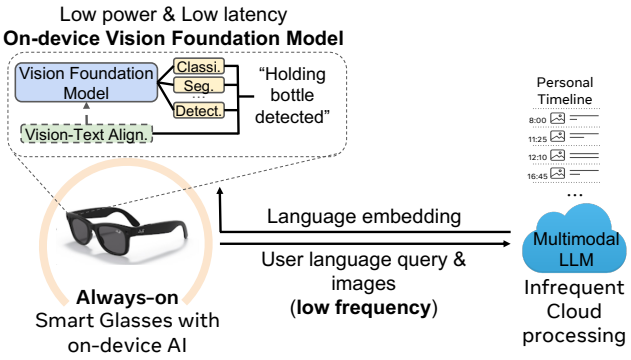

Figure 1. Always-on smart glasses with on-device VFM.

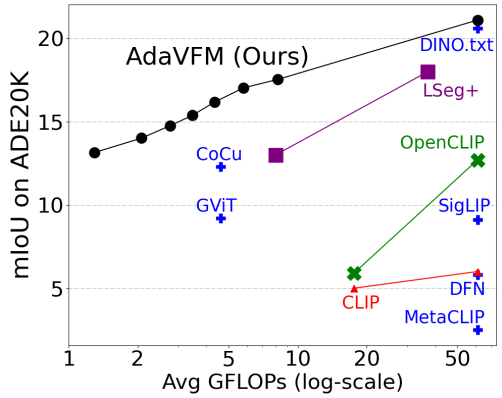

Figure 2. End-to-end mIoU comparison with prior work for open-vocabulary ADE20K segmentation [74]. Our design achieves significant improvement in accuracy-efficiency trade-off compared with state-of-the-art baselines, improving mIoU by up to 5.2% or reducing FLOPs by up to 77.9% over the best prior models.

1. Introduction

Recent advances in smartphones [19, 44], wearable devices [1, 33], augmented and virtual reality [2, 7], robotics [21, 62], and autonomous driving [36, 65] have accelerated the push toward enabling AI directly on edge devices. Modern multimodal AI models power a variety of vision and language applications, making efficient on-device deployment essential for real-time interaction and high-quality user experiences.

These trends have led to growing interest in *always-on contextual AI*—continuous, multimodal, context-aware systems that run persistently under tight energy and latency constraints [25, 32, 41, 52, 59]. As illustrated in Figure 1, smart-glass scenarios exemplify a collaborative edge–

*This preprint is currently under submission to CVPR'26. It is provided solely for faculty application review and should not be distributed.

DINOv2	ViT-S	ViT-B	ViT-L	ViT-g
#params	22M	86M	304M	1.8B
FLOPs	4.6G	17.6G	61G	700G
IN1K [12]	79.0 (-4.4)	82.1 (-1.3)	83.5 (+0.1)	83.4
food101 [4]	89.1 (-5.6)	92.8 (-1.9)	94.3 (-0.4)	94.7
cal101 [16]	97.0 (-0.6)	96.1 (-1.5)	97.5 (-0.1)	97.6
pets [47]	95.1 (-1.6)	96.2 (-0.5)	96.6 (-0.1)	96.7

Table 1. DINOv2 [46] results across datasets. Numbers in parentheses show drops relative to ViT-g. Simple tasks (Cal101, Pets) exhibit small drops (≤ 1.6) as model sizes decrease, while moderate/complex tasks (IN1K, Food101) show larger drops (up to 5.6).

cloud paradigm: the edge device executes a lightweight **vision foundation model** for real-time perception, while tasks requiring heavy computation or broad world knowledge are offloaded to a cloud-based multimodal large language model (MM-LLM).

Deploying vision foundation models (VFMs) on edge devices, however, remains challenging—and enabling *language-aligned* VFMs is even more difficult. These models are large, computation-intensive, and designed for high-capacity vision tasks, while edge devices operate under strict compute, memory, and power limits. Many interactive applications additionally impose stringent latency requirements; for example, contextual AI systems must respond within sub-second timescales (typically 200–1000 ms) to maintain perceived interactivity and usability [2, 23, 55, 68]. Directly executing full-scale language-aligned VFMs under such constraints is infeasible, and prior attempts that simply shrink model capacity [54, 61] often suffer significant performance degradation (see Section 2.3). Consequently, continuously invoking such VFMs on the edge remains prohibitively expensive despite their strong utility.

We observe that not all edge tasks require invoking a full language-aligned VFM. Simpler tasks can be handled effectively by smaller models, whereas only more complex tasks necessitate large-scale backbones. As shown in Table 1, DINOv2 [46]—a representative vision foundation model—exhibits minimal performance degradation on simpler datasets such as Pets and Cal101 when using smaller model variants, while more challenging datasets incur substantially larger drops.

A similar pattern emerges in open-vocabulary segmentation. In Fig. 3, two VFMs with 50M and 300M parameters are evaluated on ADE20K [74] under two settings: the original 150-class segmentation task and a grouped 9-class task using coarse super-classes (Section 6.2). While both models achieve comparable accuracy on the simpler 9-class task, their performance diverges significantly on the full 150-class benchmark. These results highlight the importance of ***adaptive model selection*** on the edge based on ***task difficulty***, rather than relying on a single, fixed model configuration.

In this paper, we leverage *the sensitivity of open-vocabulary tasks to model capacity* to enable efficient deployment of language-aligned vision foundation models (VFMs) on edge devices. We introduce *runtime adaptive selection*, which dynamically chooses the most suitable execution scheme for language-aligned VFMs based on task requirements and resource availability. To support this, we integrate Neural Architecture Search (NAS) into the VFM architecture and develop a two-stage training pipeline for NAS-based VFM distillation and vision-text alignment, enabling practical open-vocabulary use cases on the edge.

To further exploit this adaptive mechanism, we design a runtime management agent powered by a large language model (LLM). Although invoked only at low frequency, the agent effectively infers scene context, analyzes task properties, and synthesizes meta-information from user and environmental inputs. Using this information, it recommends an optimized execution scheme for the edge model, balancing accuracy and efficiency in dynamic real-world scenarios.

In summary, we present **AdaVFM**, a unified runtime execution framework with a high-quality training pipeline featuring: (*i*) NAS-integrated adaptive VFM deployment for efficient edge execution; (*ii*) LLM-based semantic understanding for improved text embeddings; and (*iii*) LLM-guided runtime subnet selection to fully exploit VFM adaptiveness. End-to-end evaluations (Figure 2 and Tab. 4) show that AdaVFM achieves a significantly better accuracy-efficiency trade-off than prior work [15, 20, 26, 28, 49, 64, 66, 67, 72], with up to 7.9% IN1K and 5.2% ADE20K accuracy gains and up to 77.9% FLOPs reduction.

