Title

Adapting Multimodal Systems to Inference Time Variations

ISBN

9798186372370

Author

Wu, Jason

Publication Date

2026-08-18

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Adapting Multimodal Systems to Inference Time Variations

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Jason Wu

2026

ABSTRACT OF THE DISSERTATION

Adapting Multimodal Systems to Inference Time Variations

by

Jason Wu
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2026
Professor Mani B. Srivastava, Chair

Multimodality has elevated the utility of modern machine learning models. Through data-driven deep learning, multimodal neural networks learn generalizable and robust features, providing increased performance and capabilities while promising greater resilience to changing environments. Despite such capabilities, the increased complexity of these networks renders them susceptible to other variations that can manifest at inference time – ranging from dynamics during a system's deployment to its active runtime. These weaknesses of multimodal networks have not been adequately addressed in existing work, resulting in severe inefficiencies or subpar performance when subject to inference-time variations in real-world environments. This dissertation examines the weaknesses of multimodal networks across a variety of inference-time variations and diverse datasets, demonstrating that the efficient adaptation of internal weights constitutes a powerful and highly generalizable solution.

First, we examine the challenge of deployment-time sensor perspective shift in distributed multimodal localization systems. Existing multimodal deep neural networks do not tolerate novel sensor perspectives arising during deployment, suffering from large accuracy degradations. We introduce FlexLoc [1], which combats sensor perspective shift by conditioning a

portion of network weights on the sensors' pose information. FlexLoc improves zero-shot localization under unseen configurations of sensor pose by almost 50% in comparison to baselines. *Second*, we propose ADMN [2], which further explores inference-time adaptation by addressing variations that manifest during *runtime* in real-world multimodal systems, which may necessitate per-sample adaptation. Across a wide range of downstream tasks (e.g., audio-visual classification, gesture detection), we showcase how intelligent allocation of resources across modalities is critical for addressing runtime variations, ensuring optimal usage of limited computational resources. *Third*, we introduce SWAN [3], which extends the innovations of ADMN into the complex realm of multi-object detection in autonomous driving. We jointly explore adaptation to three forms of runtime variations – changes in modality quality, platform resources, and complexity of the input sample itself. Moreover, we demonstrate how such a system can be practically realized on a real-world edge device. *Finally*, moving away from dedicated, task-specific models, we introduce CRAFT [4], an efficient method of adapting a general purpose multimodal foundation model at deployment time to an arbitrary downstream task. During this, we highlight the unique challenges associated with this class of neural networks. Ultimately, by leveraging the shared paradigm of context-aware model adaptation, these works collectively demonstrate that adjusting neural network parameters according to the state of the world is a highly effective strategy for mitigating inference-time variations.

The dissertation of Jason Wu is approved.

Jonathan Kao

Cho-Jui Hsieh

Suhas Diggavi

Mani B. Srivastava, Committee Chair

University of California, Los Angeles

2026

To all who supported me along this journey

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to acknowledge my advisor, Professor Mani Srivastava, for all his advice and feedback. I have grown so much as a researcher as a result of your guidance. Additionally, I would like to extend my gratitude to the members of my thesis committee – Prof. Suhas Diggavi, Prof. Cho-Jui Hsieh, Prof. Jonathan Kao – for their input.

I would also like to thank all the lab members at NESL, as well as all of my collaborators over the years. In particular, I am grateful for my first mentor Dr. Ziqi Wang, who guided me through research when I was an undergraduate student. My accomplishments are a result of your unwavering patience and selflessness.

I am grateful for the agencies that have provided funding for my research. My research has been sponsored in part by the DEVCOM Army Research Laboratory (award W911NF1720196), the Air Force Office of Scientific Research (awards FA95502210193 and FA95502310559), and the National Institutes of Health (award 1P41EB028242). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-23-F-0014 in the amount of $86,400.

VITA

2020–2023	B.S Electrical Engineering, University of California, Los Angeles
2023-2024	M.S Electrical and Computer Engineering, University of California, Los Angeles
2025	Summer Research Intern at AWS AI Labs
2023-Present	Ph.D. Student in Electrical and Computer Engineering, Univerisity of California, Los Angeles

PUBLICATIONS

Han, L., Yang, K., Wang, O., **Wu, J**., Quan, P., Dong, G., ... Others. (2026). TS-Skill: A Benchmark for Evaluating Analytical Skills in Time-Series Question Answering. arXiv Preprint arXiv:2605. 24703. (Under Review for Neurips 2026 Dataset and Benchmarks)

Wu, J., Jin, S.-K. S., Yuan, Y., Wigness, M., Kaplan, L. M., Qiu, H., & Srivastava, M. (2026). SWAN: World-Aware Adaptive Multimodal Networks for Runtime Variations. arXiv Preprint arXiv:2604. 26181. (Appearing in ECCV 2026)

Wu, J., Zhao, T., Liu, C., Cai, J., Zhang, Z., Li, Z., ... Wu, J. (2026). Decoupling Vision and Language: Codebook Anchored Visual Adaptation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 36957–36967.

Wu, J., Yuan, Y., Yang, K., Kaplan, L., & Srivastava, M. (2026). ADMN: A Layer-Wise Adaptive Multimodal Network for Dynamic Input Noise and Compute Resources. Advances in Neural Information Processing Systems, 38, 68757–68790.

Ouyang, X., **Wu, J.**, Kimura, T., Lin, Y., Verma, G., Abdelzaher, T., & Srivastava, M. (2025). Mmbind: Unleashing the potential of distributed and heterogeneous data for multimodal learning in iot. Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems, 491–503.

Wu, J., Wang, Z., Ouyang, X., Jeong, H. L., Samplawski, C., Kaplan, L. M., ... Srivastava, M. (2024). FlexLoc: Conditional Neural Networks for Zero-Shot Sensor Perspective Invariance in Object Localization with Distributed Multimodal Sensors. 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 8563–8570. IEEE

Dong, G., **Wu, J.**, de Gortari Briseno, J., Singh, A. D., Feng, J., Sarker, A., ... Srivastava, M. (2024). RefreshChannels: Exploiting dynamic refresh rate switching for mobile device attacks. Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services, 359–371.

Wu, J., Wang, Z., Sarker, A., & Srivastava, M. (2023). Acuity: Creating realistic digital twins through multi-resolution pointcloud processing and audiovisual sensor fusion. Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation, 79–92.

CHAPTER 1

Introduction

As modern machine learning systems are tasked with increasingly advanced duties, many systems have transitioned from leveraging a single modality towards utilizing multiple modalities for superior performance [10–13]. A modality is defined as a data channel containing a unique statistical and structural characteristic typically obtained through capturing a real-world signal. For instance, audio, images, and infrared depth maps, owing to their unique features arising from capture of distinct real-world signals, constitute independent modalities. Conversely, altering the position or orientation of a camera has no bearing upon its modality. Multimodal systems leverage information from two or more modalities to perform a given downstream task, whereas unimodal systems are constrained to operating over a single modality's data.

The recent explosion of popularity in multimodality has occurred in the context of *multimodal machine learning systems* [14–16]. These systems rely upon neural networks to extract task-relevant features from the rich, frequently high dimensional multimodal data, replacing the traditional reliance upon hand-crafted features [17]. The *data-driven training stage* consists of a neural model, a multimodal training dataset, and a predefined loss function, where the model adjusts its internal parameters to minimize the loss over the training dataset. Neural networks are particularly well suited for processing multimodal data, owing to their ability to discover complex *intra-modality* (i.e., features extracted from a single modality) and *inter-modality* (i.e., information obtained from jointly processing several modalities) relationships that can greatly elevate performance [18]. For instance, a camera-radar multi-

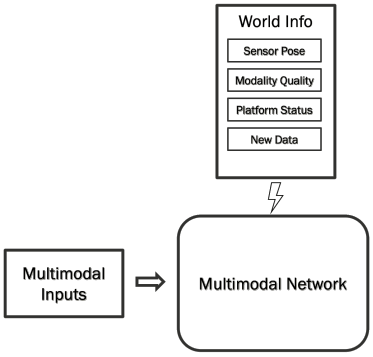

Figure 1.1: Overall framework for inference-time adaptation

modal system can extract self-contained *intra-modality* features such as color and vibration from camera and radar, respectively. Additionally, it can also reason over the information from both modalities through *inter-modality* processing to arrive at a more informed conclusion. Beyond improving performance, multimodality provides additional benefits by enhancing system *robustness* [19]. Through *intra and inter-modality processing*, multimodal systems are less susceptible to external factors (e.g., low light) corrupting a single modality, retaining accuracy in the face of environmental corruptions. Thus, learned *intra and inter-modality* relationships provide both elevated performance and robustness by fully utilizing the richness of multimodal information.

Due to the aforementioned performance and robustness of multimodal neural network systems, they are naturally frequently deployed in environments that take full advantage of these capabilities. They thrive in highly dynamic scenes (e.g., autonomous driving [13, 20], battlefields [21], emergency response [22]) in which the state of the world is subject to frequent changes, necessitating the presence of multiple modalities. Despite these advantages, current multimodal neural networks remain surprisingly brittle when deployed. These multimodal neural models typically only provide robustness towards degraded input data for the specific metric of accuracy. Nevertheless, the dynamic environments in which they purportedly excel carry a host of additional variations beyond noisy input data. When existing multimodal networks are presented with such variations, they suffer from substantial degradation of sys-

tem performance – through either diminished accuracy or inefficient usage of computational resources. In fact, the increased complexity arising from multimodality actually results in unique vulnerabilities absent from unimodal networks.

These variations, broadly referred to as *inference-time variations*, arise in two distinct stages of system operation:

- **Deployment Time:** The system is moved from its training environment to a new operational environment. Examples of deployment-time changes encompass new sensor locations, environmental conditions, computing platforms, or even novel tasks and objectives.

- **Runtime**: During active operation in its deployment environment, the system must contend with variations and changes in the state of the world. Such variations typically occur on a faster time scale, possible necessitating per-sample adaptation. Examples include shifting modality quality, unstable computational resource availability, or intermittent data loss.

These inference-time variations expose a critical, underexplored gap in multimodal systems – the advantages of additional modalities come at the cost of additional avenues for deployment and runtime variations.

This dissertation explores minimizing the effect of inference-time variations on multimodal systems through a general framework – altering the internal weights and structure of the multimodal neural network in accordance with the current state of the world (Figure 1.1). This framework, implemented across a variety of inference-time variations and diverse datasets, results in *adaptive and world-aware multimodal networks* that retain high performance in realistic operational settings. Outlined in Figure 1.2, the dissertation examines such variations in successive steps of increasing model and task complexity, beginning with dedicated single-target localization networks and ending with general purpose multimodal foundation models. Here is an overview of the remainder of the thesis:

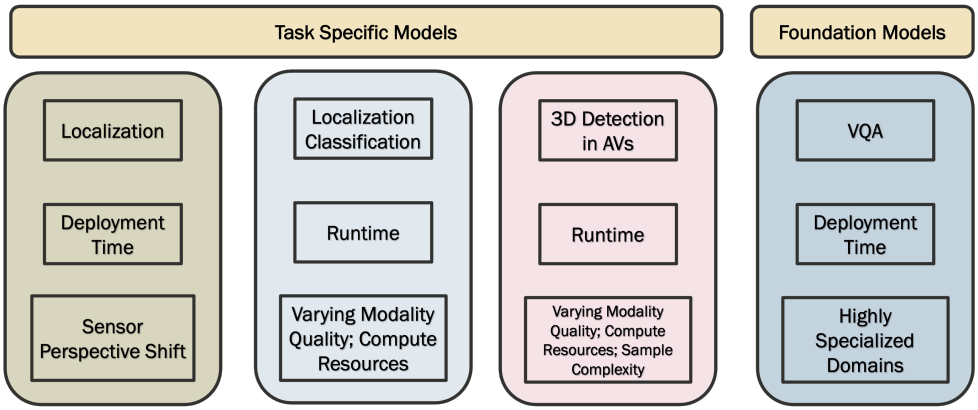

Figure 1.2: Thesis Outline. We explore increasingly complex models and tasks, ending with general purpose foundation models on VQA. We also consider several forms of deployment and runtime variations.

1.1 Sensor Perspective Shift

First, in Chapter 2, we begin with an exploration into deployment time sensor perspective shift in distributed multimodal localization systems. These localization systems typically fuse environmental sensor feeds from diverse spatial locations and modalities to estimate target positions in some coordinate system [23–25]. Owing to the high dimensionality and complexity of the input data, end-to-end neural networks are frequently used to perform the localization by training on a high quality dataset [26, 27]. One underexplored challenge involves sensor perspective shift – due to factors such as incidental contact, maintenance, or migration to a new environment, the sensors' position and orientation within a global coordinate system will inevitably change.

We find that existing networks are inherently brittle to deployment-time sensor perspective shift, where the internalized weights are unable to adapt to a new configuration of sensor poses. When presented with a novel configuration of sensor perspectives not present in the training dataset, existing localization systems can incur localization errors of over 1 meter, compared to only 20 cm of error on a previously seen viewpoint. Although there exists re-

cent work integrating sensor position into neural networks [28–33], they neglect to examine perspective invariance in the context of distributed, multimodal localization networks.

We propose *FlexLoc* [1], which incorporates auxiliary node position and orientation (pose) information directly into the multimodal neural network. We utilize *conditional neural networks* [34–36], which derive a portion of the main multimodal network's weights from auxiliary input (i.e., sensor pose). Through exposure to several configurations of sensor pose during training time, the network learns how pose impacts the final localization result, enabling it to generalize to an unseen configuration of sensor poses during inference time.

We introduce two conditional neural network mechanisms to achieve perspective invariance: Conditional 1D Convolution (CondConv) to transform extracted spatial features with sensor pose information, and a lightweight alternative, Conditional Layer Normalization (CLN), which derives normalization scaling and shifts directly from the sensor pose. Evaluated on the indoor GDTM [27] and a newly collected noisy outdoor dataset (GQ-Husky), FlexLoc achieves zero-shot generalization to completely unseen sensor perspectives, improving localization accuracy by nearly 50% over unconditional baselines with negligible parameter overhead. When viewed in the context of the general framework (Figure 1.1), FlexLoc alters the main multimodal network according to the current world state (through a conditional neural network), mitigating the effect of deployment time sensor perspective shift.

1.2 Runtime Variations in Single-Instance Tasks

Although FlexLoc demonstrated great utility under deployment-time variations, it is unclear whether the general framework of neural network adaptation is similarly helpful against runtime variations, which can occur on a per-sample basis. In Chapter 3, we perform a deep dive into two frequently occurring *runtime variations*, *varying modality quality of information* and *dynamic computational resource availability*. As the environment around the system changes, it invariably changes the quality of the input data, the effect of which varies

across modality. For instance, low-light due to cloudy weather or nighttime conditions can drastically reduce the utility of the camera modality, whereas other sensors such as radar or LiDAR are completely unaffected. Thus, the system must contend with variable modality QoI (quality-of-information) across its input samples. Moreover, the platform itself is not immune to variations either. Factors such as thermal throttling, multi-tenancy, or energy restrictions can impose limitations on the availability of computational resources for a given input sample.

The bulk of existing multimodal networks employ *static provisioning* [14, 16, 37–40], in which multimodal inputs proceed through a fixed architecture without any regard to the nature of the input. Regardless of the utility of the modality or its degree of corruption, the exact same computation will be performed for every input sample. The drawbacks of such an architecture are evident. In the case where a modality provides little information, valuable computational resources are wasted processing on its data, leading to inefficient usage of potentially limited computational resources. *We hypothesize that allocating computation in accordance to the relative modality QoI of each sample can substantially improve the model's performance.*

The statically provisioned nature of existing multimodal networks also inhibits adaptation to variable computational resource availability. Such networks greatly struggle when available computational resources dip below the minimum required to run the model in a desired time frame. We find that statically provisioned networks are not amenable to partial execution – executing a portion of the network due to insufficient resources greatly diminishes downstream accuracy. Though solutions exist in the *unimodal* case through Early-Exit [41–43] or subnetworks [44, 45], there are surprisingly few solutions in the multimodal setting. The few works exploring model adaptation in the multimodal setting [46–50] are incompatible with the combination of the two runtime variations.

We introduce ADMN [2], which addresses the runtime variations of *varying modality quality of information* and *dynamic computational resource availability* in multimodal neu-

ral networks. ADMN leverages a powerful multimodal controller that decides on an optimal allocation of resources across modalities according to real-time modality QoI and a user-specified resource budget. Thus, ADMN can adapt to a sudden decrease in one modality's quality by allocating resources away from it, and can also respond to changes in the platform's capabilities by allocating fewer total resources. ADMN is evaluated across several *single-instance tasks*, ranging from localization on the camera-depth *GDTM* dataset [27] to classification on the audio-visual *AVE* [51] dataset. In these tasks, the target cardinality is invariant across all samples ($N = 1$), which bounds the model's prediction space to a single, static vector of coordinates or category logits. ADMN approaches full-network accuracy while slashing latency by up to 60% and floating-point operations (FLOPs) by up to 75%.

1.3 Runtime Variations in Multi-Instance Tasks

While ADMN successfully addresses environment and hardware-driven variations through dynamic layer allocation, its reliance on single-instance samples raises concerns on applicability to complex tasks such as multi-object detection and localization [52–54]. Moreover, ADMN is unaware of the *complexity* of the input sample, and will always allocate the full budget of resources, lacking mechanisms to scale the computation according to sample difficulty. This is a critical source of inefficiency – easier samples that do not require the full budget will needlessly consume valuable computational resources. Finally, ADMN lacks an implementation on actual edge hardware with runtime accelerators such as TensorRT [55] that are critical when deploying modern machine learning stacks.

To bridge this gap, Chapter 4 introduces SWAN (Sample and World-Aware Multimodal Network) [3], which extends the core principles of dynamic adaptation into the highly complex realm of multi-object 3D detection for autonomous driving. Similarly to ADMN, SWAN employs a differentiable controller to allocate resources across modalities according to relative modality QoI. However, we find that ADMN's controller, relying on simple straight-

through [56] gradient propagation, struggles under complex multi-object detection scenarios. Thus, we propose to leverage *NeuralSort* [57] gradient propagation to better model resource allocation, and observe that it outperforms ADMN's straight-through controller.

Moreover, unlike ADMN, which always exhausts its allocated budget regardless of the sample difficulty, SWAN performs optimization *within a budget*. It leverages a feature-conditioned layer-skipping module and a spatial token pruner to scale down computation on "simpler" inputs. By explicitly modeling sample complexity alongside modality quality and hardware budgets, SWAN transitions the thesis framework from localized, single-instance tasks into a comprehensive paradigm for robust, adaptive edge perception in dynamic multi-object environments. SWAN outperforms related baselines by up to 9 NDS and 14.5 mAP while achieving up to a 49% reduction in FLOPs compared to a fully-provisioned network.

1.4 Transitioning to Foundation Models

Thus far, the thesis has revolved purely around task-specific multimodal networks – dedicated neural models trained and tested on one singular task, whether it is object localization, classification, or multi-object detection. In recent years, the paradigm has shifted away from task-specific networks towards general purpose multimodal foundation models [38, 58–60], which are trained on internet-scale data [61, 62]. With a singular set of weights, such models can be deployed on a breadth of diverse tasks, ranging from machine translation [63], visual question answering [64–66], summarization [67, 68], sentiment analysis [69], or object detection [70]. We focus specifically on one form of multimodal foundation model, the large vision language model (LVLM), which combines language understanding with the ability to comprehend visual content.

Despite the general-purpose nature of these LVLMs, they degrade in performance when tested in highly specific domains that are underrepresented in the training dataset [71–73]. Particularly, when subjected to curated vision-centric tasks, LVLMs can underperform

human accuracy by more than 50% [74], necessitating *deployment-time adaptation* to these edge domains and tasks. Existing methods of deployment-time adaptation involve fine-tuning the model on domain-specific data through modifications in the LLM [75] or projector [66,71]. Unfortunately, existing techniques can suffer from the following issues:

- **High Computational Requirements:** All existing methods require the entire LVLM to be loaded into GPU memory regardless of which modules are frozen or unfrozen. The forward and backward passes must traverse through the LLM, which can be billions or even trillions of parameters large [60]. This places an unreasonable strain upon computing infrastructure, and restricts model adaptation to only the few with sufficient resources.
- **Degradation of Language Capabilities:** Fine-tuning the model on domain-specific data runs a high risk of model degradation. The LVLM is likely to overfit to the distribution of the fine-tuning dataset, memorizing superficial patterns and resulting in a degradation of linguistic capabilities [76].

In Chapter 5, we propose CRAFT [4], an highly efficient method of deployment time LVLM adaptation to novel vision-centric domains. In CRAFT, we fine-tune *only the discrete visual encoder* of the LVLM, ensuring that the linguistic capabilities are untouched. Moreover, we address the challenge of high computational requirements by proposing a novel *surrogate model fine-tuning* strategy. During surrogate fine-tuning, the large LLM is replaced with a significantly smaller, frozen surrogate that allows the vision encoder to learn in the presence of language supervision. At inference time, the surrogate is replaced with the original backbone, yet the improved vision encoder's capabilities remain. Across 10 benchmarks, CRAFT improves domain-specific performance by an average of 13.51% points while preserving strong linguistic capabilities, outperforming related baselines in accuracy and efficiency.

Chapter 6 summarizes the key findings of this dissertation, and proposes some future research directions centered around adaptive multimodal neural networks. Finally, Chapter 7 outlines the contributions of each author on the main works presented in this thesis.

