AdaDINO: Context-Adaptive DINO-Distilled Vision Foundation Models for Efficient Open-Vocabulary Edge Inference

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

Always-on contextual AI runs language-aligned vision foundation models (VFMs) on edge devices, where the on-device model is the dominant continuous compute cost under strict latency and power limits. Due to an observed low-frequency shift in scene context and its relevant vocabulary, we present AdaDINO, an adaptive framework that makes on-device VFM inference efficient by matching execution to the current scene and task. We build on a known phenomenon, that the accuracy drop of shrinking model sizes depends on the task, and turn it into task-level adaptive execution. AdaDINO integrates neural architecture search (NAS) into a language-aligned VFM backbone distilled from DINOv2, training a single family of subnets for efficient execution during runtime. A multimodal large language model (LLM) on the cloud, invoked at low frequency, refines the candidate class set from scene context, while a learned selector activates the least-cost subnet predicted to retain a target fraction of accuracy. With the backbone and semantic pipeline held fixed, learned selection alone reduces average compute by 37% over the best fixed subnet at equal segmentation accuracy. Across zero-shot classification and open-vocabulary segmentation, AdaDINO establishes a strong accuracy-efficiency frontier, improving over evaluated models of comparable sizes by up to 7.9% in acc@1 on IN1K and 5.2% mIoU on ADE20K, and reducing average FLOPs by up to 74.9% at similar accuracy.

1 Introduction

Advances in smartphones, wearable devices (Abbaspourazad et al. 2024), augmented and virtual reality (Abrash 2021), robotics, and autonomous driving are pushing multimodal AI onto edge devices, driving interest in *always-on contextual AI*—continuous, context-aware systems running persistently under tight energy and latency constraints (Hoang 2025). Such systems already ship as real products. Smart glasses like Meta Ray-Ban (Meta Platforms 2023) enable contextual AI by forwarding multimodal queries to the cloud, costing multi-second responses and short battery life.

Industry (such as Meta) is converging on an edge-cloud split as in Fig.1: the edge runs a lightweight **vision foundation model** (VFM) for real-time perception, while heavy or knowledge-intensive tasks go to a cloud multimodal LLM (MM-LLM) (Li et al. 2023). The split works because context changes far more slowly than frames arrive: egocentric scenes and activities typically persist for minutes or longer (Grauman et al. 2022, 2024).

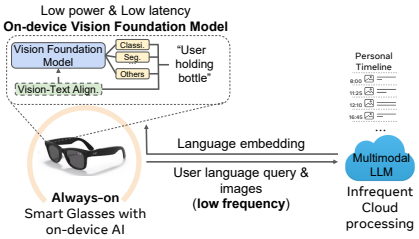

Figure 1: Motivating deployment scenario: always-on smart glasses with an on-device VFM.

Building this stack end to end is an industry-scale effort. We focus on one important layer of this stack that stands on its own as a machine-learning problem: the dominant and continuous compute cost of the always-on on-device vision model. Language-aligned VFMs are large and computation-intensive, but edge devices operate under strict compute, memory, and power limits with interaction budgets of 200–1000 ms (Stauffert, Niebling, and Latoschik 2016). Direct VFM execution under such constraints is infeasible, and prior attempts that simply shrink model capacity (Wu et al. 2023b) often suffer large performance drops (see §2.3).

However, not all tasks need a full VFM—simpler tasks suit smaller models. As in table 7, DINOv2 degrades little on simpler datasets such as Pets (Parkhi et al. 2012) and Cal101 (Fei-Fei, Fergus, and Perona 2007) with smaller models (drops of at most 1.6), while harder datasets incur larger drops (up to 5.6).

The same pattern holds in open-vocabulary segmentation: in Fig. 2, 50M and 300M VFMs are comparable on a grouped 9-class ADE20K task but diverge significantly on the full 150-class task (details in §E). This motivates ***adaptive model selection*** on the edge based on ***task difficulty***, rather than a single fixed configuration.

In this paper, we study a representative VFM family—DINOv2 (Oquab et al. 2024) and the CLIP-style DINO.txt (Jose et al. 2025) framework—and *adapt it for the edge*. We leverage ***the sensitivity of open-vocabulary tasks to model capacity*** for efficient edge deployment of DINO-based VFMs. A fixed edge VFM overkills: it spends full capacity on easy vocabularies and compares against a broad class set even when context rules out most classes. We introduce

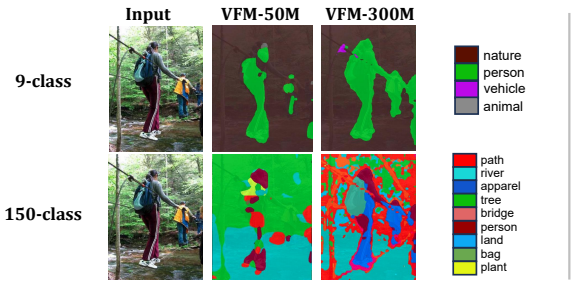

Figure 2: Open-vocabulary segmentation on ADE20K (Zhou et al. 2019, 2017) using 50M and 300M-parameter VFMs (300M: DINO.txt; 50M: similarly distilled and fine-tuned). Results are shown for both the original 150-class and the grouped 9-class setting (grouping based on WordNet (Fellbaum 1998), described in §E). **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 gives *blurred* ones. **Right**: Going from 300M to 50M costs only 0.95 mIoU on the 9-class task but 3.84 on the 150-class task.

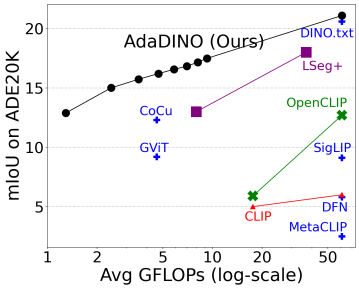

Figure 3: End-to-end open-vocabulary ADE20K segmentation. AdaDINO improves mIoU by up to 5.2% and cuts FLOPs by up to 74.9% over prior models.

task-level adaptive selection, which chooses the execution scheme fitting the current task and resources, and integrate Neural Architecture Search (NAS) into a two-stage pipeline of DINOv2 distillation and DINO.txt-style vision-text alignment. In principle the pipeline generalizes to any distillable VFM with a CLIP-style framework, though we instantiate it on DINO as a representative, well-performing choice.

To improve edge model efficiency, an LLM-powered runtime manager, invoked at low frequency, infers scene context, filters out implausible classes, and supplies an execution scheme under current scene or vocabulary. AdaDINO targets one layer in the full system stack—making the on-device VFM efficient using this runtime manager. When to refresh it, and how the full device behaves over long deployments, belong to the surrounding stack and lie outside the ML scope.

We present **AdaDINO**, an adaptive edge VFM with a training pipeline and a proof-of-concept deployment on real edge hardware. It brings DINO-family VFMs to the edge via task-context adaptation on standard deployable components. Our contributions are: (*i*) NAS-integrated adaptive VFM deployment for efficient edge execution; (*ii*) LLM-based semantic understanding for refined class sets and text embeddings; and (*iii*) learned task-level subnet selection, which alone cuts average compute by more than a third at matched accuracy. Evaluations (Fig.3, Tab.3) show that AdaDINO achieves a better accuracy-efficiency trade-off than prior work (Ghiasi et al. 2022; Jose et al. 2025; Radford et al. 2021; Fang et al. 2024; Ilharco et al. 2021; Xing et al. 2023; Xu et al. 2024, 2023a; Zhai et al. 2023; Roth et al. 2023; Mirza et al. 2023; Khattak et al. 2025).

2 Background and Related Work

This section summarizes the work most relevant to AdaDINO. An extended discussion of related work in each topic is provided in §J.

2.1 Always-on Contextual AI on Edge Devices

Wearable edge platforms are motivating *always-on contextual AI* that operates continuously under strict power, thermal, and memory constraints. Persistent offloading is impractical at wearable hardware budgets (Lane et al. 2016), so a lightweight *on-device vision foundation model* handles real-time perception such as classification and segmentation, while multimodal reasoning and broad world knowledge are delegated to a cloud-based multimodal LLM (Fig.1). Building such a device end-to-end—silicon, optics, batteries, interaction—is a division-scale industry effort, and deployed stacks already isolate the always-on perception model as a distinct layer from the intermittent cloud reasoning around it. We take that industry partition as given (§1, §J.1) and address the layer it exposes: making the dominant always-on compute cost of on-device vision models spend only what the current task requires.

2.2 Vision Foundation Models

Vision foundation models offer unified representations that generalize across tasks and domains, typically trained at scale with self-supervised learning (SSL). Recent advances such as DINOv2 (Oquab et al. 2024) and iBOT (Zhou et al. 2022a) learn transferable features that rival or surpass those obtained via supervised pretraining methods like CLIP. Large-scale VFMs, however, incur substantial compute, memory, and energy costs. We present *adaptive* VFMs that bring strong accuracy-efficiency trade-offs to the edge devices.

2.3 Contrastive Learning for VFMs

Contrastive learning methods such as CLIP (Radford et al. 2021) enable zero-shot and open-vocabulary capabilities by aligning visual and textual representations. A standard CLIP architecture consists of a Vision Encoder and a Text Encoder, both too costly for resource-constrained edge devices. Pruning or shrinking them often causes large accuracy loss.

A key property of CLIP for edge deployment is the ***differing frequencies*** of Vision Encoder and Text Encoder calls. In always-on contextual AI (Fig. 1), prompts or scenes change infrequently, while 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 latency.

Efficient VFMs and CLIP. Prior work lowers VFM cost either *statically*—distilling or shrinking CLIP-style encoders into compact backbones (Vasu et al. 2024)—or through a *different notion of adaptiveness* than ours, in which a single fixed model spends more or less compute according to per-input difficulty, e.g., via token pruning and merging (Liang et al. 2022; Bolya et al. 2023; Wu et al. 2025) or token/block halting (Yin et al. 2022). All of these react to per-sample *confidence* within a *single, fixed, closed-set* model. Our AdaDINO adapts along an orthogonal axis: it switches among *open-vocabulary* subnets conditioned on *scene- and task-level context* rather than per-frame confidence, and stays composable with the above, which can still operate within whichever subnet it activates.

2.4 Neural Architecture Search

NAS provides a principled framework for adaptive-capacity models suited to edge deployment. Its key merit is support for ***runtime subnet selection*** over sub-architectures of varying computational cost: at inference, a subnet is selected by task complexity or resources, enabling flexible accuracy-efficiency trade-offs. Weight-sharing supernets (Cai et al. 2020) and runtime width switching in slimmable networks (Yu et al. 2019; Li et al. 2021b) made this practical. Rather than a new supernet-training algorithm, our contribution is an OFA-style supernet made compatible with foundation-model distillation and vision-text alignment, driven by scene-conditioned vocabulary refinement and text-set-conditioned subnet selection.

3 Method

3.1 Overall System Design

We show the AdaDINO runtime system in Fig.4. Our optimization target is the on-device vision model: given a low-frequency context signal, make it spend only what the current task needs. The design has two complementary components: (1) an **adaptive vision foundation model** deployed on the *edge* device, running always-on for high-frequency real-time vision inference; and (2) a **cloud-side runtime manager**, invoked infrequently, where a multimodal LLM provides scene and context understanding and text embeddings, and a learned selector picks the edge subnet. Together they adapt open-vocabulary inference as scenes and tasks change. AdaDINO operates at the *task-context* granularity: each context update defines a candidate class set that stays active across many subsequent frames. The context signal is supplied at low frequency, and deciding when to refresh it is an orthogonal deployment problem outside our ML scope.

3.2 On-Edge Adaptive VFM

Execution Flow. As shown in Fig. 4, the cloud-side multimodal LLM receives a temporally sparse stream of images and/or user text interactions to infer the *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. 5a, the manager produces two outputs: (*i*) a set of *text embeddings* for the selected concepts, and (*ii*) an *execution scheme* computed by the learned selector of Step III in §3.3 from the refined class set, telling which edge subnet to use. The edge-side Vision Encoder operates continuously on every frame, computing cosine similarity between vision and cloud-provided text embeddings for real-time open-vocabulary inference.

Selective Execution Scheme. As illustrated on the left side of Fig. 4, AdaDINO enables dynamic adaptation on the edge through a NAS-based supernet space that exposes multiple execution schemes with different computational costs. At runtime, the system selects a subnet from this space based on the prediction of the manager, which reflects current task difficulty and latency-accuracy requirements. The chosen subnet stays active for the next few rounds of inferences.

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-style pipeline, similar to LiT (Zhai et al. 2022) and DINO.txt. 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.

3.3 LLM-Guided Efficient Runtime Execution

A central component of AdaDINO is the **LLM-guided runtime manager**, invoked at low frequency (e.g., every few minutes). As in Fig.5a, this manager is responsible for: (*i*) giving a *semantic understanding* of the current scene to guide the Text Encoder, and (*ii*) selecting *efficient* execution schemes of the Vision Encoder via a learned selector (Step III).

From the low-frequency contextual signals described above, 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. 5a. 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 open-vocabulary scene-related targets.

Step I: Scene Annotation Generation. For controlled and reproducible evaluation, we treat each image in public benchmarks as one context instance: the image is fed to a cloud-side MM-LLM, which produces a concise scene annotation within three words (e.g., *"airport"* or *"office"*). See §F.1 for implementation details. This protocol isolates the effect of semantic context on model execution, validated one context at a time. In deployment, one annotation is reused across temporally adjacent frames at low frequency (e.g., every few minutes) shown in Fig.5a, a schedule the projections below assume rather than measure.