2. Background and Related Work

2.1. Always-on Contextual AI on Edge Devices

The emergence of wearable edge platforms is driving the need for *always-on contextual AI* that can operate continuously under strict power, thermal, and memory constraints. As illustrated in Fig. 1, the system relies on a lightweight *on-device vision foundation model* for real-time perception tasks such as classification, detection, and segmentation, thanks to recent progress in efficient models on mobile hardware [24, 37, 40, 56]. These visual representations are aligned with language on-device to support seamless user interaction [8, 35], while more computationally intensive tasks—such as multimodal reasoning, long-term memory, and access to broad world knowledge—are delegated to a cloud-based multimodal LLM [3, 13, 57]. This distributed edge–cloud design reflects limitations of continuous cloud streaming and aligns with prior work showing that persistent offloading is impractical for wearable devices due to thermal and energy constraints [25, 32, 43]. Instead, sparse visual summaries and infrequent user queries are transmitted to the cloud, enabling the construction of a personal timeline in the spirit of memory-augmented multimodal

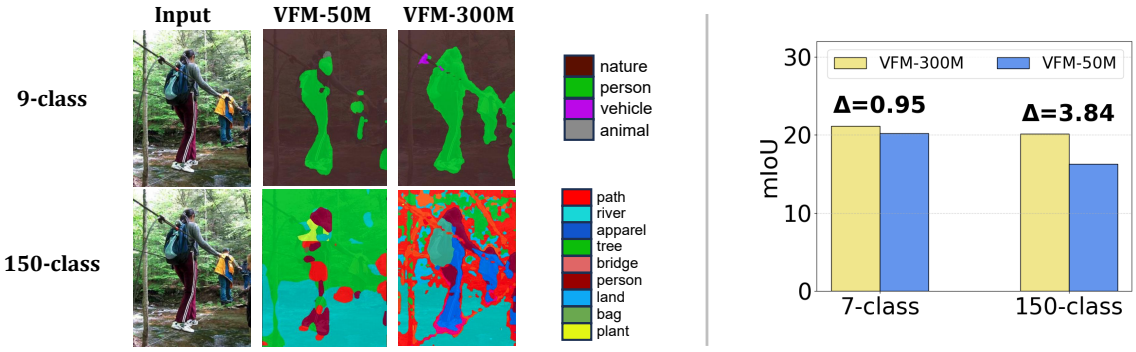

Figure 3. Open-vocabulary segmentation results on ADE20K [74] using 50M and 300M-parameter VFMs (300M: DINO.txt [28]; 50M: similarly distilled and fine-tuned). Results are shown for both the original 150-class and the grouped 9-class setting (grouping based on WordNet [17], described in Section 6.2). **Left**: Both models perform similarly on the simpler 9-class task, but in the complex 150-class setting, VFM-300M yields *clear object boundaries* while VFM-50M produces *blurred* ones. **Right**: Simpler tasks incur minimal performance drop (0.95) with smaller models, whereas complex tasks show a larger drop (3.84).

systems [6, 14]. Consequently, the always-on contextual AI requires efficient, multimodal, and text-aware foundation models capable of running on the edge while collaborating seamlessly with stronger cloud-hosted LLMs.

2.2. Vision Foundation Models

Vision foundation models have become a central paradigm in contemporary computer vision, offering unified representations that generalize effectively across a wide spectrum of tasks and domains. These models are typically trained at scale using self-supervised learning (SSL) objectives, which exploit the intrinsic structure of visual data to learn from large, unannotated image corpora. Without explicit supervision, SSL-based foundation models can leverage virtually unlimited data sources, yielding robust and semantically rich visual embeddings. Recent advances such as DINOv2 [46], MAE [22], and iBOT [75] demonstrate the effectiveness of SSL in learning transferable features that rival or surpass those obtained via supervised pretraining methods like CLIP [49] or PaLI [11]. Such models have shown strong transferability, enabling a single frozen encoder to perform competitively across diverse downstream tasks without task-specific fine-tuning [58]. Nonetheless, the remarkable performance of large-scale vision foundation models often comes at the cost of substantial computational, memory, and energy requirements. Addressing these limitations, we present *adaptive vision foundation models* that are capable of achieving state-of-the-art VFM performance with edge devices.

2.3. Contrastive Learning

Contrastive learning methods such as CLIP [28, 49, 71] enable zero-shot and open-vocabulary capabilities in vision foundation models by aligning visual and textual representations. A standard CLIP architecture consists of a Vision Encoder and a Text Encoder, both of which are computationally and memory intensive, making direct deployment on resource-constrained edge devices impractical. Simply pruning or shrinking these models often leads to significant accuracy degradation [54, 61].

A key property of CLIP relevant to edge deployment is that the Vision Encoder and Text Encoder operate at fundamentally different frequencies. In always-on contextual AI systems (Fig. 1), user prompts or scenes change infrequently, whereas perception runs continuously:

- **Vision Encoder (high-frequency):** Processes every incoming frame and must meet strict low-latency requirements (typically 200–1000 ms).
- **Text Encoder (low-frequency):** Invoked only when user prompts or scenes change, tolerating much higher latency.

This frequency asymmetry creates a design opportunity: the Vision Encoder must be heavily optimized for real-time edge inference, while the Text Encoder can remain larger without impacting responsiveness. Leveraging this imbalance is crucial for deploying CLIP-style zero-shot models efficiently on edge devices.

2.4. Neural Architecture Search

Neural Architecture Search (NAS) provides a principled framework for building models with adaptive capacity, making it well suited for resource-constrained edge deployment. A key advantage is its support for ***runtime subnet selection***, where a single supernet encompasses subarchitectures of varying computational costs. At inference time, the system selects an appropriate subnet based on task complexity or available resources, enabling flexible accuracy–efficiency trade-offs. While early NAS methods used evolutionary [50] or reinforcement-learning strategies [76], later weight-sharing supernets [5, 9, 70] made NAS practical for adaptive models. In this work, we leverage NAS to develop text-aligned, adaptive vision foundation models tailored for always-on edge scenarios.

3. Method

3.1. Overall System Design

We illustrate the overall AdaVFM runtime system in Fig. 4, designed for open-vocabulary vision tasks. The system comprises two complementary components: (1) an **Adaptive Vision Foundation Model (AdaVFM)** deployed on the edge device, operating in an always-on manner to handle high-frequency vision inference; and (2) a **cloud-based multimodal LLM execution agent**, invoked only infrequently to perform scene and context understanding, model selection, and text-embedding generation. Together, these components enable efficient, adaptive, and open-vocabulary inference in dynamic real-world scenarios.

3.2. On-Edge Adaptive Foundation Model

Execution Flow. As shown in Fig. 4, the cloud-side multimodal LLM receives a temporally sparse stream of images and/or user language interactions to infer the user's *scene and context* (e.g., *driving*). Given this context, the LLM identifies relevant open-vocabulary targets (e.g., *person, cars, signs*) and, using augmented model metadata from Fig. 5, produces two outputs: (*i*) a set of *text embeddings* for the selected open-vocabulary concepts, and (*ii*) a *recommended execution scheme* indicating the appropriate model size or subnet configuration for the edge device. The edge-side Vision Encoder then operates continuously, processing every incoming frame and computing cosine similarity between the vision embeddings and the cloud-generated text embeddings to enable real-time open-vocabulary inference.

Selective Execution Scheme. As illustrated on the left side of Fig. 4, AdaVFM enables dynamic adaptation on the edge through a NAS-based design space that exposes multiple execution paths with different computational costs. At runtime, the system selects an appropriate subnet from this search space based on the LLM-recommended execution scheme, which reflects current task difficulty, scene dynamics, and latency-accuracy requirements. This selection determines which subnet is activated for the next rounds of inferences. By invoking lighter or heavier subnets as needed, the edge device can balance responsiveness and accuracy in always-on settings under strict resource constraints.

Open-Vocabulary Vision Tasks. With the subnet selected and text embeddings supplied by the cloud, the Vision Encoder performs open-vocabulary vision tasks using a CLIP-like contrastive pipeline, similar to LiT [71] and DINO.txt [28]. Each incoming frame is encoded into visual tokens and compared against the text embeddings using cosine similarity. For zero-shot classification, the comparison is performed using the global [CLS] token, whereas for open-vocabulary dense tasks such as segmentation, patch-level embeddings are contrastively matched to produce per-pixel similarity maps. This unified contrastive interface enables flexible zero-shot understanding across both sparse and dense recognition tasks.

