ARIA: Optimizing Vision Foundation Model Inference on Heterogeneous Mobile Processors for Augmented Reality

Chanyoung Jung*
Yonsei University
Seoul, Republic of Korea
cy.jung@yonsei.ac.kr

Jeho Lee*
Yonsei University
Seoul, Republic of Korea
jeholee@yonsei.ac.kr

Gunjoong Kim
Yonsei University
Seoul, Republic of Korea
gunjoong.kim@yonsei.ac.kr

Jiwon Kim
Uppsala University
Uppsala, Sweden
jiwon.kim@angstrom.uu.se

Seonghoon Park
Yonsei University
Seoul, Republic of Korea
park.s@yonsei.ac.kr

Hojung Cha†
Yonsei University
Seoul, Republic of Korea
hjcha@yonsei.ac.kr

Abstract

Mobile Augmented Reality (AR) applications demand high-quality, real-time visual prediction, including pixel-level depth and semantics, to enable immersive and context-aware user experiences. Recently, Vision Foundation Models (VFMs) offer strong generalization capabilities on diverse and unseen data, supporting scalable mobile AR experiences. However, deploying VFMs on mobile devices is challenging due to computational limitations, particularly in maintaining both prediction accuracy and real-time performance. In this paper, we present ARIA, the first system that enables on-device inference acceleration of a VFM. ARIA employs the heterogeneity of mobile processors through a parallel and selective inference scheme: full-frame prediction is periodically offloaded to a processor with high parallelism capability like GPU, while low-latency updates on dynamic regions are conducted via a specialized accelerator like NPU. Implemented and evaluated using mobile devices, ARIA achieved significant improvements in accuracy and deadline success rate on diverse real-world mobile AR scenarios.

CCS Concepts

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Computing methodologies** → **Computer vision**.

Keywords

Mobile Augmented Reality, Vision Foundation Model, On-device AI, Heterogeneous Mobile Processors

ACM Reference Format:
Chanyoung Jung, Jeho Lee, Gunjoong Kim, Jiwon Kim, Seonghoon Park, and Hojung Cha. 2025. ARIA: Optimizing Vision Foundation Model Inference on Heterogeneous Mobile Processors for Augmented Reality. In *The 23rd Annual International Conference on Mobile Systems, Applications and Services (MobiSys '25), June 23–27, 2025, Anaheim, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3711875.3729161

*Co-primary authors with equal contribution.
†Corresponding author.

MobiSys '25, June 23–27, 2025, Anaheim, CA, USA

ACM ISBN 979-8-4007-1453-5/2025/06
https://doi.org/10.1145/3711875.3729161

1 Introduction

Mobile Augmented Reality (AR) applications involve fundamental vision tasks to extract pixel-level information, such as depth and semantics, from camera frames. For example, depth information is essential for the realistic placement of virtual objects, reflecting physical relationships (e.g., occlusion) with real-world objects. Additionally, semantic information facilitates context-aware AR experiences, e.g., providing auxiliary visual cues highlighting safety-critical areas like crosswalks to assist visually impaired individuals. Recent AR frameworks [15, 33] have employed Deep Neural Network (DNN) models to retrieve such information from camera frames accurately. For an immersive and responsive user experience, these frameworks have also supported real-time processing, e.g., 30 frames per second (FPS), of models on mobile devices.

In AR applications, the environment in which users interact is continuously changing due to the movements of the users or surrounding objects. However, traditional DNN models in existing frameworks are typically designed for limited use cases. For example, ARCore's Scene Semantics API only supports outdoor scenarios [18]. Moreover, these models must be robust against domain shifts, such as changes in lighting conditions, weather, and spatial complexity. In general, these models are trained on specific datasets with limited environmental contexts, making them susceptible to accuracy degradation when exposed to domain shifts during mobile AR sessions. One simple approach to addressing the issue is to deploy multiple models trained on different datasets, which is impossible for mobile devices with limited memory capacity. Recent studies have focused on employing domain adaptation techniques [10, 43], which update the model parameters to adapt to environmental changes. However, these methods incur significant latency overhead on mobile devices, reducing their computational budget to perform the main inference tasks.

A promising alternative is to utilize models trained on large-scale datasets encompassing diverse environmental conditions. Vision Transformers (ViTs) [9] significantly outperformed traditional vision models, such as Convolutional Neural Networks (CNNs), in handling large-scale datasets through their ability to capture complex spatial relationships in images. ViTs trained on large-scale datasets have led to the emergence of Vision Foundation Models

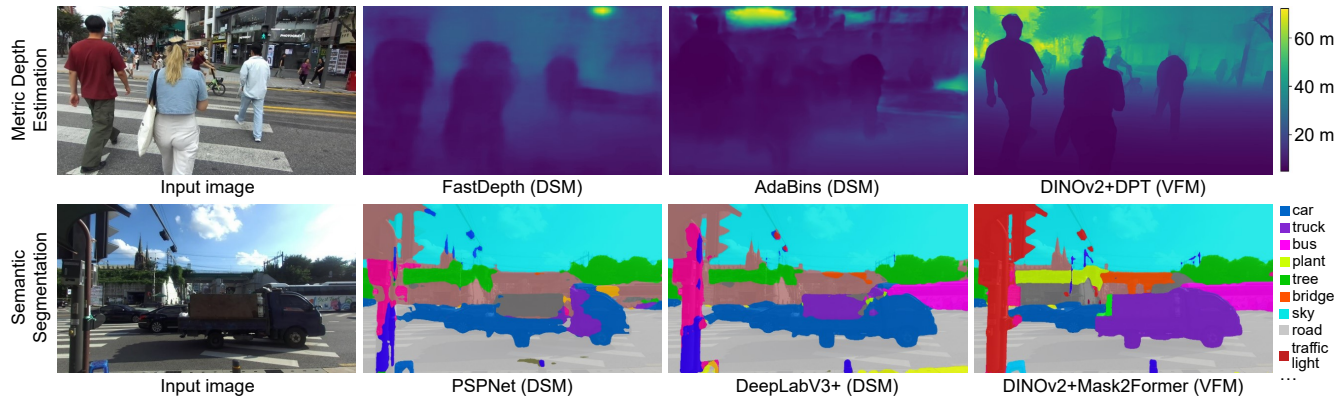


Figure 1: Predictions from domain-specific models (DSMs) and vision foundation models (VFMs) on unseen MARV dataset.

(VFMs). VFMs such as Segment Anything [24] and Depth Anything [58], typically consisting of hundreds of millions of parameters, demonstrate strong generalization (zero-shot) capabilities on data beyond their training sets (see Figure 1). This suggests that leveraging VFMs for mobile AR applications is a promising direction to enable scalable and immersive experiences.

Despite such opportunities, efforts to successfully deploy VFMs on mobile devices remain largely unexplored. Our preliminary observations indicate that maintaining high computational fidelity—such as input resolution and arithmetic precision—is crucial for preserving the generalization ability of VFMs on unseen data. However, due to the operational characteristics of ViTs, the larger data sizes resulting from high computational fidelity introduce prohibitive inference latency on mobile processors. Although mobile processors with high parallelism capabilities, such as GPUs, can alleviate the issue, meeting the stringent latency requirement remains challenging. Conversely, lowering computational fidelity enables significant inference acceleration on processors like NPUs optimized for small-sized data processing with specialized hardware. This achieves the strict latency requirement but at the expense of a noticeable accuracy drop. As a result, these single-processor baselines, high-fidelity operations on GPUs or low-fidelity operations on NPUs, fall short of satisfying both high-accuracy and low-latency requirements.

In this paper, we present ARIA, a system to enable resource-efficient on-device inference of VFMs for immersive and scalable mobile AR applications. The goal of ARIA is to maintain high-quality prediction of VFMs in real-time, e.g., 30 FPS, on mobile devices. The key idea of ARIA is to employ the unique capabilities of heterogeneous mobile processors in a parallel and selective manner. Specifically, high-resolution predictions on camera frames are periodically generated using GPU, suitable for processing large-scale inputs. Concurrently, fast, low-resolution operations on NPU are selectively applied to dynamic regions within the frames, where frequent updates are essential, e.g., moving objects. This approach effectively combines GPU's capacity for full-frame prediction with NPU's efficiency in providing local, up-to-date inferences, opening up new opportunities for high-quality, real-time VFM processing.

Despite the potential of utilizing heterogeneous processors, realizing the goal of achieving both high accuracy and responsiveness on mobile devices faces several challenges. First, accurately identifying and efficiently processing dynamic regions within camera frames is non-trivial, especially without relying on compute-intensive pixel-level analysis like optical flow or coarse-grained block-level matching [55]. Second, simply combining local updates from NPU with periodic full-frame predictions from GPU introduces spatial and temporal misalignment. The visual inconsistencies severely degrade user experiences, e.g., by disrupting virtual content rendering, where a precise understanding of a scene is necessary. Lastly, maintaining real-time performance is further complicated by changes in processor performance, device motion, and scene dynamicity.

For efficient identification and processing of dynamic regions, ARIA proposes to analyze the temporal differences in each image *patch* (e.g., 16×16 pixels). This enables fine-grained identification of dynamic regions while minimizing the inclusion of irrelevant static areas, effectively bridging the gap between pixel and block granularities. Since VFMs provide the distinctive visual features of patches, ARIA leverages this information as guidance, enabling accurate identification and tracking of dynamic regions. For the second challenge of addressing spatial and temporal mismatches in decoupled predictions, ARIA introduces feature alignment modules informed by the key dynamics in mobile AR scenarios: camera motion and object motion. Finally, ARIA employs a dynamic inference scheme that adaptively adjusts the executions of GPU and NPU in response to changing runtime factors, achieving the accuracy and latency requirements of AR applications.