CHAPTER 2

FlexLoc: Conditional Neural Networks for Zero-Shot Sensor Perspective Invariance in Object Localization with Distributed Multimodal Sensors

Localization is a critical technology for various applications ranging from navigation and surveillance to assisted living. Localization systems typically fuse information from sensors viewing the scene from different perspectives to estimate the target location while also employing multiple modalities for enhanced robustness and accuracy. Recently, such systems have employed end-to-end deep neural models trained on large datasets due to their superior performance and ability to handle data from diverse sensor modalities. However, such neural models are often trained on data collected from a particular set of sensor poses (i.e., locations and orientations). During real-world deployments, slight deviations from these sensor poses can result in extreme inaccuracies. To address this challenge, we introduce `FlexLoc`, which employs *conditional neural networks* to inject node perspective information to adapt the localization pipeline. Specifically, a small subset of model weights are derived from the node poses, enabling accurate generalization to unseen perspectives with minimal additional overhead. Our evaluations on a multimodal, multiview indoor tracking dataset showcase that `FlexLoc` improves the localization accuracy by almost 50% in the zero-shot case (no calibration data available) compared to the baselines. The source code of `FlexLoc` is available in `https://github.com/nesl/FlexLoc`.

2.1 Introduction

Modern robotic and automation applications are heavily dependent on the ability to localize and track a target with a high degree of accuracy. For instance, the emerging warehouse automation industry (estimated at 23 billion USD [77]) requires knowing the location of not only the mobile robots within the warehouse, but also any packages or human beings [78]. Localization information increases productivity by enabling coordination among human and robotic workers while providing constant feedback to mobile robots to minimize positioning errors. The recent rise in smart building technology also drives the need for accurate localization. A delivery or cleaning robot within a smart building can rely on the building's localization infrastructure to guide it through a complex floor map [79]. Finally, various healthcare applications are also greatly interested in the location of a human subject, deriving behavioral biomarkers from subject location to provide early warnings for chronic diseases [80].

With these numerous groundbreaking applications all depending heavily on accurate localization, recent works have explored the utilization of both *localization infrastructure* and *egocentric localization* technologies. Localization infrastructure integrates sensors into the environment, contrasting with egocentric localization that relies upon an on-board suite of sensors to perform techniques such as SLAM [81]. Egocentric localization techniques have been well-studied and achieve exemplary performance [82–84]. Nevertheless, egocentric sensing is susceptible to occlusions and SWaP constraints (size, weight, and power) that limit the ability to carry sensors. Thus, we primarily focus on infrastructure-based localization, which does not impose requirements upon the targets. Additionally, unlike egocentric localization techniques that require instrumenting the subject, infrastructure-based localization does not require any degree of cooperation from the localization targets, resulting in greater applicability.

In this work, we explore infrastructure-based object localization with a set of distributed

(multi-view) and multimodal sensors. Existing localization infrastructures typically incorporate multiple sensor nodes with partially overlapping sensor perspectives as a **multi-view** system to improve localization accuracy and expand sensing coverage. These systems estimate the target's location in a global coordinate system with a variety of sensors, ranging from camera [26], UWB radar [85], or acoustic sensors [86]. However, in real-world scenarios, the perception of a target is heavily influenced by environmental factors, and a single sensor modality is frequently incapable of capturing sufficient information for robustness. For instance, RGB cameras can fail to provide relevant information under low-lighting conditions. Therefore, several new **multimodal** sensing systems have been proposed to enhance the reliability of indoor localization by fusing information across multiple distinct sensors [23–25]. For instance, *Zhao et al.* [24] enhances camera-based indoor localization with inertial measurement units (IMUs) and WiFi CSI signals.

To effectively reason over the vast amount of multi-view and multimodal sensor data, localization systems use large-scale end-to-end deep learning models to predict target locations. These data-driven models not only reason about the information contained within a particular modality (*intra-modal*), but also learn to fuse information across multiple sensor modalities (*inter-modal*) to achieve accurate localization [87]. End-to-end deep learning models can be primarily divided into two approaches [88] - **early-fusion** (feature-level fusion) and **late-fusion** (result-level fusion). Early fusion architectures first extract intermediate features from each sensor with unimodal feature extractor networks, after which the features are fused together and used to generate one prediction [89]. Meanwhile, late fusion approaches predict a result (e.g., target location) individually for each sensor and generate a final prediction by reconciling all the results (e.g., using a Kalman Filter). While late-fusion approaches can minimize the communication overhead in a distributed sensing system, early-fusion approaches often achieve superior performance, as the model facilitates the early collaboration of sensors with different strengths [88].

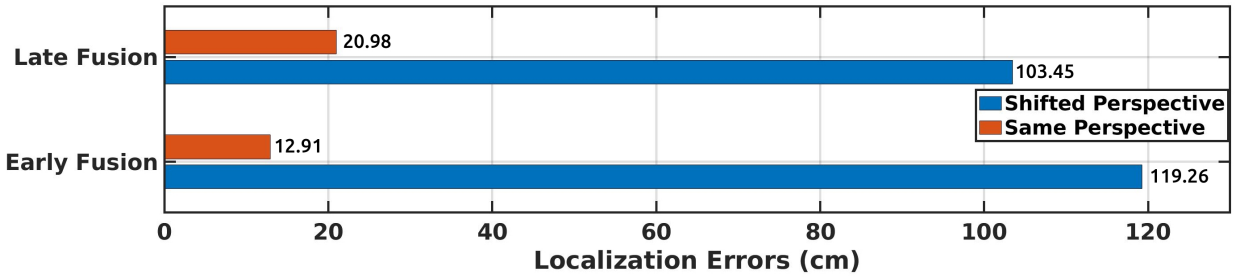

Figure 2.1: Performance degradation due to sensor perspective shift.

2.2 Problem and Motivation

Despite the high accuracy and increased robustness towards environmental factors provided by state-of-the-art multimodal learning-based systems, current approaches neglect to consider the critical issue of **sensor perspective shift**. Deep neural networks strive to learn a set of network weights that maximize the performance on a particular training dataset [90]. Consequently, the networks learn to output global coordinate predictions for the particular set of sensor positions and orientations (i.e., sensor perspectives) present within the training dataset. If the sensor perspectives do not change between training and test-time, the network will continue to accurately localize the target within the global coordinate system. When the model receives data from an *unseen configuration of sensor perspectives*, however, it fails to map the data from these new perspectives into the global coordinate system, as the weights are tuned for only the specific perspectives present within the training dataset. We illustrate this problem here through a motivational study.

We leverage GDTM [27], a multi-hour dataset we collected using distributed multimodal sensor nodes capturing a remote controlled car driving around an indoor track. In this dataset, we introduce diversity in sensor perspectives by altering the position and orientation of sensor nodes between trials. We train both an early-fusion and a late-fusion model on data from one sensor perspective, with the results shown in Figure 2.1. When we evaluate both models on new data from the *same* viewpoint, we achieve reasonable localization per-

formance, with early-fusion outperforming the late-fusion baseline. However, both models fail spectacularly when evaluated on sensor data collected from a new sensor placement, with average distance errors exceeding >1 m.

This challenge of sensor perspective shift poses a significant barrier to the practicality of localization systems. Not only can sensor pose gradually shift over time, factors such as relocation after maintenance are inevitable during the deployments. Attempting to train a new system from scratch as the sensor perspectives are changed is highly impractical due to the large amount of data required. Thus, we must **design models that can automatically adapt to sensor perspective shift without any training on the new, unseen perspective** (i.e., in a zero-shot manner). GDTM [27] attempted a solution with more than 30 cm of error on new perspectives with a suboptimal late-fusion approach: localizing the target at each distributed sensor within its *local coordinate system* and applying a hard-coded coordinate transform into global coordinates.

2.3 Proposed Method

To address this challenge, we propose `FlexLoc`, a new multimodal multi-view sensor fusion approach. FlexLoc leverages an end-to-end early fusion architecture, and is designed to address the sensor perspective shift of nodes within a distributed multimodal sensor network. As shown in Figure 2.2, the main idea involves injecting node orientation and position (pose) information as an auxiliary input into the localization network. During the training process, the model is exposed to a large variety of viewpoints, each with associated pose information, enabling it to learn how the each sensor's pose affects the global coordinate prediction. Then, during test time, the system can measure the current pose (through self-localization [91] or marker-based [92] techniques) to make accurate predictions in presence of unseen sensor viewpoints. Note that `FlexLoc` concerns itself only with *localization* rather than *tracking*, where we localize the target object for every distributed multimodal sample, rather than

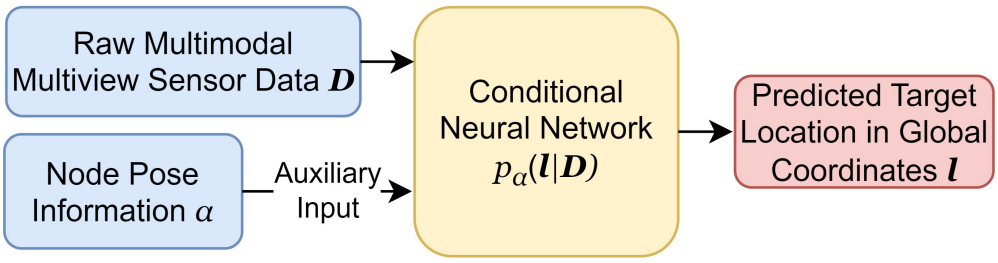

Figure 2.2: Illustration of our key idea to enable perspective invariant object localization. Unlike existing approaches that are unaware of sensor perspectives, we inject node pose information into the network through conditional neural networks.

integrating temporal information into the network itself [93].

Specifically, `FlexLoc` builds upon a feature-level (early) fusion architecture primarily consisting of backbones, adapters, a transformer encoder, and an output head. From a high level, the backbones (implemented using ResNets and ViTs [94]) extract features from raw sensor data. The features are mapped to fixed-sized vectors by the adapters and then aggregated. A multi-layer transformer encoder then processes the aggregated features. Finally, the output head yields the predicted global coordinates. To inject the node pose information, we design and insert **Conditional Neural Networks** into the aforementioned architecture. The key idea is to augment the main neural network with a small subset of additional weights conditioned on auxiliary inputs (sensor pose in our scenario). These dynamic weights are generated from the auxiliary input using a lightweight neural network during both training and inference-time. After model training, the conditional neural networks can leverage supplementary auxiliary information during inference-time, thereby achieving greater robustness over the unconditional models relying solely on the original sensor data input.

Conditional neural networks have elevated the performance of applications such as Visual Question Answering or Speech Enhancement [34–36]. With `FlexLoc`, we seek to transfer this idea to the sensor perspective shift problem, which contains increased complexity due to the fragility of the learned 3D spatial relationships. Specifically, we create two variants

of conditional neural networks. First, drawing inspiration from the multiplicative nature of coordinate transforms, we propose a *Conditional 1D Convolution* (CondConv) whose convolutional weights are conditioned on the sensor pose. These CondConv layers are inserted after the adapter to transform the extracted sensor features with the sensor pose information. Second, we propose a lightweight design for injecting the pose information through *Conditional Layer Normalization* (CLN), given the pervasive existence of normalization layers in the backbone models. We replace the learnable parameters in the normalization layers, i.e., scale (γ) and offset (β), with values derived from the pose. The lightweight CLN approach is the better choice when computing resources are limited, while the CondConv method performs better in various complex scenarios.

To evaluate `FlexLoc`, we leverage our previously collected multi-modal multi-view dataset for indoor geospatial vehicle localization [27]. Our dataset contains a total of nine-hours of data collected from three nodes that are equipped with four sensor modalities (RGB camera, depth camera, mmWave radar, and microphone), with 22 groups of unique sensor node perspectives (one perspective is an arrangement of 3 nodes). Our results in Section 2.9 show that our approach can achieve good zero-shot performance on data collected in unseen perspectives while also exhibiting test-time robustness to different combinations of sensor modalities.

2.4 Contributions

In summary, we make the following key contributions:

- We propose two methods for improving test-time performance of localization models on unseen sensor perspectives: Conditional 1D Convolution and a lightweight design based on Conditional Layer Normalization. This work constitutes the first exploration into applying conditional neural networks to address test-time robustness in localization models.

- We independently integrate the proposed two conditional methods into an early fusion architecture for multimodal multi-view learning. We then perform a detailed study investigating the performance of the two methods.
- `FlexLoc` involves a minimal number of additional parameters (at most 0.2% of model parameters), and its simple implementation allows for easy integration within existing neural networks
- `FlexLoc` shows good zero-shot generalization performance to various perspectives while also retaining good performance in the absence of various sensor modalities.
- We collect a new noisy outdoor dataset with imprecise node pose estimations (*GQ-Husky*), and showcase that `FlexLoc` outperforms baselines on both the noisy, outdoor *GQ-Husky* and the clean, indoor *GDTM* [27] dataset.

2.5 Related Work

2.5.1 Multimodal Deep Learning

Processing multimodal data through neural networks is a well established field broadly separated into two methods: early (feature-level) fusion and late (decision-level) [14]. Early Fusion techniques combine information from different modalities at the beginning of the network, processing all modalities together to make one unified prediction [95, 96]. In contrast, late fusion approaches make an independent per-modality prediction and then unify all the independent predictions. Some works [97] add a third category of hybrid fusion where each modality is independently processed into a feature embedding before aggregated. We treat hybrid fusion as an extension of the early-fusion technique. State-of-the-art techniques commonly apply early fusion over late fusion due to its ability to facilitate greater information exchange among modalities [14, 27, 88]. *Nagrani et.al* devised a transformer-based audiovisual early fusion approach, introducing a set of multimodal bottleneck tokens to facilitate

fusion and promote condensing of information [14]. Similarly, *Ho et. al* employed early fusion for object localization by utilizing independent neural networks to extract embeddings from each modality, and fusing the embeddings with a transformer decoder [27].

2.5.2 Multimodal Learning for Localization and Tracking

Machine learning algorithms based on handcrafted features and deep neural networks have been applied to indoor localization and tracking applications. For example, *Samplawski et al.* [26] used three different cameras to predict the distribution of the object's location, and designed a *multi-view* late fusion neural network to fuse the prediction of different cameras. Although they demonstrated excellent tracking results, the solution utilizes only RGB cameras and will fail under low-lighting conditions. Therefore, several new multimodal sensing systems have been proposed to leverage the strengths of different sensor modalities to enhance the reliability of localization [23–25]. For instance, [23] proposed a method to locate an indoor mobile phone user based on multi-view and multi-modal measurements, but incurs a 1-meter error. Recently, *Chen et al.* proposed a self-supervised learning approach that integrated spatial and structural information into the pretraining process to encompass factors such as node pose. However, this self-supervised method must be fine-tuned on a small amount of labeled data from the target perspective, whereas `FlexLoc` is a purely zero-shot approach [31].

Aside from [27] and [31], there exists minimal work in the field of multimodal and multiview localization through localization infrastructures. Existing work typically revolves around either unimodal multi-view localization infrastructures [98, 99], multimodal single-view localization infrastructures [100,101], or localization through egocentric sensors [52,53]. The few multimodal and multiview works [102,103] do not contain shifting sensor viewpoints or a large diversity of multimodal sensors. Thus, we were unable to perform evaluations on different datasets or find similar architectures that attempted to address the challenge of perspective shift in multimodal multi-view localization.

2.6 Perspective-Aware Neural Networks

Although there is a scarcity of distributed, multimodal localization networks that tackle the challenge of perspective invariance, there are several works that attempt to incorporate node position and orientation into neural networks. *Yang et al.* incorporates camera perspective information into the video generation pipeline to allow users to control camera movement and object motion, while well-known image synthesis techniques such as NERF [28] and 3DGS [29] create novel images from camera pose. Aside from synthesis, [30] concatenates camera pose information to the input to improve monocular depth estimation. Finally, perspective information has also been integrated into foundation model pretraining in [31–33]. These works reveal the increasing interest in incorporating sensor position into neural networks.

2.7 Conditional Neural Networks

Conditional neural networks incorporate additional auxiliary information during the training of the network to enhance the robustness or accuracy of predictions, demonstrating excellent performance in applications such as Visual Question Answering or Speech Enhancement [34–36]. For example, De Vries et. al [34] introduces conditional batch normalization to modulate convolutional feature maps of visual processing with a linguistic embedding, which significantly improves the performance of visual question-answering tasks. Zhang et. al [35] integrates a deep autoencoder with neural noise embedding for speech enhancement through conditional layer normalization, which improves the generalization of the trained model to various noisy speech inputs. Therefore, by training with a small portion of additional weights conditioned on the auxiliary input, the conditional neural networks improve the model accuracy and generalization ability. `FlexLoc` is the only work that applies conditional neural networks to the task of perspective invariant localization.

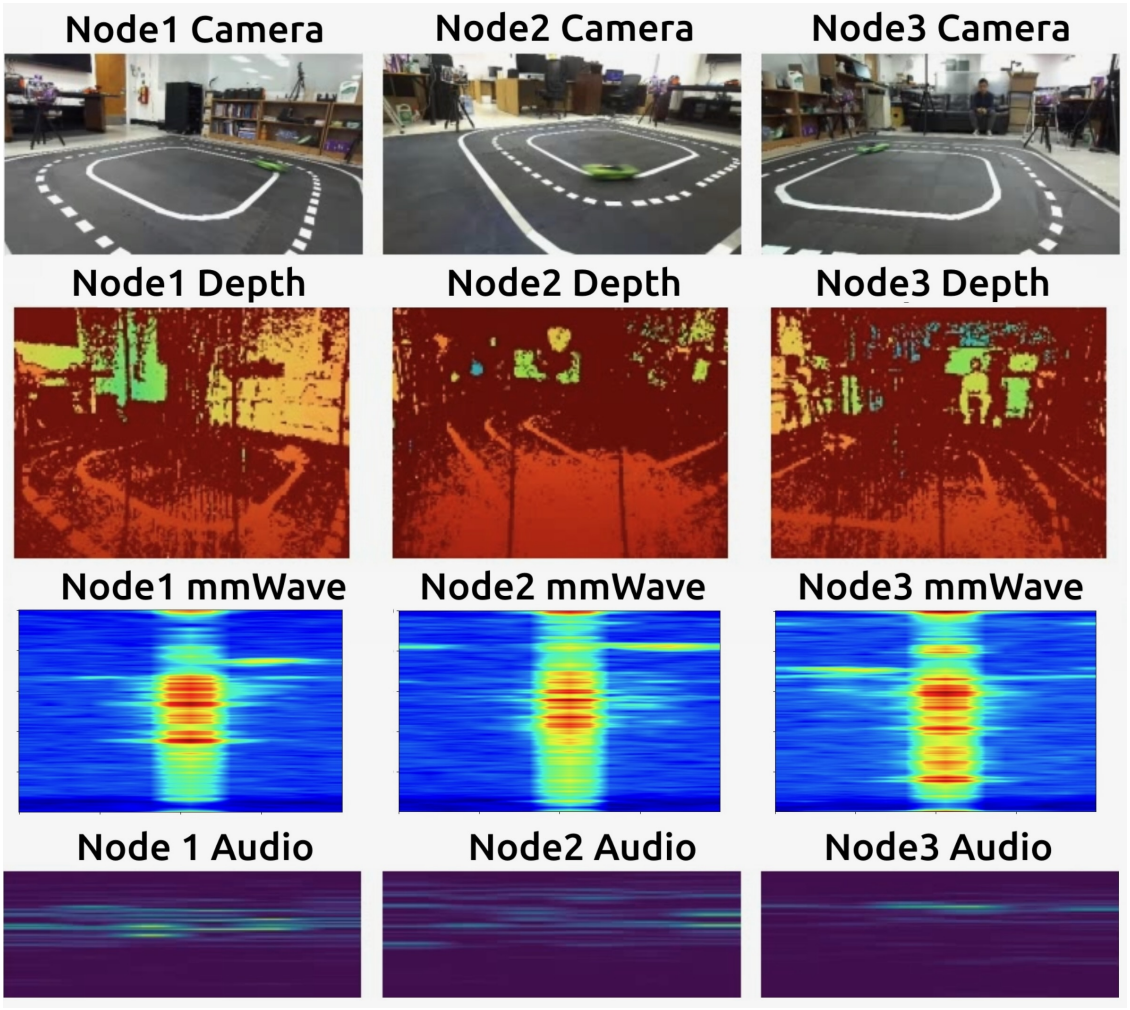

Figure 2.3: Examples of multi-modal sensor data collected by three nodes.

2.8 Methodology

2.8.1 Multimodal Multiview Datasets

One significant barrier to investigating the sensor perspective shift problem is the scarcity of publicly available multimodal, multiview datasets not only containing shifting perspectives of sensors, but also with ground truth sensor pose information. Thus, we collect and release two datasets (one indoor, one outdoor) to facilitate greater research into this field.

2.8.2 Indoor GDTM Dataset

We collected a nine-hour multimodal indoor tracking dataset, namely *GDTM* [27], to research the sensor perspective shift problem. *GDTM* is composed of three data collection nodes placed at various locations around an indoor race track to localize a remote-controlled car. Each node is composed of camera, depth, mmWave, and audio modalities (Fig. 2.3). Additionally, the position and orientation of these nodes are adjusted during data collection to generate a large diversity of sensor viewpoints (perspectives of the three nodes). We utilize this variety of viewpoints to develop our perspective invariant model. *GDTM* contains 22 unique viewpoints, allowing us to construct a dataset containing 13 views in the training dataset (105 minutes), 4 views in the validation set (35 minutes), and 5 views for testing (55 minutes). All three sets contain both circular motions and random trajectories of the robot car moving in the scene.