Step II: Semantic Understanding of a Scene. Zero-shot classification and open-vocabulary segmentation in CLIP-style models require passing candidate class names into the

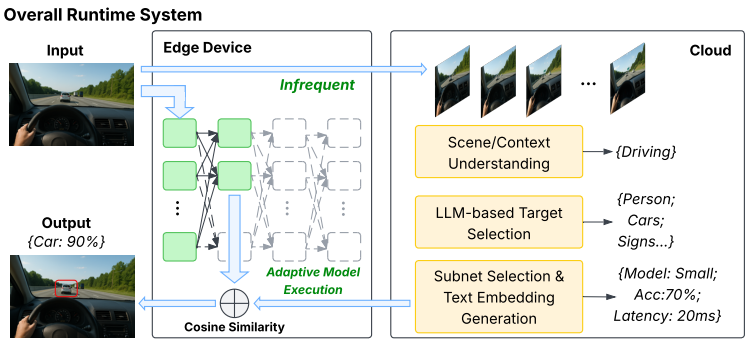

Figure 4: Overview of the proposed AdaDINO. **Left**: Edge-side execution, where the adaptive Vision Encoder follows cloud-side manager instructions to select an efficient execution scheme and perform vision-text contrastive inference. **Right**: Cloud-side execution, where the manager uses scene/context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

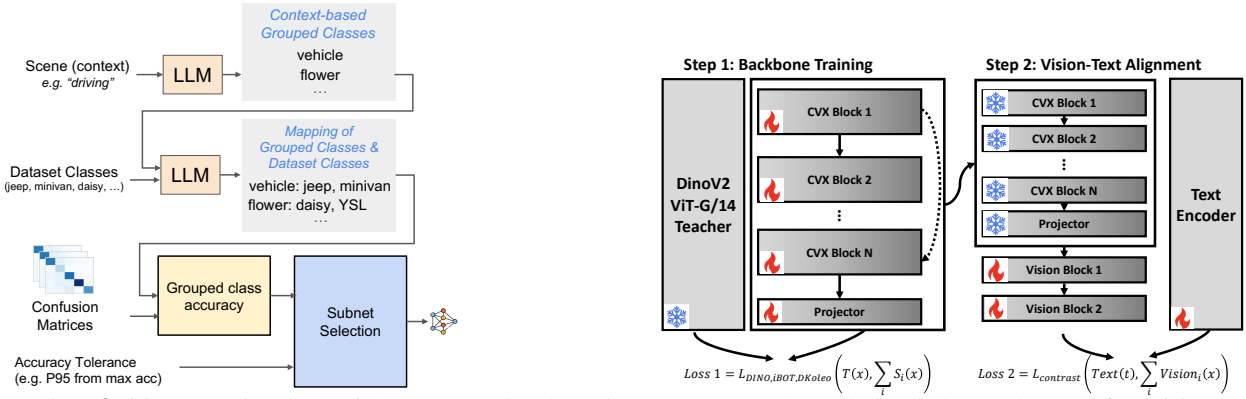

Figure 5: **Left (a)**: Operation flow of the LLM-assisted runtime manager (subnet choice follows Alg. 1). **Right (b)**: Overview of the training pipeline. The vision backbone is first distilled from a foundation model (DINOv2), followed by DINO.txt-style vision-text alignment. Both stages employ NAS and sandwich sampling (Yu and Huang 2019).

Text Encoder. Instead of blindly enumerating the entire vocabulary, our runtime manager uses LLM-driven semantic reasoning to filter out implausible classes based on the task context, reducing noise and resolving ambiguous class definitions. For example, given a scene context *"sports field"*, the manager selects *"field"* rather than *"grass"* as the more contextually appropriate phrase. This filtering step corresponds to the first stage of Fig.5a, where dataset classes are aligned to grouped semantic categories.

Step III: Difficulty-Based Model Selection. The size and composition of the class set reflect the *difficulty* of an open-vocabulary task, which can guide the runtime selection of an appropriately sized model for efficient edge execution.

A selector *learns* this ability in an open-vocabulary way from the candidate class names, following CLIP-style methods that use text embeddings as open-vocabulary classifiers. Specifically, the class names C are embedded by the Text Encoder and summarized as $d = \textsc{SetStats}(\textsc{TextEnc}(C))$, an order-/size-independent statistic of their pairwise similarities, inspired by permutation-invariant set representations (Zaheer et al. 2017). A capacity-monotone gradient-boosted regressor then predicts the *accuracy-retention ratio* $\hat{r}_M \approx \mathrm{Acc}(M, C)/\mathrm{MaxAcc}$ of each subnet M, following prior learning-to-rank/budgeted-prediction formulations (Li, Burges, and Wu 2007; Xu, Weinberger, and Chapelle 2012).

Algorithm 1 Learned Scene-Aware Subnet Selection

Require: class names C; tolerance α; subnet M & cost $c(\cdot)$
1: $d \leftarrow \textsc{SetStats}(\textsc{TextEnc}(C))$
2: $\hat{r}_M \leftarrow \textsc{Retention}(d, M) \quad \forall M$
3: **return** $\arg\min\{\, c(M) : \hat{r}_M \geq \alpha \,\}$ **else** $M_{\max}$

Alg. 1 returns the least-cost subnet with cost $c(M)$ predicted to satisfy $\hat{r}_M \geq \alpha$, which can generalize to unseen class sets. The selector is agnostic to where the class set comes from: an LLM is used in our prototype because it handles open-ended scene descriptions naturally, but other context providers can drive the same selector if necessary.

Conceptually Reducing System Overhead. Whereas prior contextual AI systems forward every multimodal query to the cloud (§1), our system invokes cloud MM-LLM at a much lower frequency (a few minutes). As a conceptual example, the cloud-side call has a 6.4s P95 latency on our prototype hardware (§C), small relative to the refresh interval. Only vision tasks run in real time on-device. This design incurs sustainable cost on the efficient edge subnet. The resulting battery-life and cloud-cost gains are first-order projections

under this cadence rather than measured results, reported as deployment motivation in §J.1.

System Robustness. Cloud unavailability does not break AdaDINO: the edge VFM falls back to the largest subnets, which remain more efficient than prior baselines (see §5).

4 Training

We integrate NAS into the DINO-based VFM, as illustrated in Fig.5b, following the motivation of §1—a trend widely observed in prior work (Lu et al. 2021) and verified in our open-vocabulary setting (§E). Our goal is to train a single *supernet*, with subnets of varying computational costs selected at runtime.

Stage 1: Backbone Distillation. We use OFA-NAS (Cai et al. 2020), training a supernet over a wide range of widths and depths. As in Fig.5b (left), we distill a DINOv2 ViT-G/14 teacher into a ConvNeXt-based (Liu et al. 2022; Woo et al. 2023) supernet. ConvNeXt is used for edge deployability: its convolutional operators map onto fixed-function NPUs far better than the ViT teacher's attention (§5.2, §C), and its structure keeps NAS integration much more tractable than transformer search spaces (Serianni and Kalita 2023; Li et al. 2021a).

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). We optimize a DINO-style distillation loss $\mathcal{L}_1 = \sum_i \mathcal{L}_{\text{DINO,iBOT,Koleo}}\big(T(x), S_i(x)\big)$, where $T(\cdot)$ is teacher features and $S_i(\cdot)$ is outputs of sampled subnets.

Stage 2: Vision-Text Alignment. Following the CLIP-style alignment procedure in DINO.txt, we freeze the trained ConvNeXt blocks (snowflake icons in Fig. 5b, right), and train only the vision blocks that map visual features into the text embedding space. This stage uses a contrastive loss $\mathcal{L}_2 = \sum_i \mathcal{L}_{\text{contrast}}\big(\text{Text}(t), \text{Vision}_i(x)\big)$. This two-stage pipeline ensures that the NAS supernet is compatible with CLIP-style open-vocabulary inference. Neither stage is tied to this pair: conceptually, any distillable VFM teacher and CLIP-style alignment objective fit the same pipeline, with DINOv2 and DINO.txt the instance we build and evaluate.

Stage 3: Selector Training. The retention predictor of Alg. 1 is trained offline under leave-scene-out cross-validation. We measure each subnet's accuracy once per scene, then regress the retention ratio $\text{Acc}(M, C)/\text{MaxAcc}$ from the class-set statistics d and a normalized capacity descriptor of M (FLOPs, width, depth, parameters), weighting each scene by its image count since retention measured on few images is noisy. As tree ensembles pull predictions toward the mean, we calibrate the outputs with isotonic regression on a held-out split, so a fixed tolerance α still separates easy from hard tasks at inference. This stage is one-time and negligible in cost next to Stages 1-2 (§G).

Sandwich Sampling for Supernet Optimization. To efficiently train the NAS search space, we follow the sandwich sampling rule (Yu et al. 2020). In each iteration, we sample the *largest* and *smallest* subnets—corresponding to the two extremes in Tab. 1—along with 2 additional subnets sampled uniformly from intermediate configurations. We compute gradients for all sampled subnets and average them to update shared weights, letting one supernet support a wide range of subnets with minimal accuracy gaps.

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 1: Our NAS search space for the vision backbone. Depth denotes the number of layers and Dim. the dimensionality of each block.

Training Details. For backbone distillation, we follow DINOv2 within our NAS framework, using the LVD-142M dataset (Oquab et al. 2024). 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 and train on the MetaCLIP dataset (Xu et al. 2024) 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. For the selector, we train 400 iterations with a learning rate of 0.05 and 15 leaves per tree. Supernet training is a one-time cloud-side cost (§B), amortized across deployments. All subnets share one checkpoint, so a device stores a single model.

5 Evaluation

5.1 Experimental Setup for Inference

We evaluate zero-shot classification and open-vocabulary segmentation, two core tasks of contextual AIs.

Zero-Shot Classification. We evaluate on ImageNet-1K (IN1K) (Deng et al. 2009), Food101 (Bossard, Guillaumin, and Van Gool 2014), DTD (Cimpoi et al. 2014) and Pets, following the inference procedure described in DINO.txt (§3.2).

Open-Vocabulary Segmentation. We use ADE20K. The effectiveness metric is mean Intersection-over-Union (mIoU).

Metrics for Efficiency. We use floating-point operations (FLOPs) and parameter count (#params) to quantify the edge Vision Encoder's computational cost and memory usage (cloud-side per-update costs: §3.3).

In addition, we acquire *real-world* latency and energy usage as additional efficiency metrics, by deploying models on a test silicon featuring an ARM Ethos-U55 NPU (Arm® 2022) fabricated in 7nm FinFET, a common industry setup (e.g., (Skillman and Edsö 2020; Zhao et al. 2025)). See §C for hardware details. The reported results under these metrics in §I include averaged measured overhead of runtime wrapper, subnet switching, and other related costs.

Evaluated NAS Subnets. From the wide-spread NAS search space, we select five representative subnets described in

Tab.2. TINY, SMALL and LARGE correspond to ConvNeXt-v2-N, -T and -S, respectively, while the remaining subnets are interpolated to span the entire search space.

5.2 Comparison with State-of-the-Art

Zero-Shot Classification. We compare AdaDINO with state-of-the-art static and dynamic/adaptive baselines in Tab.3. The static baselines are compared in the recent NoLA (Imam et al. 2024), and we use its settings. The dynamic/adaptive baselines are reported by PatchRank (Wu et al. 2025). Our approach improves over the comparably sized and larger static baselines and all reported dynamic baselines: LARGE (50M) gains at least 7.9% IN1K acc@1 over the 86M static models and 13.8% over the 86M dynamic/adaptive ones. The only exceptions are OpenCLIP (Ilharco et al. 2021) and DINO.txt, where the performance gap is primarily due to ***model size differences*** (50M LARGE vs. 304M OpenCLIP/DINO.txt). Scene/context information is excluded in Tab. 3 as it is not suitable for object-oriented datasets. As published baselines use their own pretraining data and objectives, Tab. 3 and Fig. 3 characterize end-to-end deployment trade-offs, while §6 isolates each of our components under a fixed backbone and semantic pipeline.

Our models are also substantially more efficient in execution: the ViT-based baselines rely on an 86M ViT-B (423 ms, 19.9 mJ) or a 304M ViT-L (1119 ms, 52.8 mJ), at least 2.3× the latency and 2.4× the energy of our subnets (Tab. 2).

Open-Vocabulary Segmentation. Beyond segmentation accuracy, we evaluate the accuracy-efficiency trade-off of our design against baselines. Fig. 3 presents an end-to-end comparison, showing that AdaDINO achieves clear gains in the accuracy-efficiency (FLOPs) trade-off. Accuracy-latency/energy trade-offs show similar trends, which are given in §I.

6 Ablation and Analysis

AdaDINO's accuracy-efficiency gains come from three sources, as shown in Fig.6: a better-trained vision backbone (red curve vs the substantially weaker prior standalone models in Fig.3), LLM-based runtime scene understanding that improves open-vocabulary accuracy (blue vs red), and runtime selection of efficient subnets (black vs blue). Each comparison below changes one component at a time, so the benefit of each source is measured with the others held fixed.