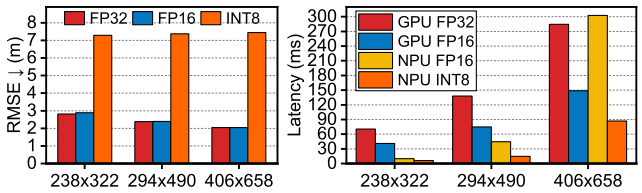
The key contributions of ARIA are as follows:

- To the best of our knowledge, ARIA is the first system designed to accelerate high-quality VFM inference on mobile devices, enabling scalable, immersive AR experiences.
- Along with a comprehensive study of VFM inference workloads on heterogeneous mobile processors, we propose a novel parallel and selective VFM inference scheme with solutions that address unique challenges.
- ARIA is fully implemented as an end-to-end system using commodity mobile devices and extensively evaluated on a self-collected dataset for mobile visual AR tasks.

Table 1: Prediction performance on unseen datasets.

Metric depth estimation on our MARV dataset			
Type	**Model (Training Dataset)**	**RMSE ↓**	**Params.**
DSM	FastDepth [63] (NYUv2)	6.61	4.0M
	AdaBins [2] (KITTI)	5.53	78.3M
VFM*	DINOv2+DPT	2.04	24.8M
Semantic segmentation on Cityscapes dataset			
Type	**Model (Training Dataset)**	**mIoU ↑**	**Params.**
DSM	PSPNet [65] (ADE20K)	0.36	13.7M
	DeepLabV3+ [4] (ADE20K)	0.45	43.7M
VFM*	DINOv2+Mask2Former	0.56	57.2M

***VFMs are pre-trained on a large corpus of existing datasets, and then are fine-tuned to Hypersim [42] and ADE20K [66] datasets to support MDE and SS, respectively.**



(a) Accuracy (H×W) (b) Latency (H×W)

Figure 2: Accuracy and latency comparison across different computational fidelities and mobile processors on Samsung Galaxy S24.

(a) FP32 (b) FP16 (c) INT8

Figure 3: Visual comparison of metric depth estimation.

2 Background and Motivation

2.1 Recent Advances in Vision Models

With the advent of Transformers in general-purpose language models, their extension to vision tasks has become pervasive. Vision Transformers (ViTs) divide images into small patches (e.g., 16×16 pixels), similar to words in language Transformers, and use the attention mechanism to capture long-range dependencies between these patches [9]. ViTs offer more effective global context understanding compared to traditional CNNs, enhancing prediction accuracy across a range of vision tasks.

Vision Foundation Models (VFMs), developed alongside ViT training techniques using large-scale datasets, have emerged as a new mainstream in foundation models (FMs) for general-purpose visual understanding tasks. A line of VFMs, such as CLIP [40], supports tasks that require text-image understanding, like image captioning, which is known as multimodal FMs. Another line of VFMs especially focuses on visual modality, providing high-quality, dense visual predictions. The DINO [3, 36] series, pioneers of these models, utilizes self-supervised learning to minimize an extensive labeling effort for large-scale data. These models offer rich visual features that are highly versatile for various downstream vision tasks. Recent VFMs use these models as encoders and train task-specific decoders to support various fine-grained vision tasks. For instance, Depth Anything [52] was trained on 62M images, supporting general-purpose depth estimation. Similarly, Meta's Segment Anything [24] shows strong image segmentation performance for any type of object.

2.2 Advantages of VFMs in Mobile AR

In real-world mobile AR scenarios, users typically navigate environments with varying lighting conditions and surrounding objects. However, traditional models trained on domain-specific datasets often exhibit poor generalization abilities in new scenarios that were not included during training. Unlike these domain-specific models (DSMs), VFMs effectively handle environmental shifts by providing superior zero-shot predictions on unseen data, without requiring additional model fine-tuning.

To compare the generalization capabilities of VFMs and DSMs, we conducted experiments on two key vision tasks in mobile AR: metric depth estimation (MDE) and semantic segmentation (SS). To quantify the zero-shot prediction accuracy, we used a self-collected MARV dataset (see Section 6.1) and the public Cityscapes dataset [7] for MDE and SS, respectively. VFMs used DINOv2 [36] as a visual feature encoder, with DPT [41] and Mask2Former [6] as decoders to produce final results for MDE and SS, respectively. For comparison, we selected popular DSMs for each vision task, varying in their number of parameters and training datasets.

Table 1 shows the zero-shot prediction performance of the models on unseen datasets. The results indicate that VFMs achieved an average of 2.1× and up to 3.2× higher accuracy compared to DSMs. Moreover, despite having 3.1× fewer parameters than AdaBins, DINOv2+DPT outperformed by a large margin of 2.7× in accuracy, highlighting the importance of training with large-scale datasets. Example results shown in Figure 1 demonstrate that VFMs produce visual predictions with the highest quality. These predictions enhance downstream AR functionalities, such as placements of virtual overlays or user interactions, where fine-grained and accurate predictions of physical objects are crucial.

2.3 VFM Inference on Mobile Processors

To enable truly immersive mobile AR applications, the VFM inference task should be performed in real-time, e.g., 30 FPS. However, VFMs demand high computational resources on mobile devices, due to their large number of parameters (see Table 1) and compute-intensive ViT operations. To analyze the complexity of on-device VFM inference workloads, we conducted preliminary experiments using modern mobile processors such as GPU and NPU widely adopted for NN inference acceleration. For our experiments, we used a Samsung Galaxy S24 with Adreno 750 GPU and Hexagon NPU.

In fine-grained visual prediction tasks, computational fidelity, such as input resolution and arithmetic precision, is a critical factor that influences model accuracy and inference speed on mobile processors. Figure 2 shows the impact of changes in computational fidelity on (i) MDE accuracy of DINOv2+DPT on MARV

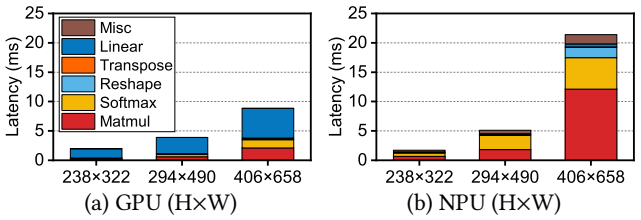


Figure 4: Latency breakdown of a FP16 ViT layer on Samsung Galaxy S24.

Table 2: Memory access breakdown (MB) for a NPU FP16 ViT layer on Samsung Galaxy S24.

H×W	Matmul	Softmax	Reshape	Transpose	Linear	Misc
238×322	31.99	4.15	2.72	12.06	26.66	28.60
294×490	73.82	13.00	3.39	19.22	64.67	51.84
406×658	1664.55	395.80	35.36	37.37	225.83	202.52

dataset and (ii) latency on different mobile processors. NPUs are highly optimized for accelerating integer (INT) operations, with NPU INT8 achieving an average 5.1× speedup compared to other settings. However, ViTs include operations sensitive to INT data format, such as attention and layer normalization, leading to significant accuracy degradation compared to floating-point (FP) operations [32, 54]. Moreover, the accuracy drops become more severe at higher resolutions as the quantization errors are amplified due to an increased number of such operations. For example, as shown in Figure 2a, NPU INT8 resulted in a 3.6× accuracy drop at a resolution of 406×658, where Figure 3 shows the visual comparison.

Fortunately, modern mobile NPU architectures have increasingly supported FP operations (e.g., FP16 on Qualcomm Hexagon NPU) to meet the growing demands [46]. Despite being hardware-optimized for accelerating INT operations, they can achieve real-time FP16 computation on low-resolution inputs. For example, as shown in Figure 2b, NPU FP16 incurs a latency of 10.0 ms at a low resolution of 238×322. However, the latency of NPU FP16 increased sharply with higher input resolution—a limitation that GPUs can alleviate due to their better parallel processing capabilities for handling large-scale data. Specifically, at a high resolution of 406×658, GPU FP16 achieves 2.0× lower latency compared to NPU FP16.

Figure 4 shows the latency breakdowns of a single FP16 ViT layer on both GPU and NPU, further demonstrating the hardware differences. Notably, the increasing latency of key operations in the attention mechanism—MatMul and Softmax—is much more significant on NPU compared to GPU. This is due to the complexity of inter-patch computations in the attention mechanism, scaling quadratically as $O(N^2)$, where N denotes the number of input patches in ViTs, combined with the limited parallelism and on-chip memory capacity of NPUs. Consequently, high-resolution attention operations become memory-bound on NPUs, leading to substantial latency overhead for memory read/write. Table 2 further quantifies this issue, showing the rapid increase in memory access at higher input resolutions. Specifically, Matmul and Softmax operations at 406×658 require 52.0× and 95.4× more memory access compared to 238×322, respectively.

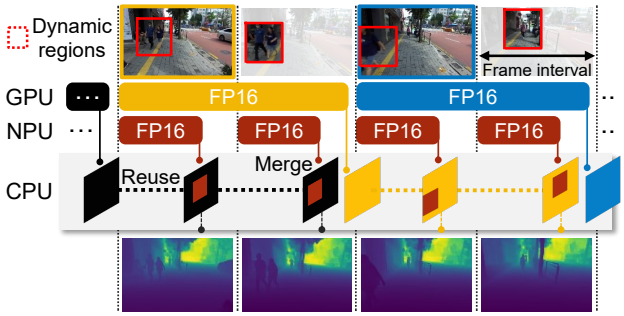


Figure 5: Parallel and selective execution using heterogeneous mobile processors.