We quantified the similarity between perspectives V_1 and V_2 by computing the *structural similarity index measure* (SSIM) of the camera data: (1) we take a frame of camera data from each sensor node in V_1 and V_2, resulting in three images in each view. (2) We compute pair-wise SSIM $\mathbf{S} \in \mathrm{R}^{3\times3}$. (3) We take $\max(\mathbf{S})$, the SSIM of the most similar image pair, as the similarity between V_1 and V_2. We introduce a two-stage procedure to ensure that the viewpoints used for testing are different from those for training and validation. First, we calculated the similarity score between V_{train}, V_{val}, and V_{test} to ensure they are all smaller than 0.60. Moreover, we manually performed a visual inspection to double-check.

2.8.3 Outdoor GQ-Husky Dataset

One significant drawback of the *GDTM* dataset is that the high quality data may not be representative of a realistic deployment scenario. The indoor, laboratory environment of *GDTM* ensured consistent lighting, minimal interference, and little scene variation between trials. These ideal characteristics are difficult to guarantee in a real-world setting, and thus

`FlexLoc` should be evaluated on noisy data to showcase its generality. Furthermore, the ground truth node pose in *GDTM* is provided by a millimeter-accurate motion capture system. These motion capture systems are expensive and difficult to set up, raising the question of whether `FlexLoc` can function with less precise self-localization techniques (e.g., AprilTags [92]).

In order to address these concerns, we collect the outdoor *GQ-Husky* dataset and evaluate `FlexLoc` on it. *GQ-Husky* was collected at the ARL Graces Quarters site with the same suite of sensors as *GDTM* (camera, depth, mmWave, and audio modalities with three identical nodes). However, the data collection was performed outdoors in a roughly 10 m $\times$ 10 m space with a Husky robot (990 mm $\times$ 698 mm $\times$ 381 mm) [104] as the tracking target. For each trial, we would change the orientation and position of each node and capture distributed, multimodal data of the Husky robot moving throughout the scene, with potential interfering objects moving in the background. Data collection occurred over three separate days with a variety of lighting conditions (early morning, afternoon, evening). Since it was impossible to set up a motion capture system in this setting, we utilized AprilTags [92] for self-localization. The AprilTag was placed in the center of the scene, allowing each node to calculate their orientation and position in the AprilTag coordinate frame from a single RGB image. Additionally, the lack of a motion capture system also complicated acquisition of the Husky robot's ground truth location. We leveraged a high-precision GPS to provide ground truth information on the Husky robot in world latitude and longitude coordinates, which were post-processed and converted into the AprilTag's coordinate frame.

The *GQ-Husky* dataset contains roughly 50 minutes of data from 26 different sets of node poses (recall each set is an arrangement of three nodes). We split *GQ-Husky* into a training split with 22 viewpoints, and a testing split with 4 different perspectives, allowing us to evaluate whether `FlexLoc` would retain its performance with imprecise self-localization and noisier scene data. Note that due to the large experiment scene (100 m^2 for *GQ-Husky* vs 9 m^2 for *GDTM*) , and a significantly larger tracking target, it is expected for the localization

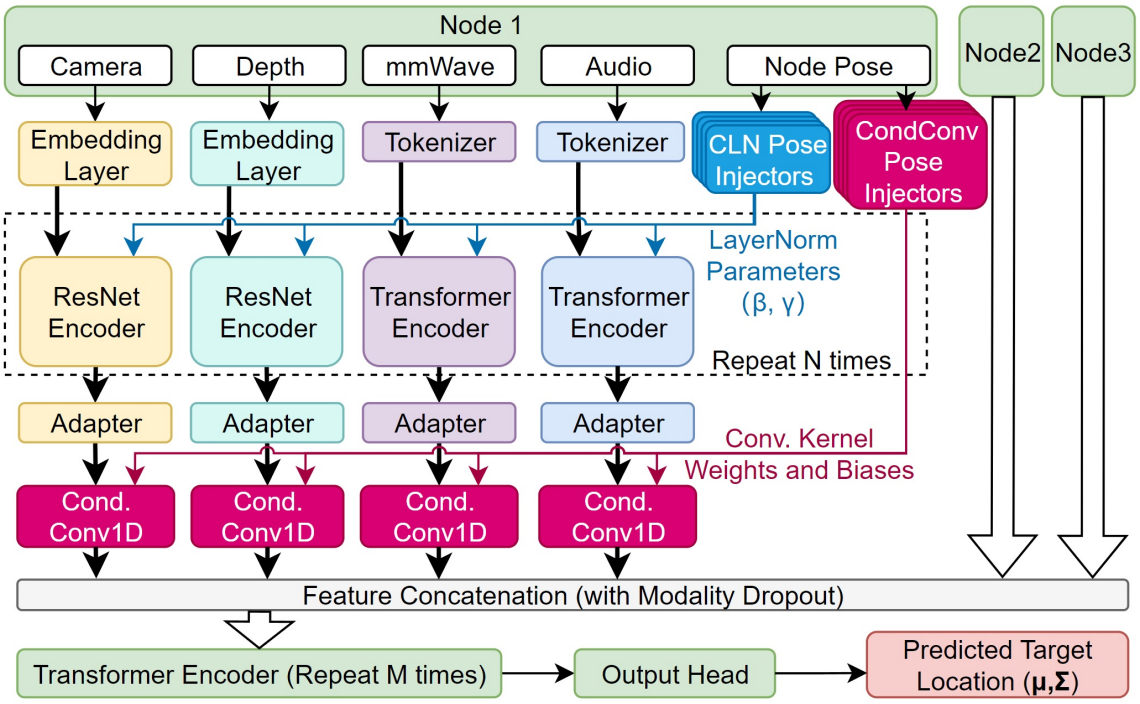

Figure 2.4: Complete `FlexLoc` architecture containing Conditional Convolution and Conditional Layer Normalization. Conditional Convolution derives its 1D convolutional kernel weights from the sensor pose and transforms the extracted features of the sensor data. Conditional Layer Normalization is a more lightweight design integrated into the backbones, where we replace the learnable parameters γ and β with values derived from sensor pose.

errors to be larger in *GQ-Husky* when compared to GDTM.

2.8.4 Overall Model Architecture

Figure 2.4 showcases the overall architecture of `FlexLoc`, which outputs predictions of the target location in pre-established global coordinates with the help of sensor pose information. First, `FlexLoc` utilizes *backbones* shared across all the nodes to process each modality independently with the goal of extracting key information (embeddings) pertinent to the localization task. The *adapters* (shared among nodes) accept these embeddings from the backbones as input and perform modality-specific processing to transform the distinct

feature vectors into vectors of the same dimension. The output of the adapters are then processed by *Conditional Convolution* to introduce pose information. Afterwards, a *transformer encoder* performs cross-modal information fusion. Finally, an *output head* utilizes the information in the joint embedding to make a unified prediction of the target's location in the pre-established global coordinate system. As an alternative to Conditional Convolution, we also propose a lightweight conditional neural network based on Conditional Layer Normalization, which is integrated into backbones to incrementally introduce pose information into the final embeddings of different sensor modalities (drawn in the same figure to save space).

Specifically, our backbones are composed of either Residual Convolutional Networks (ResNet) [105], or transformer encoders following the work on Vision Transformers [94]. ResNet backbones are a good candidate for processing visual data (camera and depth information) due to their ability to extract information about key objects in the scene while ignoring irrelevant background information. In contrast, when processing mmWave and audio sensor data, we employ a simpler stack of transformer encoders due the relative lack of background content.

Moreover, Conditional Convolution or Conditional Layer Normalization are strategically inserted into this architecture to incorporate pose information. Conditional Convolution operates on the extracted embeddings containing dense information regarding the object's position, injecting the sensor pose information through a series of 1D convolution operations. In contrast, Conditional Layer Normalization serves as a lightweight method of incorporating pose information into the early stages of data processing. We replace the traditional Batch Normalization and Layer Normalization components within each backbone with Conditional Layer Normalization that derives its parameters from the sensor pose information. We evaluate the performance of the two conditional methods within Section 2.9.

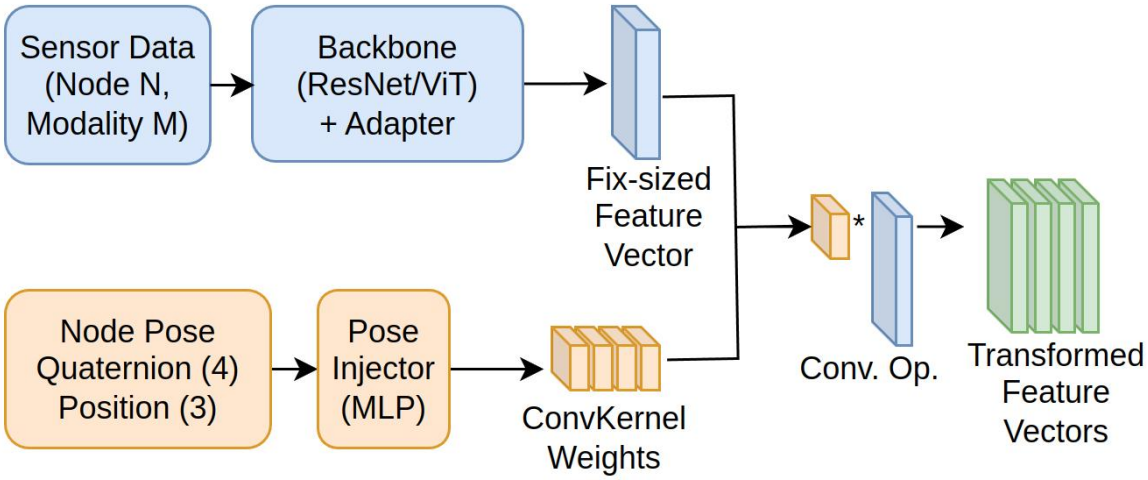

Figure 2.5: Implementation and Integration of Conditional 1D Convolution.

2.8.5 Conditional Convolution

To effectively inject the sensor pose information, we explore transforming the feature vectors of a particular node (outputs of the adapters in Fig. 2.4) with a *conditional 1D convolution*. Given an input vector $\mathbf{x} \in \mathbb{R}^N$ from the intermediate features of M sensor modalities $\mathbf{X} = \{\mathbf{x}_0, ..., \mathbf{x}_{M-1}\} \in \mathbb{R}^{M \times N}$, 1D convolution with K kernels of size S generates an output $\mathbf{y} \in \mathbb{R}^{K \times N}$ according to Equation 1, where $*$ refers to the convolutional operator,

$$\mathbf{y}_k = b_k + \mathbf{x} * \mathbf{w}_k, \quad \mathbf{w_k} \in \mathbf{W}, b_k \in \mathbf{b}. \tag{2.1}$$

Conditional 1D convolution replaces the learnable weights $\mathbf{W} \in \mathbb{R}^{K \times S}$ and biases $\mathbf{b} \in \mathbb{R}^K$ with parameters generated from the auxiliary input (sensor pose), fusing the information within $\mathbf{X}$ with additional context. Figure 2.5 provides a detailed snapshot of how conditional convolution is integrated into the overall system. The node pose is represented by a set of 7 values, including four quaternion values representing rotation and (x, y, z) position. These values are expanded by a series of linear layers into a higher dimension, from which we extract the weights and biases for the K convolutional kernels of length S. These adaptive kernels are convolved with the feature vector to generate K new feature vectors with position information.

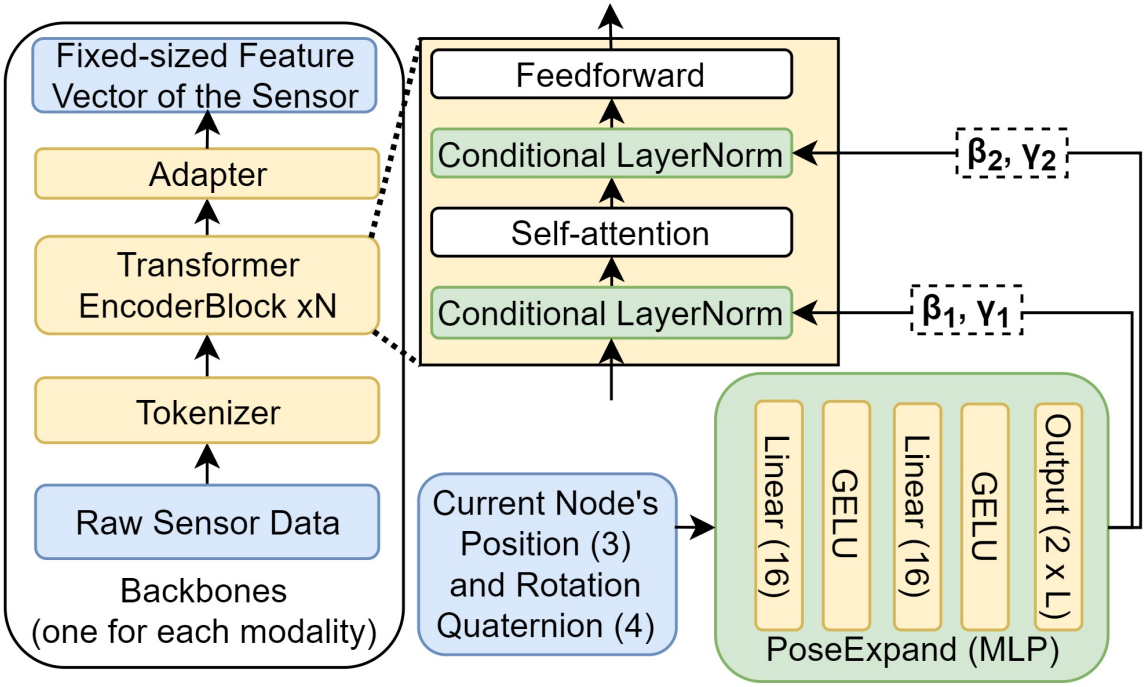

Figure 2.6: Implementation and Integration of Conditional Layer Normalization.

2.8.6 Conditional Layer Normalization

Layer Normalization [106] is a feature normalization technique that reduces training time and improves generalization of neural networks [107]. It performs normalization with the mean and standard deviation computed over elements of the same batch, while also introducing a learnable *per element scale and bias* to improve performance. Equation 2 outlines applying Layer Normalization to a single vector $\mathbf{x} \in \mathbb{R}^N$ from a batch of intermediate feature vectors $\mathbf{X} = \{\mathbf{x}_0, ..., \mathbf{x}_{B-1}\} \in \mathbb{R}^{B \times N}$:

$$\mathbf{y} = \boldsymbol{\gamma} \cdot \frac{\mathbf{x} - \mu}{\sigma} + \boldsymbol{\beta}, \;\; \mu = \frac{1}{N} \sum_{j=0}^{N-1} x_j, \;\; \sigma = \sqrt{\frac{1}{N} \sum_{j=0}^{N-1} (x_j - \mu)^2} \tag{2.2}$$

with $\boldsymbol{\gamma} \in \mathbb{R}^N$ and $\boldsymbol{\beta} \in \mathbb{R}^N$ acting as *per element* scale and shift operations.

We implement *Conditional Layer Normalization* (CLN) by replacing the learnable $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ parameters with values derived from the node's pose information. Figure 2.6 illustrates how the layer normalization blocks contained within each transformer encoder of the backbone

are replaced with their adaptive counterparts. This same process occurs with the Batch Normalization layers in the ResNet. These adaptive weights are derived from a *separate* set of linear layers (PoseExpand). PoseExpand encodes the sensor pose into an vector of dimension 16, from which the unique γ and β value of each Layer Normalization block are obtained. Note that to reduce the number of additional parameters, we use *one single* γ scalar and *one single* β scalar to modify the entire embedding instead of performing per element scale and shifts. Furthermore, in vanilla LayerNorm, the $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ parameters are shared across the batch B. In CLN, however, data entries within a batch are not guaranteed to come from the same sensor viewpoint. As a result, we compute one γ_i and one β_i scalar for each entry i in the batch. Thus, with an input of $\mathbf{X} \in \mathbb{R}^{B \times N}$, every element of x_{ij} from $\mathbf{X}$ will be normalized by μ_i and σ_i , then scaled and shifted by γ_i and β_i.

2.8.7 Training Strategies

2.8.8 Backbone Pretraining

We utilize a set of pre-trained weights to imbue our backbones with an understanding of how to process the input sensor data. In the camera and depth ResNet-18 backbones, we load a checkpoint pre-trained on ImageNet1K. As we designed the Transformer Encoder backbones ourselves, we chose to pre-train those backbones on 5 minutes of data from a viewpoint in the training dataset. The pre-training process involved separately training each modality's backbone to ensure it has an understanding of how to identify the car from sensor data.

2.8.9 Sensor Dropout

Furthermore, we also utilized a *modality dropout* process during model training to fully realize the robustness of the rich multimodal input. We discovered that training the model with all modalities active resulted in an over-reliance on the most informative modalities. To combat this, we established a 50% chance for any modality to be "dropped-out" in a

	View 1		View 2		View 3		View 4		View 5		Average	
Model Architecture	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
FlexLoc (CondConv)	**17.10**	0.96	**14.45**	0.68	**13.30**	1.06	**11.89**	1.07	**14.67**	2.03	**14.28**	1.16
FlexLoc-light (CLN)	20.50	0.55	24.88	2.78	21.98	4.04	24.38	11.90	18.45	2.33	22.04	4.32
Baseline-early (Brute Force)	32.61	3.57	24.66	3.27	26.52	3.40	24.25	7.38	26.29	5.78	26.86	4.68
Baseline-late (Hard-coded Transform)	33.69	4.97	25.36	1.16	20.93	0.05	32.97	1.23	24.71	1.94	27.53	1.87

Table 2.1: Euclidean Distance Error (in cm, averaged across 3 seeds), `FlexLoc` vs baselines

given batch during training, effectively forcing the model to learn with various subsets of modalities. We analyze this in greater detail within Section 2.9.3.

2.9 Evaluations

We evaluate a variety of models on the multiview dataset (Section 2.8.1) to showcase the performance benefit of our conditional networks. We are only concerned with predicting the target's 2-dimensional coordinate, as the target's elevation within the global coordinate system is unchanged. Thus, the primary metric for quantifying the performance is *Average Euclidean Distance* between the predicted location and the ground truth. We evaluated all our models on five *unseen sensor perspectives that were not present in training.*

2.9.1 Comparison with Existing Methods

We test our two conditional models and compare them against two baseline models:

1. *Unconditional Early Fusion*: This is essentially `FlexLoc` with no conditional layers.
2. *Local Coordinate Late Fusion* [27]: Each modality and node pair localizes the object in the node's local coordinate system, after which the predictions are transformed into a global coordinate system via a known transform matrix and merged with a Kalman Filter.

Table 2.1 showcases that all the conditional models perform better against the baseline models. In particular, Conditional Convolution (CondConv) achieves the best performance with at least 46% improvement over the baseline methods. In fact, the localization error of CondConv is comparable to models that are *trained and tested on the same viewpoint*. In Section 2.1, we trained and tested an early-fusion and late-fusion model on one single sensor viewpoint, achieving 12.91 cm and 20.98 cm of error, respectively. CondConv achieves an impressive 32% improvement over the single-perspective late fusion model and comes within 10% of the single-perspective early fusion model. These results exemplify the robustness of CondConv towards completely unseen sensor perspectives. As this is our best performing model, we term `FlexLoc` as the network containing only CondConv.

While Conditional Layer Normalization (CLN) does not achieve the same level of performance as Conditional Convolution, it can still provide a modest 17.9% improvement over the baseline early fusion model lacking conditional layers. We hypothesize that its worse performance compared to CondConv is due to the more challenging task of incorporating pose information into the backbone itself. One key advantage of CLN lies in its lightweight nature, where it employs far fewer parameters than CondConv (analyzed in Section 2.9.4). Thus, we term CLN as `FlexLoc`-Light. It is also worth noting that `FlexLoc`-Light encounters an unlucky seed where the model performance degrades, which contributed to the high standard deviation, highlighting that CLN may exhibit less stable training than CondConv.

Figure 2.7 showcases a visual depiction of each method's localization error in a CDF. We observe a significant boost in performance between `FlexLoc` and the other methods, while `FlexLoc`-Light maintains a smaller yet notable improvement over the two baseline methods. Another sizable advantage of `FlexLoc` lies in its low 90% error at only 25 cm, which is 20 cm less error than the next closest baseline method.

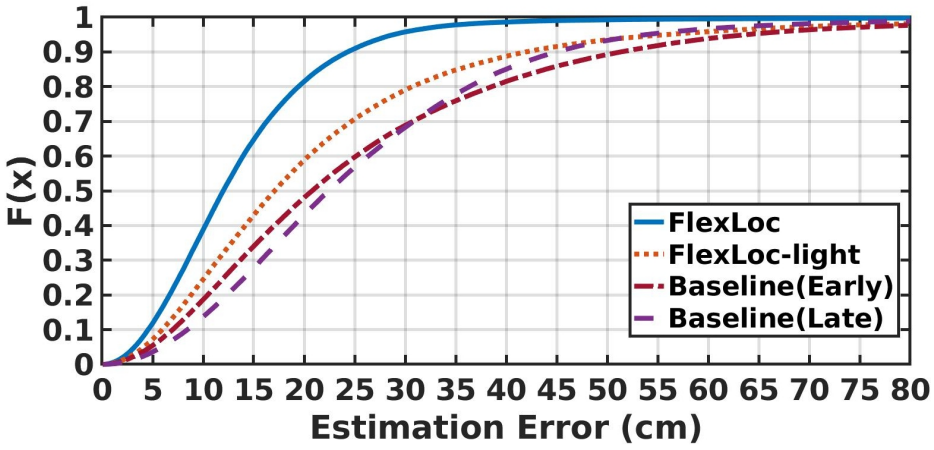

Figure 2.7: Cumulative distribution function (CDF) plot of frame-by-frame localization error of different methods.