Our two runtime mechanisms read only class names and subnet capacities, not backbone internals, so in principle they compose on any VFM backbone—a conceptual generalization we do not evaluate: a stronger one would enter as a better red curve, with filtering and selection trends unchanged.

6.1 Stronger NAS-Enabled Vision Backbone

Compared with DINO.txt-based standalone-distilled models, our NAS-based subnets achieve near-identical accuracy—e.g., LARGE reaches 78.8% on IN1K versus 79.1% for its standalone counterpart, and stays within 0.9% of the standalone model across all ten datasets—so weight sharing preserves the accuracy of independently trained models. Full details are in table 5.

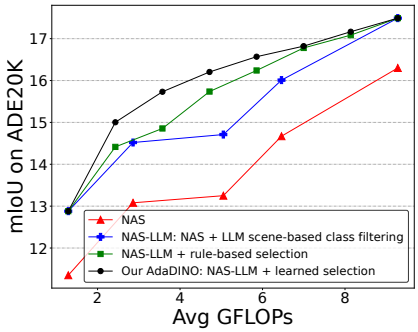

Figure 6: Ablation of accuracy-efficiency gains. Red: our vision backbone with one globally fixed subnet per point and no semantic context; blue: the same family plus LLM-based class filtering, still with fixed subnets; green: the same semantic pipeline with a heuristic rule-based selector; black: with our learned selector. Green and black isolate the selector, sharing the same model family and semantic inputs (§6.3).

6.2 LLM-Assisted Runtime Scene Understanding

An LLM-based manager on the cloud assists AdaDINO by providing semantic understanding to improve the Text Encoder. We evaluate this process by first generating scene annotations on ADE20K with a LLM, then using a LLM to ***filter out*** impossible class names and ***resolve ambiguity*** by selecting contextually proper terms. The refined class set is provided to AdaDINO for open-vocabulary segmentation. See §F for implementation of the cloud-side manager.

Scene Annotation Generation. We compare LLM-generated scene annotations with the ground-truth labels in ADE20K. As in Tab. 4, a sufficiently powerful LLM such as Llama4 (Meta AI 2025) clearly outperforms the vanilla setting where no scene generation and class filtering is used. Moreover, one concise annotation per scene approximates the accuracy of ground-truth scene labeling in ADE20K.

Another observation is that *smaller* models *benefit more* from the semantic filtering, with up to 1.53 mIoU improvement—a good fit for the always-on edge setting.

Class Name Filtering. Similarly, we ablate different commercial LLMs, prompts and implementation choices for class filtering in §F.3. LLM-based filtering yields substantial segmentation gains for AdaDINO—up to 1.76 mIoU over prior baselines and 1.53 mIoU over non-LLM Vanilla—and a visualization there shows more regular, coherent, semantically consistent masks.

Out-of-Distribution Scenes. Open-vocabulary tasks often encounter scenes outside any predefined training set, which calls for the *open-ended* semantic reasoning of an LLM-based manager. An OOD ablation in §F.5 shows that our open-set scene classifier remains robust while closed-set ones degrade significantly as fewer runtime scenes are known in advance.

6.3 Runtime Subnet Selection

Fig.6 studies how the cloud-side manager selects an edge-side VFM subnet by task difficulty, comparing our learned selector (black, §3.3) with a heuristic rule-based selector

Subnet	Depths	Dimensions	#params	FLOPs	Latency	Energy
MIN	[3,3,9,3]	[48,96,192,384]	6.2M	1.29G	25ms	1.2mJ
TINY	[3,3,9,3]	[72,144,288,576]	16.1M	2.86G	52ms	2.4mJ
SMALL	[3,3,9,3]	[96,192,384,768]	28.7M	5.05G	89ms	4.1mJ
BASE	[3,3,15,3]	[96,192,384,768]	35.9M	6.46G	120ms	5.5mJ
LARGE	[3,3,27,3]	[96,192,384,768]	50.3M	9.28G	182ms	8.4mJ

Table 2: Five representative NAS subnets selected from the search space with sizes matching ConvNeXt-v2 variants (-N/-T/-S).

	Model	#param	Food101	DTD	Pets	IN1K
AdaDINO	LARGE	50M	81.7	58.3	93.3	73.3
	BASE	36M	80.5	57.9	93.3	72.4
	SMALL	29M	79.2	57.2	92.8	71.3
	TINY	16M	78.3	56.3	92.1	69.8
	MIN	6M	73.8	54.1	91.4	65.5
Static	DINO.txt (2025)	304M	93.4	67.5	95.3	81.6
	OpenCLIP (2021)	304M	-	55.6	83.9	74.0
	CLIP (2021)	86M	-	42.9	85.0	61.9
	Waffle (2023)	86M	-	51.0	88.1	63.5
	LaFTer (2023)	86M	-	46.1	82.7	64.2
	ProText (2025)	86M	-	50.7	89.0	64.9
	NoLA (2024)	86M	-	56.1	89.3	65.4
	TinyCLIP (2023b)	45M	-	-	-	62.7
	MobileCLIP (2024)	11M	-	-	-	67.8
Dynamic	EViT (2022)	86M	80.3	43.3	87.1	58.1
	A-ViT (2022)	86M	82.3	43.5	83.2	57.6
	ToMe (2023)	86M	82.4	41.5	87.2	58.3
	PatchRank (2025)	86M	84.0	44.3	87.7	59.5

Table 3: Zero-shot classification. AdaDINO ***outperforms*** the comparably sized (45–86M) and larger static baselines, and all reported 86M dynamic baselines, by at least 7.9% IN1K acc@1. OpenCLIP and DINO.txt are much larger (304M vs. our $\leq$50M) and MobileCLIP (11M) much smaller. Baselines use their published pretraining setups.

(green, §H) and a fixed-subnet baseline without runtime selection (blue). All other parts are kept identical across the three settings.

Runtime subnet selection ***substantially improves*** the accuracy–efficiency trade-off over using fixed subnets: at 15.0 mIoU, our learned selector reduces compute from 5.4 to 3.4 GFLOPs (37%), while the rule-based selector requires 4.2 GFLOPs. The learned selector also ***consistently outperforms*** the heuristic in the intermediate operating range. Its prediction error, per-tolerance violation rates, and cost against an oracle selector are quantified in §G.

Which Subnet is Selected? Fig.7 shows the fraction of selected subnets in ADE20K under different tolerance α, optimized for FLOPs usage. Even with strict accuracy requirements ($\alpha = 0.98$), some scenes still select smaller subnets. This necessitates ***adaptive runtime selection***, where certain task contexts can be efficiently handled by smaller models.

7 Conclusion

We present AdaDINO, a context-adaptive framework for efficient edge DINOv2-distilled VFMs that combines a NAS-enabled model family with an LLM-assisted runtime manager to refine the active vocabulary and select the least-cost subnet by task context. With backbone and semantic pipeline held fixed, learned selection alone cuts average compute by

	Llama4	Llama3.2	GT	Vanilla
MIN	12.88 (+1.53)	11.7	13.11	11.35
TINY	14.52 (+1.44)	13.68	14.54	13.08
SMALL	14.71 (+1.46)	13.83	14.71	13.25
BASE	16.01 (+1.34)	15.29	16.13	14.67
LARGE	17.49 (+1.19)	16.9	17.53	16.30

Table 4: ADE20K mIoU under different *scene annotation* strategies. First two columns use LLM-generated labels at runtime. GT uses ground-truth labels. Vanilla stands for not using scene for class filtering (red curve in Fig.6). Differences over Vanilla are shown in parentheses for Llama4, the scene annotator in our final design (blue in Fig.6). LLM-based semantic understanding ***markedly improves*** over Vanilla and ***approximates*** ground-truth scenes.

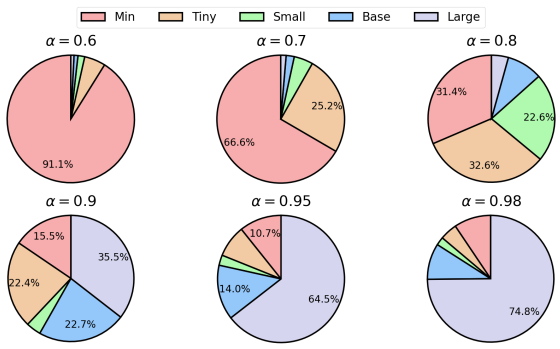

Figure 7: Subnet selection across α. Larger α imposes stricter accuracy requirements, leading to selection of larger subnets.

37%, and overall it improves the accuracy-efficiency trade-off by up to 5.2% mIoU or 74.9% fewer FLOPs—task-level adaptation built on standard, well-optimized components that stay deployable on edge accelerators. Next steps include context-refresh scheduling and alternative context providers.

References

Abbaspourazad, S.; Elachqar, O.; Miller, A.; Emrani, S.; Nallasamy, U.; and Shapiro, I. 2024. Large-scale Training of Foundation Models for Wearable Biosignals. In *The Twelfth International Conference on Learning Representations*.

Abrash, M. 2021. Creating the Future: Augmented Reality, the next Human-Machine Interface. In *2021 IEEE International Electron Devices Meeting (IEDM)*, 1–11.

Alayrac, J.-B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; Ring, R.; Rutherford, E.; Cabi, S.; Han, T.; Gong, Z.; Samangooei, S.; Monteiro, M.; Menick, J.; Borgeaud, S.; Brock, A.; Nematzadeh, A.; Sharifzadeh, S.; Binkowski, M.; Bar-

reira, R.; Vinyals, O.; Zisserman, A.; and Simonyan, K. 2022. Flamingo: a visual language model for few-shot learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713871088.

Arm®. 2022. Arm Ethos-U55 microNPU Description. https://www.arm.com/products/silicon-ip-cpu/ethos/ethos-u55.

Banitalebi-Dehkordi, A.; Vedula, N.; Pei, J.; Xia, F.; Wang, L.; and Zhang, Y. 2021. Auto-Split: A General Framework of Collaborative Edge-Cloud AI. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*.

Bolukbasi, T.; Wang, J.; Dekel, O.; and Saligrama, V. 2017. Adaptive Neural Networks for Efficient Inference. In *Proceedings of the International Conference on Machine Learning (ICML)*, 527–536.

Bolya, D.; Fu, C.-Y.; Dai, X.; Zhang, P.; Feichtenhofer, C.; and Hoffman, J. 2023. Token Merging: Your ViT But Faster. In *International Conference on Learning Representations (ICLR)*.

Bossard, L.; Guillaumin, M.; and Van Gool, L. 2014. Food-101 – Mining Discriminative Components with Random Forests. In *European Conference on Computer Vision*.

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2020. Once-for-All: Train One Network and Specialize it for Efficient Deployment. In *International Conference on Learning Representations*.

Chen, L.; Zaharia, M.; and Zou, J. 2023. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *arXiv preprint arXiv:2305.05176*.

Chen, L.; Zhao, H.; Liu, T.; Bai, S.; Lin, J.; Zhou, C.; and Chang, B. 2024. An Image is Worth 1/2 Tokens After Layer 2: Plug-and-Play Inference Acceleration for Large Vision-Language Models. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Chen, M.; Lin, M.; Li, K.; Shen, Y.; Wu, Y.; Chao, F.; and Ji, R. 2023a. CF-ViT: A General Coarse-to-Fine Method for Vision Transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Chen, X.; Wang, X.; Changpinyo, S.; Piergiovanni, A.; Padlewski, P.; Salz, D.; Goodman, S.; Grycner, A.; Mustafa, B.; Beyer, L.; et al. 2023b. PaLI: A Jointly-Scaled Multilingual Language-Image Model. In *The Eleventh International Conference on Learning Representations*.

Cho, S.; Shin, H.; Hong, S.; Arnab, A.; Seo, P. H.; and Kim, S. 2024. CAT-Seg: Cost Aggregation for Open-Vocabulary Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chu, X.; Zhang, B.; and Xu, R. 2021. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on computer vision*, 12239–12248.

Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; and Vedaldi, A. 2014. Describing Textures in the Wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Ding, D.; Mallick, A.; Wang, C.; Sim, R.; Mukherjee, S.; Ruhle, V.; Lakshmanan, L. V. S.; and Awadallah, A. H. 2024. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. In *International Conference on Learning Representations (ICLR)*.

Driess, D.; Xia, F.; Sajjadi, M. S. M.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; Huang, W.; Chebotar, Y.; Sermanet, P.; Duckworth, D.; Levine, S.; Vanhoucke, V.; Hausman, K.; Toussaint, M.; Greff, K.; Zeng, A.; Mordatch, I.; and Florence, P. 2023. PaLM-E: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Fan, Y.; Ma, X.; Wu, R.; Du, Y.; Li, J.; Gao, Z.; and Li, Q. 2025. VideoAgent: A Memory-Augmented Multimodal Agent for Video Understanding. In Leonardis, A.; Ricci, E.; Roth, S.; Russakovsky, O.; Sattler, T.; and Varol, G., eds., *Computer Vision – ECCV 2024*, 75–92. Cham: Springer Nature Switzerland. ISBN 978-3-031-72670-5.

Fang, A.; Jose, A. M.; Jain, A.; Schmidt, L.; Toshev, A. T.; and Shankar, V. 2024. Data Filtering Networks. In *The Twelfth International Conference on Learning Representations*.

Fedus, W.; Zoph, B.; and Shazeer, N. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research (JMLR)*, 23(120): 1–39.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2007. 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.