2.4 Summary

We first observed that FP operations are essential to avoid severe accuracy loss in VFMs. We did not consider full-precision operations (i.e., FP32) as they offer minor accuracy improvements over FP16. While NPU FP16 enabled real-time VFM inference at low resolutions, it naturally suffered from accuracy degradation due to image resizing. High-resolution FP16 operations address this issue, and the increased latency of processing larger data is alleviated to some extent with the strong parallelism capabilities of GPU but still fails to meet the stringent latency requirements (e.g., 30 FPS). As a result, these single-processor baselines—low-resolution operations on NPU or high-resolution operations on GPU—cannot satisfy the demands for both accuracy and latency. This motivates us to design a system that effectively combines unique characteristics of NPU and GPU on floating-point operations to meet the requirements.

3 Opportunities and Challenges

3.1 Opportunities

Based on our observations, we derived a key idea to achieve real-time performance of a VFM without a significant accuracy loss, as depicted in Figure 5. A GPU periodically generates high-resolution visual predictions for the entire camera frames, which are reused in the next few frames. Concurrently, an NPU selectively processes dynamic regions (e.g., moving objects) within each frame, where frequent, up-to-date predictions are essential. Note that these local predictions are generated without the loss of details caused by image resizing, avoiding the accuracy drops observed in Figure 2a. Finally, the local updates from NPU are integrated into the latest full-frame prediction from GPU. This strategy effectively delivers high-quality VFM predictions in real-time, by leveraging the strengths of heterogeneous mobile processors: the GPU's capability to handle large-scale input data and the NPU's efficiency in small-data processing.

3.2 Challenges

Realizing the above strategy to enable real-time processing of VFMs on resource-constrained mobile devices is accompanied by the following three challenges:

Accurate and efficient processing of dynamic regions. For a reliable update of dynamic regions in the latest full-frame prediction,

Figure 6: Inconsistencies in decoupled predictions (NPU: dynamic regions highlighted by a box, GPU: entire frame).

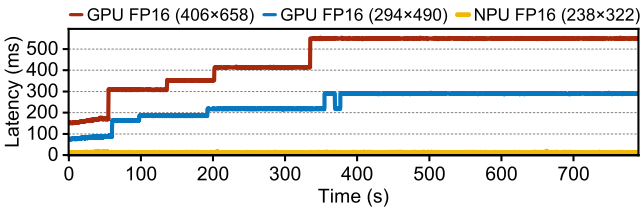


Figure 7: Runtime traces of inference latencies on Samsung Galaxy S24. The two GPU executions were profiled in separate experiments, while NPU was executed concurrently with both.

accurate identification of those regions is critical. However, employing DNN-based object detection and tracking or dense optical flow can result in significant latency on mobile devices. Furthermore, existing selective inference methods [49, 55] rely on rectangular blocks to identify the dynamic regions as shown in Figure 5. Such coarse-grained selection mechanisms often include irrelevant static areas, reducing efficiency in dynamic region processing.

Spatial and temporal inconsistencies across decoupled predictions. The local updates from NPU and the latest full-frame prediction from GPU must be seamlessly merged for subsequent AR functionalities. However, processing only a fraction of the frame risks underutilizing a ViT's ability for global understanding, potentially leading to spatial inconsistencies between local and full-frame predictions. For example, as shown in Figure 6, simply merging these decoupled predictions results in visual inconsistency. Additionally, since full-frame predictions on GPU are performed periodically, the merged results without considering scene changes caused by camera motion—common in mobile AR—lead to temporal mismatches. Therefore, a method to effectively address these spatial and temporal inconsistencies is essential.

Dynamic changes in processor performance, camera motion, and scene complexity. Achieving real-time execution of a VFM on mobile devices becomes even more challenging due to various runtime factors. First, continuously running the high-resolution VFM inference on mobile GPU often leads to performance degradation due to system-level decisions such as thermal throttling. As shown in Figure 7, the latency of GPU FP16 is increased up to 3.7× during runtime due to the thermal issue. This can be mitigated by reducing the processing resolution of GPU FP16 or even eliminated through low-resolution execution on NPU. Second, continuous scene changes due to camera or object motion must be carefully considered to meet the accuracy and latency requirements of AR vision tasks. For example, rapid scene changes caused by abrupt

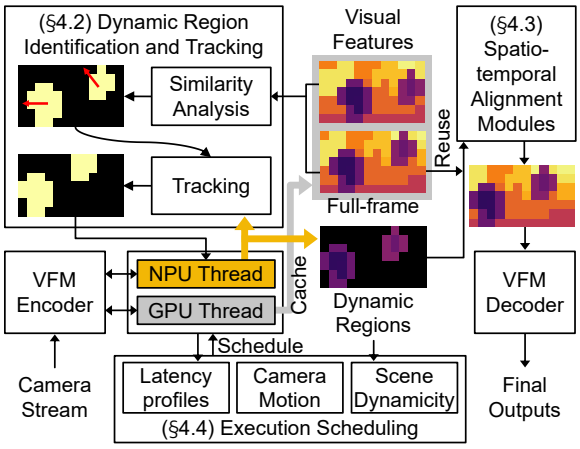


Figure 8: Overview of ARIA.

camera motion make periodic full-frame predictions from GPU severely outdated, where large portions of them are not reusable.

4 Design of ARIA

4.1 System Overview

Motivated by our key insights, we present ARIA, a system that supports real-time, on-device VFM inference for mobile devices. The design goal of ARIA is to leverage heterogeneous mobile processors to achieve both inference speedup and accuracy improvement of a VFM for mobile AR applications. Additionally, ARIA aims to be generic, supporting various VFMs for critical AR vision tasks.

Figure 8 shows the system architecture of ARIA, which follows the baseline workflow in Figure 5: parallel execution of periodic, full-frame prediction on GPU and real-time update for dynamic regions on NPU. ARIA utilizes patch-based representations to process the dynamic regions in a fine-grained manner and seamlessly merges the results with the latest full-frame prediction. To efficiently and accurately identify the dynamic regions, ARIA employs temporal similarity analysis based on visual features of patches, which are the intermediate results of VFMs (Section 4.2.1). The identified dynamic regions are propagated across frames by our tracking method until the next full-frame prediction is available from GPU (Section 4.2.2). NPU extracts visual features for the dynamic regions in each frame, which are then merged into the latest full-frame feature map through spatio-temporal alignment modules (Section 4.3). The aligned feature map is then processed by a task-specific decoder of VFM, generating the final prediction. During runtime, ARIA dynamically adapts to scene changes, abrupt camera motion, and thermal throttling by scheduling the execution of GPU and NPU to meet the demands of mobile AR applications (Section 4.4).

4.2 Transformer-guided Dynamic Region Identification and Tracking

Leveraging the NPU's efficiency in small-data processing to selectively update the dynamic regions in the latest full-frame prediction helps maintain both high accuracy and responsiveness. To enable

precise and efficient processing of dynamic regions, a fine-grained identification method is required that minimizes the inclusion of irrelevant static regions. The key idea of ARIA is to determine dynamic regions by analyzing temporal differences at the granularity of *patches*, which are the basic processing units of ViTs. This patch-based approach is lightweight due to its use of low-dimensional representations compared to pixels, e.g., 29×47 patches in a 406×658 image, and provides a natural way to leverage the patches' visual features generated by VFMs.

4.2.1 Dynamic region identification. To identify dynamic patches, ARIA utilizes the patch embedding features generated from the VFM encoder. As input patches pass through a sequence of ViT layers in the encoder, such as attention, their embedding features inherently capture contextual information (e.g., how patches are related to each other) along with the visual and spatial characteristics. Leveraging these distinctive and rich features makes it easy to detect temporal differences in patches, improving identification robustness.

Figure 9 shows dynamic patch identification between the two consecutive frames processed by GPU, i.e., frames $t - \Delta_{\text{GPU}}$ and t. To identify dynamic patches $P_{\text{Dyn}}^{(t)}$, ARIA uses cosine similarity to determine the patch locations in the target frame t that exhibit significant temporal differences compared to the previous frame $t - \Delta_{\text{GPU}}$:

$$P_{\text{Dyn}}^{(t)} = \left\{ i \mid sim\left(f_i^{(t-\Delta_{\text{GPU}})\rightarrow(t)}, f_i^{(t)}\right) < \tau \right\}, \tag{1}$$

where $f_i^{(k)}$ denotes the embedding features of patch location $i \in P^{(k)}$ at frame k. Threshold τ is for ignoring dynamic patches with minor changes, thereby adjusting the NPU's workload to process these patches. Temporal differences in each patch location can occur not only from object movement but also from the camera's motion $\theta^{(t-\Delta_{\text{GPU}})\rightarrow(t)}$. To compensate for camera motion and thereby only consider object motion, ARIA warps the patches in $f^{(t-\Delta_{\text{GPU}})}$ into the target frame t based on the estimated patch flow (Section 4.3.1).

As seen in the embedding features visualized in Figure 9, the foreground objects and the backgrounds can be easily distinguished at the encoder phase of VFMs [3], making significant temporal differences in object boundaries. However, updating only these boundary patches using NPU may fail to fully capture the changes in object appearances. To ensure up-to-date predictions in the inner regions of the moving objects, ARIA uses a convex hull algorithm to group the identified dynamic patches and fill each group.