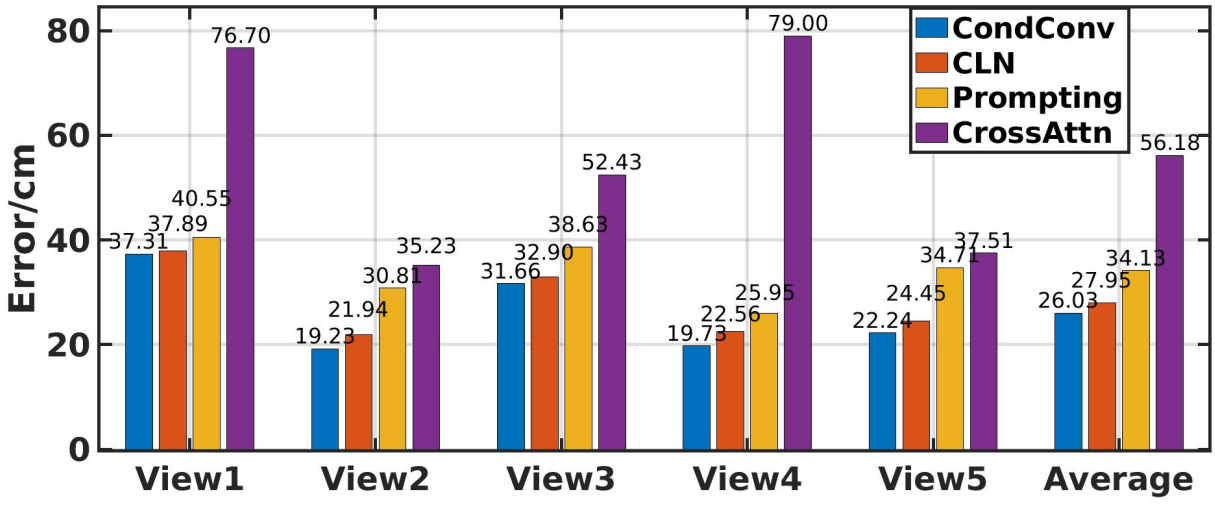

Figure 2.8: Comparisons of different methods to inject pose information.

2.9.2 Different Methods of Fusing Conditional Information

We also compare `FlexLoc`'s performance against various existing methods of integrating sensor pose information to showcase the performance benefit of the conditional layers. As the techniques we utilize require transformer encoders, we shift to an alternate model where the ResNets are removed and all the backbones are composed of transformer encoders. We implement and evaluate two networks inspired from adjacent works targeting perspective shift in other contexts, while stressing that these works do not attempt to address the issue of sensor pose shift for *localization*:

1. *Cross Attention Pose Fusion* [108]. This work incorporates camera pose into a diffusion model through cross-attention for improved video generation. We utilize this concept to fuse camera pose information through cross attention layers integrated into the transformer backbones.

2. *Pose Concatenation with Input* [30]. This work concatenates camera pose information into the input to improve monocular depth map prediction on unseen camera viewpoints. We implement their technique by encoding the camera pose as a single token that is concatenated to the sensor data tokens to be fed into the transformer backbones.

In Figure 2.8, we observe that integrating pose information through `FlexLoc` (CondConv) and `FlexLoc`-Light (CLN) achieves approximately 20% improvement over the next best method of concatenating prompts to the input. Furthermore, the Cross Attention method performs poorly with an error of 56 cm. The inferior performance of the other methods exemplifies the critical impact of carefully selecting the right method for incorporating node pose information.

	From Pretrained Backbones	From Scratch
FlexLoc (CondConv)	**14.28**	14.92
FlexLoc-light (CLN)	**22.04**	29.14

Table 2.2: Effect of Pretraining

2.9.3 Impact of different training strategies

In Table 2.2, we analyze the impact of loading pre-trained weights to initialize the network. In Section 2.8.7, we introduced a pretraining strategy where we loaded pre-trained weights into the backbone to aid in identifying the car. When training CLN without this strategy, the model performance degrades by 7.1 cm (24%), highlighting the importance of starting from a good set of weights for CLN. In contrast, CondConv was not adversely impacted with a performance loss of only 0.7 cm. The difference between these two models is likely due to CLN's position within the backbone, where it is closely coupled with the backbone weights and thus highly sensitive to their values. CondConv is placed *after* the backbone layers and is more insulated from the backbone weights.

Figure 2.9 illustrates the impact of the *sensor dropout* training method outlined in Section 2.8.7. Without sensor dropout, the model over-relies on the camera modality and largely ignores other modalities, incurring catastrophic error when testing on combinations of modalities without camera (150 cm for depth and 60 cm for mmWave). Applying sensor dropout increases the robustness of the model to missing modalities, allowing it to maintain good localization performance despite the loss of the critical camera modality.

2.9.4 Number of Extra Network Parameters

Both of the conditional components utilize a set of linear layers to derive their weights from the provided pose information. CondConv utilizes a wider set of linear layers with a total of 52992 parameters, while CLN is more lightweight with only 3328 parameters. However, CLN

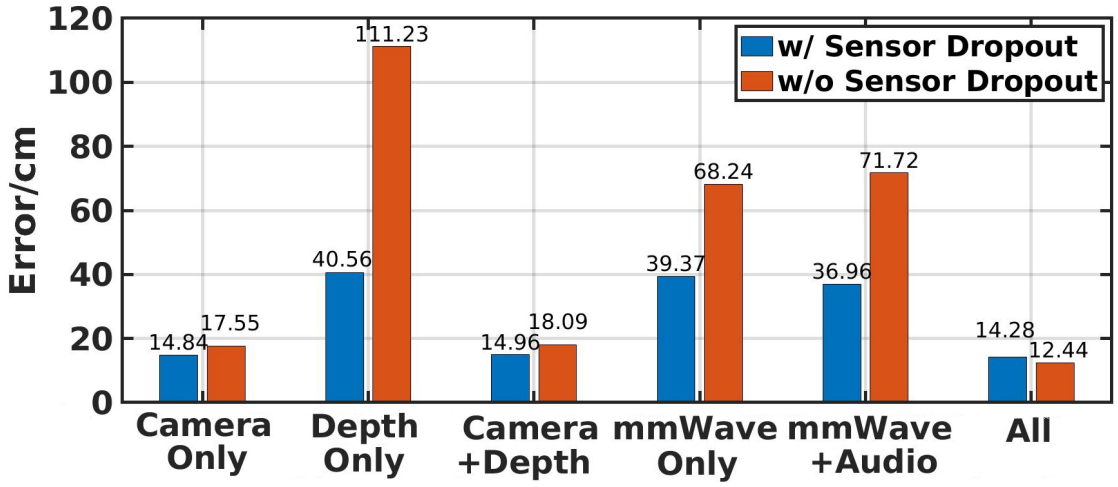

Figure 2.9: `FlexLoc` performance with and without dropout when only a subset of sensing modalities are available.

replaces the BatchNorm2D layers within the ResNet-18 architecture that each have more learnable γ and β parameters (Section 2.8.6), resulting in *fewer model parameters compared to a network without conditional layers.* Table 2.3 illustrates that CondConv results in 0.19% more model parameters and a computational overhead of 0.57% more Multiply-and-

	Performance	Additional Overhead	
Models	Localization Error	Parameters	MACS (M)
FlexLoc (CondConv)	14.28 (-46.84%)	53904 (+0.19%)	95.46 (+0.57%)
FlexLoc-light (CLN)	22.04 (-17.96%)	-11788 (-0.04%)	-54.04 (-0.32%)
CondConv + CLN	16.11 (-40.02%)	42116 (+0.15%)	41.28 (+0.24%)

Table 2.3: Additional overhead incurred by `FlexLoc` in terms of parameters and computations.

	View 1	View 2	View 3	View 4	Averaged
FlexLoc (CondConv)	**102.5**	**53.3**	**58.3**	**59.3**	**64.0**
Baseline-Early	194.5	183.8	224.1	173.8	190.5

Table 2.4: `FlexLoc`'s performance on GQ-Husky, Average Distance Error (cm)

Accumulate operations (MACs). In contrast, CLN removes parameters from the network, containing 0.04% *fewer* parameters and 0.32% *fewer* MACs. In smaller networks where these parameters and operations are more significant, CLN will be a better choice than CondConv despite the worse performance. Interestingly, unifying both CondConv and CLN into the same network resulted in worse performance than CondConv alone. This is likely because CLN and CondConv integrate similar information into the network, causing a saturation of performance, and the increased difficulty of learning with two conditional components hurts performance. Due to the similar overhead and worse performance of the unified model compared to CondConv, we choose to keep CLN and CondConv as separate architectures.

2.9.5 Performance on GQ-Husky

We evaluate `FlexLoc`'s performance on the *GQ-Husky* dataset to demonstrate its applicability to more complex and noisier datasets. The average distance error of the CondConv method in comparison to the Baseline Early-Fusion technique is showcased in Table 2.4. We can see that `FlexLoc`'s advantages are amplified on *GQ-Husky*, where it achieves *one-third* of the baseline's average distance error, revealing that injecting node pose information through conditional neural networks is an effective method for sensor perspective shift. Moreover, this performance in spite of background noise and imprecise sensor pose information demonstrates a degree of robustness in `FlexLoc`.

2.10 Limitations, Future Work, and Discussions

2.10.1 Sensor Pose Estimation from Sensing Data

While *GQ-Husky* helped eliminate the reliance on the OptiTrack motion capture system present within the *GDTM* dataset, it still requires for an AprilTag to be present during inference time. Thankfully, this can be performed as a calibration step, where an AprilTag is placed into the scene to allow the nodes to perform self-localization and can be subsequently be removed. Despite this, future work can explore extending `FlexLoc` to more settings by leveraging markerless self-localization techniques such as iNERF [91], or with on-board IMU and UWB sensors [109]. This enables deployment in environments where it is infeasible to place an AprilTag for self-localization.

Additionally, although the less-precise AprilTag pose information in *GQ-Husky* still yielded exemplary results, we did not explicitly examine the relationship between sensor pose errors and downstream accuracy. Future work can examine the degree to which sensor-pose information can be corrupted before it provides no benefit over the pose-agnostic baseline, coupled with error detection methods to decide when to incorporate pose information.

2.10.2 Extension to More Comprehensive Environments

While we achieved exemplary localization results in the presence of sensor perspective shift, we acknowledge that the setting of both the *GDTM* and *GQ-Husky* datasets (single car/robot driving in an area) are fairly simplistic. The background objects present in *GQ-Husky* demonstrate a degree of robustness to interfering objects, but other factors such as `FlexLoc`'s ability to generalize to different tracking targets or environment shift remain unexplored. Furthermore, it will be interesting to evaluate `FlexLoc`'s performance in the case of *multi-object localization*, which can provide critical information to applications such as robotic swarms [110].

2.10.3 Comparison to Late Fusion Architectures

While we primarily employed an early-fusion strategy due to superior performance, late fusion can be a superior option in certain scenarios. In a distributed computing setting, late fusion has its traditional advantage of flexible composition (independent predictions enable a dynamic number of sensors) and minimal communication overhead by exchanging only the final predictions. We sought to absorb these advantages of late fusion into the design of `FlexLoc`. First, we used a transformer encoder architecture to enable flexible composition, as transformers are structured to process arbitrary lengths of features. Second, `FlexLoc` leverages a backbone-adapter-fusion architecture. Each node can choose to either stream the raw sensor data to the central server or compute the features (256 length embeddings) locally and then stream the features to reduce communication overhead. We envision future work to investigate optimal strategies based on the capacity of the nodes and servers.

2.11 Conclusions

In this thesis, we propose `FlexLoc`, a multi-view multimodal object localization system that is robust to sensor perspective shifts. `FlexLoc` employs *conditional neural networks* to inject node perspective information into the localization pipeline, which can automatically adapt to unseen sensor perspectives without additional calibration data, achieving zero-shot generalization. Our evaluations on both indoor and outdoor datasets showcase that FlexLoc can outperform existing techniques with minimal additional overhead, while maintaining its performance in spite of noisy sensor pose information.

CHAPTER 3

ADMN: A Layer-Wise Adaptive Multimodal Network for Dynamic Input Noise and Compute Resources

In FlexLoc, adapting the model itself effectively addresses the challenge of sensor perspective shift at *deployment time*. Next, we consider the space of *runtime variations*, which occur on a comparatively faster timescale – possibly necessitating adaptation on a per-sample basis. Existing systems often leverage multimodal deep neural networks in these dynamic scenarios due to the robustness afforded by multiple sensing modalities. Nevertheless, they struggle with varying compute resource availability (due to multi-tenancy, device heterogeneity, etc.) and fluctuating quality of inputs (from sensor feed corruption, environmental noise, etc.). Statically provisioned multimodal systems cannot adapt when compute resources change over time, while existing dynamic networks struggle with strict compute budgets. Additionally, both systems often neglect to consider the impact of variations in modality quality. Consequently, modalities suffering substantial corruption may needlessly consume resources better allocated towards other modalities. We propose ADMN, a layer-wise **A**daptive **D**epth **M**ultimodal **N**etwork capable of tackling both challenges - it adjusts the total number of active layers across all modalities to meet strict compute resource constraints, and continually reallocates layers across input modalities according to their modality quality. Our evaluations showcase ADMN can match the accuracy of state-of-the-art networks while reducing up to 75% of their floating-point operations. We release our code at `https://github.com/nesl/ADMN`.

3.1 Introduction

Background: Multimodal deep learning systems fusing sensory data from various modalities are the standard for accurate, robust sensing [111, 112]. Accordingly, these multimodal systems are invaluable in highly dynamic environments, where a given input modality's quality-of-information (QoI) can vary drastically across samples. Low QoI data is corrupted, noisy, or otherwise degraded in a manner reducing its informativeness. Fluctuations in a modality's QoI can occur slowly (e.g., lighting conditions over the day) or rapidly (e.g., battlefield settings or unstable sensor feeds).

Challenges: Although multimodal deep learning systems are generally robust to variable QoI, a key challenge surrounds the *computational efficiency*. Most state-of-the-art multimodal networks employ *static provisioning* in which multimodal inputs are processed by a fixed architecture [16, 37] regardless of their individual utility. Consequently, valuable resources may be wasted on low-QoI modalities. Recent work has explored dynamic networks that train policy networks to reduce computation for *easy samples* [43, 46, 47, 49]. Unfortunately, explicit consideration of highly dynamic QoI variations among modalities has been largely overlooked. *We hypothesize that flexibly allocating computational resources among modalities in accordance with each modality's QoI on a per-sample basis can substantially improve model performance in compute-limited settings.*

Additionally, current works also neglect the challenge of *dynamic compute resource availability*. The environments where multimodal systems are most relevant often impose *temporally variable but strictly bounded* compute budgets. The maximum budget varies with time according to factors such as thermal throttling, energy fluctuations, or multi-tenancy, and cannot be exceeded at any given moment. Neither statically provisioned models nor dynamic networks are equipped to function under such constraints. Most dynamic networks optimize for *average-case efficiency*, without mechanisms to constrain the worst-case compute beyond the total cost of the full network. The only partially compatible works are

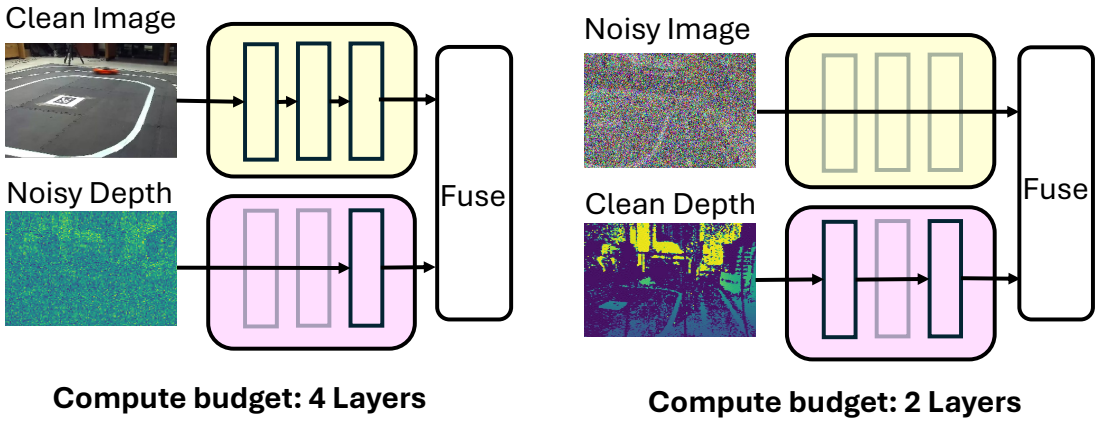

Figure 3.1: Overview of ADMN. Variable depth backbones adapt to both changing compute resources and input noise characteristics

those performing *model selection with gating networks* [46, 47, 113], which can be adapted to varying compute resource availability by training a set of models for each budget. Aside from requiring an unreasonable amount of training resources, it also impedes the standard practice of loading pretrained weights prior to finetuning [114], as there do not exist publicly available pretrained weights for every possible compute budget. *We hypothesize that a single network initialized with pretrained weights, which can dynamically adjust its resource usage, offers an effective solution to the challenge of fluctuating compute resources.*

Proposed Solution: We propose ADMN, an **A**daptive **D**epth **M**ultimodal **N**etwork jointly tackling the challenges of adaptation to both dynamic compute resources and variable QoI inputs. While these challenges are agnostic to the multimodal fusion method (e.g., data-level [15], embedding-level [115], and late [26]), we focus specifically on embedding-level fusion due to applicability and ease of implementation. Figure 3.1 provides a high-level depiction of ADMN. Following the standard for embedding-level fusion, each modality is processed with an independent backbone before undergoing fusion with a transformer encoder.

First, ADMN addresses the challenge of *dynamic compute resources* through adaptive backbones containing adjustable layer configurations. With the same set of model weights, ADMN

activates a subset of backbone layers according to the available compute resources. We ensure compatibility with pretrained weight initializations by injecting the LayerDrop technique [45] into both the backbone pretraining and multimodal network finetuning processes. This requires the adaptation of the unimodal, text-only LayerDrop technique to not only Vision Transformers, but also multimodal networks. Such a strategy produces a novel multimodal network whose backbones are resilient to missing layers at test-time, allowing for the usage of fewer layers during resource scarcity.

Second, given an established layer budget, `ADMN` addresses the challenge of *dynamic QoI* by adapting the choice of selected backbone layers according to each modality's QoI. `ADMN` learns a multimodal controller on top of the adaptive backbones to perform layer allocation. Unlike existing works, we explicitly structure the controller around dynamic QoI by introducing *corruption-aware supervision* into the training loss. However, as it is reliant on the presence of labeled corruption information during training, we propose an alternative *autoencoder-based initialization* (`ADMN_AE`) to emphasize the QoI without corruption labels. The controller is trained end-to-end for a particular layer budget through Gumbel-Softmax Sampling [116] and the straight-through estimator [56].

We benchmark `ADMN` against several baselines to demonstrate its ability to preserve accuracy while simultaneously minimizing energy and latency costs. `ADMN` is tested on both multimodal localization and classification tasks with realistic sensor corruptions representing dynamic QoI, reinforcing its generality and applicability. `ADMN` can match the performance of larger state-of-the-art models while reducing FLOPS by up to 75% and latency by up to 60%. We release our code at `https://github.com/nesl/ADMN`. Our contributions are summarized as below:

- We present the first layer-wise adaptive multimodal network where resource allocation among modality backbones is dictated by QoI characteristics and temporally variable but strictly bounded compute budgets at inference time for every sample. We evaluate `ADMN` with realistic sensor corruptions modeling dynamic QoI.

- We create a general framework for training layer-adaptive multimodal networks by injecting the LayerDrop technique into both unimodal backbone pretraining and multimodal finetuning, and systematically evaluate it across various modality combinations and datasets.

- We design a multimodal perceptual controller that explicitly attends to modality QoI through either *corruption-aware supervision* when training-time corruption labels are provided, or *autoencoder-based initialization* when they are not.

- We introduce a low-complexity method of training a multimodal controller that meets strictly bounded compute budgets instead of reducing average-case computation.

3.2 Related Work

Early Exiting in Unimodal Networks. Early Exiting has been explored extensively in unimodal networks to improve the *average-case inference efficiency* [41–43]. Methods like DeeBERT [41] and PABEE [42] use confidence thresholds to halt computation for simpler inputs. However, extending such techniques to dynamic multimodal QoI and strictly bounded, variable compute budgets is nontrivial. Specifically, they neglect to consider the impact of low-QoI samples even in the *unimodal* setting, while `ADMN` addresses the complex interplay of dynamic *multimodal* QoI. Moreover, Early Exiting is unable to constrain the *worst-case* inference computation beyond the total network cost, clashing with the need to accommodate strictly bounded compute budgets.