Fellbaum, C. 1998. *WordNet: An electronic lexical database*. MIT press.

Figurnov, M.; Collins, M. D.; Zhu, Y.; Zhang, L.; Huang, J.; Vetrov, D.; and Salakhutdinov, R. 2017. Spatially Adaptive Computation Time for Residual Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gebru, T.; Krause, J.; Wang, Y.; Chen, D.; Deng, J.; and Fei-Fei, L. 2017. Fine-grained car detection for visual census estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Georg, M.; Tanzer, G.; Uboweja, E.; Hassan, S.; Shengelia, M.; Sepah, S.; Forbes, S.; and Starner, T. 2025. 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)*, 13897–13906.

Ghiasi, G.; Gu, X.; Cui, Y.; and Lin, T.-Y. 2022. Scaling open-vocabulary image segmentation with image-level labels. In *European conference on computer vision*, 540–557. Springer.

Google. 2026. Gemini Developer API Pricing. https://ai.google.dev/gemini-api/docs/pricing. Accessed 2026-07-04.

Goswami, R. G.; Krishnamurthy, P.; LeCun, Y.; and Khorrami, F. 2025. 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)*, 6930–6939.

Grauman, K.; Westbury, A.; Byrne, E.; Chavis, Z.; Furnari, A.; Girdhar, R.; Hamburger, J.; Jiang, H.; Liu, M.; Liu, X.; et al. 2022. Ego4D: Around the World in 3,000 Hours of Egocentric Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Grauman, K.; Westbury, A.; Torresani, L.; Kitani, K.; Malik, J.; Afouras, T.; Ashutosh, K.; Baiyya, V.; Bansal, S.; Boote, B.; et al. 2024. Ego-Exo4D: Understanding Skilled Human Activity from First- and Third-Person Perspectives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gu, X.; Lin, T.-Y.; Kuo, W.; and Cui, Y. 2022. Open-vocabulary Object Detection via Vision and Language Knowledge Distillation. In *International Conference on Learning Representations (ICLR)*.

Gupta, T.; and Kembhavi, A. 2023. Visual Programming: Compositional Visual Reasoning Without Training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations (ICLR)*.

Han, Y.; Huang, G.; Song, S.; Yang, L.; Wang, H.; and Wang, Y. 2022. Dynamic Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(11): 7436–7456.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2022. Masked Autoencoders Are Scalable Vision Learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.

Hoang, M. L. 2025. A Review of Developments and Metrology in Machine Learning and Deep Learning for Wearable IoT Devices. *IEEE Access*.

Hooker, S. 2021. The Hardware Lottery. *Communications of the ACM*, 64(12): 58–65.

Hu, W.; Dou, Z.-Y.; Li, L. H.; Kamath, A.; Peng, N.; and Chang, K.-W. 2024. Matryoshka Query Transformer for Large Vision-Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Huang, G.; Chen, D.; Li, T.; Wu, F.; van der Maaten, L.; and Weinberger, K. Q. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *International Conference on Learning Representations (ICLR)*.

Ilharco, G.; Wortsman, M.; Carlini, N.; Taori, R.; Dave, A.; Shankar, V.; Namkoong, H.; Miller, J.; Hajishirzi, H.; Farhadi, A.; et al. 2021. Openclip. *Zenodo*.

Imam, M. F.; Marew, R. F.; Hassan, J.; Fiaz, M.; Aji, A. F.; and Cholakkal, H. 2024. CLIP meets DINO for Tuning Zero-Shot Classifier using Unlabeled Image Collections. *arXiv preprint arXiv:2411.19346*.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2704–2713.

Jitkrittum, W.; Gupta, N.; Menon, A. K.; Narasimhan, H.; Rawat, A. S.; and Kumar, S. 2023. When Does Confidence-Based Cascade Deferral Suffice? In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jose, C.; Moutakanni, T.; Kang, D.; Baldassarre, F.; Darcet, T.; Xu, H.; Li, D.; Szafraniec, M.; Ramamonjisoa, M.; Oquab, M.; et al. 2025. Dinov2 meets text: A unified framework for image-and pixel-level vision-language alignment. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 24905–24916.

Kang, Y.; Hauswald, J.; Gao, C.; Rovinski, A.; Mudge, T. N.; Mars, J.; and Tang, L. 2017. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 615–629.

Kavukcuoglu, K. 2025. Gemini-2.5: Our Most Intelligent AI Model. Google DeepMind Blog.

Khattak, M. U.; Naeem, M. F.; Naseer, M.; Van Gool, L.; and Tombari, F. 2025. Learning to prompt with text only supervision for vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 4230–4238.

Kim, S.-Y.; Chung, D.-o.; Lee, K.; Lee, C.; and Huh, J. 2023. Low-power always-on camera (AoC) system with workload offloading to CMOS image sensor. In *2023 IEEE International Conference on Consumer Electronics (ICCE)*, 1–2. IEEE.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images.

Lane, N. D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Jiao, L.; Qendro, L.; and Kawsar, F. 2016. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 1–12.

Laskaridis, S.; Venieris, S. I.; Almeida, M.; Leontiadis, I.; and Lane, N. D. 2020. SPINN: Synergistic Progressive Inference of Neural Networks over Device and Cloud. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom)*.

Li, B.; Weinberger, K. Q.; Belongie, S.; Koltun, V.; and Ranftl, R. 2022. Language-driven Semantic Segmentation. In *International Conference on Learning Representations (ICLR)*.

Li, C.; Tang, T.; Wang, G.; Peng, J.; Wang, B.; Liang, X.; and Chang, X. 2021a. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12281–12291.

Li, C.; Wang, G.; Wang, B.; Liang, X.; Li, Z.; and Chang, X. 2021b. Dynamic Slimmable Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, E.; Zhou, Z.; and Chen, X. 2018. Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy. In *Proceedings of the 2018 Workshop on Mobile Edge Communications*, MECOMM'18, 31–36. New York, NY, USA: Association for Computing Machinery. ISBN 9781450359061.

Li, J.; Li, D.; Savarese, S.; and Hoi, S. 2023. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Li, J. N.; Zhang, Z. J.; and Ma, J. 2025. OmniQuery: Contextually Augmenting Captured Multimodal Memories to Enable Personal Question Answering. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25. New York, NY, USA: Association for Computing Machinery. ISBN 9798400713941.

Li, P.; Burges, C. J. C.; and Wu, Q. 2007. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. In *Advances in Neural Information Processing Systems*, volume 20, 897–904.

Liang, F.; Wu, B.; Dai, X.; Li, K.; Zhao, Y.; Zhang, H.; Zhang, P.; Vajda, P.; and Marculescu, D. 2023. Open-Vocabulary Semantic Segmentation with Mask-adapted CLIP. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liang, Y.; Ge, C.; Tong, Z.; Song, Y.; Wang, J.; and Xie, P. 2022. Not All Patches Are What You Need: Expediting Vision Transformers via Token Reorganizations. In *International Conference on Learning Representations (ICLR)*.

Lin, R.; Weng, P.; Wang, Y.; Ding, H.; Han, J.; and Wang, F. 2025. 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)*, 1429–1438.

Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11976–11986.

Lu, P.; Peng, B.; Cheng, H.; Galley, M.; Chang, K.-W.; Wu, Y. N.; Zhu, S.-C.; and Gao, J. 2023. Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lu, Z.; Sreekumar, G.; Goodman, E.; Banzhaf, W.; Deb, K.; and Boddeti, V. N. 2021. Neural Architecture Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9): 2971–2989.

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Maji, S.; Kannala, J.; Rahtu, E.; Blaschko, M.; and Vedaldi, A. 2013. Fine-Grained Visual Classification of Aircraft. Technical report.

Matsubara, Y.; Levorato, M.; and Restuccia, F. 2022. Split Computing and Early Exiting for Deep Learning Applications: Survey and Research Challenges. *ACM Computing Surveys*, 55(5): 1–30.

Mehta, S.; and Rastegari, M. 2022. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer. In *International Conference on Learning Representations*.

Meng, L.; Li, H.; Chen, B.-C.; Lan, S.; Wu, Z.; Jiang, Y.-G.; and Lim, S.-N. 2022. AdaViT: Adaptive Vision Transformers for Efficient Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Menon, S.; and Vondrick, C. 2023. Visual Classification via Description from Large Language Models. In *International Conference on Learning Representations (ICLR)*.

Meta AI. 2025. The Llama 4 Herd: The Beginning of a New Era of Natively Multimodal AI Innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Accessed April 5, 2025.

Meta Platforms, I. 2023. Meta Ray-Ban Smart Glasses. https://www.meta.com/ai-glasses/ray-ban-meta/. A series of AI-enabled smart glasses combining camera, audio, and voice-controlled Meta AI features, developed by Meta Platforms in partnership with Ray-Ban.

Mirza, M. J.; Karlinsky, L.; Lin, W.; Possegger, H.; Kozinski, M.; Feris, R.; and Bischof, H. 2023. Lafter: Label-free tuning of zero-shot classifier using language and unlabeled image collections. *Advances in Neural Information Processing Systems*, 36: 5765–5777.

Ong, I.; Almahairi, A.; Wu, V.; Chiang, W.-L.; Wu, T.; Gonzalez, J. E.; Kadous, M. W.; and Stoica, I. 2025. RouteLLM: Learning to Route LLMs from Preference Data. In *International Conference on Learning Representations (ICLR)*.

OpenAI. 2025. Introducing GPT-5. https://openai.com/index/introducing-gpt-5/. Large language model.

OpenAI. 2026. OpenAI API Pricing. https://developers.openai.com/api/docs/pricing. Accessed 2026-07-04.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; et al. 2024. DINOv2: Learning Robust Visual Features without Supervision. *Transactions on Machine Learning Research Journal*, 1–31.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, 3498–3505. IEEE.

Pratt, S.; Covert, I.; Liu, R.; and Farhadi, A. 2023. 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*, 15691–15701.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PmLR.

Rao, Y.; Zhao, W.; Liu, B.; Lu, J.; Zhou, J.; and Hsieh, C.-J. 2021. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y. L.; Tan, J.; Le, Q. V.; and Kurakin, A. 2017. Large-scale evolution of image classifiers. In *International conference on machine learning*, 2902–2911. PMLR.

Riquelme, C.; Puigcerver, J.; Mustafa, B.; Neumann, M.; Jenatton, R.; Susano Pinto, A.; Keysers, D.; and Houlsby, N. 2021. Scaling Vision with Sparse Mixture of Experts. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Roth, K.; Kim, J. M.; Koepke, A. S.; Vinyals, O.; Schmid, C.; and Akata, Z. 2023. Waffling around for performance: Visual classification with random words and broad concepts. In *Proceedings of the IEEE/CVF international conference on computer vision*, 15746–15757.

Seneviratne, S.; Hu, Y.; Nguyen, T.; Lan, G.; Khalifa, S.; Thilakarathna, K.; Hassan, M.; and Seneviratne, A. 2017. A survey of wearable devices and challenges. *IEEE Communications Surveys & Tutorials*, 19(4): 2573–2620.

Serianni, A.; and Kalita, J. 2023. Training-free Neural Architecture Search for RNNs and Transformers. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2522–2540. Toronto, Canada: Association for Computational Linguistics.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations (ICLR)*.

Shen, Y.; Song, K.; Tan, X.; Li, D.; Lu, W.; and Zhuang, Y. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shrivastava, A.; Selvaraju, R. R.; Naik, N.; and Ordonez, V. 2023. Clip-lite: Information efficient visual representation learning with language supervision. In *International Conference on Artificial Intelligence and Statistics*, 8433–8447. PMLR.

Skillman, A.; and Edsö, T. 2020. A Technical Overview of Cortex-M55 and Ethos-U55: Arm's Most Capable Processors for Endpoint AI. In *2020 IEEE Hot Chips 32 Symposium (HCS)*, 1–20.

Stauffert, J.-P.; Niebling, F.; and Latoschik, M. E. 2016. Reducing application-stage latencies for real-time interactive systems. In *2016 IEEE 9th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, 1–7. IEEE.

Surís, D.; Menon, S.; and Vondrick, C. 2023. ViperGPT: Visual Inference via Python Execution for Reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Teerapittayanon, S.; McDanel, B.; and Kung, H. T. 2016. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. In *International Conference on Pattern Recognition (ICPR)*.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training Data-Efficient Image Transformers & Distillation Through Attention. In *International Conference on Machine Learning (ICML)*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vasu, P. K. A.; Pouransari, H.; Faghri, F.; Vemulapalli, R.; and Tuzel, O. 2024. MobileCLIP: Fast Image-Text Models through Multi-Modal Reinforced Training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15963–15974.

Wang, X.; Yu, F.; Dou, Z.-Y.; Darrell, T.; and Gonzalez, J. E. 2018. SkipNet: Learning Dynamic Routing in Convolutional Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Wang, Y.; Huang, R.; Song, S.; Huang, Z.; and Huang, G. 2021. Not All Images Are Worth 16x16 Words: Dynamic Transformers for Efficient Image Recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, Y.; Jiang, X.; Cheng, D.; Li, D.; and Zhao, C. 2024. Learning Hierarchical Prompt with Structured Linguistic Knowledge for Vision-Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Wang, Y.; Lv, K.; Huang, R.; Song, S.; Yang, L.; and Huang, G. 2020. Glance and Focus: A Dynamic Approach to Reducing Spatial Redundancy in Image Classification. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Woo, S.; Debnath, S.; Hu, R.; Chen, X.; Liu, Z.; Kweon, I. S.; and Xie, S. 2023. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16133–16142.