3.3. LLM-Guided Efficient Runtime Execution

A central component of our system is the **LLM-guided runtime management agent**, invoked at low frequency during execution (e.g., every few minutes). As shown in Fig. 5, this agent is responsible for: (*i*) producing a *semantic understanding* of the current scene to guide the Text Encoder, and (*ii*) selecting *efficient* model configurations and execution schemes for the Vision Encoder.

The agent runs in the cloud and periodically receives contextual signals such as temporally sparse image samples and/or user language interactions. From these inputs, the LLM infers high-level scene context (e.g., *"driving"*) and generates a set of *context-based grouped classes*, as illustrated in the top block of Fig. 5. These coarse concepts are then mapped to fine-grained categories guided by vision datasets (e.g., *vehicle* $\rightarrow$ {*jeep, minivan*}) using a second LLM query, yielding a refined set of open-vocabulary targets relevant to the scene. In our experiments, we incorporate categories from ImageNet [12] and ADE20k [74], organized through WordNet [17].

Semantic Understanding of a Scene. Zero-shot classification and open-vocabulary segmentation in CLIP-style models [28, 49, 71] require passing candidate class names into the Text Encoder. Instead of blindly enumerating the entire vocabulary, our runtime agent uses LLM-driven semantic reasoning to filter out implausible classes based on the context. This reduces noise, improves precision, and resolves ambiguous class definitions. For example, given a scene context *"sports field"*, the agent selects *"field"* rather than *"grass"* as the more contextually appropriate phrase. This filtering step corresponds to the first stage of Fig. 5, where dataset classes are aligned to grouped semantic categories.

Difficulty-Based Model Selection. The size and composition of the selected class set directly reflect the *difficulty* of the current open-vocabulary task. Using precomputed confusion matrices (Fig. 5, bottom-left) and an accuracy tolerance threshold, the agent estimates the expected performance of candidate model configurations. It then selects the most efficient subnet from the NAS search space whose grouped-class accuracy meets the required tolerance. This process reduces the effective task complexity and enables adaptive selection of lighter or heavier Vision Encoder subnets depending on the scene.

Implementation Details. In practice, due to dataset constraints, we simulate scene context using ground-truth annotations (e.g., ADE20K [74] location labels such as *"airport"* or *"art_gallery"*). These annotations serve as surrogates for real-world contextual signals—such as GPS metadata, user utterances, or device histories—which would be

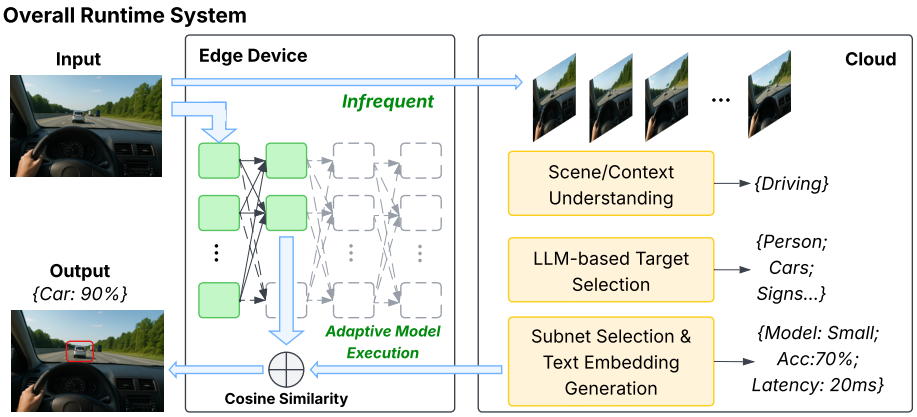

Figure 4. Overview of the proposed AdaVFM. **Left**: Edge-side execution, where the adaptive Vision Encoder follows cloud agent instructions to select an efficient execution scheme and perform vision-text contrastive inference. **Right**: Cloud-side execution, where the agent uses scene/context information to generate semantic understanding for the Text Encoder and execution guidance for the Vision Encoder.

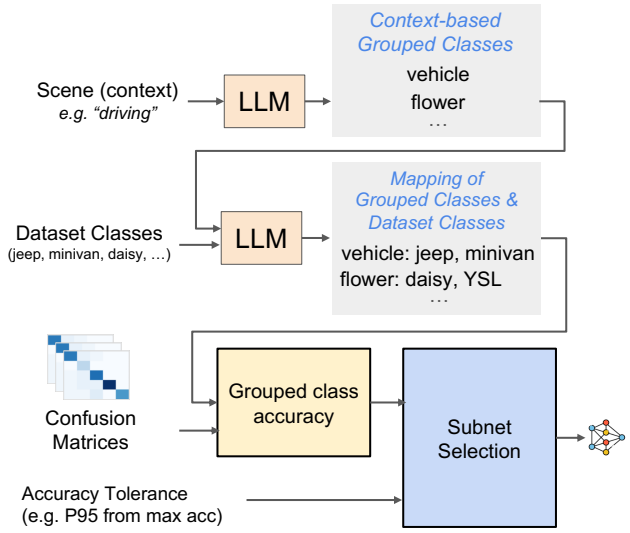

Figure 5. Operation flow: LLM-based runtime management agent.

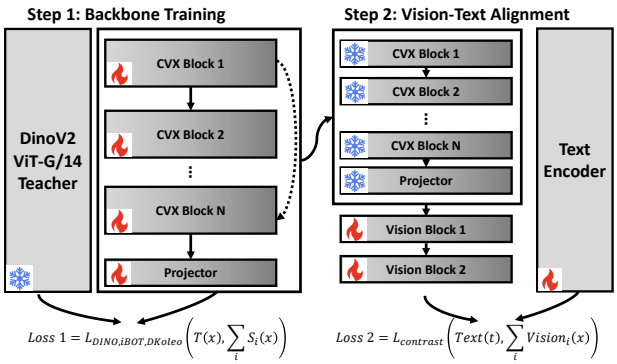

Figure 6. Overview of the training pipeline. The vision backbone is first distilled from a foundation model (DINOv2 [46]), followed by vision-text alignment using CLIP. Both stages employ NAS and sandwich sampling [69].

available in deployed scenarios. Extending the agent to incorporate richer user activity and historical usage traces remains an avenue for future work.

4. Training

To enable efficient and adaptive visual inference on edge devices, we integrate Neural Architecture Search (NAS) into the vision foudation models, as illustrated in Fig. 6. The key motivation is that simpler recognition tasks can often be handled by smaller networks with minimal accuracy degradation—a trend widely observed in prior work (e.g., [38]) and further verified in our open-vocabulary setting (see Section 6.2). Leveraging this observation, our goal is to train a single *supernet* from which subnets of varying computational cost can be selected at runtime.

Stage 1: Backbone Distillation. We adopt the Once-for-All (OFA) NAS framework [5], which trains a supernet spanning a wide range of model widths and depths. As shown in Fig. 6 (left), we distill a DINOv2 ViT-G/14 teacher [46] into a ConvNeXt-based [60] supernet. ConvNeXt is selected for its parameter and compute efficiency on edge devices, and because its convolutional structure makes NAS integration substantially more tractable than in transformer-based search spaces [34, 53]. To ensure consistent output dimensionality across all subnets, we add a lightweight linear projector on top of the ConvNeXt backbone. During this stage, all ConvNeXt blocks and the projector are trainable (indicated by flame icons in the figure), and we optimize a DINO-style distillation loss:

$$\mathcal{L}_1 = \mathcal{L}_{\text{DINO,iBOT,Koleo}}\big(T(x), \sum_i S_i(x)\big),$$

where $T(\cdot)$ denotes teacher features and $S_i(\cdot)$ represents the outputs of sampled subnets.