Unimodal Sub-Networks. Instead of performing input-aware network adaptation, other works prune a single large network at runtime to accommodate varying computational resources [44, 45]. *Once-For-All* [44] proposed an algorithm pruning a large network across depth, width, kernel size, and resolution, showcasing competitive performance against state-of-the-art neural architectural search baselines. *LayerDrop* [45] introduced a layer-wise

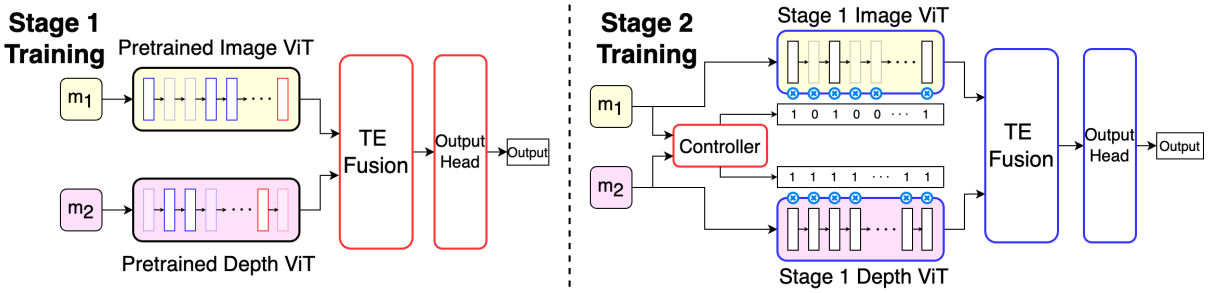

Figure 3.2: ADMN architecture. [Gray box]: dropped layer, [Blue box]: frozen layer, [Red box]: tunable layer. TE: Transformer Encoder.

dropout rate during pretraining for on-demand layer reduction at test time. We leverage LayerDrop as the foundation for ADMN, but emphasize that ADMN's contributions can easily be applied to other inference-time pruning techniques.

Dynamic Inference for Multimodal Systems. To address the inefficiency of static multimodal networks, dynamic networks [46–50] leverage input-dependent forward paths. DynMM [46] and DynaFuse [113] train a set of expert networks with different modalities and perform network selection at inference time. AdaMML [47] and Listen to Look [48] leverage multimodal information to eliminate temporal redundancy in videos. ACF [49] dynamically replaces certain modules with lightweight networks according to the input for greater efficiency. Despite these advances, existing methods (1) overlook significant QoI variations arising from corruption, (2) utilize or discard entire modalities without fine-grained control, and (3) fail to consider *fixed* resource budgets. ADMN addresses these gaps by introducing two methods of performing layer allocation with an explicit focus on relative modality QoI, while also accounting for a strictly bounded compute budget.

3.3 Methodology

3.3.1 Problem Description

In real-world multimodal systems, sensor data suffers from *dynamic QoI variations*. Environmental factors such as inclement weather and lowlight can corrupt the utility of a modality, while sensor-specific factors such as camera ISO or audio gain also play a factor. Additionally, environmental QoI corruptions may unequally affect modalities. For example, fog may affect vision (heavily) and depth (partially), but not audio. Furthermore, we also impose a strictly bounded, time-varying compute constraint to emulate factors such as thermal throttling or variable available energy. Thus, the goal of `ADMN` is to perform correct modality resource allocation according to the diverse sensor QoI variations, all while ensuring that the total allocated resources adhere to a strict budget.

A typical embedding-level fusion multimodal network is illustrated on the left side of Figure 3.2. It extracts unimodal embeddings with modality-specific backbones, condenses them into one joint embedding via self-attention with a *Transformer encoder*, and obtains an output from the joint embedding. `ADMN` builds upon this architecture by proposing two novel advancements. First, we introduce a *layer-wise adaptive multimodal network* comprised of dynamic modality backbones, thus enabling inference-time adaptation to any layer budget. Second, `ADMN` proposes a QoI-aware controller that dynamically allocates the layer budget optimally among modality backbones (i.e., transformer layers), allowing it to greatly outperform static models with the same layer budget.

3.3.2 `ADMN` Architecture

Figure 3.2 shows the task-agnostic architecture of `ADMN`, which involves a two-stage training process.

Stage 1: LayerDrop Finetuning. We first initialize each unimodal backbone with a

set of general weights (e.g., ImageNet weights) pretrained with LayerDrop [45]. Subsequently, the multimodal network is finetuned with LayerDrop on the desired task while freezing the earlier backbone layers (shown in blue) to prevent overfitting. Stage 1's objective is to adapt the pretrained weights to the specific target task, while also training the later Fusion and Output layers to accommodate diverse embeddings arising from various backbone layer configurations.

Stage 2: Controller Training. We freeze all the Stage 1 network weights and train a controller with either *corruption-aware supervision or autoencoder initialization* to allocate a fixed budget of L layers in accordance to each sample's relative modality QoI. From the multimodal inputs, the controller outputs a discrete sequence summing to L representing the selection of backbone layers.

3.3.3 Stage 1: LayerDrop Finetuning

We adapt LayerDrop [45], originally designed for *unimodal text transformers*, to support arbitrary multimodal transformers through integration into unimodal pretraining and multimodal finetuning.

3.3.3.1 Vanilla LayerDrop

LayerDrop [45] enabled on-demand test-time depth reduction of textual transformers by training with a layer-wise dropout rate. By stochastically dropping out the transformer layers, the network is trained to function with only a subset of layers. For an inference-time layer budget L, they introduced an "every-other" dropout strategy where layers are dropped in an alternating fashion starting from the middle. We refer readers to the original work for greater detail.

3.3.3.2 Multimodal LayerDrop

`ADMN` extends LayerDrop to Vision Transformer-style networks (ViT) [94] by first integrating them into the unimodal backbone pretraining process. We pretrain 12-Layer visual and audio backbones on the ImageNet-1k [117] and AudioSet [118] datasets, respectively. Rather than performing supervised training, which suffers from convergence issues, we pretrain both backbones with the Masked Autoencoder (MAE) approach [119, 120]. We enforce a LayerDrop rate of 0.2 in each layer of the ViT encoder while fixing the decoder. By employing LayerDrop in MAE pretraining, the model learns to reason in the presence of missing layers.

Subsequently, the MAE pretrained weights are loaded into the appropriate backbones of a multimodal (e.g., vision-depth) neural network to be finetuned on a specific corrupted dataset. As shown in Figure 3.2, the majority of the early backbone layers are frozen during finetuning, with later layers adjusting to the new task. A LayerDrop rate of 0.2 is maintained during the finetuning process in all the backbones, allowing the remaining learnable layers (Fusion and Output) to adapt to the countless combinations of missing layers. This process ultimately creates a task-specific multimodal network supporting adaptive backbone layer configurations at inference time.

Full-Backbone LayerDrop. One unique challenge with *multimodal* LayerDrop is the need to subject individual backbones to extreme dropout conditions. While dropping all layers in a *unimodal* network is rare, one may frequently drop all layers of a heavily corrupted modality in a *multimodal* network. Unfortunately, the typical LayerDrop training ratio of 0.2 in a standard 12 layer ViT is unlikely to expose the network to such a case during training. To remedy this, we employ full-backbone dropout during training time, establishing a 10% chance that all layers of a given modality's backbone will be dropped out independently of the 0.2 LayerDrop rate.

3.3.4 Stage 2: Controller Training

The controller selectively activates backbone layers in the frozen Stage 1 network according to the relative modality QoI and total layer budget L on a per-sample basis to minimize task loss, as shown in Figure 3.2. Ideally, the controller should also remain lightweight in operations, latency, and size.

3.3.4.1 Controller Architecture

In contrast to previous works [43, 46, 50] which train controllers to reduce computation in *the average case with no regard to variable QoI or fixed resource budgets*, `ADMN`'s controller is explicitly constructed around both factors. Figure 3.3 reveals the structure of the controller. First, we downsample the inputs and pass them through modality specific lightweight convolutional networks. These convolutions aim to produce embeddings containing information solely regarding each modality's QoI, which is sufficient from low-resolution data. With M modalities, the convolutional networks produce M total noise embeddings, which are fused by a transformer encoder into one embedding e_{corr} representing the QoI of every input modality. From e_{corr}, we can obtain a set of raw logits

$$\pi = \text{MLP}_{\pi}(e_{\text{corr}}) \text{ where } \pi \in \mathbb{R}^{C},\ C = \sum_{i=1}^{M} |b_i| \tag{3.1}$$

where $|b_i|$ is the total number of backbone layers for modality m_i. In essence, π represents an allocation of L available layers among C total backbone layers, with values dependent on the QoI characteristics of every input sample.

3.3.4.2 QoI-Aware Training

Ideally, the convolutional layers will automatically focus on each modality's QoI to provide correct layer allocations from the task-specific loss $\mathcal{L}_{model}$. Nevertheless, as we later show in Section 3.4.4, supervision with purely the task loss is insufficient, where the controller fails

to properly attend to QoI and outputs unsuitable layer allocations. Thus, we introduce two methods enabling QoI-awareness: *corruption-aware supervision* when the training dataset contains QoI labels, and *autoencoder-based initialization* (termed as `ADMN_AE`) when they do not.

Corruption-Aware Supervision. We supervise the controller with an additional corruption loss $\mathcal{L}_{corr}$. If the corruption is quantifiable (e.g., camera ISO, Gaussian Noise), we predict a corruption vector $\hat{\sigma}_m = \text{MLP}_{\text{corr}}(e_{\text{corr}})$ containing the corruption values for all modalities. The corruption loss is the MSE loss $\mathcal{L}_{corr} = \sum_{i=1}^{M} |\hat{\sigma}_{m_i} - \sigma_{m_i}|$, where σ represents the ground truth corruption value. If the corruption is difficult to quantify but can be grouped into K categories (e.g., weather annotations), we obtain a set of logits $f \in \mathbb{R}^K$, where $f = \text{MLP}_{\text{corr}}(e_{\text{corr}})$, and set $\mathcal{L}_{corr}$ as the cross-entropy loss between f and the category labels. We optimize over the joint loss $\mathcal{L}_{total} = \mathcal{L}_{model} + \mathcal{L}_{corr}$, which explicitly instructs the model to attend to the modality QoI.

Autoencoder-Based Initialization. In many scenarios, there is a lack of any ground truth QoI information. The key intuition of the *autoencoder-based initialization* approach is that the compression and reconstruction objectives of autoencoders result in a well-organized latent space that group similar samples together [121, 122]. Since the QoI corruptions are significant enough to impact downstream accuracy, they are vital to the reconstruction objective. Consequently, an autoencoder trained on variable QoI data is likely to structure the latent space in a manner that reflects each sample's QoI properties. Figure 3.3 illustrates the structure of `ADMN_AE`'s autoencoder pretraining. The *encoder* is comprised of the controller's perceptual components (i.e., convolution layers and fusion transformer), while the *decoder* performs reconstruction from e_{corr} with modality specific deconvolution layers. With autoencoder pretraining, the controller's perceptual layers learn to attend to input modality QoI without the need for any QoI labels. We then load the encoder weights into the controller, freeze them to retain QoI understanding, and learn the layer MLP allocations with the singular loss $\mathcal{L}_{model}$.

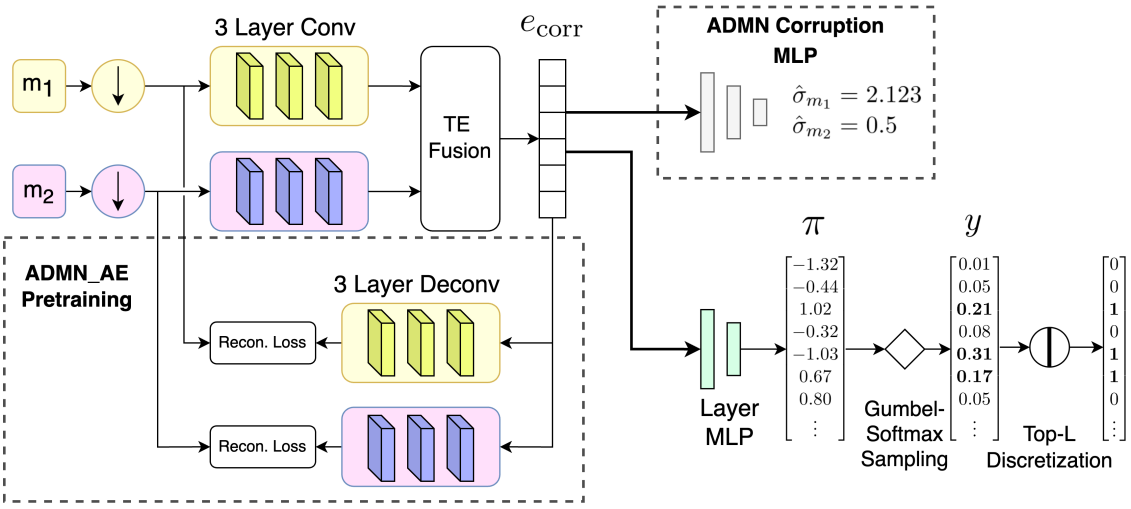

Figure 3.3: Detailed depiction of the ADMN controller.

3.3.4.3 Differentiable Layer Selection

Although the logits π provide information on which layers to select, activating a given layer is a binary decision. Such categorical decisions interrupt the flow of gradients and prevent end-to-end training. Traditionally, unimodal early-exit networks [43] employ Gumbel-Softmax Sampling [116] to approximate a categorical distribution while retaining differentiability. These networks employ decision networks *at every layer* of the model and leverage Gumbel-Softmax Sampling to decide whether to execute that particular layer. Coupled with a resource-aware loss, these techniques can train dynamic networks that reduce computation in the *average case on easy samples*. However, ADMN substantially differs from previous dynamic networks in that ADMN predicts (1) the entire allocation of layers at the *beginning* of model execution that (2) sum to a hard layer budget L. Simply applying Gumbel-Softmax Sampling to ADMN is insufficient, as the approach cannot select L total layers among all modality backbones.

We devise a method to select L total layers in a differentiable manner. Initially, we perform standard Gumbel-Softmax sampling with a temperature of 1 to obtain y. This allows multiple high-value logits to be represented in the resulting probability distribution,

better accommodating the selection of L layers. This is accomplished at the expense of the highly desirable near-discrete behavior of the low-temperature Gumbel-Softmax. We introduce a subsequent *top-L discretization* on y to activate only L layers, and maintain gradient flow through the *straight-through estimator* [56]. The downstream layers receive the discretized value, while the gradients received are copied over to the continuous logits y, allowing for gradient flow.

3.4 Evaluations

3.4.1 Experimental Setup

3.4.1.1 Datasets and Corruptions

The GDTM localization dataset [115] involves distributed nodes each containing modalities RGB, depth, mmWave radar, and multichannel audio. The target is a remote-controlled car on an indoor track. The MM-Fi [123] human activity recognition dataset contains 40 subjects, 27 total activities, and modalities RGB, depth, mmWave, and WiFi. We focus on the visual modalities (RGB, depth) for these two datasets. Finally, we evaluate on the audiovisual AVE [51] dataset with 4143 videos covering 28 classes, from which we use both RGB and audio modalities. These datasets do not naturally contain varying modality QoI, so we synthetically corrupt the modalities on a per-sample basis to simulate realistic sensor corruptions as follows:

Gaussian Noise: We add $N(0, \sigma_{i_j})$ to each modality i's input data. Each modality defines a set of N_i standard deviations $\{\sigma_{i_1}, \sigma_{i_2}, ...\sigma_{i_{N_i}}\}$ from which σ_{i_j} is drawn for each sample. This setting can represent systems with unstable links injecting different levels of noise, or sensors with different settings such as camera ISO levels. We apply this to the *RGB and depth modalities of the GDTM and MM-Fi datasets with* $N_i = 4$.

Rain: For the *outdoor RGB samples of the AVE Dataset*, we also explore simulating rainfall

and haze corruption, following the technique in [124].

Lowlight: We mimic lowlight corruption through a combination of gamma correction, color shift, and additive noise [125]. We create moderately and severely impaired lowlight *RGB samples from GDTM* while leaving the IR depth unchanged. However, under normal lighting (RGB unchanged), we emulate a light-saturated IR depth sensor by utilizing a frame of sunlight saturated IR depth. We also add lowlight corruption to both *rainy outdoor and clean indoor RGB samples of the AVE Dataset.*

Blur: We employ Gaussian Kernel blurring with two kernel sizes to emulate different severity of blur on *RGB images from the GDTM.*

Background Noise: We mix in wind audio for outdoor events and sound from a standing fan for indoor events to corrupt the *AVE dataset's audio modality.*

3.4.1.2 Baselines

Upper Bound: The full 12 layers are allocated to each backbone (no dropout), representing the maximum layer budget.

Naive Allocation: Given a layer budget L and M modalities, we naively allocate $\frac{L}{M}$ layers to each backbone following "every-other" allocation.

Image/Depth/Audio Only: All L layers are allocated to one modality, valid only for $L \leq 12$

Naive Scratch: We train a *new network from scratch without LayerDrop for every layer budget* L on the downstream task with each backbone containing $\frac{L}{M}$ layers.

Modality Network Selection (MNS): Most existing dynamic networks ignore fixed resource budgets and cannot be directly compared to `ADMN`. We adapt the network selection technique from [46, 47, 113], which train a set of "expert models" and use a gating network to select among them to create the *MNS* baseline. This technique is also frequently referred to as "mixture-of-experts". *MNS* trains an individual unimodal network for each modal-

	GDTM Dataset											
	Gaussian				**Lowlight**				**Blur**			
Method	6	8	12	16	6	8	12	16	6	8	12	16
Upper Bound	29.6				18.8				9.4			
ADMN	51.4	39.0	**33.1**	30.3	49.5	23.9	**18.0**	**17.3**	**11.8**	**11.2**	**9.8**	9.9
ADMN_AE	53.6	38.4	33.5	**29.4**	**35.3**	**22.8**	18.7	17.6	14.0	11.4	9.8	**9.6**
MNS	39.3	**32.1**	57.5	73.0	117.5	83.9	117.6	116.8	109.5	114.2	117.7	74.0
Naive Alloc.	112.5	97.6	46.9	31.0	90.3	67.3	27.1	17.7	81.4	50.6	12.5	9.8
Naive Scratch	**39.1**	36.7	40.4	37.3	117.5	84.6	117.6	117.5	117.8	116.2	117.7	93.6
Image Only	67.5	53.2	53.7	—	56.8	45.6	46.2	—	15.4	11.2	10.3	—
Depth Only	85.0	61.7	58.6	—	72.1	64.9	63.4	—	25.7	22.6	22.3	—

Table 3.1: GDTM Localization Error (cm) (↓) with 6-16 Layers of Budget

ity and one multimodal network where *each model is trained from scratch to conform to a specific layer budget*. In contrast, ADMN requires only the training of one single backbone network with LayerDrop, scaling much more effectively with the number of budgets, while also reaping the advantage of pretrained weight initializations.

Essentially, *Upper Bound*, *Naive Allocation*, and *Image/Depth/Audio Only* are *allocation baselines* operating on the same finetuned Stage 1 backbone, while *Naive Scratch* and *MNS* train new networks.

3.4.2 Main Results

GDTM Localization Error. Table 3.1 highlights that both ADMN (corruption supervised) and ADMN_AE (autoencoder pretrained) drastically outperform all the allocation baselines regardless of the corruption, especially at small layer budgets. Under Lowlight with 8 layers of budget, ADMN and ADMN_AE localize within 5 cm of the upper bound, while the next best

	MMFI (Gaussian)			
Method	6	8	12	16
Upper Bound	44.44%			
ADMN	35.03%	**39.25%**	41.92%	43.31%
ADMN_AE	**36.16%**	38.43%	**42.18%**	**44.40%**
MNS	3.70%	3.70%	3.70%	3.70%
Naive Alloc.	5.56%	12.96%	29.01%	42.90%
Naive Scratch	3.70%	3.70%	3.70%	3.70%
Image Only	18.83%	18.52%	23.15%	—
Depth Only	17.59%	30.86%	32.72%	—

Table 3.2: MMFI Accuracy (↑)

	AVE (Rain + Background Noise)			
Method	6	8	12	16
Upper Bound	71.19%			
ADMN	**57.07%**	**62.95%**	**67.48%**	66.60%
ADMN_AE	52.95%	62.30%	66.77%	66.67%
MNS	25.81%	26.56%	17.21%	13.34%
Naive Alloc.	36.16%	46.89%	65.71%	**67.95%**
Naive Scratch	23.94%	26.56%	18.70%	13.22%
Image Only	47.76%	54.11%	57.85%	—
Audio Only	36.41%	40.52%	42.64%	—

Table 3.3: AVE Classification Accuracy (↑)

baseline incurs almost 27 cm of additional localization error. The performance difference is the smallest under Blur, which we attribute to the strength of the visual backbone. While the heavily blurred RGB images are uninformative to the human eye, the visual backbone can still produce accurate localizations, and the controller correctly prioritizes image for layer allocation despite heavy blur. This exemplifies a strength of ADMN – learning allocations from the task loss prevents errors from preconceived human bias. In comparison to Naive Scratch and MNS, ADMN handily outperforms them with the exception of 6 and 8 layer Gaussian corruption. Due to the need to train networks from scratch (cannot leverage pretrained weights for arbitrary layer budgets), these two baselines struggle to converge with deeper networks and more complex corruptions. ADMN_AE's great performance also demonstrates that knowledge of the ground truth corruption is unnecessary.