4.2.2 Dynamic region tracking. Unfortunately, dynamic patch identification using embedding features relies on the GPU's full-frame analysis, which has a processing interval Δ_{GPU} longer than the frame interval. Simply resizing each frame to a low resolution and extracting embedding features using NPU is not feasible due to excessive overheads and the possible identification failures caused by loss of details. Therefore, ARIA proposes calculating motion information (e.g., moving direction and distance) for each identified dynamic region (i.e., group of dynamic patches) and tracking their locations across continuous frames, until the next full-frame feature embeddings become available. This is divided into two sub-tasks: (i) matching dynamic patches between the two consecutive frames processed by GPU (frames $t-\Delta_{\text{GPU}}$ and t) and (ii) estimating

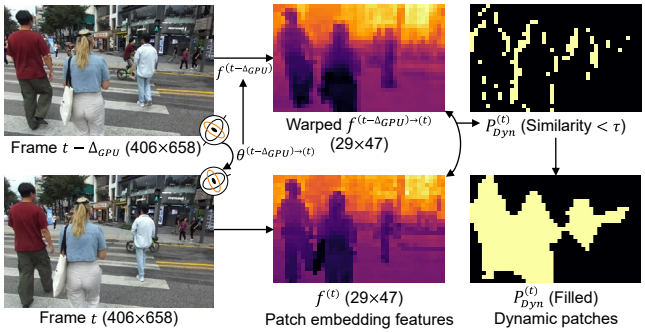


Figure 9: Transformer-guided dynamic region identification (patch size: 14×14, τ: 0.9).

the future locations of dynamic regions by applying their motion information.

For the first sub-task of dynamic patch matching, ARIA identifies the corresponding patches $P_{\text{Dyn}}^{(t-\Delta_{\text{GPU}})}$ at frame $t-\Delta_{\text{GPU}}$ for dynamic patches $P_{\text{Dyn}}^{(t)}$ at frame t as follows:

$$P_{\text{Dyn}}^{(t-\Delta_{\text{GPU}})} = \left\{ j \mid \arg\max_j sim\left(f_j^{(t-\Delta_{\text{GPU}})\rightarrow(t)}, f_{i_{\text{Dyn}}}^{(t)}\right) \right\}, \quad \forall i_{\text{Dyn}} \in P_{\text{Dyn}}^{(t)}, \tag{2}$$

where $j \in P^{(t-\Delta_{\text{GPU}})}$ denotes each patch location at frame $t-\Delta_{\text{GPU}}$. For the second sub-task of estimating locations of dynamic regions during the next full-frame processing interval, ARIA calculates the motion vector $(\Delta x, \Delta y)$ for each matched pair of dynamic patches based on their displacements. Then, at each incoming frame, the locations of dynamic regions are estimated by applying the average motion vectors of the corresponding patches. The tracked dynamic regions and their motion vectors are continuously updated based on the newly extracted embedding features using NPU.

4.3 Dynamics-aware Spatio-temporal Alignment of Decoupled Predictions

The final results of a VFM should be generated by ensuring temporal and spatial alignment of decoupled predictions—local updates of dynamic regions from NPU and the latest high-resolution prediction from GPU. First, a natural approach to achieving temporal alignment is to use warping based on camera motion information. Achieving reliable pixel-level warping requires accurate depth information [30], but the information is not always available since ARIA's goal is to support various vision tasks beyond depth estimation. The simplistic 2D planar transformation results in visual mismatches with the actual scene due to parallax artifacts [28, 64], disrupting an immersive AR experience. Second, a straightforward method for spatial alignment is to merge local updates for dynamic regions into the warped full-frame prediction using interpolation [60]. However, this is vulnerable to alignment errors in dynamic regions, particularly in cases of involving large displacements caused by fast-moving objects near the camera.

ARIA proposes an accurate spatio-temporal feature alignment method, which accounts for the two types of dynamics, i.e., camera

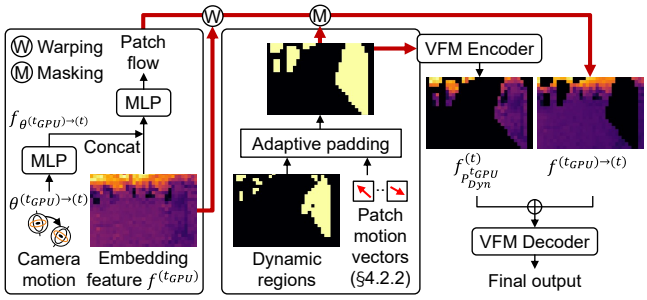


Figure 10: Spatio-temporal feature alignment.

and object movements. Similar to Section 4.2, this approach relies on intermediate patch features shared by VFMs rather than the final predictions (e.g., depth maps), scalable to various vision tasks. Figure 10 shows how the proposed spatio-temporal feature alignment modules are integrated into a VFM. First, each patch in the latest full-frame feature map is warped into the current frame based on patch flow estimation (Section 4.3.1). Next, the VFM encoder extracts embedding features for dynamic regions judiciously extended by our adaptive padding (Section 4.3.2), supporting seamless integration with the warped full-frame feature map. The aligned embedding features are then passed to the task-specific decoder, producing the final prediction results. In the following, we describe the details of our alignment modules.

4.3.1 Patch flow estimation for temporal alignment. The latest full-frame feature map generated by GPU at frame t_{GPU} is reused at each incoming frame t, where $t_{\text{GPU}} < t$. Leveraging the temporal similarity of continuous frames is a common approach to reduce the computational workload of complex DNN execution [20, 55]. To compensate for camera motion-induced temporal inconsistencies, ARIA introduces a patch flow estimation module. This module uses a Multi-Layer Perceptron (MLP) to predict the *patch flow*, representing the temporal displacement of each patch caused by the camera's motion. First, the camera motion $\theta^{(t_{GPU})\to(t)}$ is transformed into a latent embedding through an MLP to match the dimensionality of the patch embedding features. The camera motion embedding is concatenated with the patch embedding features and then passed through another MLP to generate the patch flows. Finally, the patches are warped using the predicted flows. These MLPs are optimized to predict accurate patch flows by learning the temporal disparities based on the distinctive visual cues for patches provided by VFMs.

4.3.2 Adaptive padding for spatial alignment. ARIA replaces the dynamic regions in the warped full-frame feature map with features newly extracted by NPU. Instead of employing additional neural components, ARIA addresses spatial inconsistencies using an adaptive padding module. This module adds vertical and horizontal padding of size s_p around each dynamic region, enabling VFMs to extract features with larger receptive fields. To do so, the relationships between dynamic regions and surrounding areas (e.g., relative depth scales) are well captured. To address the large displacement from fast-moving objects where the initial padding cannot cover, ARIA applies additional margins based on the moving trajectories (i.e., motion vectors) of dynamic regions acquired by our tracking method (Section 4.2.2). This allows for new observations on areas that were previously occluded by moving objects.

4.4 Execution Scheduling

Mobile AR applications have diverse requirements in terms of execution latency and prediction accuracy of vision tasks. The system's runtime scheduler aims to ensure that the inference latency L of a VFM on a mobile device does not exceed the given latency target L_T (e.g., 33 ms for 30 FPS), while maximizing the prediction accuracy A. The objective function of the scheduler is formulated as follows:

$$\max A, \quad \text{s.t.} \quad L \le L_T. \tag{3}$$

The design of the scheduler includes the following components: (i) defining the unique runtime factors of mobile AR scenarios that impact system performance, (ii) developing a heterogeneous multi-processor inference engine capable of adapting to the changes in these factors, and (iii) establishing an execution policy to satisfy the objective.

Understanding the relationship between the workload characteristics using heterogeneous mobile processors and the runtime factors in mobile AR is critical to achieving the scheduler's goal. As we discussed in Section 3, selectively inferring dynamic regions on NPU exhibits low and stable latency, meaning the effective latency L for generating the final output of a VFM is determined by NPU. Therefore, NPU's workload must be configured to satisfy the objective $L \le L_T$ on a given device, considering *scene dynamicity* attributed to the number and speed of dynamic patches. Unlike NPU, the GPU's full-frame inference workload exhibits variable latency profiles at runtime due to thermal throttling. Prolonged GPU execution frequency reduces the areas of the latest full-frame feature map that can be reused, and this issue is exacerbated by abrupt camera motion (e.g., large rotational movement). Consequently, the GPU's workload must be adjusted by accounting for both *camera motion* and *thermal throttling*.

We developed a heterogeneous mobile-processor inference engine capable of adapting to the dynamic runtime factors. The processor assignments and execution flow of system components are detailed in Figure 11. Full-frame VFM encoder inference is delegated to GPU, while NPU handles other neural networks, such as the dynamic region encoder. The CPU is responsible for the remaining tasks (e.g., camera motion estimation) and orchestrating the overall inference pipeline—managing input data and scheduling GPU/NPU execution. During the offline phase, the engine profiles the latency of full-frame encoder inference across multiple resolution levels to prepare for runtime adjustments. This resolution scaling mechanism compensates for GPU performance fluctuations (due to thermal throttling) and abrupt camera motion. Additionally, within the CPU-NPU execution flow, the engine determines the maximum number of the dynamic region encoder cycles that can be executed within the remaining latency budget, considering L_T and profiled latencies of other static operations. Since most mobile DNN inference frameworks do not support dynamic input shape [35], each execution cycle has a fixed size of input patches.