Wu, C.; Yin, S.; Qi, W.; Wang, X.; Tang, Z.; and Duan, N. 2023a. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. *arXiv preprint arXiv:2303.04671*.

Wu, C.-E.; Lin, J.; Hu, Y. H.; and Morgado, P. 2025. Patch Ranking: Token Pruning as Ranking Prediction for Efficient CLIP. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 5842–5851.

Wu, K.; Peng, H.; Zhou, Z.; Xiao, B.; Liu, M.; Yuan, L.; Xuan, H.; Valenzuela, M.; Chen, X. S.; Wang, X.; et al. 2023b. Tinyclip: Clip distillation via affinity mimicking and

weight inheritance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 21970–21980.

Wu, Z.; Nagarajan, T.; Kumar, A.; Rennie, S.; Davis, L. S.; Grauman, K.; and Feris, R. 2018. BlockDrop: Dynamic Inference Paths in Residual Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wu, Z.; Xiong, C.; Ma, C.-Y.; Socher, R.; and Davis, L. S. 2019. AdaFrame: Adaptive Frame Selection for Fast Video Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiao, J.; Hays, J.; Ehinger, K. A.; Oliva, A.; and Torralba, A. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, 3485–3492. IEEE.

Xing, L.; Huang, Q.; Dong, X.; Lu, J.; Zhang, P.; Zang, Y.; Cao, Y.; He, C.; Wang, J.; Wu, F.; and Lin, D. 2025. PyramidDrop: Accelerating Your Large Vision-Language Models via Pyramid Visual Redundancy Reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xing, Y.; Kang, J.; Xiao, A.; Nie, J.; Shao, L.; and Lu, S. 2023. Rewrite caption semantics: Bridging semantic gaps for language-supervised semantic segmentation. *Advances in Neural Information Processing Systems*, 36: 68798–68809.

Xu, H.; Xie, S.; Tan, X.; Huang, P.-Y.; Howes, R.; Sharma, V.; Li, S.-W.; Ghosh, G.; Zettlemoyer, L.; and Feichtenhofer, C. 2024. Demystifying CLIP Data. In *The Twelfth International Conference on Learning Representations*.

Xu, J.; Hou, J.; Zhang, Y.; Feng, R.; Wang, Y.; Qiao, Y.; and Xie, W. 2023a. Learning open-vocabulary semantic segmentation models from natural language supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2935–2944.

Xu, J.; Liu, S.; Vahdat, A.; Byeon, W.; Wang, X.; and De Mello, S. 2023b. Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xu, M.; Zhang, Z.; Wei, F.; Hu, H.; and Bai, X. 2023c. Side Adapter Network for Open-Vocabulary Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xu, Z.; Nguyen, K. D.; Mukherjee, P.; Bagchi, S.; Chaterji, S.; Liang, Y.; and Li, Y. 2025. Learning to Inference Adaptively for Multimodal Large Language Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Xu, Z. E.; Weinberger, K. Q.; and Chapelle, O. 2012. The Greedy Miser: Learning under Test-Time Budgets. In *Proceedings of the 29th International Conference on Machine Learning*.

Yang, C.; An, Z.; Huang, L.; Bi, J.; Yu, X.; Yang, H.; Diao, B.; and Xu, Y. 2024. CLIP-KD: An Empirical Study of CLIP Model Distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, L.; Han, Y.; Chen, X.; Song, S.; Dai, J.; and Huang, G. 2020. Resolution Adaptive Networks for Efficient Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yin, H.; Vahdat, A.; Alvarez, J.; Mallya, A.; Kautz, J.; and Molchanov, P. 2022. A-ViT: Adaptive Tokens for Efficient Vision Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu, J.; and Huang, T. S. 2019. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1803–1811.

Yu, J.; Jin, P.; Liu, H.; Bender, G.; Kindermans, P.-J.; Tan, M.; Huang, T.; Song, X.; Pang, R.; and Le, Q. 2020. Bignas: Scaling up neural architecture search with big single-stage models. In *Computer Vision–ECCV 2020: 16th European Conference, Part VII 16*, 702–717. Springer.

Yu, J.; Yang, L.; Xu, N.; Yang, J.; and Huang, T. 2019. Slimmable Neural Networks. In *International Conference on Learning Representations (ICLR)*.

Yu, Q.; He, J.; Deng, X.; Shen, X.; and Chen, L.-C. 2023. Convolutions Die Hard: Open-Vocabulary Segmentation with Single Frozen Convolutional CLIP. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yuan, Z.; Xue, C.; Chen, Y.; Wu, Q.; and Sun, G. 2022. PTQ4ViT: Post-Training Quantization for Vision Transformers with Twin Uniform Quantization. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R.; and Smola, A. J. 2017. Deep Sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Zeng, L.; Chen, X.; Zhou, Z.; Yang, L.; and Zhang, J. 2020. CoEdge: Cooperative DNN Inference with Adaptive Workload Partitioning over Heterogeneous Edge Devices. *IEEE/ACM Transactions on Networking*.

Zhai, X.; Mustafa, B.; Kolesnikov, A.; and Beyer, L. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, 11975–11986.

Zhai, X.; Wang, X.; Mustafa, B.; Steiner, A.; Keysers, D.; Kolesnikov, A.; and Beyer, L. 2022. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 18123–18133.

Zhang, H.; Zhang, P.; Hu, X.; Chen, Y.-C.; Li, L. H.; Dai, X.; Wang, L.; Yuan, L.; Hwang, J.-N.; and Gao, J. 2022. Glipv2: Unifying localization and vision-language understanding. *Advances in Neural Information Processing Systems*, 35: 36067–36080.

Zhang, Y.; Fan, C.-K.; Ma, J.; Zheng, W.; Huang, T.; Cheng, K.; Gudovskiy, D.; Okuno, T.; Nakata, Y.; Keutzer, K.; and Zhang, S. 2025. SparseVLM: Visual Token Sparsification for Efficient Vision-Language Model Inference. In *International Conference on Machine Learning (ICML)*.

Zhao, Y.; Chen, J.; Zhang, S. Q.; Sarwar, S. S.; Stangherlin, K. H.; Gomez, J. T.; Seo, J.-S.; De Salvo, B.; Liu, C.; Gibbons, P. B.; and Li, Z. 2025. H4H: Hybrid Convolution-Transformer Architecture Search for NPU-CIM Heterogeneous Systems for AR/VR Applications. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, ASPDAC '25, 1133–1141. New York, NY, USA: Association for Computing Machinery. ISBN 9798400706356.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017. Scene Parsing through ADE20K Dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhou, B.; Zhao, H.; Puig, X.; Xiao, T.; Fidler, S.; Barriuso, A.; and Torralba, A. 2019. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3): 302–321.

Zhou, J.; Wei, C.; Wang, H.; Shen, W.; Xie, C.; Yuille, A.; and Kong, T. 2022a. iBOT: Image BERT Pre-Training with Online Tokenizer. In *International Conference on Learning Representations (ICLR)*.

Zhou, K.; Yang, J.; Loy, C. C.; and Liu, Z. 2022b. Conditional Prompt Learning for Vision-Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, K.; Yang, J.; Loy, C. C.; and Liu, Z. 2022c. Learning to Prompt for Vision-Language Models. *International Journal of Computer Vision (IJCV)*, 130(9): 2337–2348.

Zoph, B.; and Le, Q. 2017. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*.

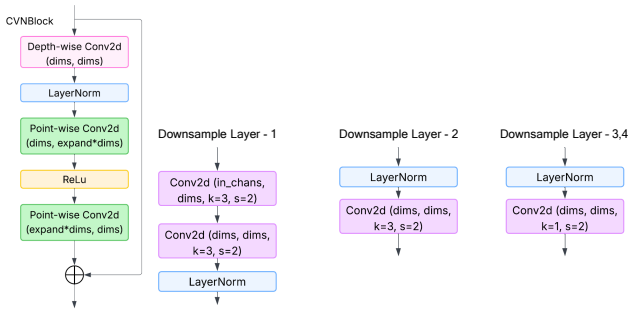

Figure 8: **(a) Left**: Selective ConvNeXt-v2 blocks. **(b) Right**: Downsample layers. The block widths ($dims$) are ***selectable*** during training and runtime.

A Architecture of Basic Blocks

A.1 ConvNext-v2 Blocks

We use ConvNeXt-v2 (Woo et al. 2023) with ***selective*** capacity as the core building block of our model, as shown in Fig. 8a. All block widths ($dims$) are selectable during training and runtime, enabling the ***adaptive*** behavior of our model. We also replace GELU with ReLU for better compatibility and efficiency on edge devices.

A.2 Downsample Layers

We provide the basic structures of the downsampling layers we use in Fig. 8b. As with the ConvNeXt blocks, the block widths ($dims$) are ***selectable*** and 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 Training Infrastructure

Our models are trained on a cluster consisting of 48 AMD EPYC 7282 CPUs and 128 NVIDIA A100 PCIe GPUs, each with 80GB of GPU memory. All reported experimental results, including both accuracy and efficiency measurements, are averaged over at least three independent runs.

C Hardware Platform and Evaluation Setup

We adopt the ARM Ethos-U55 (Arm® 2022; Skillman and Edsö 2020) as a representative edge Neural Processing Unit (NPU). The test silicon (Fig. 9) is fabricated in 7 nm FinFET and reflects the compute design of edge AR/VR devices such as Meta Ray-Ban (Meta Platforms 2023). It operates at 560 MHz, delivers $>$128 GOP/s, occupies $\sim$7.8 mm^2, and provides on-chip memory with 4 GB/s aggregate bandwidth.

The device runs the Zephyr real-time operating system. We deploy and evaluate various DNN layer types—standard, depth-wise, and point-wise convolutions; fully connected layers; attention and other transformer components; as well as activations and others—on the NPU using the ARM Ethos-U Vela toolchain. The measured metrics of efficiency include execution latency and energy consumption of models. The reported results in the main paper and supplementary material include averaged overheads of runtime wrappers, subnet switching, and other related costs.

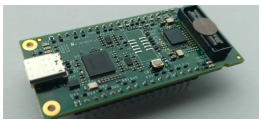

Figure 9: Our ARM Ethos-U55 test silicon.

On-Device Memory Footprint. Because every operating point is a subnet of one weight-sharing supernet, the device holds a single checkpoint rather than a separate model per size. The supernet spans 6.2–52.3M parameters (table 1); at its largest it is about 52 MB at 8-bit integer precision and 105 MB at 16-bit—the two integer widths the Ethos-U55 datapath executes. A smaller subnet activates only a subset of these weights, so parameter memory scales down with the selected width and depth, to roughly 6 MB for MIN at 8-bit. Covering the same range of operating points with independent standalone backbones would instead cost the sum of their sizes, whereas the shared supernet keeps on-device storage at that of its single largest member while still exposing the full accuracy–efficiency frontier at runtime. Activation and working-set memory depend on input resolution and the toolchain's buffer allocation and lie outside this parameter budget.

Standard Components by Design. AdaDINO is built from established, well-optimized components—a ConvNeXt-style convolutional backbone, weight-sharing NAS, foundation-model distillation, and a CLIP-style alignment recipe—rather than new operators. For an always-on edge target this is a deployment choice, not a gap in novelty. A fixed-function NPU such as the Ethos-U55 accelerates only the operators its compiler supports: the Vela toolchain maps supported layers to the NPU and leaves the rest to run on the far slower embedded CPU (Arm® 2022; Skillman and Edsö 2020). This is a concrete case of the *hardware lottery*, where an idea prevails largely by fitting the available hardware and tooling rather than on merit alone (Hooker 2021), so an operator without mature kernel and compiler support is handicapped whatever its accuracy. Theoretical cost only compounds this, since FLOPs is a loose proxy for on-device latency, which also turns on memory access and platform support (Ma et al. 2018). Favoring standard, widely-optimized primitives keeps the whole model family deployable and fast on the target, whereas novel operators often fall back or fail to map and never ship. The contribution is accordingly not a new backbone but the task-context adaptation on top—semantic class filtering and learned subnet selection from the class vocabulary—which is independent of the backbone and improves as a stronger, equally deployable one becomes available.

D Per-Subnet Per-Task Accuracy of the NAS-Distilled Vision Backbone

Table 5 reports the accuracy of the five representative subnets drawn from our NAS-distilled vision backbone across ten vision datasets. Except IN1K (kNN), all datasets use logreg classifiers. Compared with standalone-distilled models (last column), the largest NAS-based subnet (LARGE) exhibits only ***marginal*** drops—at most 0.9% (on cars) and 0.3% on IN1K—confirming that weight sharing across the supernet incurs negligible cost at the high-capacity end. Sensitivity to model size, however, varies substantially across datasets: fine-grained tasks such as cars and Food101 degrade sharply for the smallest subnet (MIN: -18.4% and -11.1%), whereas coarse-grained tasks such as Caltech-101 and Pets stay within a few points. This heterogeneity is precisely what motivates adaptive, task-conditioned subnet selection at the edge rather than a single fixed model.

E Granularity of User-Defined Classes is an Indicator of Task Difficulty

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) (Deng et al. 2009) and ADE20K (Zhou et al. 2019). 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 (Fellbaum 1998). For ADE20K, we also evaluate two settings: the original 150-class segmentation and a 9-class segmentation based on WordNet grouping with granularity similar to the 7-class IN1K.