Block	Dim.	Depth	Stride
Downsample-1	$48 \sim 96$	1	4
ConvNext-Block-1	$48 \sim 96$	3	1
Downsample-2	$96 \sim 192$	1	2
ConvNext-Block-2	$96 \sim 192$	3	1
Downsample-3	$192 \sim 384$	1	2
ConvNext-Block-3	$192 \sim 384$	$9 \sim 27$	1
Downsample-4	$384 \sim 768$	1	2
ConvNext-Block-4	$384 \sim 768$	3	1
#params	$6.2\text{M} \sim 52.3\text{M}$		
FLOPs	$1.29\text{G} \sim 9.28\text{G}$		

Table 2. Our NAS search space for the vision backbone. Depth denotes the number of layers in each block; Dim. represents the dimensionality of each block.

Stage 2: Vision-Text Alignment. Next, following the CLIP-style alignment procedure in DINO.txt [28], we freeze the trained ConvNeXt blocks and projector (indicated by snowflake icons in Fig. 6, right), and train only the lightweight vision blocks that map visual features into the text embedding space. This stage uses a contrastive loss:

$$\mathcal{L}_2 = \mathcal{L}_{\text{contrast}}\big(\text{Text}(t), \sum_i \text{Vision}_i(x)\big).$$

This two-stage pipeline ensures that the NAS supernet is compatible with CLIP-style open-vocabulary inference.

Sandwich Sampling for Supernet Optimization. To efficiently train the NAS search space, we follow the sandwich sampling rule [69]. In each iteration, we sample the *largest* and *smallest* subnets—corresponding to the two extremes in Table 2—along with 2 additional subnets sampled uniformly from intermediate configurations. We compute gradients for all sampled subnets and average them to update shared weights. This training strategy enables the supernet to support a wide range of subnets with minimal accuracy gaps between supernet and stand-alone models.

Training Details. For backbone distillation, we follow DINOv2 [46] within our NAS framework, using the LVD-142M dataset. We train the supernet for 625k iterations with a batch size of 4096 and a learning rate of 0.0002. For vision–text alignment, we adopt the setup of DINO.txt v2 [28] and train on the MetaCLIP dataset under the same NAS framework. This stage is trained for 100k iterations with a batch size of 4096 and a learning rate of 0.0004.

5. Evaluation

5.1. Experimental Setup for Inference

Zero-Shot Classification. We evaluate zero-shot classification on the ImageNet-1K (IN1K) [12], Food-101 [4], Describable Textures Dataset (DTD) [10] and Oxford Pets [47]. For inference, we follow the procedure described in DINO.txt [28]: at test time, the class names are fed into the Text Encoder to obtain text embeddings, which are then compared with the global image descriptors produced by the Vision Encoder using cosine similarity.

Subnet	Depths	Dimensions	#params	FLOPs
MIN	[3,3,9,3]	[48,96,192,384]	6.2M	1.29G
TINY	[3,3,9,3]	[72,144,288,576]	16.1M	2.86G
SMALL	[3,3,9,3]	[96,192,384,768]	28.7M	5.05G
BASE	[3,3,15,3]	[96,192,384,768]	35.9M	6.46G
LARGE	[3,3,27,3]	[96,192,384,768]	50.3M	9.28G

Table 3. Five representative NAS subnets selected from the search space described in Section 4 and Tab. 2, with sizes corresponding to model variants (-N/-T/-S) in ConvNeXt-v2 [60].

Open-Vocabulary Segmentation. We evaluate open-vocabulary segmentation on the ADE20K dataset [74], which includes scene annotations [73] utilized by our LLM-based runtime agent. The evaluation metric for effectiveness is mean Intersection-over-Union (mIoU).

Metrics for Efficiency. In this paper, we use the number of floating-point operations (FLOPs) and the number of model parameters (#params) to quantify execution efficiency in terms of computational cost and memory usage.

Evaluated NAS Subnets. Given the vast number of possible subnets in the NAS search space, we select five representative subnets to illustrate the flexibility and performance of our NAS model, as shown in Table 3. TINY, SMALL and LARGE correspond to ConvNeXt-v2-N, -T and -S [60], respectively, while the remaining subnets are interpolated to span the full NAS search space.

5.2. Comparison with State-of-the-Art

Zero-Shot Classification. We compare AdaVFM with state-of-the-art baselines in Table 4. Our approach achieves significant improvements over all baselines except DINO.txt [28], where the performance gap is primarily due to ***model size differences*** (50M LARGE vs. 300M DINO.txt). Scene and context information are excluded in Table 4 since they are not available for all datasets.

Open-Vocabulary Segmentation. Beyond segmentation accuracy, we evaluate the effectiveness-efficiency trade-off of our design against baselines. Using scene/context information from ADE20K [74], the LLM-based runtime agent can contribute to both semantic understanding and adaptive execution. Figure 2 presents an end-to-end comparison, showing that AdaVFM achieves significant gains in the effectiveness-efficiency trade-off.

6. Ablation Study

6.1. NAS-Enabled Vision Backbone

As described in Section 4, we integrate Neural Architecture Search (NAS) into the distillation of the vision backbone. Table 5 reports results for five representative subnets

	Model	#param	Food101	DTD	Pets	IN1K
AdaVFM	MIN	6M	73.8	54.1	91.4	65.5
	TINY	16M	78.3	56.3	92.1	69.8
	SMALL	29M	79.2	57.2	92.8	71.3
	BASE	36M	80.5	57.9	93.3	72.4
	LARGE	50M	81.7	58.3	93.3	73.3
Baselines	CLIP [49]	86M	-	42.9	85.0	61.9
	CuPL [48]	86M	-	48.0	87.2	63.4
	Waffle [51]	86M	-	51.0	88.1	63.5
	LaFTer [42]	86M	-	46.1	82.7	64.2
	ProText [30]	86M	-	50.7	89.0	64.9
	NoLA [27]	86M	-	56.1	89.3	65.4
	OpenCLIP [26]	304M	-	55.6	83.9	74.0
	DINO.txt [28]	304M	93.4	67.5	95.3	81.6

Table 4. Zero-shot classification results. AdaVFM **outperforms all baselines** by at least 7.9% IN1K acc@1 except [26, 28], mainly due to large difference in model sizes ($\leq$50M vs. 300M ViT-L).

Dataset	NAS-based Distillation					Standalone
	MIN	TINY	SMALL	BASE	LARGE	LARGE
IN1K [12]	69.4(-9.4)	74.0	76.6	77.7	78.8(-0.3)	79.1
cal101 [16]	95.2(-1.3)	95.5	96.0	96.6	96.5(+0.3)	96.2
Pets [47]	91.4(-3.3)	92.9	93.9	94.6	94.7(+0.2)	94.5
Cifar10 [31]	93.7(-4.4)	95.8	97.1	97.9	98.1(-0.1)	98.2
DTD [10]	76.2(-4.8)	79.4	80.4	80.7	81.0(-0.3)	81.3
SUN397 [63]	69.7(-6.9)	73.2	75.3	76.2	76.6(0.0)	76.6
aircraft [39]	61.9(-9.0)	66.1	68.9	69.8	70.9(+0.1)	70.8
Cifar100 [31]	78.7(-10.2)	83.1	86.1	87.9	88.9(-0.2)	89.1
Food101 [4]	77.4(-11.1)	82.6	86.1	87.5	88.5(-0.1)	88.6
cars [18]	66.8(-18.4)	76.9	82.1	84.3	85.2(-0.9)	86.1

Table 5. Performance of our NAS-distilled vision backbone across multiple vision tasks. Except IN1K (kNN), all other datasets use logreg as classifiers. Compared with standalone-distilled models (last column), the NAS-based model shows only ***marginal*** drops (differences in parentheses under LARGE). Sensitivity to model size varies across datasets (differences under MIN), showing the importance of adaptive model selection based on task complexity.

selected from the backbone. Compared with standalone-distilled models, our NAS-based subnets achieve comparable accuracy. Moreover, since these subnets share weights within the NAS supernet, they enable runtime selection on edge devices. We also observe that sensitivity to model size varies across datasets, reinforcing the principle of adaptive VFM execution at the edge.