To achieve the goal of maximizing accuracy while adhering to the target latency L_T, the inference engine employs two scheduling policies for the decoupled VFM encoder inference workloads:

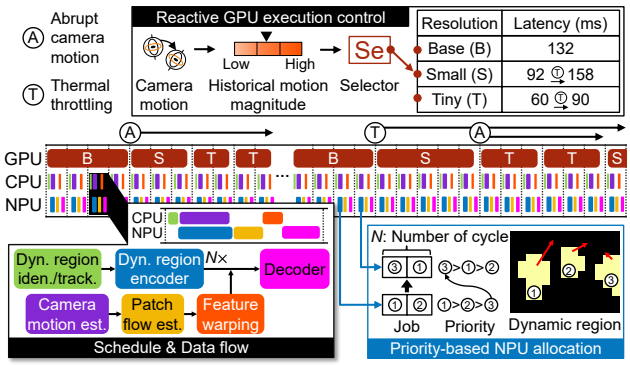


Figure 11: Heterogeneous multi-processor inference engine.

priority-based NPU allocation and reactive resolution selection for GPU execution. First, as not all dynamic regions in a frame can be processed within the predefined number of cycles, the engine determines their NPU delegation order based on priority. The priority of the dynamic region is decided by its moving speed, i.e., the magnitude of the motion vector calculated via dynamic region tracking. The rationale behind this is that up-to-date predictions for fast-moving objects are critical to minimizing user discomfort caused by visual misalignment between the objects and their virtual overlays [27]. Dynamic regions that are not processed in the current frame will be prioritized in the next frame. These regions are compensated by our tracking and adaptive padding methods, considering their updated locations and moving trajectories.

Second, as shown in the example runtime traces in Figure 11, the engine adjusts GPU's full-frame inference workload in response to camera motion and thermal throttling. When abrupt camera motion causes rapid scene changes, the engine resizes the frame resolution to a lower level, reducing the latency of GPU. The resolution selection criterion is based on the history of the camera motion magnitude during the user session. The magnitude is calculated as a weighted sum of translation and rotation magnitudes, where rotation is more prioritized since it causes drastic scene changes compared to translation. The engine maintains the historical magnitudes over a certain period and divides them into levels equal to the number of resolution levels, enabling a simple mapping between the current motion magnitude and the target resolution. Meanwhile, since predicting the performance changes based on a device's thermal states is impractical, the engine monitors the latencies of resolution levels and updates their profiles when significant deviations due to thermal throttling are detected. Although these reactive policies are feasible, exploring a more sophisticated policy that involves the intricate modeling of the relationship between the system performance and camera motion/thermal states remains a valuable area for future work.

5 Implementation

We implemented ARIA using C++ (Android NDK v25) and Java (Android API). We used Qualcomm Snapdragon platforms due to their popularity and floating-point computation capabilities with NPU. All neural networks in ARIA were converted to TensorFlow Lite (TFLite) 2.16.0 [17] models and executed using the TFLite C++ library. We used TFLite GPU Delegate [16] for GPU inference and Qualcomm Neural Network (QNN) Delegate [39] for NPU inference.

In our current implementation, the initial resolution levels for full-frame inference on the GPU are the same as those depicted in Figure 2, but they can be adjusted as needed. The patch size used for dynamic region identification and tracking relies on the input patch size of a given VFM, typically 14×14 or 16×16 pixels. The MLPs used for patch flow estimation were developed in PyTorch and trained on a subset of our MARV dataset, consisting of 25K images across multiple videos with diverse camera motion patterns. The training loss was calculated as a mean squared error (MSE) between the actual feature map and the feature map warped to the target frame using estimated patch flows. For camera motion estimation, we employed the Camera Pose API from ARCore [1].

6 Evaluation

We evaluated the prediction accuracy and system performance of ARIA on a large-scale dataset for mobile AR vision tasks. Experiments were conducted using Samsung Galaxy smartphones with Qualcomm Snapdragon 8 Gen APs, including Galaxy S22 (8 Gen 1 with Adreno 730 GPU and Hexagon 780 DSP) and S24 (8 Gen 3 with Adreno 750 GPU and Hexagon NPU).

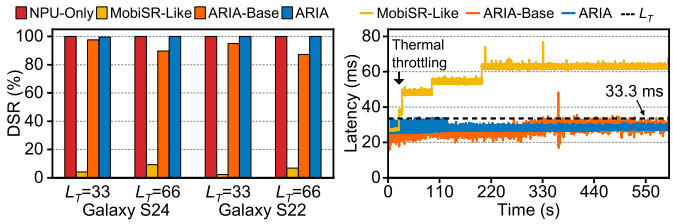
6.1 Evaluation Setup

Dataset. We created a custom dataset due to the lack of publicly available datasets that (i) support multiple visual prediction tasks for mobile AR, (ii) capture realistic camera movements across diverse environments, and (iii) provide 6 degrees of freedom (6DoF) camera pose information. To collect the data, we built a sensor rig using a smartphone and a ZED 2 stereo depth camera [44]. We collected video sequences from various indoor and outdoor spaces, including streets and campuses, capturing diverse camera motions and object interactions. During collection, the smartphone estimated camera poses using ARCore [1], while the ZED 2 camera produced ground-truth depth maps. For each captured RGB image, we generated pseudo-labels for pixel-level semantics using a cloud-scale VFM (556M parameters) with DINOv2 encoder (ViT-Large) and Mask2Former decoder, fine-tuned on the ADE20K dataset with 150 semantic classes. The resulting dataset, called MARV (Mobile AR Vision) dataset, contains 83K pose-labeled images with pixel-level depth and semantics. We excluded the images used for training the MLPs in patch flow estimation and conducted experiments on the remaining 58K test images.

Models. We used two VFMs in Table 1, both utilizing a DINOv2 (ViT-Small) encoder. For the decoder generating final predictions, we used the official DPT for metric depth estimation (MDE), and re-implemented Mask2Former for semantic segmentation (SS) to meet the memory constraints of mobile devices. These VFMs were not fine-tuned on our MARV dataset, as the goal was to assess their zero-shot accuracy on diverse AR scenes.

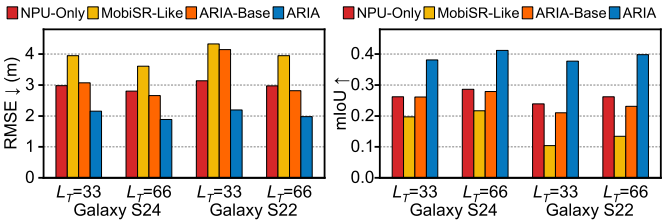
Metrics. For the evaluation, we used two metrics:

- **Deadline success rate (DSR):** DNN inference tasks in AR applications have diverse latency requirements, considering the constraints like device's display refresh rate [27]. We measured DSR as the ratio of frames that meet their latency



(a) Deadline success rate (b) End-to-end latency on S24

Figure 12: DSR and end-to-end latency.



(a) Metric depth estimation (b) Semantic segmentation

Figure 13: Visual prediction accuracy.

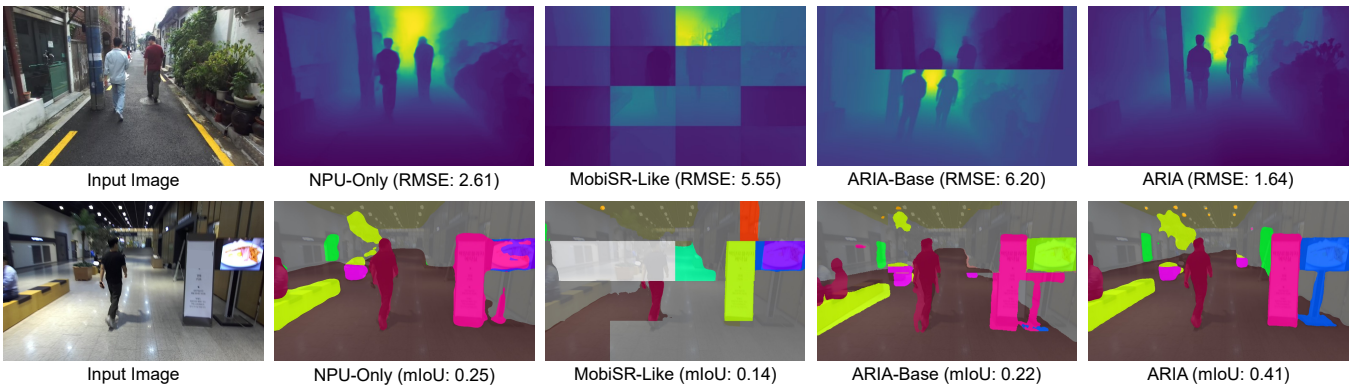


Figure 14: Example VFM predictions on MARV dataset with Samsung Galaxy S24 under L_T = 33.3 ms.

target L_T for running inference tasks to the total number of executed tasks during the experiments.

- **Prediction accuracy:** We used root mean squared error (RMSE) for MDE and mean intersection over union (mIoU) for SS as accuracy metrics. In real AR scenarios, frames with missing inference deadlines negatively impact user experiences, e.g., by disrupting virtual content rendering. Such cases should be treated as failures (zero accuracy), but we ignore these for a clearer comparison.