Classification Accuracy. Tables 3.2 and 3.3 depict the classification accuracy for the MM-Fi and AVE Datasets, respectively, from 6 to 16 Layers. We observe similar trends as the GDTM dataset, but notice weaker results for the Naive Scratch and MNS baselines. These classification tasks with changing environments and temporal dependencies are considerably more complex than the single-car GDTM localization task. Consequently, the

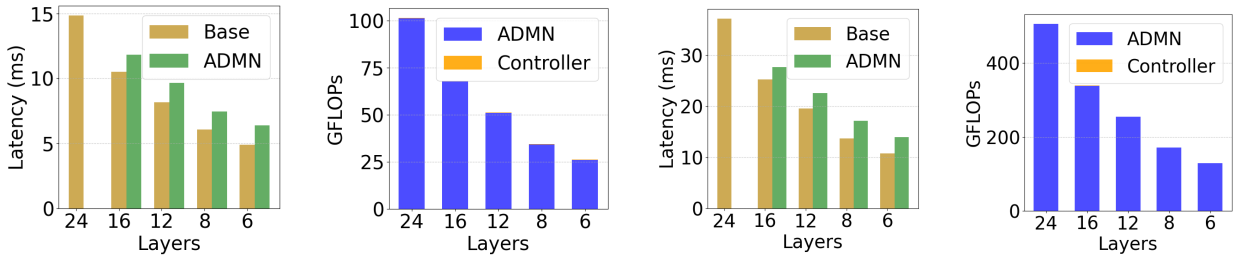

Figure 3.4: Latency (ms) and GFLOPs vs Layers for GDTM (left) and MM-Fi (Right)

accuracy of networks trained from scratch suffers. This serves to illustrate the importance of our LayerDrop training process – it enables the adaptation of pretrained networks to any arbitrary layer budget.

Latency and FLOPs. We show the meaningful reduction in latency (ms) and Giga-Floating Point Operations (GFLOPs) of `ADMN` in Figure 3.4. Although the controller accounts for a significant proportion of the latency at smaller layer budgets ($\sim 20\%$), the high throughput at these latencies (> 150 fps for GDTM) surpasses most sensor sampling rates. Moreover, `ADMN`'s controller utilizes a negligible amount of FLOPs. The controller constitutes about 1% and 0.2% of the model's total operations for GDTM and MM-Fi, respectively, at *the fewest allocation of 6 layers.* When viewing Tables 3.1, 3.2, and 3.3 in context of these metrics, we can observe the significance of `ADMN`. For instance, under Blur corruption in GDTM, `ADMN` localizes within ~ 2 cm of the Upper Bound while reducing latency by $\sim$60% and FLOPs by $\sim$75%.

3.4.3 Extended Evaluations

Three Modalities. To ensure that `ADMN` generalizes to more than two modalities, we perform localization on GDTM with RGB, depth, and mmWave radar modalities. Figure 3.5 shows a CDF of the error on a dataset where each modality has a 50% likelihood of suffering extreme Gaussian Noise. `ADMN` continues to perform correct allocations with three heterogeneous modalities.

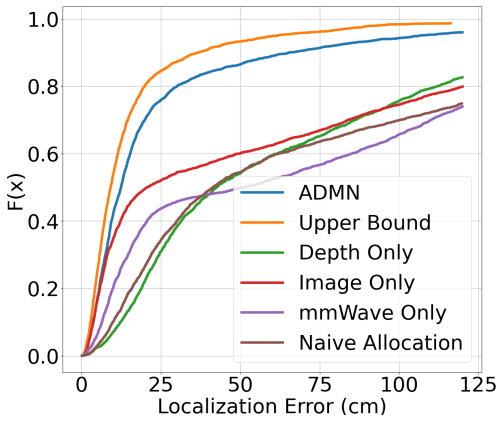

Figure 3.5: Localization Error for GDTM with Three Modalities

Unequal Backbone Computation. For each AVE dataset sample, we process eight image frames and one large audio spectrogram. Thus, the visual backbone consumes approximately *three times* the FLOPs of the audio backbone. We neglected this in Table 3.3 by assuming that all backbones were equally resource intensive. In Table 3.4, we present results in terms of *Audio Layer Budgets*, where activating one layer in the image backbone costs *three audio layers*. In comparison to Naive Alloc., which activates the same number of **real** layers per modality (e.g., Naive Alloc. 12 layers is three image and three audio layers), `ADMN` performs superior allocations accounting for unequal computation.

Table 3.4: Unequal AVE Classification Accuracy (↑), budget in Audio Layers

	AVE (Rain + Background Noise)			
Method	12	16	24	32
Upper Bound			71.19%	
`ADMN`	51.83%	64.34%	66.46%	67.68%
Naive Alloc.	36.16%	46.89%	65.71%	67.95%

Table 3.5: Impact of supervision on AVE Classification Accuracy (↑)

	AVE (Rain + Background Noise)			
Method	6	8	12	16
Upper Bound			71.19%	
`ADMN`	57.07%	62.95%	67.48%	66.60%
`ADMN_AE`	52.95%	62.30%	66.77%	66.67%
Task Loss `ADMN`	46.20%	57.13%	64.53%	67.50%

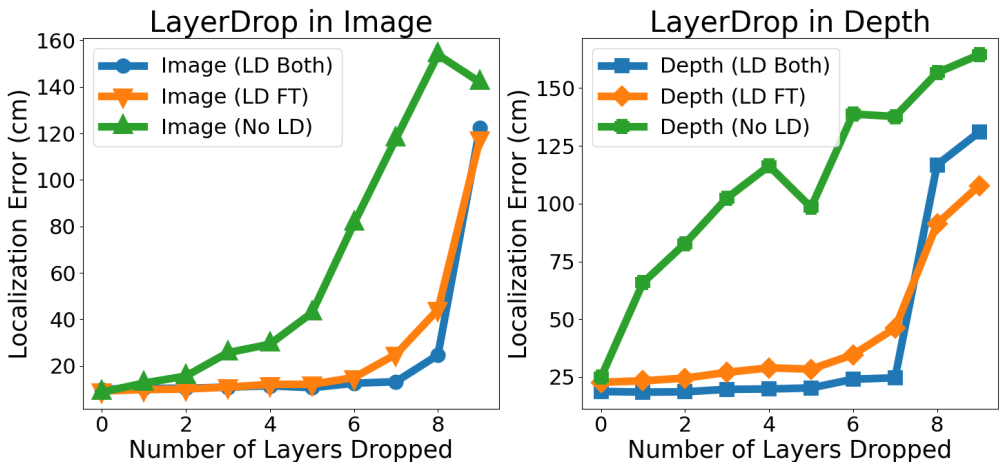

Figure 3.6: Effect of LayerDrop on 12-layer unimodal image and depth localization networks, evaluated on GDTM.

3.4.4 Ablation Studies

Efficacy of LayerDrop. LayerDrop is integrated into two stages – initial MAE pretraining on ImageNet and Stage 1 Finetuning. Figure 3.6 showcases the localization error on the GDTM dataset for separate unimodal depth and image networks. We observe that adding LayerDrop during finetuning (LD FT) has the greatest impact, but benefit is observed when added into both stages (LD Both). The ability to drop many layers without significant degradation confirms that LayerDrop is compatible with the ViT architecture, MAE pretraining, and is also effective when finetuned on another task.

Impact of Controller Supervision. In Table 3.5, we omit both the autoencoder pretraining and the corruption supervision to get a variant of `ADMN` supervised purely by the Task Loss only. The inferior results show the importance of utilizing either corruption supervision or autoencoder pretraining.

Autoencoder Latent Space. In Figure 3.7, we visualize how `ADMN_AE`'s autoencoder performs corruption-aware clustering in the latent space through a t-SNE plot. We note that this behavior is learned purely through a data driven manner, and *no information on*

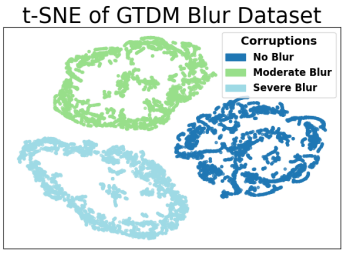

Figure 3.7: t-SNE of the autoencoder for different levels of RGB Blur on GDTM Blur

the corruptions were provided during autoencoder training. The obvious clustering of the various corruptions reveals how `ADMN_AE` is an effective replacement for explicit corruption supervision.

3.5 Limitations and Discussions

Dynamic Compute Constraints: One drawback is that the `ADMN` controller is trained only for one particular layer budget L. Consequently, we must train and load multiple controllers for *dynamic test-time compute requirements* induced by factors such as thermal throttling. Nevertheless, the training overhead for the controllers is low, enabling a controller to be trained for every layer budget. The controller also constitutes only 2.3% and 3.2% of the total network parameters in the GDTM and MM-Fi tasks, respectively, allowing for easy storage on disk.

Batched Inference: Since `ADMN` obtains the layer allocation prior to backbone execution, future work can exploit this for efficient batched inference. Classic adaptive models are incompatible with batched inference due to per-layer execution decisions for each sample. However, with `ADMN`, we can run an initial profiling stage and group samples into *sub-batches* based on similar layer allocation. For instance, samples with high depth noise and low image noise activate similar layers and can be grouped together.

Fusion with Early Exit: While `ADMN` and unimodal Early-Exit methods tackle funda-

mentally different problems, the two techniques can be combined for further efficiency gains. ADMN always allocates L layers across all the modalities. However, on simple inputs, all L layers may not be necessary, allowing for Early-Exit techniques to improve performance.

3.6 Conclusion

This chapter outlines ADMN, a multimodal network capable of dynamically adjusting the number of active Transformer layers across modalities according to the quality of each sample's input modalities. Through this continual reallocation of computational resources, ADMN can match the accuracy of far larger networks while utilizing a fraction of their operations. Additionally, the dynamic backbones of ADMN are also well suited for scenarios with adaptive compute, ranging from heterogeneous deployment devices to fluctuating energy availability. We demonstrated the superiority of ADMN compared to other baselines across both classification and localization tasks.

CHAPTER 4

SWAN: World-Aware Adaptive Multimodal Networks for Runtime Variations

Despite the efficacy of the system introduced in ADMN, there remain several practical challenges not addressed. Though ADMN considers changes in modality quality and available platform resources, it is unable to adapt to the complexity of the samples. Consequently, ADMN always allocates the full budget of layers, regardless of potential redundancy on "easier" samples. Moreover, it is unclear how well ADMN generalizes to real-world systems running on edge hardware and subject to complex downstream tasks.

In this chapter, we consider an expanded set of runtime variations: changes in modality quality, available platform resources and overall sample complexity (new). Current multimodal networks struggle with such fluctuations – adaptive networks cannot adhere to a strict compute budget, controller-based networks neglect to consider input complexity, and statically provisioned networks fail at all the above. Consequently, they do not extract maximum utility from the expended computational resources. We present SWAN (**S**ample and **W**orld-**A**ware Multimodal **N**etwork), the first adaptive multimodal network that accomplishes all three goals. SWAN employs a quality-aware controller to assign resources among modalities according to a variable user-specified maximum budget. Within this budget, an adaptive gating module further optimizes efficiency by scaling layer utilization according to sample complexity. For further gains, SWAN also employs a token dropping module that masks semantically irrelevant multimodal features before performing detections. We evaluate SWAN in the domain of autonomous driving with complex multi-object 3D detection,

reducing FLOPs by up to 49% with minimal degradation. Moreover, we demonstrate practicality on edge hardware with high performance runtime engines. The code is available at `https://github.com/nesl/SWAN`.

4.1 Introduction

Real world systems frequently contend with *runtime variations* as the world changes during system operation. A camera-based object detector may encounter varying environmental conditions that drastically alter the utility of its input modality. Similarly, a system deployed on a mobile robot will encounter diverse computational resource availability owing to thermal throttling or power limitations. Despite the large class of runtime variations, the most prevalent variations can be largely grouped into three categories – input modality quality of information (QoI) (*e.g.* sensor failure, dynamic weather), sample complexity (*e.g.* more or less sensing targets), and platform dynamics (*e.g.* thermal throttling).

A system's ability to adjust to such variations determines its utility in realistic deployments. One particularly relevant scenario involves the broad space of autonomous vehicles (AVs), ranging from small mobile robots to road-certified vehicles. The majority of AVs will encounter different environmental lighting or weather, and must exhibit robustness to resulting variations in input QoI. The AV's platform capabilities can also vary, where faster traveling speeds, low battery capacity, or thermal throttling can affect the amount of computation available for the current input sample. Finally, adapting computation based on sample complexity is also an important consideration. Since AVs operate on edge devices with limited resources, reducing computation on "easier" samples can translate to meaningful reductions in latency and energy consumption. The inability to adapt to any one of these variations can severely hamper the safety or utility of a given AV.

Existing works often deal with each of these variations separately. First, deep multimodal neural networks have emerged as a promising method for combating variable QoI. By virtue

of aggregating information across multimodal sensors, such networks mitigate environmental corruptions in the input data through redundancy. Thus, multimodal networks are the de facto standard in AVs [39, 40, 126]. Although they provide robustness against variable QoI for the metric of accuracy, the additional complexity of these networks can complicate adaptation to other runtime variations. Next, to meet dynamic computational budgets, reconfigurable neural networks modulate network computation during inference time. Once-For-All [44], LayerDrop [45], and ADMN [2] train adaptive networks where modules or layers can selectively be omitted at inference time for budget flexibility. Finally, input-adaptive systems incorporate knowledge of the complexity of the input to flexibly increase or reduce the computation performed on each sample [43,127,128]. For instance, AdaViT [43] dynamically removes attention heads, tokens, and layers from the network according to the nature of the input.

However, current literature falls short of proposing a system that *jointly addresses* these dynamics. The key difficulty lies in the compounding complexity that arises when these strategies are combined. To illustrate, once multimodality is introduced to combat dynamic QoI, adapting network utilization to input complexity requires consideration of shared information across modalities. Similarly, techniques that adapt network computation to a maximum budget are difficult to incorporate with those implementing per-sample complexity adaptation. We propose `SWAN`, an adaptive multimodal network that addresses all three runtime variations – changes in modality QoI, maximum compute, and sample complexity. Although this problem is universal across any domain deploying multimodal neural networks in real-world environments, we focus specifically on autonomous vehicles (AVs) due to the high task complexity (multimodal, multi-object 3D detection), robust evaluation frameworks, and high quality datasets.

First, we train a *layer-wise adaptive multimodal network* based on the existing CMT [40] AV detection framework to allow for flexible allocation of layers during inference-time. Next, we introduce a controller that determines the optimal allocation of layers across modalities

for the current budget and relative modality QoI. The controller is trained through *NeuralSort* [57] gradient approximation, enabling an understanding of relative importance among layers which existing controller-based techniques [2] fail to model. Finally, `SWAN` also minimizes resource usage within the user-specified maximal budget, enabling additional reductions in compute and latency. We present two complementary innovations – a feature-aware *SkipGate* module that conditionally executes the next layer, and a token dropout mechanism that filters out redundant tokens before the detection head. Both methods are conditioned upon the input, enabling efficiency on simpler inputs while retaining flexibility on more complex samples.

We evaluated `SWAN` on the popular nuScenes dataset [53] with modality QoI degradation simulated through the MultiCorrupt [129] pipeline. Across various maximum allowable budgets, `SWAN` outperforms related baselines by up to 9 NDS and 14.5 mAP while achieving up to a 49% reduction in FLOPs compared to a fully-provisioned network. `SWAN`'s controller correctly prioritizes high-QoI modalities under compute constraints, while the SkipGate and token pruning modules significantly reduce network computation. Additionally, we benchmarked `SWAN` on an Nvidia Jetson Orin edge device and with real-world data to further demonstrate its practicality. Finally, we outline and address various challenges when considering real-world deployments of `SWAN` (e.g., runtime engines, detecting platform conditions). We provide the code at `https://github.com/nesl/SWAN`.

In summary, our main novel contributions are as follows:

1. We introduce the first multimodal network that modulates resource allocation across modalities in accordance to modality QoI, sample complexity, and a user-defined maximum budget

2. `SWAN`'s lightweight, QoI-aware controller selects an optimal layer configuration given a specified budget, while retaining end-to-end differentiability through the *NeuralSort* gradient estimation technique

3. We additionally integrate a feature-based layer dropping module (*SkipGate*) that is conditioned on the output of the controller, and filter out tokens prior to the detection head for further efficiency gains

4. We perform evaluations on synthetically corrupted and real-world data, while also running on edge hardware to verify efficiency gains

5. We thoroughly discuss, then address the practical limitations of deploying `SWAN` on a real-world platform

4.2 Related Work

Multimodal AV Neural Networks. Owing to strict robustness and accuracy demands of autonomous driving, many AV perception stacks rely on multimodal deep neural networks [39, 126, 130–132]. For example, BEVFusion [39] combines information across LiDAR and camera in the BEV space, improving detection performance by over 19 NDS and 5 NDS relative to the best camera and LiDAR unimodal detectors, respectively. Additionally, these multimodal networks also showcase greater robustness to environmental noise when compared to their unimodal counterparts [126]. Adapting these multimodal networks to runtime variations that AVs commonly encounter is an open challenge.

Input-Aware Deep Neural Networks. Input-aware deep networks modify the network behavior according to the complexity of the input – saving layers on simpler inputs while leveraging full network capacity for more difficult samples [43, 127, 133–135]. AdaVIT [43] introduces a highly flexible ViT in which attention heads, layers and patches are flexibly executed according to input complexity. Early-Exit techniques [127, 134, 135] terminate execution once a confidence threshold is reached, enabling reduced computation on simpler inputs. These networks are incompatible with the strict limits on allowable computation that can occur in AV settings. Existing adaptive networks optimize the *average case computation*

with no guarantees on worse-case.

Adaptive Multimodal Networks. Although the majority of adaptive networks are unimodal, there exist a few adaptive *multimodal* networks. Several recent works [47, 136, 137] explored adaptive selection strategies, choosing to activating modalities or neural experts based on the input. ADMN [2] presents a more granular adaptation with a QoI-aware controller activating layers within a single multimodal network to meet a specified budget. However, they neglect to consider adaptation *within* a particular compute budget, while also being limited to toy networks with simple objectives (*e.g.* single object localization). Conversely, `SWAN` employs a flexible, budget-aware adaptation policy in the complex realm of AVs.

4.3 Methodology

`SWAN` accomplishes three objectives: it adapts the network to a user-specified maximum budget, decides an optimal allocation of resources among modalities for that budget, and then minimizes resource usage according to the input. In AV scenarios, this is particularly critical as available computational resources change over time (*e.g.* thermal throttling, energy constraints), coupled with rapidly changing modality QoI (*e.g.* changing environment).

Fig. 4.1 showcases the overall architecture of `SWAN`. Aside from the standard neural components present in most AV detection networks (discussed in the next section), we introduce three new components. First, `SWAN` learns a *QoI-aware controller* that intelligently assigns layers to modality backbones according to their marginal utility. The controller is trained to accommodate a diverse set of layer budgets, and will maximize its usage of a specified budget at inference time. Next, to ensure that valuable resources are not needlessly consumed if the budget is set too high, we integrate our *SkipGate* module into the multimodal network. Based on the feature vectors output by the prior layer and additional contextual information, SkipGate will determine whether to execute the current layer, enabling a reduction of

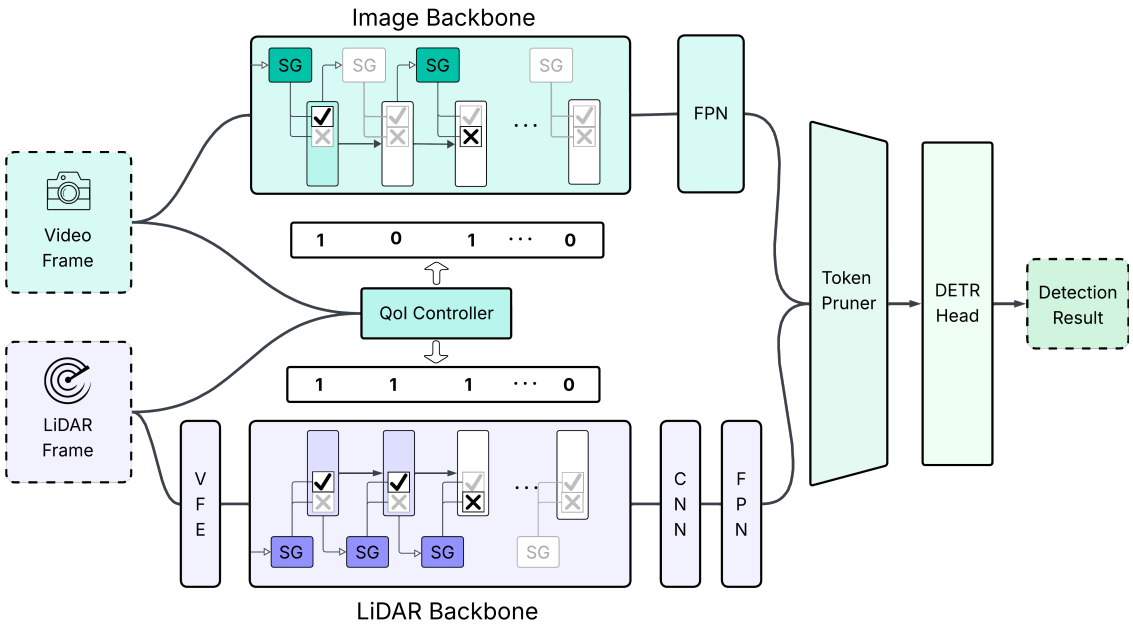

Figure 4.1: SWAN architecture. The QoI controller allocates resources among the backbones according to the input QoI and the current budget. For each selected layer, a modality-shared SkipGate module decides whether to actually execute the layer. The unimodal features are filtered by a token pruning module before the detection head.

computation within the outer controller's allocation. Finally, since not all areas of the input are equally important to the detection task, we implement a *layer-dropping* algorithm prior to the detection head that focuses computation on the semantically meaningful tokens.