6.2. Granularity of User-Defined Classes

In close-vocabulary tasks, the main sources of variation are determined by the datasets and workloads. In contrast, for open-vocabulary tasks, an additional factor—how users define and execute the task at runtime—can influence task difficulty, alongside the underlying dataset distribution.

In this section, we study the impact of how users define ***class names*** and their ***granularities***. We conduct experiments on ImageNet-1K (IN1K) [12] and ADE20K [74]. For IN1K, we evaluate two settings: the original 1000-class classification and a 7-class setting using very general class names. The hierarchical grouping of class names is based on synsets in WordNet [17]. For ADE20K, we also evaluate

Dataset	MIN	TINY	SMALL	BASE	LARGE
IN1K [12]	65.5	69.8	71.3	72.4	73.3
IN-WN7	73.2	73.8	74.1	74.1	74.3
ADE [74]	11.35	13.08	13.25	14.67	16.30
ADE-WN9	19.01	19.62	19.38	20.23	20.22

Table 6. Acc@1 on IN1K [12] and mIoU on ADE20K [74]. IN1K and ADE denote the original datasets, while IN-WN7 and ADE-WN9 represent the 7-class and 9-class grouped versions based on WordNet [17] as described in Section 6.2. Coarse-grained class groupings (7- and 9-class) exhibit smaller performance drops as model size decreases, indicating reduced task difficulty.

	GPT5	GPT5-mini	Gemini2.5	GPT5-Top25
MIN	13.11 (+1.76)	12.75	12.57	13.35
TINY	14.54 (+1.46)	14.28	14.19	14.38
SMALL	14.71 (+1.46)	14.41	14.37	14.48
BASE	16.13 (+1.46)	15.77	15.79	15.80
LARGE	17.53 (+1.23)	17.13	17.20	16.82
ViT-L	21 (+0.86)	20.68	20.68	19.91

Table 7. mIoU on ADE20K [74] with different LLM-based semantic filtering methods. Incorporating LLM semantics **notably improves** open-vocabulary segmentation for both AdaVFM and DINO.txt baseline [28], with *smaller* models benefiting the most.

two settings: the original 150-class segmentation and a 9-class segmentation based on WordNet grouping with granularity similar to the 7-class IN1K setting.

As shown in Table 6, while the original classification and segmentation tasks experience significant performance drops with smaller model sizes, the 7- or 9-class tasks with coarse-grained labels maintain strong performance even with smaller inference models. These results demonstrate that user-defined class granularity has a significant impact on the difficulty of open-vocabulary tasks. For simpler tasks, smaller models can achieve substantially higher efficiency while maintaining nearly the same accuracy. This observation is also visualized in Figure 3.

Consequently, NAS-enabled subnet selection is crucial for enabling efficient inference on edge devices across tasks of varying complexity.

6.3. Agent-Assisted Runtime Scene Understanding

As introduced in Section 3.3, an LLM-based agent is employed on the cloud to assist the AdaVFM, by providing semantic understanding to improve the Text Encoder. In this section, we simulate this process using scene information [73] from the ADE20K dataset [74]. For each scene, we use a commercial LLM to ***filter out*** impossible class names and ***resolve ambiguity*** by selecting contextually appropriate terms. The refined semantic set is then provided to AdaVFM for open-vocabulary segmentation.

As shown in Table 7, incorporating LLM-based grouping and filtering of vocabulary yields substantial segmentation gains for *both* our adaptive AdaVFM and non-adaptive

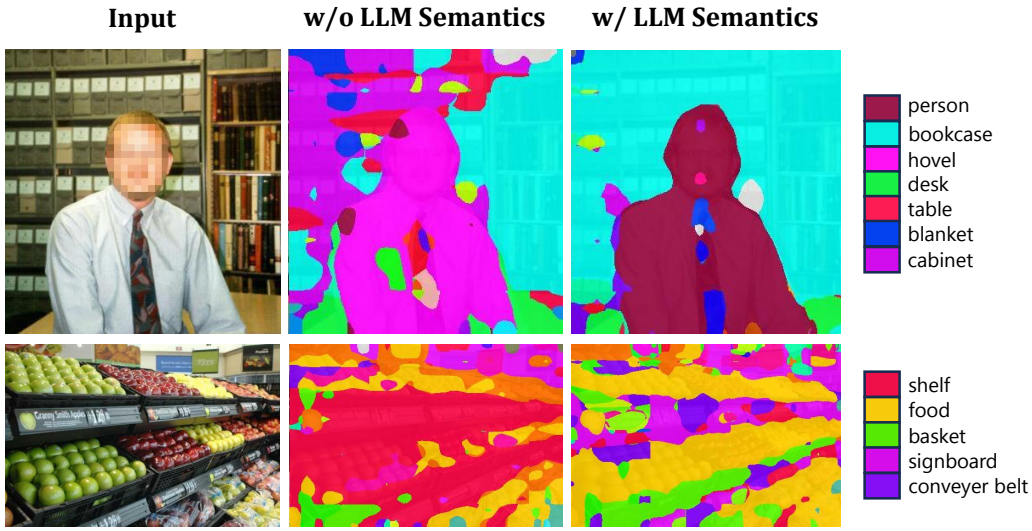

Figure 7. Open-vocabulary segmentation on ADE20K [74], comparing results with and without LLM-based scene understanding. LLM-based semantic understanding yields more *regular and coherent* segmentation masks.

baselines (e.g., [28]). Our approach achieves up to 1.76 mIoU improvement over non-LLM methods. Figure 7 further shows that LLM-driven semantic understanding produces more regular, coherent, and semantically consistent segmentation masks.

Another observation is that *smaller* models *benefits more* from the semantic filtering. Our approach achieves up to 1.76 mIoU improvement for small NAS subnets and up to 1.23 mIoU for large subnets. This demonstrates that our AdaVFM design is extremely suitable for edge intelligence.

Among the commercial LLMs we evaluated, GPT-5 [45] achieves the best performance. Using alternative LLMs (e.g., Gemini-2.5 [29]) or smaller variants (GPT-5-mini) results in noticeable performance degradation. The first three methods in Table 7 yield an indefinite number of candidate classes based on scene context, while the fourth method restricts predictions to a fixed top-25 class subset, which leads to poorer performance.

6.4. Agent-Assisted Runtime Model Selection

In AdaVFM, the execution of edge language-aligned VFM is selected based on the context. A hardware execution mode is specified first—for example, high-performance mode always requires the maximal accuracy (of the largest subnet); balanced mode requires estimated $\geq 90\%$ of maximal accuracy; and power-saving mode requires $\geq 80\%$. Given the current context, the cloud LLM computes the optimal subnet that satisfies the mode requirements in the current scene and instructs the edge VFM accordingly. This selection leverages a scene-wise confusion matrix obtained during training on large datasets (LVD-142M [46]).