Baselines. We used three baselines for comparative analysis. All baselines were implemented to meet a predefined latency target L_T on a given device. We selected L_T values of 33.3 ms (30 FPS) and 66.6 ms (15 FPS), reflecting the stringent latency requirements of mobile AR applications. The baselines include:

- **NPU-Only:** A single-processor baseline that relies on NPU FP16 operations. The input resolutions were decided based on the profiled latency and L_T. We did not consider GPU-based baseline because the resolutions adhering to the strict L_T settings on GPU were no larger than those on NPU, as shown in Figure 2b.
- **MobiSR-Like:** MobiSR [29] was originally designed for super-resolution tasks but can be generalized to other vision tasks. We modified MobiSR to process image blocks using different mobile processors except for CPU due to its excessive VFM inference latency. Image blocks are allocated to the processors based on scene dynamicity, i.e., the number of dynamic patches.
- **ARIA-Base:** A baseline version of ARIA without its solutions, where the overview is shown in Figure 5. For dynamic region identification, this baseline utilizes image block matching [55]. The latest prediction from GPU is aligned to the current frame using simple 2D planar transformation and the NPU's results on dynamic regions are merged using bilinear interpolation.

6.2 Overall Performance

Deadline success rate. We analyzed DSR across different devices and latency targets, as shown in Figure 12a. We only reported DSR for MDE, as no significant differences were observed between MDE and SS. The results indicated that MobiSR-Like and ARIA-Base exhibited lower DSRs compared to other methods. MobiSR-Like missed the deadline for 93.1–97.7% of frames under different settings. For each frame, MobiSR-Like offloads the inference tasks of some image blocks to GPU. As shown in Figure 12b, GPU execution caused thermal throttling, rapidly increasing the end-to-end latency—the total time required to generate the final result for each frame. In contrast, the end-to-end latencies of ARIA-Base were decided by CPU and NPU, eliminating the effects of GPU performance degradation through concurrent execution. However, it processes computationally intensive tasks like image block matching [55] for dynamic region identification using CPU (e.g., 8–10 ms on Galaxy S24). Mobile CPUs are also prone to thermal issues, where the effects are exacerbated by heat generation from GPU [37]. During runtime, this resulted in 2.4% of frames failing to meet the strict 33.3 ms deadline on Galaxy S24. ARIA, on the other hand, achieved an

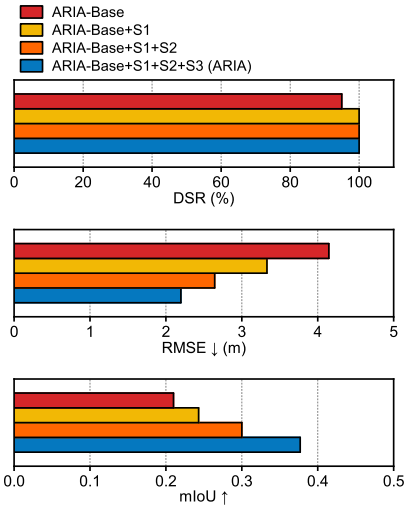


Figure 15: Comparison of DSR, MDE accuracy, and SS accuracy achieved by ARIA and its variants on Samsung Galaxy S22 under L_T = 33.3 ms.

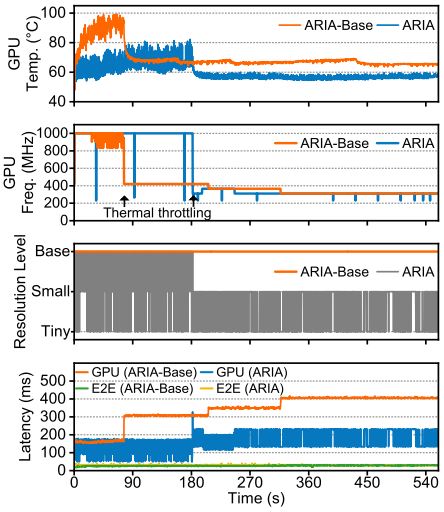


Figure 16: Traces of GPU temperature, GPU frequency, resolution level for GPU execution, and latency on Samsung Galaxy S24 under L_T = 33.3 ms.

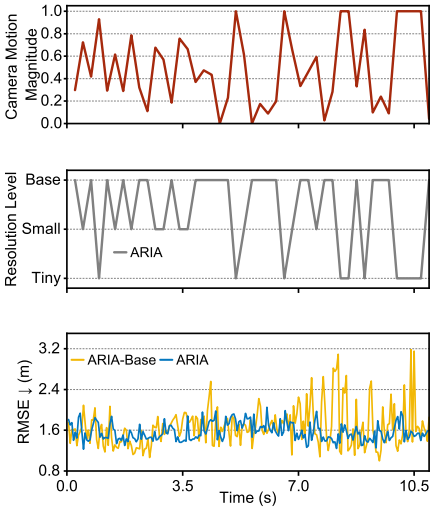


Figure 17: Sampled traces of camera motion magnitude, resolution level for GPU execution, and MDE accuracy on Samsung Galaxy S24 under L_T = 33.3 ms.

average DSR of 99.9% across all settings. By minimizing CPU overhead associated with dynamic region identification, ARIA ensured consistent performance under stringent deadlines.

Visual prediction accuracy. We validated the visual prediction accuracy of MDE and SS, as shown in Figure 13. As we discussed in Section 2.3, maintaining zero-shot capabilities of VFMs requires full-frame floating-point execution. However, the limited capacity of NPU-Only to handle large-scale attention operations necessitates compromises in image resolutions to meet the latency constraints, e.g., 294×350 for MDE on Galaxy S24. Image downscaling leads to the loss of fine-grained details like object edges in final predictions, as shown in Figure 14. MobiSR-Like, which provides low-resolution predictions in a block granularity, undermines the global processing capabilities of VFMs and therefore resulted in severe visual inconsistencies, as illustrated in the example predictions on MARV dataset. Consequently, MobiSR-Like showed the lowest average accuracy across all tested configurations. ARIA overcomes these issues by offloading full-frame, high-resolution operations to GPU while leveraging NPU to provide globally aligned predictions on dynamic regions. Compared to NPU-Only and MobiSR-Like, ARIA improved accuracy by 27.6–33.4% and 45.3–50.0% for MDE and 30.5–36.7% and 47.2–72.3% for SS, respectively.

Meanwhile, ARIA-Base struggled to deliver visually consistent predictions, as shown in Figure 14. Specifically, we observed a significant accuracy loss with Galaxy S22 under 33.3 ms latency target, due to the device's limited computing capabilities. To meet this stringent latency requirement, ARIA-Base relied on lower-resolution predictions for processing dynamic regions using NPU. Also, its inefficiency in dynamic region identification further reduced the latency budget allocated for handling these regions, resulting in an extremely low resolution of 168×168 for MDE.

6.3 Ablation Study

To further analyze the effectiveness of three key solutions proposed in ARIA, we conducted an ablation study using Galaxy S22 with a latency target of 33.3 ms, which is the most challenging setup in our experiment. We isolated and evaluated the solutions incrementally added to ARIA-Base, as shown in Figure 15.

By applying dynamic region identification and tracking (S1), we observed improvements in DSR. The moderate enhancement, reducing the missing frames from 5.0% to 0.009%, is crucial, as such delayed frames are highly noticeable to users in subsequent AR functionalities [27], adversely affecting the immersive experience. The reduced latency is mainly attributed to the efficient dynamic region identification, which leverages low-dimensional embedding feature maps. Compared to ARIA-Base, this method also improved accuracy by 20.0% for MDE and 13.4% for SS. By employing a fine-grained and precise identification method, it minimized the inclusion of irrelevant static regions and generated predictions on dynamic regions using NPU, a crucial factor for enhancing accuracy. Further incorporating both spatio-temporal alignment modules (S2) and execution scheduler (S3) proposed in ARIA improved accuracy by 34.0% for MDE and 35.6% for SS, demonstrating the efficacy of our system design.

6.4 System Robustness

ARIA adjusts the execution of GPU and NPU to meet the accuracy and latency requirements of user applications under varying runtime conditions. To analyze ARIA's robustness, we collected runtime traces on Galaxy S24 with a latency target of 33.3 ms. The traces were based on a 600-second video in MARV dataset, capturing traversal across a diverse range of environments. As shown in Figure 16, ARIA maintains stable, low-latency performance throughout the entire sequence. While ARIA-Base quickly heated GPU and suffered from subsequent performance degradation, ARIA kept the

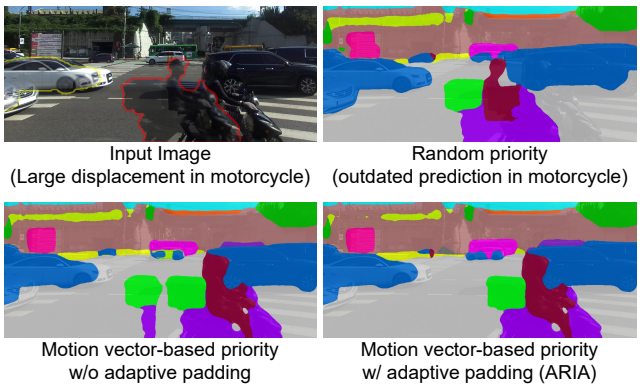


Figure 18: Efficacy of priority-based NPU allocation.

temperature lower and prevented sudden performance drops by adaptive resolution scaling for full-frame GPU execution. Moreover, as shown in Figure 17, ARIA's resolution scaling effectively managed varying camera motion, preserving high prediction accuracy even under rapid pose changes. In contrast, ARIA-Base continued to rely on a high-resolution input (Base) for GPU execution, causing the latest full-frame prediction to be outdated, thereby reducing accuracy under these conditions. The delay was further exacerbated by thermal throttling, leading to temporal misalignment between GPU and NPU outputs, as shown in Figure 14.