4.3.1 Layer-Adaptive Multimodal AV Network

Architecture. SWAN is built upon the CMT [40] multimodal camera and LiDAR architecture. CMT processes each modality with a modality specific network, and adds 3D positional encodings to the unimodal features prior to fusion with a transformer decoder. We select

the Swin-Tiny ViT [138] as the camera encoder and FlatFormer Transformer [139] as the LiDAR encoder. Following FlatFormer and CMT, we utilize a Dynamic Voxel Feature Encoder (VFE), a SECOND-based convolutional backbone to process our dense LiDAR features, and independent feature pyramid networks to merge multi-scale features (Fig. 4.1). The bulk of the computation occurs in the transformer encoders; these auxiliary networks are lightweight and are not targeted for adaptation.

Training Procedure. CMT presents a multimodal fusion framework for AV detection and does not concern itself with flexible runtime adaptation. We incorporate LayerDrop [45] into network training by introducing a layer dropout rate (*i.e.* 0.2) within FlatFormer and Swin backbones. Over the training process, each layer is stochastically dropped out according to the rate, thereby conditioning the multimodal network to function with a subset of backbone layers.

4.3.2 QoI-Aware Controller

Controller Architecture. The controller allocates a specified budget of layers across the adaptive-depth backbones in the multimodal network. To ensure proper allocation, the controller must be aware of the relative input modality QoI. In Fig. 4.2, we provide a detailed overview of the controller.

First, the controller learns to identify the QoI characteristics of the input samples according to Eq. (4.1):

$$\mathbf{z} = \text{Concat}_{m \in M}(f_{\theta_m}(\mathbf{x}_m)) \ ; \ \mathcal{L}_{\text{env}} = \text{CE}(\text{MLP}_{\text{env}}(\mathbf{z}), y_{\text{env}}) \tag{4.1}$$

where f_{θ_m} is a lightweight convolutional network for modality $\mathbf{x}_m$, y_{env} is the ground truth corruption label (*e.g.* darkness), and CE is the cross-entropy loss. $\mathcal{L}_{env}$ ensures the information within $\mathbf{z}$ represents the QoI information.

Next, the controller maintains a library of fixed sinusoidal positional embeddings $\mathcal{E} =$

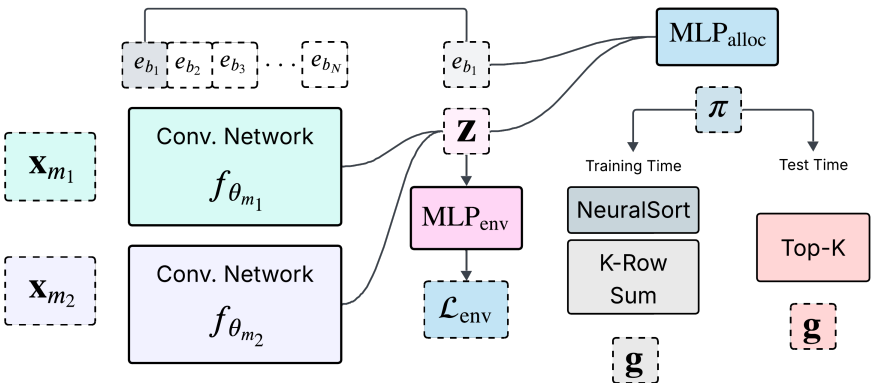

Figure 4.2: `SWAN` controller. Lightweight convolutional networks extract the QoI of the input modalities, and *NeuralSort* maintains end-to-end differentiability during training

$\{\mathbf{e}_b \in \mathbb{R}^d \mid b \in \mathcal{B}\}$ corresponding to a set of N user-specified layer budgets $\mathcal{B} = \{b_1, b_2, \ldots, b_N\}$. During training, we sample a budget $b \sim \mathrm{Uniform}(\mathcal{B})$ for each input, ensuring that the controller is compatible with each budget. At inference time, the user specifies the budget b. The selected embedding $\mathbf{e}_b$ is concatenated with $\mathbf{z}$ to produce $\tilde{\mathbf{z}}$ with QoI and budget awareness.

Finally, the controller selects an allocation of layers according to Eq. (4.2).

$$\boldsymbol{\pi} = \mathrm{MLP}_{\mathrm{alloc}}(\tilde{\mathbf{z}}) = [\boldsymbol{\pi}_1 \parallel \boldsymbol{\pi}_2 \parallel \cdots \parallel \boldsymbol{\pi}_M] \tag{4.2}$$

where each sub-vector $\boldsymbol{\pi}_m \in \mathbb{R}^{L_m}$ contains the logits used to determine the activation state of the L_m layers in the backbone of modality m. At inference time, we simply activate the layers corresponding to the b largest logits across $\boldsymbol{\pi}$. At training time, however, this sampling operation is non-differentiable and requires a different approach.

Training the Controller. The critical challenge when designing adaptive networks is maintaining differentiability during the training process. While methods such as Reinforcement Learning allow learning over non-differentiable operations, recent works [2, 43] prefer Gumbel-Softmax [116] or Straight-Through [56] gradient approximation techniques for end-to-end learnability, citing the convergence struggles of reinforcement learning. However, we find that the straight-through propagation approach in ADMN results in uncertain layer

selections that perform poorly on complex AV datasets (Tab. 4.1).

In `SWAN`, we integrate the differentiable *NeuralSort* sorting algorithm into our controller to address this challenge [57]. Given a vector $\boldsymbol{\pi} \in \mathbb{R}^n$, NeuralSort produces a permutation matrix where the i-th row is defined as

$$\hat{P}_{sort(\boldsymbol{\pi})}[i,:](\tau) = \text{softmax}\left(\frac{(n+1-2i)\boldsymbol{\pi} - A_{\boldsymbol{\pi}}\mathbf{1}}{\tau}\right) \tag{4.3}$$

where $A_{\boldsymbol{\pi}}$ represents the matrix of absolute pairwise differences $|\boldsymbol{\pi}_i - \boldsymbol{\pi}_j|$ and τ is a temperature parameter that controls the smoothness of the relaxation. Intuitively, $\hat{P}_{sort(\boldsymbol{\pi})}[i,j]$ represents a soft assignment weight approximating the event that item j is the i-th largest element in $\boldsymbol{\pi}$.

During training, `SWAN` adds stochasticity through a standard Gumbel distribution to produce $\tilde{\boldsymbol{\pi}} = \text{Gumbel}(0,1) + \boldsymbol{\pi}$, and applies *NeuralSort* to obtain $\hat{P}_{sort(\tilde{\boldsymbol{\pi}})}$. With the previously sampled budget b, we calculate a soft-selection mask $\mathbf{g} \in \mathbb{R}^L$ (where $L = \sum_{m \in M} L_m$) by aggregating the first b rows of the relaxed permutation matrix:

$$\mathbf{g} = \sum_{i=1}^{b} \hat{P}_{\text{sort}(\tilde{\boldsymbol{\pi}})}[i,:] \tag{4.4}$$

To apply the budget constraint across modalities, we partition $\mathbf{g}$ into modality-specific subvectors $[\mathbf{g}_1 \parallel \mathbf{g}_2 \parallel \cdots \parallel \mathbf{g}_M]$. The forward pass for the l-th layer of modality m is then weighted by its corresponding gate $g_{m,l}$:

$$\mathbf{h}_{m,l} = g_{m,l} \cdot f_{m,l}(\mathbf{h}_{m,l-1}) + (1 - g_{m,l}) \cdot \mathbf{h}_{m,l-1} \tag{4.5}$$

where $\mathbf{h}_{m,l}$ is the output embedding of layer l and $f_{m,l}$ represents the current backbone layer.

Functionally, by annealing the temperature value τ during training, we can strike a balance between high *exploration* at the start of training, and *exploitation* during the latter stage. When τ is small (*e.g.* 0.1), the behavior of *NeuralSort* becomes near discrete, minimizing the domain shift between soft logits during training and hard-gating during inference.

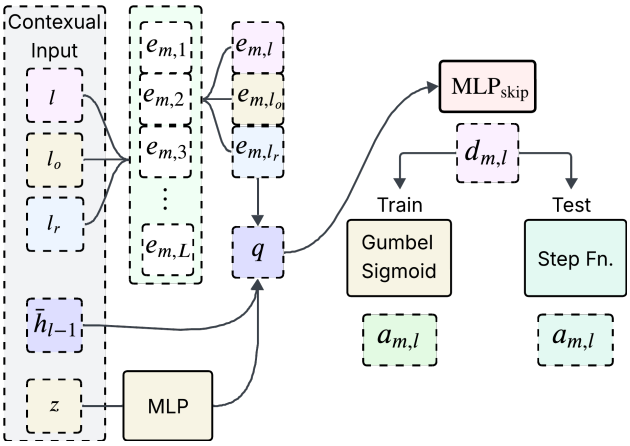

Figure 4.3: SkipGate uses contextual information and the previous layer output for conditional execution

We supervise this process with the detection loss, allowing the model to learn layer allocations under various QoIs and compute budgets for ideal detection performance.

4.3.3 *SkipGate* Layer Dropout Module

SkipGate Architecture. While the controller effectively allocates b layers across all modalities, it will always consume the maximum allotted budget. Realistically, the user is likely to simply set a maximum compute budget that the system should not exceed. Consequently, adaptation within the layer budget can result in meaningful reductions in compute, energy, and latency.

In Fig. 4.3, we provide an overview of the proposed *SkipGate* module, which conditionally executes a layer selected by the QoI controller. We initialize individual *SkipGate* modules for each modality, and share the same module among a modality's backbone layers. First, each SkipGate module maintains a fixed library of sinusoidal embeddings $\mathcal{E}_m = \{\mathbf{e}_{m,l} \in \mathbb{R}^d\}$ where $l \in \{0, 1 \dots L_{max}\}$ and L_{max} represents the maximum number of layers across all modality backbones. The SkipGate module converts the following contextual input integers into sinusoidal embeddings: the current layer in the backbone (l), the number of controller

selected backbone layers remaining (l_r), and the number of layers assigned to the other modality (l_o). Additionally, the SkipGate module also receives the noise embeddings $\mathbf{z}$ from the controller. The need for all this contextual information underscores the difficulty of designing a SkipGate module in this setting – it is heavily influenced by the modality QoI and the decisions of the controller.

From this contextual information, SkipGate obtains logits $d_{m,l}$ representing the likelihood of executing a particular layer l according to Eq. (4.6).

$$\mathbf{q} = (\bar{h}_{l-1,m}) \parallel e_{m,l} \parallel \text{MLP}_z(\mathbf{z}) \parallel e_{m,l_r} \parallel e_{m,l_o} \quad ; \quad d_{m,l} = \text{MLP}_{\text{skip}}(\mathbf{q}) \tag{4.6}$$

where $\bar{h}_{l-1,m}$ is the averaged output features of a previous backbone layer. During inference time, we apply thresholding by executing only the layers with positive logit values.

SkipGate Training. Similar to the training of the QoI controller, layer selection is a non-differentiable operation. However, SkipGate only needs to make a single decision – whether to execute the current layer or not. Thus, we do not require the ordered property of NeuralSort and can resort to a simpler *Gumbel-Sigmoid* operator to obtain the soft-probabilities $a_{m,l}$:

$$a_{m,l} = \sigma\left(\frac{d_{m,l} + g_1 - g_2}{\tau}\right) \quad ; \quad g_1, g_2 \sim \text{Gumbel}(0,1) \tag{4.7}$$

As we decrease the temperature τ, the behavior approaches a discrete $(0, 1)$ selection while retaining differentiability. We then utilize $a_{m,l}$ to weigh the contributions of the current layer:

$$\mathbf{h}_{m,l} = a_{m,l} \cdot f_{m,l}(\mathbf{h}_{m,l-1}) + (1 - a_{m,l}) \cdot \mathbf{h}_{m,l-1} \tag{4.8}$$

We add a utilization penalty that encourages the module to minimize its layer usage, implemented as a *hinge loss* scaled by a constant β_m:

$$\mathcal{L}_{\text{skip}} = \sum_{m \in M} \sum_{l=1}^{L_m} \frac{\text{ReLU}(d_{m,l} + \beta)}{\beta_m} \tag{4.9}$$

$\mathcal{L}_{\text{skip}}$ serves two purposes – it penalizes large logit values, forcing the module to only activate the useful layers, and it cuts off the loss after $d_{m,l}$ reaches $-\beta$, preventing the module from "cheating" by making logits indefinitely negative.

4.3.4 DETR Head Token Pruning

The detection head converts a large number of modality tokens from the backbones into bounding box and classification results. With average LiDAR and image resolutions, the detection head may process over thirty-thousand tokens. Despite this, many of the tokens may not be semantically meaningful in context of the downstream detection task, corresponding to irrelevant details such as the sky or background trees.

To remedy this, we propose a token dropping algorithm that eliminates these unnecessary tokens. Given the backbone output embeddings $\mathbf{h_m} \in \mathbb{R}^{H \times W \times D}$, we obtain a set of weights $\mathbf{w_m} \in \mathbb{R}^{H \times W}$, where $\mathbf{w_m} = \sigma(f_{\beta_m}(h_m))$. f_{β_m} is a small two-layer convolutional network that preserves the input shape through padding, and σ is the sigmoid function. We directly discretize $\mathbf{w_m}$ into $\tilde{\mathbf{w}}_\mathbf{m}$ by applying a rounding operation, and apply the straight through estimator for gradient propagation. The training process is supervised with an additional utilization loss $\mathcal{L}_{tp} = \sum_{m \in M} \sum_{i \in H, j \in W} (\tilde{w}_{m,i,j})$.

4.3.5 Joint Training Details

In this section, we provide greater details on the overall training process. *First*, we train our multimodal network with LayerDrop and standard detection loss $\mathcal{L}_{\text{detection}}$, producing a layer-adaptive multimodal AV network. *Next*, we freeze the main network's parameters and train the *QoI-aware controller* on the corrupted nuScenes dataset with loss $\mathcal{L} = \mathcal{L}_{\text{detection}} + \alpha_1 \mathcal{L}_{\text{env}}$, where $\mathcal{L}_{\text{env}}$ is from Eq. (4.1). Afterwards, we introduce the *SkipGate* modules while freezing all other parameters, and train with loss $\mathcal{L} = \mathcal{L}_{\text{detection}} + \alpha_2 \mathcal{L}_{\text{skip}}$, where $\mathcal{L}_{\text{skip}}$ is defined in Eq. (4.9). We anneal the temperature τ during training. Finally, we train for token pruning

with loss $\mathcal{L} = \mathcal{L}_{\text{detection}} + \alpha_3 \mathcal{L}_{\text{tp}}$. α_2 and α_3 allow for control over the degree of layer skipping and token pruning by trading off higher accuracy with greater efficiency.

4.4 Main Results

4.4.1 Main MultiCorrupt Results

Datasets and Baselines. The standard nuScenes dataset [53] contains only a limited number of rainy and dark corruptions. Since `SWAN` targets varying modality QoI, we used the *MultiCorrupt* [129] generation pipeline to simulate five new conditions by corrupting only the clean (sunny, dry) samples of the nuScenes dataset. *Lidar Beamsreducing* occurs when sensor degradation or power constraints restrict the number of emitted beams, *Camera Fog* simulates a foggy scene where camera is heavily impacted while high-power LiDARs exhibit robustness, *Darkness* heavily impacts camera while LiDAR is unaffected, *LiDAR Motionblur* and *Camera Motionblur* occur when vehicle vibrations or improper mounting lead to motion artifacts. We emphasize that these evaluations were merely a conduit to display `SWAN`'s robustness towards dynamic QoI – we did not aim to exhaustively evaluate every possible environmental corruption.

We compared our base AV network against several popular AV detection models: LiDAR-only SST [140], image-only PETR [141], and multimodal BEVFusion [39]. Moreover, to simulate platform dynamics, we apply `SWAN` under different budgets of L total backbone layers. Under this constraint, we implement two additional baselines – Naive Allocation where each modality receives $\frac{L}{2}$ layers, and also the controller-based, QoI-aware ADMN [2] scheme. For greater transparency into `SWAN`, we progressively evaluate the integration of each module, from the controller to token pruning. Unless otherwise specified, `SWAN` refers to SWAN-PSC from Tab. 4.1.

Key Takeaways. We summarize the key observations from Tab. 4.1, followed by a more comprehensive analysis in the subsequent sections. First, `SWAN`'s controller-based alloca-

Method	LiDAR Beamsreduce		Camera Fog		Camera Motionblur		Dark		LiDAR Motionblur		Compute ↓	
	NDS ↑	mAP ↑	NDS	mAP	NDS	mAP	NDS	mAP	NDS	mAP	GFLOPs	Latency
Base Network	41.80	28.97	67.57	61.30	67.65	61.31	66.85	59.45	63.93	58.32	651.00	124.84
BEVFusion	32.08	10.77	68.38	62.89	69.49	64.88	68.28	62.56	58.18	53.60	630.73	118.76
PETR	35.21	32.63	22.17	14.72	20.32	12.84	23.24	14.92	35.21	32.63	1100.15	44.31
SST	22.06	1.38	66.42	58.65	66.43	58.65	66.43	58.65	54.82	42.73	372.85	92.79
Naive 16	39.48	25.40	67.48	61.17	67.21	60.85	66.70	59.37	63.02	56.79	556.41	120.93
SWAN-C 16	40.97	27.74	67.31	60.62	67.31	60.55	67.2	60.49	63.25	57.3	597.55	121.59
SWAN-SC 16	39.01	28.37	66.18	58.16	66.18	58.46	65.86	58.48	63.03	56.94	455.66	116.17
SWAN-PSC 16	36.42	25.43	65.08	57.43	64.95	57.20	64.72	57.34	61.40	55.18	428.37	83.42
ADMN 16	40.12	26.78	67.54	61.16	67.28	61.15	66.89	60.15	62.84	56.65	584.01	123.21
Naive 8	32.92	16.16	64.88	56.74	64.39	56.34	64.74	56.47	59.33	50.97	426.56	106.09
SWAN-C 8	36.76	26.08	64.57	56.67	64.83	56.81	65.92	57.83	58.37	49.21	445.01	110.33
SWAN-SC 8	36.77	26.07	64.56	55.95	65.12	57.02	66.05	58.33	58.35	49.13	400.63	109.03
SWAN-PSC 8	34.42	23.23	63.99	55.95	64.20	56.24	64.94	57.14	56.82	47.60	373.42	77.15
ADMN 8	36.35	21.04	65.86	57.90	65.69	57.79	65.68	57.74	55.74	46.52	438.09	110.21
Naive 6	29.59	9.99	61.40	51.02	61.73	52.10	61.34	50.74	54.78	44.17	394.31	102.11
SWAN-C 6	34.33	22.61	63.77	54.72	64.55	55.95	64.28	55.80	56.19	45.18	400.43	107.09
SWAN-SC 6	34.30	22.60	64.45	55.78	65.24	57.20	65.50	57.58	54.41	41.98	372.62	106.51
SWAN-PSC 6	31.99	19.87	63.96	55.92	64.36	56.47	64.52	56.66	55.00	44.08	345.07	74.37
ADMN 6	32.55	15.95	63.06	53.71	62.94	53.82	63.12	53.99	48.05	35.29	407.78	105.81
Naive 4	24.71	3.88	57.83	44.70	57.71	45.02	56.73	43.62	45.82	29.45	362.00	98.81
SWAN-C 4	29.17	16.12	64.47	55.80	64.77	56.49	64.46	56.13	54.38	41.97	358.90	103.73
SWAN-SC 4	29.16	16.11	64.45	55.78	64.72	56.48	64.45	56.10	54.33	41.94	358.90	104.33
SWAN-PSC 4	27.92	14.18	63.95	55.90	64.95	55.91	63.74	55.69	54.86	43.99	331.41	72.21
ADMN 4	26.33	7.48	55.69	42.04	60.55	49.66	58.08	46.47	47.82	32.20	369.65	102.02

Table 4.1: `SWAN` against baselines across diverse modality QoI and compute budgets. The three SWAN variants progressively integrate the controller (C), SkipGates (S), and token pruner (P). Metrics reported are NDS and mAP with averaged median latency (ms) and Giga-Floating Point Operations (GFLOPs) on an Nvidia RTX 4090

tion (SWAN-C) substantially increases mAP and NDS compared to similar baselines under compute-limited settings (*e.g.* four layers). Next, the SkipGate (SWAN-SC) module's abil-

ity to drop layers reduces the model FLOPs by up to 18% below the Naive baseline with marginal decreases in mAP and NDS. Finally, the token pruning module (SWAN-PSC) filters out a large portion of tokens prior to the detection head, reducing the SWAN-SC latency by a maximum of 30.8%.

SWAN Detection Performance. In the first row of Tab. 4.1, we compare the base multimodal AV detection network with all 20 layers (8 in LiDAR, 12 in image) against multimodal (BEVFusion), LiDAR-only (SST), and camera-only (PETR) networks. Our multimodal network displays enhanced robustness against environmental corruptions compared to the unimodal networks: when either LiDAR in SST or camera in PETR is corrupted, the lack of redundancy causes model performance to drop. Additionally, our base multimodal network is on-par with the efficient BEVFusion's performance.