Fig. 8 presents an ablation study on the functionalities of the cloud LLM agent. With identical text embeddings,

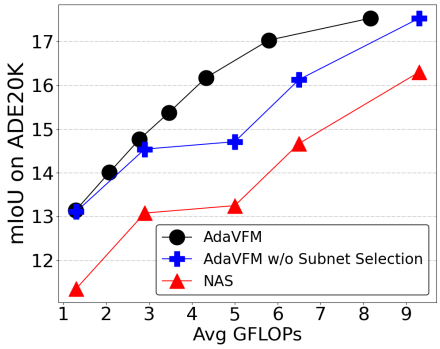

Figure 8. Impact of the LLM runtime agent on open-vocabulary segmentation on ADE20K [74]. **NAS** uses text-aligned subnets without the LLM agent; **AdaVFM w/o Subnet Selection** uses LLM only for semantic class filtering; **AdaVFM** uses the LLM for both semantic class filtering and adaptive subnet selection. Efficiency gains show that runtime adaptive selection is critical.

enabling adaptive selection on the edge VFM significantly improves runtime efficiency (AdaVFM vs. w/o Subnet Selection). Integrating all components yields the superior end-to-end performance of AdaVFM, as shown in Fig. 2.

7. Conclusion

We introduced AdaVFM, an adaptive framework for efficient deployment of vision foundation models on edge devices. By integrating neural architecture search (NAS) into the vision backbone and leveraging an LLM-based runtime agent, AdaVFM dynamically selects optimal execution schemes based on context and task complexity. This design enables real-time adaptation with minimal accuracy loss under strict hardware constraints. Experiments on zero-shot classification and open-vocabulary segmentation show

that AdaVFM achieves superior accuracy–efficiency trade-offs over prior VLM methods (up to 5.2% mIoU improvements or up to 77.9% reduced FLOPs in ADE20K), highlighting the importance of adaptive execution for context-aware edge intelligence.

References

[1] Salar Abbaspourazad, Oussama Elachqar, Andrew Miller, Saba Emrani, Udhyakumar Nallasamy, and Ian Shapiro. Large-scale training of foundation models for wearable biosignals. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[2] Michael Abrash. Creating the future: Augmented reality, the next human-machine interface. In *2021 IEEE International Electron Devices Meeting (IEDM)*, pages 1–11, 2021. 1, 2

[3] Jean-Baptiste Alayrac et al. Flamingo: A visual language model for few-shot learning. In *NeurIPS*, 2022. 2

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 2, 6, 7

[5] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. 3, 5

[6] Ziqiao Chang et al. Memory-augmented multimodal language models for long-horizon understanding. *arXiv preprint arXiv:2401.XXXX*, 2024. 3

[7] Jianchuan Chen, Jingchuan Hu, Gaige Wang, Zhonghua Jiang, Tiansong Zhou, Zhiwen Chen, and Chengfei Lv. Taoavatar: Real-time lifelike full-body talking avatars for augmented reality via 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10723–10734, 2025. 1

[8] Junnan Chen et al. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 2

[9] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on computer vision*, pages 12239–12248, 2021. 3

[10] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 6, 7

[11] Mostafa Dehghani, Yi Tay, Victor Mustar, Neil Houlsby, et al. Pali: A jointly-scaled multilingual language–image model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2, 4, 6, 7, 12

[13] Danny Driess et al. Palm-e: An embodied multimodal language model. In *ICLR*, 2023. 2

[14] Linxi Fan et al. Chorus: Multi-modal memory-augmented models. *arXiv preprint arXiv:2312.XXXX*, 2023. 3

[15] Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander T Toshev, and Vaishaal Shankar. Data filtering networks. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[16] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007. 2, 7

[17] Christiane Fellbaum. *WordNet: An electronic lexical database*. MIT press, 1998. 3, 4, 7, 12

[18] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017. 7

[19] Manfred Georg, Garrett Tanzer, Esha Uboweja, Saad Hassan, Maximus Shengelia, Sam Sepah, Sean Forbes, and Thad Starner. Fsboard: Over 3 million characters of asl fingerspelling collected via smartphones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13897–13906, 2025. 1

[20] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. Scaling open-vocabulary image segmentation with image-level labels. In *European conference on computer vision*, pages 540–557. Springer, 2022. 2

[21] Raktim Gautam Goswami, Prashanth Krishnamurthy, Yann LeCun, and Farshad Khorrami. Robopepp: Vision-based robot pose and joint angle estimation through embedding predictive pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6930–6939, 2025. 1

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[23] Eric Heim et al. Measuring and mitigating latency in augmented reality systems. In *ISMAR*, 2019. 2

[24] Andrew Howard et al. Searching for mobilenetv3. In *ICCV*, 2019. 2

[25] Loc Huynh et al. Edgent: Toward collaborative and on-demand deep learning in edge computing. In *MobiHoc*, 2017. 1, 2

[26] Gabriel Ilharco, Mitchell Wortsman, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, et al. Openclip. *Zenodo*, 2021. 2, 7

[27] Mohamed Fazli Imam, Rufael Fedaku Marew, Jameel Hassan, Mustansar Fiaz, Alham Fikri Aji, and Hisham Cholakkal. Clip meets dino for tuning zero-shot classifier using unlabeled image collections. *arXiv preprint arXiv:2411.19346*, 2024. 7

[28] Cijo Jose, Théo Moutakanni, Dahyun Kang, Federico Baldassarre, Timothée Darcet, Hu Xu, Daniel Li, Marc Szafraniec, Michaël Ramamonjisoa, Maxime Oquab, et al.

Dinov2 meets text: A unified framework for image-and pixel-level vision-language alignment. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24905–24916, 2025. 2, 3, 4, 6, 7, 8, 12, 13

[29] Koray Kavukcuoglu, Sundar Pichai, Demis Hassabis, Kent Walker, James Manyika, and Ruth Porat. Gemini-2.5: Our most intelligent ai model. Google DeepMind Blog, 2025. 8, 12

[30] Muhammad Uzair Khattak, Muhammad Ferjad Naeem, Muzammal Naseer, Luc Van Gool, and Federico Tombari. Learning to prompt with text only supervision for vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4230–4238, 2025. 7

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7

[32] Nicholas D. Lane et al. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *IPSN*, 2015. 1, 2

[33] Jiye Lee and Hanbyul Joo. Mocap everyone everywhere: Lightweight motion capture with smartwatches and a head-mounted camera. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1091–1100, 2024. 1

[34] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12281–12291, 2021. 5

[35] Jiyang Li et al. Q-former v2: Improved vision-language alignment for multimodal models. *arXiv preprint arXiv:2310.XXXX*, 2023. 2

[36] R.D. Lin, Pengcheng Weng, Yinqiao Wang, Han Ding, Jinsong Han, and Fei Wang. Hilots: High-low temporal sensitive representation learning for semi-supervised lidar segmentation in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1429–1438, 2025. 1

[37] Zhuang Liu et al. A convnet for the 2020s. In *CVPR*, 2022. 2

[38] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Neural architecture transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):2971–2989, 2021. 5

[39] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 7

[40] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *ICLR*, 2022. 2

[41] Sewon Min et al. A wearable multimodal assistant for context-aware understanding. In *UbiComp*, 2022. 1

[42] Muhammad Jehanzeb Mirza, Leonid Karlinsky, Wei Lin, Horst Possegger, Mateusz Kozinski, Rogerio Feris, and Horst Bischof. Lafter: Label-free tuning of zero-shot classifier using language and unlabeled image collections. *Advances in Neural Information Processing Systems*, 36:5765–5777, 2023. 7