To construct these coarse-grained settings, we group the IN1K categories into super-classes as follows. For each class name, we first locate its entry in WordNet. 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 the following set of seven super-classes.

Seven Super-classes of ImageNet-1K

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

The ADE20K dataset (Zhou et al. 2019) is processed in a similar manner, but applies a heuristic class-name optimization from DINO.txt (Jose et al. 2025) 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

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 keeping nearly the same accuracy. This observation is also visualized in fig. 2.

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

The same effect also appears at the *dataset* level. Table 7 reports zero-shot accuracy of the DINOv2 family across four classification datasets: simpler datasets such as Cal101 and Pets lose little accuracy with smaller models, whereas IN1K and Food101 degrade substantially, mirroring the granularity effect above.

F LLM-Based Runtime Scene Understanding

F.1 Prompts for Runtime Scene Annotation Generation

For open-vocabulary segmentation on ADE20K (Zhou et al. 2019), given an infrequently sampled image, we use the following prompts for Llama4 and Llama3.2 (Meta AI 2025) to generate a scene annotation.

Llama Prompt for Runtime Scene Annotation Generation

Given an input image, imagine it comes from a computer vision benchmark such as ADE20K.

1. Infer the single most appropriate scene or location category depicted in the image.
2. Use a WordNet-style noun phrase that is semantically aligned with ADE20K scene labels (e.g., "residential street", "indoor office", "natural park", "industrial warehouse").
3. Prefer generic, canonical scene names over narrative descriptions.

Dataset	NAS-based Distillation					Standalone
	MIN	TINY	SMALL	BASE	LARGE	LARGE
IN1K (Deng et al. 2009)	69.4(-9.4)	74.0	76.6	77.7	78.8(-0.3)	79.1
Cal101 (Fei-Fei, Fergus, and Perona 2007)	95.2(-1.3)	95.5	96.0	96.6	96.5(+0.3)	96.2
Pets (Parkhi et al. 2012)	91.4(-3.3)	92.9	93.9	94.6	94.7(+0.2)	94.5
Cifar10 (Krizhevsky and Hinton 2009)	93.7(-4.4)	95.8	97.1	97.9	98.1(-0.1)	98.2
DTD (Cimpoi et al. 2014)	76.2(-4.8)	79.4	80.4	80.7	81.0(-0.3)	81.3
SUN397 (Xiao et al. 2010)	69.7(-6.9)	73.2	75.3	76.2	76.6(0.0)	76.6
aircraft (Maji et al. 2013)	61.9(-9.0)	66.1	68.9	69.8	70.9(+0.1)	70.8
Cifar100 (Krizhevsky and Hinton 2009)	78.7(-10.2)	83.1	86.1	87.9	88.9(-0.2)	89.1
Food101 (Bossard, Guillaumin, and Van Gool 2014)	77.4(-11.1)	82.6	86.1	87.5	88.5(-0.1)	88.6
cars (Gebru et al. 2017)	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 vision tasks. Except IN1K (kNN), all other datasets use logreg classifiers. Compared with standalone-distilled models (last column), the NAS-based model shows only ***marginal*** drops (differences in parenthesis of LARGE). Sensitivity to model size varies across datasets (differences of MIN), highlighting adaptive model selection based on task complexity.

Dataset	MIN	TINY	SMALL	BASE	LARGE
IN1K (Deng et al. 2009)	65.5	69.8	71.3	72.4	73.3
IN-WN7	73.2	73.8	74.1	74.1	74.3
ADE (Zhou et al. 2019)	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 and mIoU on ADE20K. IN1K/ADE are original datasets, and IN-WN7/ADE-WN9 are 7-/9-class WordNet-based groupings. Coarser groupings show smaller accuracy drops as model size decreases, indicating lower task difficulty.

Output format (mandatory):
Output only one phrase.
Do NOT use a code block, do NOT use markdown, and do NOT include quotes.
Only output the raw string itself (e.g., residential street).

F.2 Prompts for Runtime Scene-Aware Filtering

For open-vocabulary segmentation on ADE20K (Zhou et al. 2019), given a scene context, we use the following prompts for GPT-5 (OpenAI 2025) and Gemini-2.5 (Kavukcuoglu 2025) to filter out implausible class names in the scene from the full set of 150 classes.

GPT-5 Prompt for Runtime Scene-Aware Filtering

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.

If multiple object names refer to ambiguous concepts, keep only one most likely to use when describing this particular scene.
For example:
- In a "sportsfield" scene, prefer "field" over "grass".
- In an outdoor scene, prefer "ground" over "floor".
- In an indoor scene, prefer "floor" over "ground".
- In an indoor scene, prefer "wall" over "building".
If there is no clear preference, you can keep both.

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

Gemini-2.5 Prompt for Runtime Scene-Aware Filtering

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.

DINOv2	#params	FLOPs	IN1K	Food101	Cal101	Pets
ViT-S	22M	4.6G	79.0 (-4.4)	89.1 (-5.6)	97.0 (-0.6)	95.1 (-1.6)
ViT-B	86M	17.6G	82.1 (-1.3)	92.8 (-1.9)	96.1 (-1.5)	96.2 (-0.5)
ViT-L	304M	61G	83.5 (+0.1)	94.3 (-0.4)	97.5 (-0.1)	96.6 (-0.1)
ViT-g	1.8B	700G	83.4	94.7	97.6	96.7

Table 7: DINOv2 results. Numbers in parentheses show drops relative to ViT-g. Simple tasks (Cal101 (Fei-Fei, Fergus, and Perona 2007), Pets (Parkhi et al. 2012)) exhibit small drops ($\leq$ 1.6) as model sizes decrease, while moderate/complex tasks (IN1K (Deng et al. 2009), Food101 (Bossard, Guillaumin, and Van Gool 2014)) show larger drops (up to 5.6).

- A total of 150 key-value pairs.
- Values must be integers 0 or 1 only.

If multiple object names refer to ambiguous concepts, keep only one most likely to use when describing this particular scene.
For example:
- In a "sportsfield" scene, prefer "field" over "grass".
- In an outdoor scene, prefer "ground" over "floor".
- In an indoor scene, prefer "floor" over "ground".
- In an indoor scene, prefer "wall" over "building".
If there is no clear preference, you can keep both.

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.

Top-k Prompt for GPT-5 for Runtime Scene-Aware Filtering

Consider computer vision tasks in a scene of $scene.
Given a scene description and a list of 150 object names, output only the top $***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.

If multiple object names refer to ambiguous concepts, keep only one most likely to use when describing this particular scene.
For example:
- In a "sportsfield" scene, prefer "field" over "grass".
- In an outdoor scene, prefer "ground" over "floor".
- In an indoor scene, prefer "floor" over "ground".
- In an indoor scene, prefer "wall" over "building".
If there is no clear preference, you can keep both.

	GPT5	GPT5-mini	Gemini2.5	Top25
MIN	12.88 (+1.53)	12.75	12.57	13.35
TINY	14.52 (+1.44)	14.28	14.19	14.38
SMALL	14.71 (+1.46)	14.41	14.37	14.48
BASE	16.01 (+1.34)	15.77	15.79	15.80
LARGE	17.49 (+1.19)	17.13	17.20	16.82
ViT-L	21.00 (+0.86)	20.68	20.68	19.91

Table 8: ADE20K (Zhou et al. 2019) mIoU with different LLM-based *class filtering* methods. Incorporating LLM ***notably improves*** open-vocabulary segmentation for both AdaDINO and DINO.txt baseline (Jose et al. 2025), with *smaller* models benefiting more. Differences over the non-LLM Vanilla setting are shown in parentheses for GPT-5.

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

F.3 End-to-End Segmentation Results of Class Name Filtering

As shown in Table 8, incorporating LLM-based grouping and filtering of vocabulary yields substantial segmentation gains for *both* our adaptive AdaDINO and non-adaptive baselines (e.g., (Jose et al. 2025)). Our approach gains up to 1.76 mIoU over prior baselines and 1.53 mIoU over non-LLM Vanilla. Fig. 10 further shows that LLM-based understanding yields more regular, coherent, and semantically consistent masks.

Among the commercial LLMs we evaluated, GPT-5 (OpenAI 2025) achieves the best performance. Using alternative LLMs (e.g., Gemini-2.5 (Kavukcuoglu 2025)) or smaller variants (GPT-5-mini) results in noticeable performance degradation. The first three methods in Tab. 8 yield an indefinite number of candidate classes based on scene context. The fourth method restricts predictions of GPT-5 to a fixed top-25 class subset, which leads to poorer performance.

F.4 Quality of Scene-Aware Class Filtering

Table 9 shows the average recall and precision for each LLM prompt used in scene-aware class name filtering in ADE20K (Zhou et al. 2019). The manager is designed as a conservative vocabulary filter rather than an object detector: removing a class that is actually present hurts far more than retaining an extra plausible one, so the prompts deliberately prioritize recall over precision. Its benefit comes from removing semantically incompatible labels and resolv-

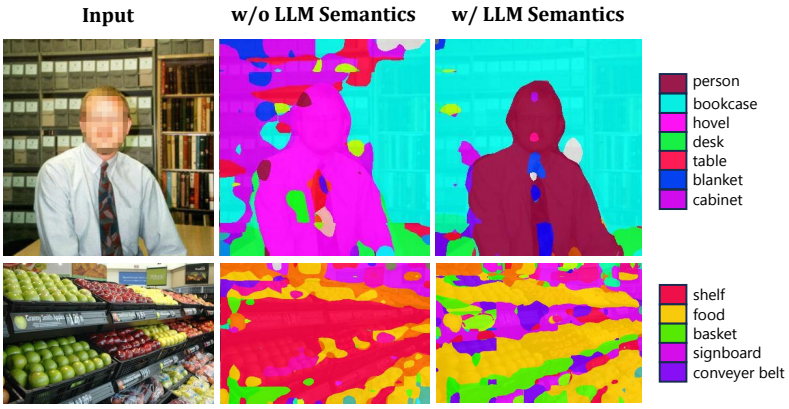

Figure 10: Open-vocabulary segmentation on ADE20K (Zhou et al. 2019) with and without LLM-based scene understanding. LLM guidance produces more *regular and coherent* masks.

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 9: Average recall and precision for each LLM prompt used in scene-aware class filtering on ADE20K (Zhou et al. 2019). Among these methods, GPT-5 achieves the best end-to-end performance for open-vocabulary segmentation.

Known scenes	AdaDINO (LLM)	Random	WordNet-Sim
80%	17.53	16.09	17.35
70%	17.53	15.17	17.27
60%	17.53	13.48	15.15
40%	17.53	12.56	15.13

Table 10: ADE20K (Zhou et al. 2019) mIoU of the LARGE subnet with LLM-based open-ended scene detection versus non-LLM scene classifiers restricted to a known subset of scenes. Despite an oracle scene-to-class lookup, the non-LLM baselines suffer noticeable drops that grow as fewer runtime scenes are known in advance.

ing dataset-specific ambiguity, not from pinpointing exactly the objects in the image. 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 (Jose et al. 2025).

F.5 LLM vs. Non-LLM Scene Detection under Out-of-Distribution Scenes

The cloud-side manager of AdaDINO relies on an LLM throughout its semantic pipeline: it understands the current scene, generates an open-ended scene label (§F.1), and then filters class names conditioned on that scene (§F.2). A natural question is whether such an LLM is necessary, or whether a lightweight non-LLM scene classifier could fulfill the same role. Non-LLM scene classifiers are efficient for closed-vocabulary settings, but AdaDINO targets ***open-vocabulary*** tasks, where runtime scenes may be out-of-distribution (OOD) with respect to any predefined scene set. Forcing them into a fixed scene set can mislead the downstream class filtering.

To validate this, we conduct an OOD open-vocabulary segmentation experiment on ADE20K (Zhou et al. 2019). The LLM-based method performs open-ended scene detection and class filtering, as in our final design. For the non-LLM baselines, we use a DINOv2-based (Oquab et al. 2024) scene classifier, but restrict its output to a subset of ADE20K scene labels, emulating deployments where only a fraction of runtime scenes are known in advance. We construct this scene subset in two ways: (*i*) *Random*, which randomly keeps a fraction of ADE20K scenes; and (*ii*) *WordNet-Sim*, which selects representative scene labels by clustering scene names with WordNet (Fellbaum 1998) similarity. The classifier maps each image to one of these subset scene labels. All non-LLM baselines are further given an ***oracle*** lookup from the predicted retained scene to its filtered class set, making them strong heuristics.

As shown in Table 10, despite the oracle lookup, the non-LLM baselines still suffer noticeable drops compared with the LLM-based method, and the gap widens as fewer runtime scenes are known in advance. These results support the need for the ***flexible semantic grounding*** of LLMs in open-vocabulary tasks: an LLM-based scene understander generalizes to OOD scenes, whereas a closed-vocabulary supervised classifier misassigns them to its fixed scene set and propagates the error into class filtering.

G Learned Selector: Implementation Details and Calibration

This section collects the implementation details of the learned selector (algorithm 1) in one place and evaluates

its retention predictor directly.