ARIA's priority-based NPU allocation ensures that the most dynamically changing regions, such as fast-moving objects, receive timely updates, as shown in Figure 18. By ranking dynamic regions according to their motion vectors, ARIA focused its limited NPU execution cycles to provide up-to-date predictions on areas that contribute most to user discomfort if left outdated, i.e., a motorcycle in Figure 18. When random priority is applied, however, predictions for these areas may become stale and visually misaligned. ARIA further integrates object motion–aware adaptive padding with our priority-based approach to mitigate the issues arising from incomplete coverage.

6.5 Runtime Overhead

Lastly, we analyzed the runtime overhead of ARIA. Figure 19 shows the end-to-end latency breakdown on Galaxy S24. Although ARIA's spatio-temporal alignment modules introduced some latency overhead, primarily due to patch flow estimation, the dynamic region identification and execution scheduling methods were designed to be efficient, resulting in negligible additional latency. Under a relaxed latency target of 66.6 ms, the number of NPU's execution cycles for the dynamic region encoder is increased, dominating the end-to-end latency. For instance, the number of cycles under the latency target of 66.6 ms is 4, compared to 1 cycle under 33.3 ms. This increase in cycles is the primary factor behind the accuracy improvements shown in Figure 13, as the extended cycles often fully capture the dynamic regions, even under conditions of high scene dynamicity.

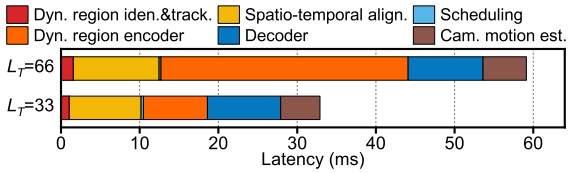


Figure 19: Breakdown of end-to-end latency on Samsung Galaxy S24 with different L_T settings.

7 Related Work

Domain adaptation for mobile DNN inference. Domain adaptation is an emerging technique that updates DNN model parameters to adapt to environmental shifts. This is particularly critical for maintaining the robustness of DNNs in applications running on mobile platforms, such as robots or AR devices, where the environment is constantly changing. Supervised adaptation techniques [5, 14, 57] apply model fine-tuning based on labels from a few data samples in unseen domains. To acquire labels of unseen data, EdgeFM [57] proposes to leverage foundation models (FM) on a cloud server, transferring their knowledge to a model deployed on edge devices. Recent adaptation methods [10, 43, 47, 48] have shown that models can adapt to domain shifts in an unsupervised manner without requiring labeled data. AdaShadow [10] is a pioneering work that applies such adaptation techniques to latency-critical applications like mobile AR. However, as recent mobile applications increasingly demand high-performance models with large numbers of parameters, fine-tuning can introduce unacceptable latency overheads. As a promising alternative, there is a growing effort to deploy FMs directly on mobile devices to enable effective zero-shot adaptation without the need for fine-tuning.

On-device inference of foundation models. Along with the success of FMs such as Large Language Models (LLMs), private, on-device inference of these models has been recently gaining attention. Initial efforts have relied on model compression techniques such as pruning [11, 34], knowledge distillation [45, 50, 51, 63], and quantization [8, 13, 32, 59]. However, compressing FMs to a mobile scale introduces the risk of diminishing their zero-shot capabilities. Post-training quantization [13, 59] alleviates the issue through calibration using data samples in the target domain but requires continuous re-calibration to cope with the environmental shifts. Early-exit [53, 67] or Mixture-of-Experts [12, 62] process only necessary components in foundational models, reducing the redundant computations. These methods, however, require additional deployment costs customizing FMs, e.g., training early-exit layers. While many of these solutions emphasize algorithmic optimization, it is equally important to consider system-level characteristics (e.g., thermal effects) and opportunities (e.g., hardware accelerators) to enhance the processing efficiency of FMs deployed on mobile devices [25]. In contrast to prior systems [54, 56] that primarily target generative FMs such as LLMs, our work focuses on discriminative vision tasks—like depth estimation and semantic segmentation—that are critical for mobile AR applications.

On-device inference acceleration using heterogeneous mobile processors. A line of works [21, 22, 31, 38] utilizes heterogeneous mobile processors to accelerate the DNN inference workloads. FYE-SR [19] splits the model computations into GPU-based floating-point operations and NPU-based integer operations, according to their sensitivity to quantization. Although this approach is promising, extension to ARIA requires much effort to minimize the possible accuracy drops of quantization-sensitive VFMs and ViT switching overheads between processors. MobiSR [29] splits the input image, rather than model layers, into blocks and judiciously processes them using multiple processors, but ignores visual quality loss due to the inconsistency in decoupled prediction results.

Selective and adaptive inference for optimized resource utilization. Selective inference schemes [20, 55, 60, 61] reduce inference latency by applying DNN computations to sample frames and reusing the results for subsequent frames. NEMO [60] employs this scheme to specifically accelerate super-resolution-enhanced video streaming using codec information. DeepCache [55] reuses the cached intermediate results of DNNs, instead of final outputs, focusing on reusable regions identified through image block matching. ARIA aligns with this method but enhances efficiency by employing finer-grained identification of non-reusable dynamic regions and offloading them to NPU for processing. Additionally, ARIA relates to adaptive inference frameworks [23, 26, 27, 49], which prioritize DNN computations for critical entities such as fast-moving objects or complex scenes, further optimizing resource utilization.

8 Discussion and Future Work

Generalizability and hardware dependence. Although our experiments focused on Qualcomm's Snapdragon platforms—chosen for their wide adoption and support for floating-point NPU operations—ARIA's core design is not restricted to a single hardware vendor. We believe that its parallel and selective inference approach can be generalized to any mobile device with heterogeneous computing processors. While higher-capacity NPUs may better accommodate high-resolution FP16 operations of VFMs, they do not obviate the need for heterogeneous processing, particularly as the size of VFMs continues to grow. Additionally, not all mobile devices include NPUs; ARIA can leverage other low-power processors (e.g., DSPs or CPU little cores) when available.

Multi-task visual prediction support. The current ARIA implementation is optimized for accelerating inference of a single VFM dedicated to a specific task, such as depth estimation or semantic segmentation. However, real-world AR applications often require simultaneous predictions from multiple vision tasks—for example, combining depth and semantics to provide both spatial and contextual understandings. Supporting such multi-task scenarios introduces new challenges in system design, particularly under stringent latency constraints. A promising future direction is to share a VFM encoder across tasks or to selectively activate task-specific components (e.g., decoders or experts), thereby minimizing redundant computations. Additionally, the scheduler could be extended to incorporate task-aware accuracy-latency trade-offs, dynamically prioritizing tasks based on the AR context—such as prioritizing semantic cues for navigation vs. depth for object placement—while managing hardware resource contention. Ensuring robust and efficient multi-task inference, without compromising ARIA's real-time performance, is a critical step toward realizing practical and fully-featured mobile AR systems.

9 Conclusion

This paper proposes ARIA, a novel system for optimizing the on-device inference of VFMs for mobile AR applications. By leveraging the unique characteristics of heterogeneous mobile processors, ARIA combines GPU-based full-frame predictions with NPU-accelerated updates on dynamic regions, achieving both high accuracy and real-time performance. Key innovations include fine-grained dynamic region identification and tracking, spatio-temporal feature alignment of decoupled predictions, and adaptive execution scheduling to handle runtime variations. Extensive experiments have shown that ARIA outperforms its baselines under diverse conditions in environments and devices, paving the way for scalable, immersive mobile AR applications.

Acknowledgments

We sincerely thank the anonymous shepherd and the reviewers for their valuable feedback in improving this paper. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00344323) and the Ministry of Education (No. RS-2024-00351030). This work was also supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by MSIT (No. RS-2018-II180532).

References

[1] ARCore. 2024. Camera. https://developers.google.com/ar/reference/java/com/google/ar/core/Camera.

[2] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. 2021. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4009–4018.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9650–9660.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*. 801–818.

[5] Minghao Chen, Hongyang Xue, and Deng Cai. 2019. Domain adaptation for semantic segmentation with maximum squares loss. In *Proceedings of the IEEE/CVF international conference on computer vision*. 2090–2099.

[6] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. 2022. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1290–1299.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3213–3223.

[8] Yifu Ding, Haotong Qin, Qinghua Yan, Zhenhua Chai, Junjie Liu, Xiaolin Wei, and Xianglong Liu. 2022. Towards Accurate Post-Training Quantization for Vision Transformer. In *Proceedings of the 30th ACM International Conference on Multimedia* (Lisboa, Portugal) *(MM '22)*. Association for Computing Machinery, New York, NY, USA, 5380–5388. https://doi.org/10.1145/3503161.3547826

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[10] Cheng Fang, Sicong Liu, Zimu Zhou, Bin Guo, Jiaqi Tang, Ke Ma, and Zhiwen Yu. 2024. AdaShadow: Responsive Test-time Model Adaptation in Non-stationary Mobile Environments. In *Proceedings of the 22nd ACM Conference*

on Embedded Networked Sensor Systems (Hangzhou, China) *(SenSys '24)*. Association for Computing Machinery, New York, NY, USA, 295–308. https://doi.org/10.1145/3666025.3699339

[11] Matteo Farina, Massimiliano Mancini, Elia Cunegatti, Gaowen Liu, Giovanni Iacca, and Elisa Ricci. 2024. MULTIFLOW: Shifting Towards Task-Agnostic Vision-Language Pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16185–16195.

[12] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research* 23, 120 (2022), 1–39.

[13] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[14] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*. PMLR, 1180–1189.

[15] Google. 2024. ARCore. https://developers.google.com/ar.

[16] Google. 2024. GPU delegates for LiteRT. https://ai.google.dev/edge/litert/performance/gpu.