[43] Sparsh Mittal. A survey on energy and thermal constraints in mobile and wearable deep learning. *IEEE Access*, 2020. 2

[44] Gyeongsik Moon, X. Weipeng, Rohan Joshi, W. Chenglei, and Takaaki Shiratori. Authentic hand avatar from a phone scan via universal hand model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2029–2038, 2024. 1

[45] OpenAI. Introducing gpt-5. https://openai.com/index/introducing-gpt-5/, 2025. Large language model. 8, 12

[46] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, pages 1–31, 2024. 2, 3, 5, 6, 8, 13

[47] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 2, 6, 7

[48] Sarah Pratt, Ian Covert, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15691–15701, 2023. 7

[49] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 2, 3, 4, 7

[50] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International conference on machine learning*, pages 2902–2911. PMLR, 2017. 3

[51] Karsten Roth, Jae Myung Kim, Andrew Koepke, Oriol Vinyals, Cordelia Schmid, and Zeynep Akata. Waffling around for performance: Visual classification with random words and broad concepts. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15746–15757, 2023. 7

[52] Christian Rupprecht et al. Passive visual monitoring for always-on perception. *arXiv preprint arXiv:2206.14883*, 2022. 1

[53] Aaron Serianni and Jugal Kalita. Training-free neural architecture search for RNNs and transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2522–2540, Toronto, Canada, 2023. Association for Computational Linguistics. 5

[54] Aman Shrivastava, Ramprasaath R Selvaraju, Nikhil Naik, and Vicente Ordonez. Clip-lite: Information efficient visual representation learning with language supervision. In *International Conference on Artificial Intelligence and Statistics*, pages 8433–8447. PMLR, 2023. 2, 3

[55] Sebastian Stein et al. The latency requirements of real-time

interactive systems. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 2

[56] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 2

[57] Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 2

[58] Yifan Wang, Alan Li, Hang Zhao, et al. Towards universal vision encoders for multi-task transfer. *arXiv preprint arXiv:2503.01234*, 2025. 3

[59] Zeyu Wang et al. Lifelong and context-aware perception for wearable ai. *arXiv preprint arXiv:2402.00111*, 2024. 1

[60] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16133–16142, 2023. 5, 6, 12

[61] Kan Wu, Houwen Peng, Zhenghong Zhou, Bin Xiao, Mengchen Liu, Lu Yuan, Hong Xuan, Michael Valenzuela, Xi Stephen Chen, Xinggang Wang, et al. Tinyclip: Clip distillation via affinity mimicking and weight inheritance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21970–21980, 2023. 2, 3

[62] Wenke Xia, Ruoxuan Feng, Dong Wang, and Di Hu. Phoenix: A motion-based self-reflection framework for fine-grained robotic action correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6981–6990, 2025. 1

[63] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010. 7

[64] Yun Xing, Jian Kang, Aoran Xiao, Jiahao Nie, Ling Shao, and Shijian Lu. Rewrite caption semantics: Bridging semantic gaps for language-supervised semantic segmentation. *Advances in Neural Information Processing Systems*, 36: 68798–68809, 2023. 2

[65] Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1602–1611, 2025. 1

[66] Hu Xu, Saining Xie, Xiaoqing Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying CLIP data. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[67] Jilan Xu, Junlin Hou, Yuejie Zhang, Rui Feng, Yi Wang, Yu Qiao, and Weidi Xie. Learning open-vocabulary semantic segmentation models from natural language supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2935–2944, 2023. 2

[68] Mengmi Xu et al. Ear: Efficient always-on recognition on wearable cameras. In *CVPR*, 2023. 2

[69] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1803–1811, 2019. 5, 6

[70] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *Computer Vision–ECCV 2020: 16th European Conference, Part VII 16*, pages 702–717. Springer, 2020. 3

[71] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18123–18133, 2022. 3, 4

[72] 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*, pages 11975–11986, 2023. 2

[73] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6, 7

[74] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 1, 2, 3, 4, 6, 7, 8, 12, 13, 14

[75] Jinghao Zhou, Xiangxiang Yu, Ping Luo, et al. ibot: Image bert pre-training with online tokenizer. In *International Conference on Learning Representations (ICLR)*, 2022. 3

[76] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2016. 3

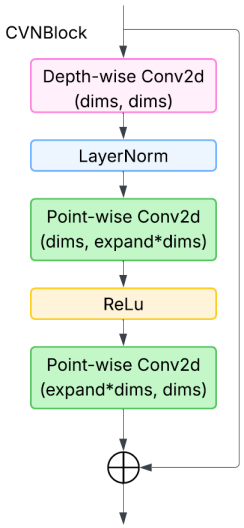

Figure 9. Basic structure of selective ConvNext-v2 blocks [60]. Block widths ($dims$) are ***selectable*** during training and runtime.

In this supplementary material, we provide additional implementation details for our design and further ablations.

A. Architecture of Basic Blocks

A.1. ConvNext-v2 Blocks

We use ConvNeXt-v2 [60] with ***selective*** capacity as the core building block of our model, as illustrated in Figure 9. All block widths ($dims$) are selectable during training and runtime, enabling the ***adaptive*** behavior of our model. We additionally replace GELU with ReLU for improved compatibility and efficiency on edge devices.

A.2. Downsample Layers

We also provide the basic structures of the downsampling layers used in AdaVFM in Figure 10. As with the ConvNeXt blocks, the block widths ($dims$) remain ***selectable*** and are adjusted according to the chosen configuration of the surrounding ConvNeXt blocks.

Downsample Layer 1 uses two consecutive 3×3-Conv2D layers with stride 2, yielding an effective downsampling factor of 4. Downsample Layer 2 uses a single 3×3-Conv2D layer with stride 2. Downsample Layers 3 and 4 instead apply 1×1-Conv2D layers with stride 2.

B. Grouped Super-Class of Datasets

In this paper, we examine how the description and granularity of user-defined class names influence the difficulty of open-vocabulary classification and segmentation. To analyze this, we group the ImageNet-1K (IN1K) [12] categories into super-classes as follows. For each class name, we first locate its entry in WordNet [17]; if it is absent, we use GPT-5 to map it to the closest WordNet term. We then traverse the WordNet synset hierarchy and identify the lowest ancestor that contains $5\% \sim 40\%$ of all IN1K classes, yielding a following set of seven super-classes.

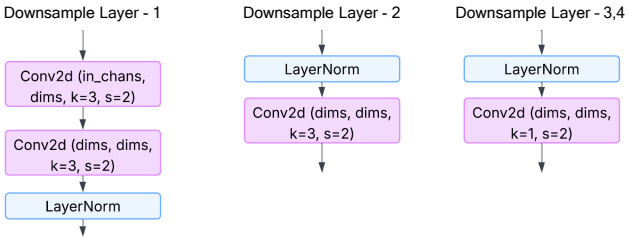

Figure 10. Basic structure of downsample layers in AdaVFM. The block widths ($dims$) are ***selectable*** during training and runtime.

Seven Super-classes of ImageNet-1K

1. animal
2. clothing
3. food
4. vehicle
5. instrumentality
6. structure
7. geological formation

The ADE20K dataset [74] is processed in a similar manner, but applies a heuristic class-name optimization from DINO.txt [28] to improve segmentation performance, resulting in the following nine super-classes.