Inputs. The class names C of the current task are embedded by the Text Encoder into 2048-dimensional vectors, precomputed and fetched by lookup, so the selector runs no text-encoder pass of its own. SETSTATS turns them into a 27-d descriptor d that is invariant to the order and size of the class set. For each of two prompt templates ("a photo of *class*" and the bare class name), the embeddings are L2-normalized and eleven statistics of their pairwise cosine-similarity matrix are taken: mean, standard deviation, min, max, and five percentiles (10/25/50/75/90) of the off-diagonal entries, plus the mean and max of the per-class nearest-neighbor similarity. The class count and its log complete the descriptor, together with three schema fields that are constant across our segmentation tasks. Class sets with fewer than two names have no pairs, so the pairwise statistics are zeroed and the size features carry the signal. Computing d averages 1 ms per class set on a single CPU core. Each candidate subnet M is described by four capacity features—FLOPs, width, depth, and parameter count—each normalized by the largest subnet's value.

Training Data and Labels. Supervision comes from the ADE20K validation scenes: each validation image carries a scene-category label, and each scene's class set is its row in a human-verified scene-to-class table over the 150-class vocabulary. Dropping scenes without a usable class set or largest-subnet measurement leaves 664 scenes covering 1992 of the 2000 validation images, or $664 \times 5 = 3320$ (scene, subnet) rows. The label of a row is the retention ratio $\mathrm{Acc}(M, C)/\mathrm{MaxAcc}$, where accuracy is the scene's mIoU restricted to its class set and pooled over its images, and MaxAcc is the same measurement for the largest subnet. Most scenes hold a single validation image (median 1, mean 3.0), so scene-level labels are noisy, and scenes are weighted by their image counts during fitting.

Public Dataset Choice and Generalization. ADE20K is, to our knowledge, the public segmentation benchmark that supplies all three ingredients this supervision needs at once: a scene-category taxonomy, dense masks over one shared vocabulary, and scene labels from which a human-verified scene-to-class table can be built. Object- or stuff-centric benchmarks such as COCO-Stuff or Pascal Context carry no comparable scene taxonomy, so they cannot furnish per-scene retention labels without substantial extra annotation, and we run every reported selector study on ADE20K. The mechanism, though, is not tied to this vocabulary: the descriptor reads only statistics of class-name text embeddings and per-subnet capacity, both defined for any class set, so a selector fit on one vocabulary transfers to another with no change to the model. The settings this is ultimately meant for are in-house, open-vocabulary deployments whose data cannot be released. ADE20K is the public proxy on which the complete scene-to-class supervision happens to be available, and the held-out results below stand in for that target.

Regression Model and Calibration. A gradient-boosted regressor maps (d, M) to the retention ratio and is constrained to be monotone in all four capacity features (400 iterations, learning rate 0.05, 15 leaves per tree). Tree ensembles pull predictions toward the mean, which would blur the separation between easy and hard tasks, so the raw outputs are calibrated with isotonic regression fit on a held-out half of each training fold. Every selector prediction reported in this paper is made under leave-scene-out cross-validation: scenes are grouped into five folds, and each scene is served by a model that never saw it, so the evaluated scenes and their class sets are unseen at fit time.

Training vs. Deployment Inputs. The supervision above is built entirely from ground truth: scenes come from the dataset's scene-category labels, and each class set is a human-verified scene-to-class row. At inference, the selector instead receives what the runtime pipeline produces: a Llama4-generated scene annotation followed by GPT-5 class filtering (§6.2). The deployed selector thus runs on LLM-generated, *open-vocabulary* class sets that never appear in its fitting data, and the descriptor accepts them unchanged because it reads only statistics of class-name text embeddings. The diagnostics below probe the predictor on held-out scenes under the ground-truth protocol, isolating predictor quality from LLM variability, while the end-to-end results in the main paper exercise the full LLM-driven path.

Runtime Behavior. The selector does not name a subnet directly. It predicts a retention ratio for every subnet, and an explicit decision rule returns the least-cost subnet whose prediction meets α, falling back to the largest subnet when none qualifies (algorithm 1). This separation lets one trained selector serve different operating tolerances without retraining. The selector runs on the cloud once per context update, and its cost is negligible next to the MM-LLM call in the same update.

Predictor Accuracy. On scenes with reliable labels (at least three validation images), the deployed predictor reaches a mean absolute error of 0.086 on the retention ratio (RMSE 0.123). Single-image scenes, whose measured retention is itself noise-dominated, inflate the raw error over all 3320 held-out rows to 0.198, still an improvement from 0.225 before calibration. Binned by predicted retention, the well-populated bins below 0.9 are conservative—measured retention exceeds the prediction—while calibrated predictions saturating near 1.0 overshoot slightly.

Meeting the Tolerance. Table 11 reports selector behavior at the six tolerances of fig. 7, with compute relative to always running the largest subnet. Both the mean per-scene retention of the selected subnets and the dataset-level pooled retention track the requested α closely, the latter meeting the tolerance at 32 of the 34 operating points of the sweep behind fig. 6 (worst shortfall 0.008 at $\alpha = 0.99$), while average compute stays 8.0–85.7% below the largest subnet. Because most scenes carry a single validation image, per-scene retention is measured under heavy label noise, so algorithm 1 is best read as selecting the cheapest subnet whose *predicted* retention reaches α: the tolerance is honored in expectation and in the dataset-level aggregate rather than as a per-scene certificate. Turning it into a per-context high-probability guarantee—predicting a lower retention quantile and selecting against that instead of the mean—is a natural extension that our formulation admits and we leave to future work.

α	scene ret.	pooled ret.	FLOPs (%)
0.6	0.842	0.749	−85.6
0.7	0.840	0.763	−84.0
0.8	0.888	0.818	−70.8
0.9	0.931	0.939	−29.3
0.95	0.947	0.968	−14.4
0.98	0.970	0.982	−8.8

Table 11: Selector behavior on the 664 held-out ADE20K scenes at the six tolerances of fig. 7. Scene ret.: mean per-scene retention of the selected subnets; pooled ret.: dataset-level mIoU ratio over all images; FLOPs: average compute relative to always running the largest subnet (negative = savings).

Oracle Comparison. We also compare against an oracle that reads each scene's measured retention and picks the cheapest qualifying subnet, incurring no violations by construction. At the strict tolerances most relevant to the always-on setting ($\alpha \geq 0.94$), the learned selector stays within 7.5% of this oracle's average compute. The margin widens at looser tolerances, where the oracle most exploits single-image label noise. Because the oracle reads the same few validation images that define the labels, it marks a lower bound on achievable compute rather than a reachable target.

H Baseline: Rule-Based Model Selection

We include a simple rule-based baseline only for comparison, which is not used in the final design of our AdaDINO. The baseline assigns each task to a subnet using a fixed lookup rule based solely on the number of class names. Tasks are ranked by class count and mapped to progressively larger subnets, while a single global parameter controls the overall allocation and traces an accuracy-efficiency frontier. Since it uses neither text embeddings nor a learned predictor, the method is inexpensive and easy to implement. However, this coarse heuristic ignores the identities and relationships of the classes, which is reflected as poorer performance in fig. 6, motivating the more fine-grained learned selector used in AdaDINO.

I Accuracy-Efficiency Trade-offs with Additional Metrics

End-to-End Comparison. In the main paper, we present the accuracy-efficiency trade-off (fig. 3) using FLOPs. Here, we additionally report results based on average execution latency and energy consumption, respectively in Fig. 11 and Fig. 12. The compared baselines are identical to those in fig. 3 (Ghiasi et al. 2022; Jose et al. 2025; Radford et al. 2021; Fang et al. 2024; Ilharco et al. 2021; Xing et al. 2023; Xu et al. 2024, 2023a; Zhai et al. 2023). They demonstrate that AdaDINO achieves significant gains in the accuracy-efficiency trade-off. Since these metrics exhibit the same trends and lead to the same main conclusions, we include them only in the Supplementary Material.

Ablation Across Additional Metrics. In the main paper, we present the ablation of AdaDINO's accuracy-efficiency gains

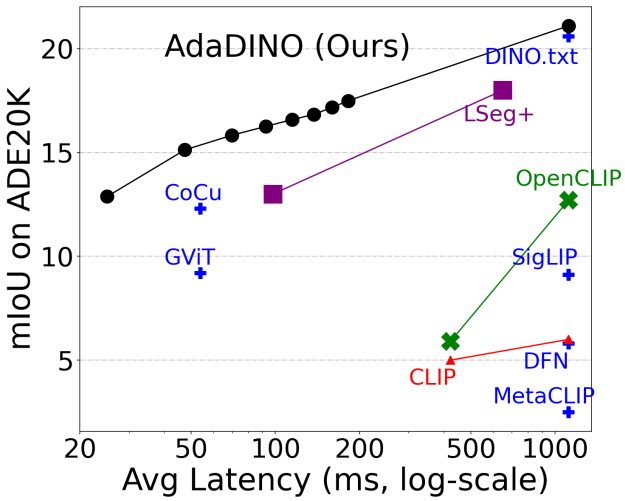

Figure 11: End-to-end trade-off between mIoU on open-vocabulary ADE20K segmentation (Zhou et al. 2019) and average execution latency.

(fig. 6) using FLOPs. Here, we additionally report the ablation based on average execution latency, energy consumption, and number of model parameters, respectively in Fig. 13, Fig. 14, and Fig. 15. The compared settings are identical to those in fig. 6, including the rule-based selection baseline described in §H. They demonstrate that each component of AdaDINO provides a clear benefit, with the learned selector consistently yielding the best trade-off. Since these metrics exhibit the same trends and lead to the same main conclusions, we include them only in the Supplementary Material.

J Extended Related Work

This section provides an extended discussion of the related work summarized in §2.

J.1 Always-on Contextual AI on Edge Devices

Recent advances in smartphones (Georg et al. 2025), wearable devices (Abbaspourazad et al. 2024), augmented and virtual reality (Abrash 2021), robotics (Goswami et al. 2025), and autonomous driving (Lin et al. 2025) have accelerated the push toward enabling AI on edge devices. The emergence of wearable edge platforms is motivating *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 and segmentation, due to recent progress in efficient models on mobile hardware (Mehta and Rastegari 2022; Liu et al. 2022). These visual representations are aligned with language on-device to support seamless user interaction (Li et al. 2023; Zhang et al. 2022), 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 (Alayrac et al. 2022; Driess et al. 2023; Touvron et al. 2023). This distributed edge-cloud design reflects limitations of continuous cloud

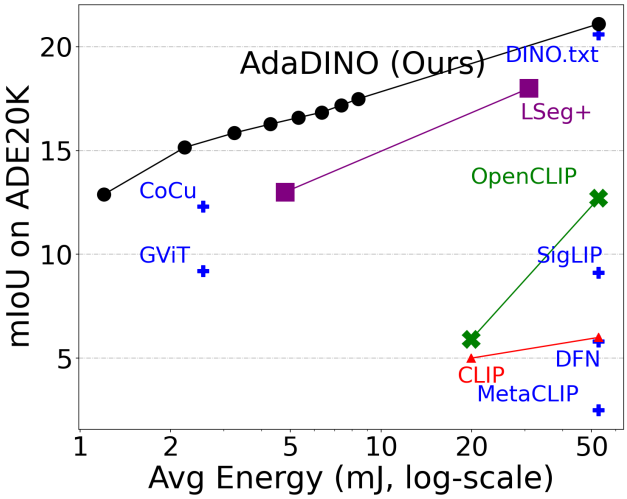

Figure 12: End-to-end trade-off between mIoU on open-vocabulary ADE20K segmentation (Zhou et al. 2019) and average energy consumption.

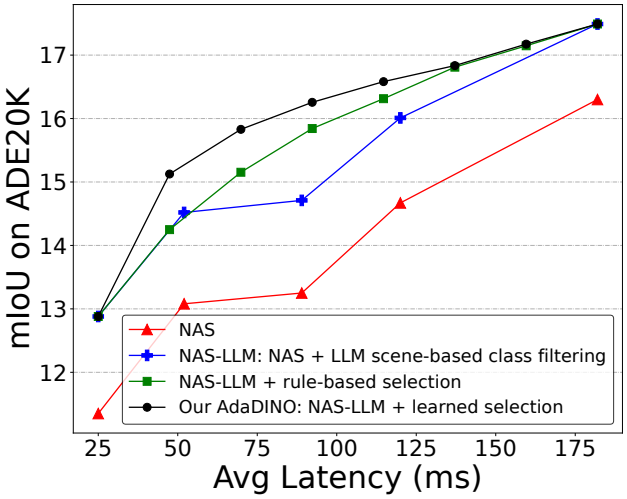

Figure 13: Ablation of AdaDINO's accuracy-efficiency gains, measured in average execution latency.

streaming and aligns with prior work showing that persistent offloading is impractical for wearable devices due to energy constraints (Lane et al. 2016; Li, Zhou, and Chen 2018; Seneviratne et al. 2017). Instead, sparse visual summaries and infrequent user queries are transmitted to the cloud, enabling the construction of a personal timeline by memory-augmented multimodal systems (Li, Zhang, and Ma 2025; Fan et al. 2025). 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.