[17] Google. 2024. LiteRT overview. https://ai.google.dev/edge/litert.

[18] Google. 2024. Scene Semantics API. https://developers.google.com/ar/develop/scene-semantics.

[19] Kai Huang, Xiangyu Yin, Tao Gu, and Wei Gao. 2024. Perceptual-Centric Image Super-Resolution using Heterogeneous Processors on Mobile Devices. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (Washington D.C., DC, USA) *(ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 1361–1376. https://doi.org/10.1145/3636534.3690698

[20] Loc N. Huynh, Youngki Lee, and Rajesh Krishna Balan. 2017. DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services* (Niagara Falls, New York, USA) *(MobiSys '17)*. Association for Computing Machinery, New York, NY, USA, 82–95. https://doi.org/10.1145/3081333.3081360

[21] Joo Seong Jeong, Jingyu Lee, Donghyun Kim, Changmin Jeon, Changjin Jeong, Youngki Lee, and Byung-Gon Chun. 2022. Band: coordinated multi-DNN inference on heterogeneous mobile processors. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services* (Portland, Oregon) *(MobiSys '22)*. Association for Computing Machinery, New York, NY, USA, 235–247. https://doi.org/10.1145/3498361.3538948

[22] Fucheng Jia, Deyu Zhang, Ting Cao, Shiqi Jiang, Yunxin Liu, Ju Ren, and Yaoxue Zhang. 2022. CoDL: efficient CPU-GPU co-execution for deep learning inference on mobile devices. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services* (Portland, Oregon) *(MobiSys '22)*. Association for Computing Machinery, New York, NY, USA, 209–221. https://doi.org/10.1145/3498361.3538932

[23] Shiqi Jiang, Zhiqi Lin, Yuanchun Li, Yuanchao Shu, and Yunxin Liu. 2021. Flexible high-resolution object detection on edge devices with tunable latency. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking* (New Orleans, Louisiana) *(MobiCom '21)*. Association for Computing Machinery, New York, NY, USA, 559–572. https://doi.org/10.1145/3447993.3483274

[24] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4015–4026.

[25] Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, and Hamed Haddadi. 2024. MELTing Point: Mobile Evaluation of Language Transformers. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (Washington D.C., DC, USA) *(ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 890–907. https://doi.org/10.1145/3636534.3690668

[26] Jeho Lee, Chanyoung Jung, Jiwon Kim, and Hojung Cha. 2024. Panopticus: Omnidirectional 3D Object Detection on Resource-constrained Edge Devices. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (Washington D.C., DC, USA) *(ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 1207–1221. https://doi.org/10.1145/3636534.3690688

[27] Jingyu Lee, Hyunsoo Kim, Minjae Kim, Byung-Gon Chun, and Youngki Lee. 2024. Maestro: The Analysis-Simulation Integrated Framework for Mixed Reality. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services* (Minato-ku, Tokyo, Japan) *(MobiSys '24)*. Association for Computing Machinery, New York, NY, USA, 99–112. https://doi.org/10.1145/3643832.3661891

[28] Kyu-Yul Lee and Jae-Young Sim. 2020. Warping residual based image stitching for large parallax. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8198–8206.

[29] Royson Lee, Stylianos I. Venieris, Lukasz Dudziak, Sourav Bhattacharya, and Nicholas D. Lane. 2019. MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors. In *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 54, 16 pages. https://doi.org/10.1145/3300061.3345455

[30] Danyang Li, Jingao Xu, Zheng Yang, Qian Zhang, Qiang Ma, Li Zhang, and Pengpeng Chen. 2022. Motion inspires notion: self-supervised visual-LiDAR fusion for environment depth estimation. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services* (Portland, Oregon) *(MobiSys '22)*. Association for Computing Machinery, New York, NY, USA, 114–127. https://doi.org/10.1145/3498361.3538918

[31] Neiwen Ling, Xuan Huang, Zhihe Zhao, Nan Guan, Zhenyu Yan, and Guoliang Xing. 2023. BlastNet: Exploiting Duo-Blocks for Cross-Processor Real-Time DNN Inference. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems* (Boston, Massachusetts) *(SenSys '22)*. Association for Computing Machinery, New York, NY, USA, 91–105. https://doi.org/10.1145/3560905.3568520

[32] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. 2021. Post-training quantization for vision transformer. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 2152, 12 pages.

[33] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. 2019. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172* (2019).

[34] Wei Niu, Xiaolong Ma, Sheng Lin, Shihao Wang, Xuehai Qian, Xue Lin, Yanzhi Wang, and Bin Ren. 2020. PatDNN: Achieving Real-Time DNN Execution on Mobile Devices with Pattern-based Weight Pruning. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 907–922. https://doi.org/10.1145/3373376.3378534

[35] onnxruntime. 2024. QNN Execution Provider. https://onnxruntime.ai/docs/execution-providers/QNN-ExecutionProvider.html.

[36] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193* (2023).

[37] Jihoon Park, Seokjun Lee, and Hojung Cha. 2018. App-Oriented Thermal Management of Mobile Devices. In *Proceedings of the International Symposium on Low Power Electronics and Design* (Seattle, WA, USA) *(ISLPED '18)*. Association for Computing Machinery, New York, NY, USA, Article 36, 6 pages. https://doi.org/10.1145/3218603.3218622

[38] Keondo Park, You Rim Choi, Inhoe Lee, and Hyung-Sin Kim. 2023. PointSplit: Towards On-device 3D Object Detection with Heterogeneous Low-power Accelerators. In *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks* (San Antonio, TX, USA) *(IPSN '23)*. Association for Computing Machinery, New York, NY, USA, 67–81. https://doi.org/10.1145/3583120.3587045

[39] Qualcomm. 2024. Qualcomm Linux TensorFlow Lite Runtime Reference. https://docs.qualcomm.com/bundle/publicresource/topics/80-70014-54/overview.html.

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

[41] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*. 12179–12188.

[42] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. 2021. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10912–10922.

[43] Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. 2023. Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11920–11929.

[44] stereolabs. 2024. ZED 2. https://www.stereolabs.com/products/zed-2.

[45] Ximeng Sun, Pengchuan Zhang, Peizhao Zhang, Hardik Shah, Kate Saenko, and Xide Xia. 2023. Dime-fm: Distilling multimodal and efficient foundation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 15521–15533.

[46] Stylianos I. Venieris, Mario Almeida, Royson Lee, and Nicholas D. Lane. 2024. NAWQ-SR: A Hybrid-Precision NPU Engine for Efficient On-Device Super-Resolution. *IEEE Transactions on Mobile Computing* 23, 3 (2024), 2367–2381. https://doi.org/10.1109/TMC.2023.3255822

[47] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2020. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726* (2020).

[48] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. 2022. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision*

and Pattern Recognition. 7201–7211.

[49] Weijun Wang, Liang Mi, Shaowei Cen, Haipeng Dai, Yuanchun Li, Xiaoming Fu, and Yunxin Liu. 2024. Region-based content enhancement for efficient video analytics at the edge. *arXiv preprint arXiv:2407.16990* (2024).

[50] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MINILM: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 485, 13 pages.

[51] Yixuan Wei, Han Hu, Zhenda Xie, Ze Liu, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, and Baining Guo. 2023. Improving clip fine-tuning performance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5439–5449.

[52] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. 2019. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 6101–6108.

[53] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. *arXiv preprint arXiv:2004.12993* (2020).

[54] Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. 2025. Fast On-device LLM Inference with NPUs. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1* (Rotterdam, Netherlands) *(ASPLOS '25)*. Association for Computing Machinery, New York, NY, USA, 445–462. https://doi.org/10.1145/3669940.3707239

[55] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. 2018. DeepCache: Principled Cache for Mobile Deep Vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (New Delhi, India) *(MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 129–144. https://doi.org/10.1145/3241539.3241563

[56] Zhenliang Xue, Yixin Song, Zeyu Mi, Xinrui Zheng, Yubin Xia, and Haibo Chen. 2024. Powerinfer-2: Fast large language model inference on a smartphone. *arXiv preprint arXiv:2406.06282* (2024).

[57] Bufang Yang, Lixing He, Neiwen Ling, Zhenyu Yan, Guoliang Xing, Xian Shuai, Xiaozhe Ren, and Xin Jiang. 2024. EdgeFM: Leveraging Foundation Model for Open-set Learning on the Edge. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems* (Istanbul, Turkiye) *(SenSys '23)*. Association for Computing Machinery, New York, NY, USA, 111–124. https://doi.org/10.1145/3625687.3625793

[58] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. 2024. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10371–10381.

[59] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 27168–27183. https://proceedings.neurips.cc/paper_files/paper/2022/file/adf7fa39d65e2983d724ff7da57f00ac-Paper-Conference.pdf

[60] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2020. NEMO: enabling neural-enhanced video streaming on commodity mobile devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 28, 14 pages. https://doi.org/10.1145/3372224.3419185

[61] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: neural video enhancement at scale. In *Proceedings of the ACM SIGCOMM 2022 Conference* (Amsterdam, Netherlands) *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 795–811. https://doi.org/10.1145/3544216.3544218

[62] Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023. Edgemoe: Fast on-device inference of moe-based large language models. *arXiv preprint arXiv:2308.14352* (2023).

[63] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. 2023. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289* (2023).

[64] Fan Zhang and Feng Liu. 2014. Parallax-tolerant image stitching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3262–3269.

[65] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2881–2890.

[66] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. 2017. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 633–641.

[67] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems* 33 (2020), 18330–18341.