Nine Super-classes of ADE20K

1. person;clothing
2. animal
3. plant
4. food
5. vehicle
6. nature;natural_environment;geological_formation
7. furniture;indoor_furniture
8. road;path
9. building;structure

C. LLM-Based Runtime Scene Understanding

C.1. Prompts for Runtime Scene-Aware Filtering

For open-vocabulary segmentation on ADE20K [74], given a scene context, we use the following prompts for GPT-5 [45] and Gemini-2.5 [29] to filter out implausible class names in the scene from the full set of 150 classes.

GPT-5 Prompt

Consider computer vision tasks in a scene of $scene. Given a scene description and a list of 150 object names, output a JSON dictionary where each key is an object name and the value is 0 if the object is highly unlikely in the scene, otherwise 1.

Requirements:
- Output only valid JSON (no explanations or text before/after).
- Keys must match exactly the provided object names.
- A total of 150 key-value pairs.
- Values must be integers 0 or 1 only.

Object names: $object_names.
Now output the JSON result only.

Gemini-2.5 Prompt

You are a JSON-producing assistant.
Consider computer vision tasks in a scene of $scene. Given a scene description and a list of 150 object names, output a JSON dictionary where each key is an object name and the value is an integer:
- 1 if the object is likely or possible to be found in the scene.
- 0 if the object is highly unlikely or impossible in the scene.

Requirements:
- Starting with {{ and ending with }}. Do NOT use markdown or triple backticks.
- Output only valid JSON (no explanations or text before/after).
- Keys must match exactly the provided object names.
- A total of 150 key-value pairs.
- Values must be integers 0 or 1 only.

Object names: $object_names.
Now output the JSON result only.

The two prompts above generate an unbounded set of candidate classes based on the scene context. We also study a method that restricts predictions to a fixed top-k class subset, using the GPT-5 prompts listed below.

Method	Recall	Precision
GPT-5	89.7	6.1
Gemini-2.5	94.7	3.2
Top-20	71.4	9.1
Top-25	77.1	8.0

Table 8. Average recall and precision for each LLM prompt used in scene-aware class filtering on ADE20K [74]. Among these methods, GPT-5 achieves the best end-to-end performance for open-vocabulary segmentation.

Top-k Prompt for GPT-5

Consider computer vision tasks in a scene of $scene. Given a scene description and a list of 150 object names, output only the top $$\boldsymbol{k}$$ most relevant objects likely to appear in this scene.

Requirements:
- Output only valid JSON (no explanations or text before/after).
- Keys must match exactly the provided object names.
- A total of 150 key-value pairs.
- Values must be integers 0 or 1 only.

Object names: $object_names.
Now output the JSON result only.

C.2. Quality of Scene-Aware Filtering

Table 8 shows the average recall and precision for each LLM prompt used in scene-aware class name filtering in ADE20K [74]. Top-k methods exhibit high precision but low recall by being over-aggressive in filtering, while Gemini-2.5 achieves high recall but low precision by being over-conservative, leaving too many irrelevant classes remaining. Both lead to poorer end-to-end performance in open-vocabulary segmentation compared to GPT-5, but still provide slight end-to-end improvements over the non-LLM baselines, such as DINO.txt [28].

D. Agent-Assisted Runtime Model Selection

D.1. Implementation Details

The pseudocode for selecting a subnet given a scene during runtime is provided in Algorithm 1. A hardware mode must be specified, with a *hardware mode parameter* α representing the required fraction of maximum accuracy achievable within the NAS search space of AdaVFM. The LLM-based agent first identifies the closest scene in a large training dataset (e.g., LVD-142M [46]) to the current scene (Step 1). It then estimates the achievable accuracy under the speci-

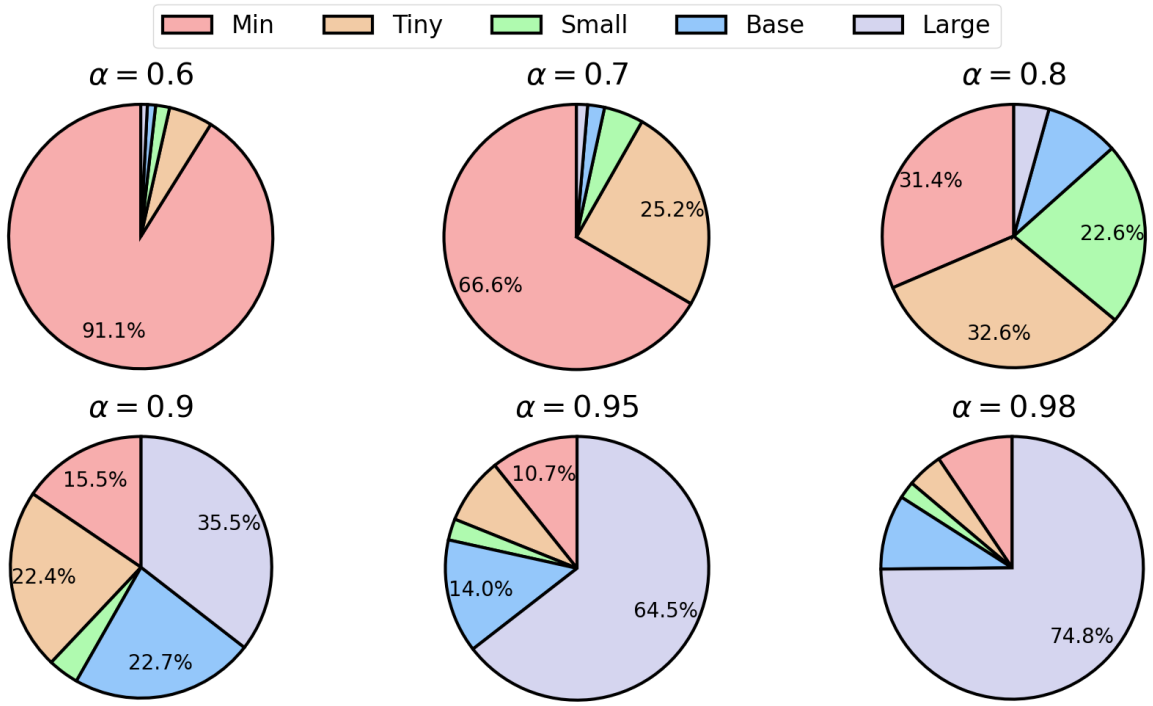

Figure 11. Fraction of selected FLOPs-optimized subnets across all ADE20K [74] scenes for different *hardware mode parameters* α. Portions below 10% are unlabeled due to space limits. Larger α imposes stricter accuracy requirements, leading to the selection of larger subnets.

Algorithm 1 LLM-based Scene-aware Subnet Selection

Require: Scene S, Hardware mode parameter $\alpha \leq 1$

1: $S' \leftarrow LLM(S, Existing\_Scenes)$
2: $MaxAcc \leftarrow Acc(Max\_Subnet, S')$
3: $Subnet \leftarrow argmin_M\{Acc(M, S') \geq \alpha \cdot MaxAcc\}$
4: **return** $Subnet$

fied hardware mode (Step 2) and selects the subnet in the NAS search space with the minimal resource cost—such as FLOPs, memory usage, latency, or energy—that satisfies the accuracy requirement (Step 3).

D.2. Selected Subnets During Runtime

Figure 11 shows the fraction of selected subnets (from the five described in our submission—MIN, TINY, SMALL, BASE and LARGE) across all scenes in ADE20K [74] under different hardware mode parameters α, optimized for average FLOPs usage. Higher α enforces stricter accuracy requirements, favoring the selection of larger subnets.

Notably, even with extremely strict accuracy requirements ($\alpha = 0.98$), some scenes still select smaller subnets, highlighting that certain contexts can be efficiently handled by smaller models and motivating the need for ***adaptive runtime selection*** at the edge.