Scope of This Work. Bringing an always-on contextual-AI wearable to market is an undertaking at the scale of entire industry divisions, engaging hundreds to thousands of engineers and researchers. Production efforts such as Meta Ray-Ban (Meta Platforms 2023) span custom silicon and accelerator design, thermal and battery engineering, optics and display, capture and sensing stacks, interaction design, and privacy infrastructure. Each of these is a hard engineering problem in its own right, and the open questions there—thermal envelopes, battery capacity, wearability—belong to that effort, not to any single paper. Within this stack, the model-execution layer is a self-contained research problem, and an important one: the always-on vision workload dominates the device's sustained compute, so its efficiency directly bounds battery life and responsiveness during contextual AI usage (§3.3). The layer also has well-defined inputs (the scene context and candidate class set), a well-defined output (the execution scheme of the edge model), and a measurable objective (the accuracy–efficiency frontier of §5). This is a scope a single research paper can treat rigorously, and precedent agrees: efficient backbones and split computing each sustain research lines of their own (§J.4). Concretely, given the edge–cloud division and the episodic context refresh that deployed systems adopt, we make the on-device vision model's cost track the current task, through LLM-based vocabulary refinement and learned subnet selection. The surrounding layers—context-refresh triggering, sensing policy, and user interaction—are supplied by the deployment and compose directly with our mechanisms. The minutes-or-longer persistence of egocentric contexts (Grauman et al. 2022, 2024) makes the low-frequency refresh cadence we assume a conservative choice.

Projected Deployment Benefits. The device- and cloud-level gains this layer enables motivate the always-on setting; we report them as first-order projections under the low-frequency refresh cadence above, not as end-to-end system measurements. Contextual-AI glasses that forward every query to the cloud (Meta Platforms 2023) report query latencies around 5 s and roughly two hours of battery life. Running perception on the edge subnet and calling the cloud MM-LLM only when the scene changes shifts the sustained cost onto the efficient on-device model: combining our measured accelerator energy (table 2) with that cloud-only behavior projects at least $5\times$ longer battery life during contextual-AI usage, counting accelerator energy alone and excluding the camera, display, communication, and CPU. Because the cloud is queried per context rather than per frame, cloud cost also becomes frame-rate independent—on the order of $1.6 per hour under public API pricing against $14 per hour-fps for cloud-only streaming, a 10–100$\times$ reduction (OpenAI 2026; Google 2026). These figures are motivation for the deployment target rather than validated results, and the refresh cadence they assume is a deployment policy that our benchmark does not itself measure.

J.2 Vision Foundation Models

Vision foundation models have become prevalent in computer vision, offering unified representations that generalize across a wide spectrum of tasks and domains. These models are typically trained at scale using self-supervised learning (SSL), which exploit the intrinsic structure of visual data to learn from large, unannotated image corpora.

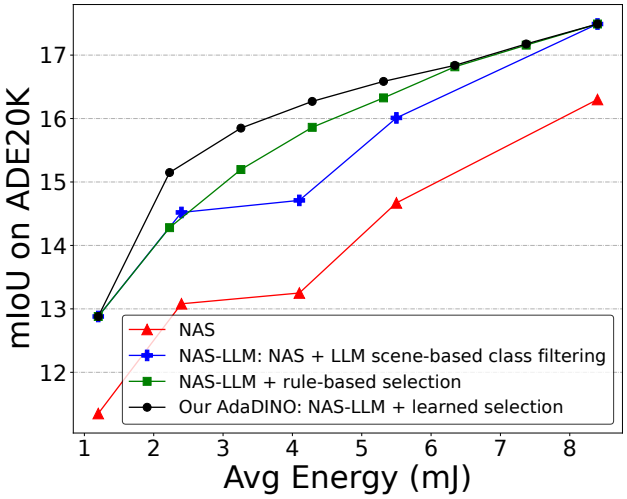

Figure 14: Ablation of AdaDINO's accuracy-efficiency gains, measured in average energy consumption.

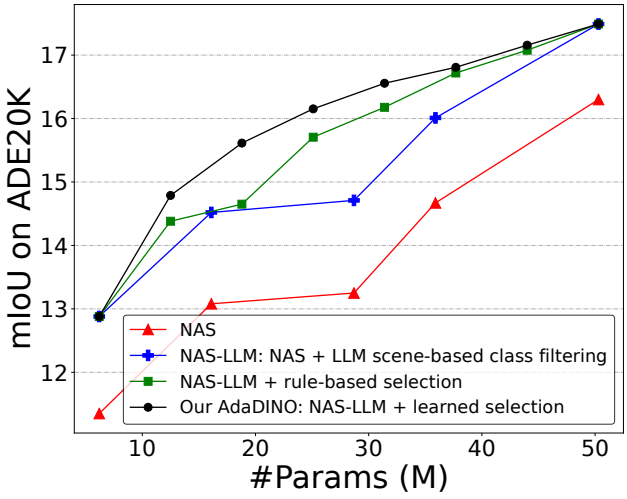

Figure 15: Ablation of AdaDINO's accuracy-efficiency gains, measured in number of model parameters.

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 (Oquab et al. 2024), MAE (He et al. 2022), and iBOT (Zhou et al. 2022a) demonstrate the effectiveness of SSL in learning transferable features that rival or surpass those obtained via supervised pretraining methods like CLIP (Radford et al. 2021) or PaLI (Chen et al. 2023b). Such models have shown strong transferability, enabling a single frozen encoder to perform competitively across diverse downstream tasks without task-specific fine-tuning. Nonetheless, large-scale vision foundation models often incur substantial computational, memory, and energy costs. Addressing these limitations, we present *adaptive vision foundation models* that achieve state-of-the-art VFM performance on edge devices.

J.3 Contrastive Learning

Contrastive learning methods such as CLIP enable zero-shot and open-vocabulary capabilities in VFMs by aligning visual and textual representations. A standard CLIP architecture consists of a Vision Encoder and a Text Encoder, both computationally and memory intensive, making deployment on resource-constrained edge devices impractical. Pruning or shrinking them often causes significant accuracy loss (Shrivastava et al. 2023; Wu et al. 2023b).

A key property of CLIP for edge deployment is the ***differing frequencies*** of Vision Encoder and Text Encoder calls. In always-on contextual AI (fig. 1), prompts or scenes change infrequently, while perception runs continuously:

- **Vision Encoder (high-frequency):** Processes every incoming frame and must meet strict low-latency requirements (typically 200–1000 ms) (Stauffert, Niebling, and Latoschik 2016; Kim et al. 2023).
- **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.

Efficient Vision Foundation Models. Prior work lowers VFM cost *statically*, distilling or shrinking CLIP-style encoders into compact, fixed backbones (Wu et al. 2023b; Vasu et al. 2024). AdaDINO instead keeps a family of subnets and picks among them at runtime. Several other efficiency directions are also related to AdaDINO but adapt along a different axis. We review them in §J.4 and clarify how their scope differs from ours.

J.4 Relation to Adjacent Directions

Several established research directions are close to AdaDINO in spirit but differ in scope. We summarize the most relevant ones below and, for each, explain why it is either outside the problem AdaDINO addresses or orthogonal to our design and composable with it. In every case the two are complementary rather than competing, so these methods are not substitutes that our evaluation would need to compare against.

Adaptive and Dynamic Inference. A large body of work makes a single network input-adaptive, spending more or less computation on each input according to its estimated difficulty (Han et al. 2022). Typical strategies include token pruning, merging, and sparsification in vision transformers (Liang et al. 2022; Bolya et al. 2023; Rao et al. 2021; Wu et al. 2025; Wang et al. 2021; Chen et al. 2023a), token or block halting (Yin et al. 2022; Meng et al. 2022), input-dependent depth and layer skipping (Wang et al. 2018; Wu et al. 2018), early-exit and anytime prediction (Huang et al. 2018; Teerapittayanon, McDanel, and Kung 2016), runtime width selection within one supernet (Yu et al. 2019; Li et al. 2021b), resolution-adaptive computation (Yang et al. 2020; Figurnov

et al. 2017; Wang et al. 2020), sparsely-gated mixtures of experts (Shazeer et al. 2017; Fedus, Zoph, and Shazeer 2022; Riquelme et al. 2021), adaptive frame selection for video (Wu et al. 2019), and budget-aware or token-reduced inference for multimodal LLMs (Xu et al. 2025; Chen et al. 2024; Zhang et al. 2025; Xing et al. 2025; Hu et al. 2024). These methods read a per-input signal, such as prediction confidence or token saliency, and adjust how much of one fixed, closed-vocabulary model to run. AdaDINO instead chooses among distinct open-vocabulary subnets from the scene and the user-defined class vocabulary, which describe the task rather than an individual frame. The two are independent, and any of these per-input mechanisms can still run inside the subnet AdaDINO activates, so they complement rather than replace our adaptive selection.

Model Compression and Quantization. Pruning, quantization, and knowledge distillation (Han, Mao, and Dally 2016; Jacob et al. 2018; Hinton, Vinyals, and Dean 2015; Touvron et al. 2021; Yuan et al. 2022; Yang et al. 2024) shrink a network to a single cheaper operating point. Our subnets are themselves distilled from a large teacher, and each one can be pruned or quantized on its own. Compression produces one smaller model, whereas AdaDINO maintains several and decides which to run for a given task.

Collaborative Edge-Cloud Inference. Split computing divides a single network between the device and the cloud, or exits early on-device, to balance latency, energy, and accuracy (Kang et al. 2017; Laskaridis et al. 2020; Matsubara, Levorato, and Restuccia 2022; Banitalebi-Dehkordi et al. 2021; Zeng et al. 2020). AdaDINO does not split the vision model: the selected subnet runs entirely on the edge, and only the infrequent decision of scene understanding and vocabulary filtering is sent to the cloud LLM. The design question is thus which computation to offload and how often, not where to cut a fixed network.

Cascades and Input-Dependent Routing. Cascades and routers escalate or dispatch each input among several separate models based on confidence or cost, in both vision and language settings (Bolukbasi et al. 2017; Jitkrittum et al. 2023; Chen, Zaharia, and Zou 2023; Ong et al. 2025; Ding et al. 2024). AdaDINO selects among subnets of one weight-sharing supernet rather than a pool of independent models, and routes on scene and task semantics for open-vocabulary perception rather than on the confidence of a fixed label set. It also decides once per scene, so no input is re-run through a larger model.

LLM-Driven Vision Systems. Several systems use an LLM as a controller that plans and calls vision tools for each query (Shen et al. 2023; Surís, Menon, and Vondrick 2023; Wu et al. 2023a; Gupta and Kembhavi 2023; Lu et al. 2023). In AdaDINO the LLM has a narrower, low-frequency role: it summarizes the scene and filters the class vocabulary for the on-device model, and stays out of the per-frame perception loop. The contribution here is the edge vision model and how it is chosen, not an agent that reasons about every input.

Open-Vocabulary Segmentation. Specialized open-vocabulary segmentation and detection models add dedicated decoders or adapters on top of large backbones to push accuracy (Li et al. 2022; Ghiasi et al. 2022; Xu et al. 2023c; Cho et al. 2024; Xu et al. 2023b; Liang et al. 2023; Yu et al. 2023; Gu et al. 2022). They aim for segmentation quality at server-scale compute, whereas AdaDINO uses open-vocabulary segmentation as one way to probe an edge backbone under a tight compute budget, following the protocol of DINO.txt (Jose et al. 2025). We report against that open-vocabulary VFM, which shares our setting. The heavier, task-specialized models sit in a different accuracy-compute regime.

Prompt and Vocabulary Design for Open-Vocabulary Models. A related line improves the text side of open-vocabulary recognition, for instance by enriching class names with LLM-generated descriptions (Pratt et al. 2023; Menon and Vondrick 2023), ensembling or reweighting prompts (Roth et al. 2023), or learning prompts and adapters (Mirza et al. 2023; Khattak et al. 2025; Zhou et al. 2022c,b; Wang et al. 2024). These act on a fixed target vocabulary and are computed offline, whereas the LLM in AdaDINO selects and filters the active vocabulary at runtime from the inferred scene. The two are complementary, and several such methods already appear as baselines in our accuracy-efficiency comparison (Xing et al. 2023; Xu et al. 2023a; Roth et al. 2023; Mirza et al. 2023; Khattak et al. 2025).

Summary. Across these directions, each is either outside the problem AdaDINO targets or a complementary optimization that composes with our subnets, so none is a substitute for the task-level, open-vocabulary model selection we introduce. The comparisons that isolate this axis—adaptive selection versus fixed backbones, and our learned LLM-guided selector versus rule-based and non-LLM alternatives—are already reported in §H, §F.5, and fig. 6. Our evaluation therefore covers the comparisons relevant to our claims.

J.5 Neural Architecture Search

Neural Architecture Search (NAS) provides a principled framework for adaptive-capacity models, suited to resource-constrained edge deployment. Its key merit is support for ***runtime subnet selection*** over sub-architectures of varying computational cost. At inference, it selects a subnet by task complexity or resources, enabling flexible accuracy-efficiency trade-offs. While early NAS methods used evolutionary (Real et al. 2017) or RL-based strategies (Zoph and Le 2017), later weight-sharing supernets (Cai et al. 2020; Chu, Zhang, and Xu 2021; Yu et al. 2020) made NAS practical for adaptiveness. Slimmable networks select widths at run time under resource budgets (Yu et al. 2019; Li et al. 2021b), an early form of multi-capacity runtime switching. These supernets usually pick a subnet to meet a hardware or latency budget. In AdaDINO, the choice is instead driven by task semantics: the LLM manager reads the scene and class vocabulary, and its learned selector picks the subnet. We use NAS to develop text-aligned, adaptive VFMs for always-on edge scenarios.