Applying `SWAN` under compute restricted regimes results in meaningful increases in detection performance. For instance, under 4 layers of budget and LiDAR Beamsreduce noise, adding the controller (SWAN-C) raises mAP from 3.88 to 16.12 compared to the Naive baseline. Similarly large NDS and mAP gains occur across other corruptions. Moreover, SWAN consistently outperforms ADMN under budget scarcity. ADMN's straight-through gradient estimation fails to learn precise layer allocations – it selects suboptimal layers within each modality's backbone (apparent under LiDAR motionblur). In contrast, `SWAN` utilizes *NeuralSort* for controller training, enabling a superior understanding of relative backbone layer importance which directly translates into improved detection performance.

Additionally, when comparing against the fully-provisioned *Base Network*, `SWAN` achieves competitive results with a fraction of the required layers. With a 4 layer budget, `SWAN` detects within 2.7 NDS and 5.4 mAP of this upper bound under Camera Motionblur. Similar results can be observed under Darkness. This highlights not only the effectiveness of our layer allocation algorithm, but also the flexiblity of our multimodal adaptive backbone.

Controller and SkipGate Layer Selection. Next, we take a deeper dive into the layer allocations outputted by the SkipGate module and controller. Tab. 4.1 shows how the Skip-

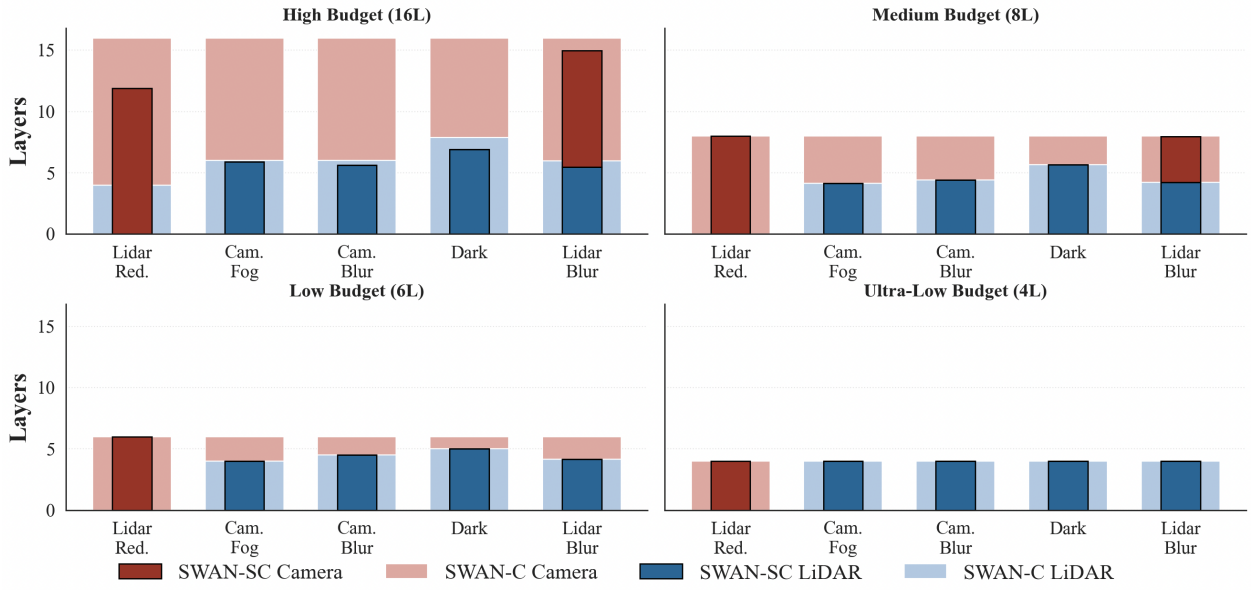

Figure 4.4: Layer selection of SWAN-C and SWAN-SC under different corruptions and budgets. Maximum budget is 20 layers

Gate module drastically reduces the compute performed in the network when the allocated budget is high without an adverse impact on detection performance. For instance, on average, SWAN-SC reduces GFLOPs by over 18% relative to the Naive baseline at a budget of 16 Layers.

Fig. 4.4 outlines the layer selection capabilities of the controller (SWAN-C) and SkipGate (SWAN-SC) modules. Under severe budget constraints (*e.g.* 4 layers), the controller favors modalities with higher relative QoI (*e.g.* LiDAR during darkness). Interestingly, under LiDAR motionblur, the controller still favors the LiDAR modality. As showcased in Tab. 4.1, diverting resources towards camera (*i.e.* Naive 4) results in significant degradation, demonstrating that the blurry LiDAR still offers superior information compared to camera. This reveals one of the key advantages of `SWAN`: rather than relying on pre-conceived human notions of how to allocate resources, `SWAN` *learns an optimal allocation from the data itself.* Furthermore, in these low budgets, the SkipGate module recognizes the resource scarcity and correctly decides to executes all layers.

The utility of the SkipGate module becomes evident under compute-rich settings (*e.g.* 16 layers). In this setting, there exists redundancy across layers, allowing the SkipGate module to eliminate certain layers to reduce computation. The SkipGate module analyzes the input to determine the layer execution – running as few as 6 layers (*e.g.* camera fog) or as many as 15 layers (*e.g.* LiDAR Blur), demonstrating its advanced understanding of how the current budget and environmental conditions affect the necessity of a given layer.

Token Pruning. We calculate that on average, SWAN-PSC retains 25.54% of image tokens and 48.84% of LiDAR tokens, contributing towards large decreases in model latency in Tab. 4.1. Fig. 4.5 depicts how the pruning module prioritizes semantically meaningful areas by removing the background LiDAR and image features. Moreover, when camera is corrupted by fog, most of the image tokens are discarded, and the few retained tokens surround the detection targets (*e.g.* vehicles) in the scene. However, we emphasize that these visualizations are an approximation. Due to downsampling and windowed attention in both transformer backbones, information is exchanged and condensed across spatial tokens, which can result in the irregular pattern seen in "Clean Camera".

Latency and FLOPs Analysis. Interestingly, despite the SkipGate module reducing layer utilization (Fig. 4.4) and minimizing FLOPs by up to 18%, the latency barely improves over the Naive 16 baseline (Tab. 4.1). Additionally, under other layer budgets, SWAN-SC can take longer to execute compared to both the Naive baseline and SWAN-C. We find that on average, the controller takes 1.5 ms to run while each SkipGate image and LiDAR module have latencies of 0.2 and 0.3 ms, respectively, failing to account for the discrepancy in the results.

We attribute this "invisible latency" to the *orchestration overhead of PyTorch* on high-performance devices. The controller and SkipGate modules require data (*e.g.* budgets, allocation results) to be transferred between the CPU and GPU, which can take fractions of a millisecond. On a high performance system (*e.g.* Nvidia RTX 4090) where the backbone layers can run in milliseconds, the orchestration latency becomes a non-negligible portion of

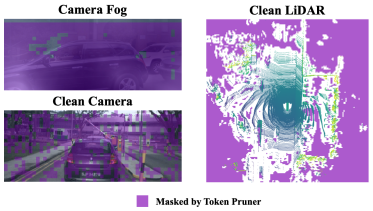

Figure 4.5: Token Pruning Visualization

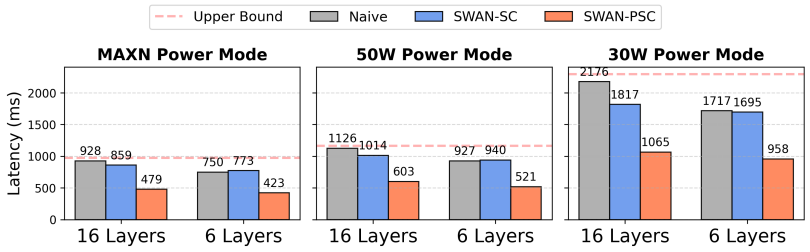

Figure 4.6: Evaluations on Nvidia Jetson Orin across camera fog corruption and different power modes

model runtime. We verify this in Section 4.4.3, where an Nvidia Jetson Orin edge device running the same SWAN models benefits by over 100 ms with the inclusion of the SkipGate modules. Additionally, we perform model compilation to mask this latency in Section 4.5.

4.4.2 Real Corrupted Data

To showcase the feasibility of SWAN on real-world noisy data, we validate it on rainy and dark subsets of the nuScenes validation dataset. We previously discarded this data to avoid interference with MultiCorrupt data synthesis. First, we perform a challenging evaluation of *sim2real transfer coupled with novel corruption generalization*. SWAN has only seen MultiCorrupt dark samples, and has never seen any rainy samples during the training of its modules. In Fig. 4.7, we showcase SWAN's performance compared to the Naive (equal) allocation baseline. Despite the domain shift, SWAN obtains superior NDS/mAP in low-compute regimes, confirming the generality of our approach.

However, SWAN suffers from low accuracy on higher budget allocations due to extreme layer dropping by the SkipGate module. We utilized heavily corrupted synthetic data to train SWAN, which likely resulted in the SkipGate model learning aggressive layer dropping. Thus, we independently finetune the SkipGate module and the subsequent token pruning on the rainy and dark subset of nuScenes train. Fig. 4.7 depicts higher layer usage and recovered NDS/mAP.

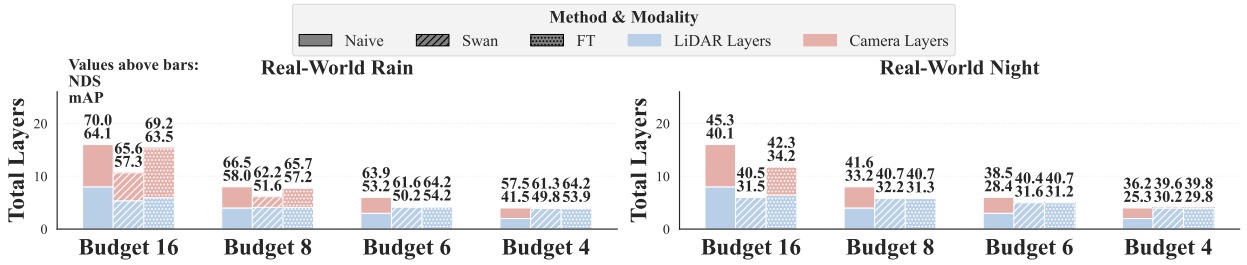

Figure 4.7: Base and Finetuned (FT) SWAN on real-world nuScenes rainy/dark data. *Bar height represents layer utilization, with NDS/mAP scores shown above in text*

The need to perform real-world finetuning to recover accuracy is not a shortcoming of SWAN – it stems from the domain shift between the synthetic Multicorrupt dataset and real-world nuScenes samples. When available, SWAN should always be trained and tested on real-world corrupted data to minimize the domain gap. Existing AV datasets prioritize high-quality sensing and lack the systematic modality failures (e.g., severe sensor faults) to fully evaluate SWAN's capabilities, necessitating the usage of the Multicorrupt dataset.

4.4.3 Edge Hardware Evaluation

We benchmark our SWAN model on an Nvidia Jetson Orin edge device on camera fog corruption in Fig. 4.6. We observe a substantial reduction in per-sample latency when introducing the SkipGate module (SWAN-SC), reinforcing our hypothesis that the Tab. 4.1 latencies on the RTX 4090 were dominated by overhead costs. Moreover, as the hardware capabilities degrade, the advantage of the SkipGate controller increases, reducing the latency compared to the Naive baseline by 7.4%, 9.9%, and 16.5% for MAXN, 50W, and 30W power constraints, respectively. When employing the full SWAN model with token pruning, we reduce latency by over 50%. Real-world AVs are likely to utilize these power-efficient edge devices, highlighting SWAN's impact in realistic deployments.

	GDTM Single Obj. Localization (Error (cm) ↓ / PyTorch Lat.)				**MM-Fi 15-Frame Human Gesture Recog.** (Accuracy (%)↑ / PyTorch Lat.)			
Method	**6 Layers**	**10 Layers**	**16 Layers**	**20 Layers**	**6 Layers**	**10 Layers**	**16 Layers**	**20 Layers**
Naive	78.1 / **4.1**	44.3 / **5.6**	36.9 / **7.9**	30.4 / 9.4	23.2 / 12.7	33.0 / **18.2**	**45.7** / 26.3	44.4 / 31.9
SWAN-C	**48.2** / 4.3 (4.7)	**36.5** / 5.8 (3.8)	**28.5** / 8.1 (2.8)	**28.3** / 9.6 (2.7)	**30.9** / 13.7 (7.4)	**39.2** / 19.3 (6.1)	43.2 / 27.5 (4.7)	**46.0** / 33.1 (3.6)
SWAN-SC	48.4 / 5.1 (25.7)	36.8 / 6.7 (20.4)	30.7 / 8.5 (7.2)	31.6 / **9.2** (-1.9)	29.6 / **12.3** (-3.3)	38.0 / 18.7 (2.9)	42.9 / **24.0** (-8.5)	41.7 / **27.1** (-15.1)

Table 4.2: 4090 Performance and PyTorch latency (ms) with percent overhead (Δ%) w.r.t Naive Alloc. Format: Perf. / Latency (Δ%).

4.5 Real-world System Deployment

In this section, we highlight and address several practical challenges associated with realizing `SWAN` on real-world edge hardware. During these evaluations, we verify that `SWAN`'s ability to simultaneously jointly address three unique runtime variations – dynamic modality QoI, variable computational resource available, and fluctuating sample complexity – is generalizable to additional datasets beyond nuScenes. First, we illustrate that `SWAN` is not only compatible with modern neural network runtime engines (e.g., TensorRT), but performing such model compilation can greatly minimize the "invisible latency" observed previously. Next, our previous evaluations abstracted the available budget in terms of available layers for simplicity, overlooking the different FLOPs and latency profiles of layers across heterogeneous backbones. We demonstrate that `SWAN` is equally effective when representing the budget as a real-world metric (e.g., total backbone latency) despite non-uniform backbone latencies. Finally, the current version of `SWAN` relies upon an external budget supplied by the user. We design a method inferring the available budget from the current system state, where a decrease in available computational resources, such as a spike in GPU usage from a background process, can be detected and compensated for by assigning a lower budget.

4.5.1 New Datasets

We perform evaluations on the MM-Fi and GDTM datasets to show generalizability beyond nuScenes. GDTM is a single target localization dataset with data from several distributed

multimodal nodes, each containing a suite of sensing modalities. We utilize the RGB camera and realsense LiDAR from all three multimodal sensors. Through the evaluations on GDTM, we verify that `SWAN` functions properly on object detection tasks the nuScenes dataset. `SWAN` was additionally evaluated on the MM-Fi human activity recognition dataset, utilizing the 15 frames of RGB vision and stereo depth for each sample. Varying levels of Gaussian noise were added to each of the samples to simulate variable modality QoI. 12 Layer ViT-B backbones were used to process the features of each modality and compress them into a [CLS] token, after which a lightweight transformer encoder performed fusion across modalities/time. Token pruning is omitted due to lack of a DETR-like module.

We showcase the results in Table 4.2. Consistent with the results on nuScenes, Tab. 4.2 shows how `SWAN` improves performance at lower budgets, both in terms of lower localization error and higher classification accuracy. When viewing the latencies incurred by adding the controller and SkipGate modules, GDTM exhibits the same "invisible latency" as nuScenes, where relatively fast backbone speeds result in orchestration overhead dominating. In contrast, MM-Fi, which sends 15 frames of data through each unimodal backbone, reaps significantly more benefits from `SWAN`, highlighting that `SWAN` is particularly effective when backbone latency is high.

4.5.2 TensorRT Implementation

We implement SWAN on a neural network runtime engine, specifically TensorRT, to accomplish the following objectives. First, it proves that `SWAN` is compatible with modern neural runtime engines, which is critical for widespread adoption given the staggering efficiency gains obtained through TensorRT model compilation. Moreover, we also validate our earlier hypothesis that the inefficiencies of `SWAN` in terms of latency are an artifact of PyTorch orchestration overhead, observing significant latency drops when running through TensorRT. When exporting the adaptive `SWAN` networks to TensorRT, careful consideration must be taken to retain the conditional behavior when converting to an intermediate ONNX format,

Metric	Configuration	Layers			
		6	**10**	**16**	**20**
Localization Error (cm)	Naive	78.06	44.29	36.85	30.37
	SWAN-C	48.24	36.50	28.47	28.31
	SWAN-SC	48.41	36.76	30.72	31.60
Latency Per Sample 4090 (ms)	SWAN-C PyTorch	4.27	5.80	8.10	9.61
	SWAN-SC PyTorch	5.13	6.73	8.45	9.18
	Percent Change	**20.22**	**15.95**	**4.32**	**-4.50**
	SWAN-C TRT	1.83	2.61	3.77	4.54
	SWAN-SC TRT	2.03	2.71	3.41	3.66
	Percent Change	**10.72**	**3.95**	**-9.70**	**-19.55**
Latency Per Sample Orin (ms)	SWAN-C PyTorch	35.70	49.22	69.19	83.05
	SWAN-SC PyTorch	41.10	54.16	68.87	75.06
	Percent Change	**15.12**	**10.04**	**-0.46**	**-9.62**
	SWAN-C TRT	10.46	15.23	22.46	27.22
	SWAN-SC TRT	11.48	15.38	19.15	20.22
	Percent Change	**9.70**	**0.97**	**-14.70**	**-25.71**

Table 4.3: Comparison of Localization Error and Latency (RTX 4090 vs. Jetson Orin) across selected layers. Percent change showcases the overhead of the SkipGate modules

after which the TensorRT builder replaces the ONNX If-Nodes with specialized TensorRT conditionals branches. Evaluations on the GDTM dataset are conducted both on a 4090 desktop GPU and an Nvidia Jetson Orin edge device running with MAXN power and fixed clocks.

First, Table 4.3 reveals the efficacy of TensorRT deployment itself, which reduces latency by over half on RTX 4090, and up to two-thirds on Orin. Such staggering efficiency gains, especially on the Orin edge device, underscore the importance of TensorRT compatibility – innovations incompatible with TensorRT deployment will be rejected for practical deployments. Beyond just being compatible with TensorRT, `SWAN` *improves when run on TensorRT*, reducing the percent overhead of the SkipGate modules by up to 15 percentage

Method	Latency Budget			
	5 ms	10 ms	14 ms	19 ms
Naive	36.81 / 16.13	45.82 / 29.45	54.78 / 44.17	59.33 / 50.97
SWAN-C	**42.13 / 29.75**	**51.58 / 43.45**	**55.76 / 46.00**	**59.96 / 52.30**

Table 4.4: nuScenes evaluations on PyTorch 4090 where the budget is supplied as total backbone latency. NDS/mAP values shown.

points as pre-compiling the model into a CUDA graph eliminates orchestration overhead. This remarkable result verifies that the puzzling inefficiency of SWAN is not an inherent flaw of its design – rather, it is simply an artifact of PyTorch's eager runtime environment. The fact that SWAN exhibits the greatest latency improvements when evaluated on an edge device and with TensorRT solidifies it as a promising choice for real-world, edge deployments.

4.5.3 Real-World Metrics

All the previous evaluations have been conducted assuming that the budget is provided in terms of layers. While LiDAR and Image layers are not equally resource intensive – image layers consume ∼2.4x more FLOPs but ∼1/3 the latency of LiDAR layers – this abstraction enabled a clear evaluation of the controller's ability to prioritize modalities based on marginal utility. Consequently, the FLOPs and latency values in Tab. 4.1 are slightly influenced by the reallocation between these asymmetric layers. However, there is nothing inherent to SWAN's design that does not allow it to operate in terms of real-world metrics – real-world costs can simply be assigned to each layer.

We confirm that SWAN is cost-metric agnostic by evaluating it on LiDAR motionblur under four latency budgets. These budgets are where Naive Allocation provides 1, 2, 3, and 4 layers per backbone, corresponding to 5, 10, 14, 19 ms, given that Flatformer and Swin layers cost 3.13 ms and 1.43 ms, respectively. SWAN's controller deftly handles asymmetric

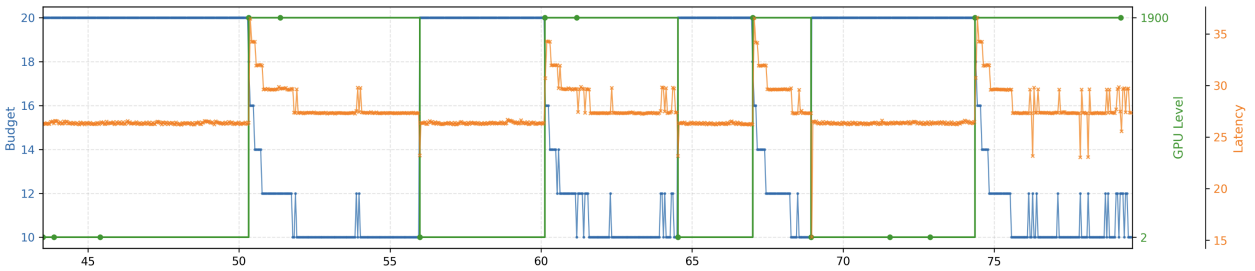

Figure 4.8: A snapshot of SWAN-C's performance when subjected to a changing background process. Budget in layers (blue), GPU Level in matrix multiply size (Green), and Latency in ms (Yellow). X-axis is in seconds

backbone latencies under these various latency budgets (Table 4.4), outperforming the Naive allocation variant at all latency budgets, with more meaningful improvement at smaller budgets. At low budgets (i.e., 5 ms and 10 ms), the controller allocates all resources towards image (Swin) due to its lower latency cost and high QoI.

4.5.4 Inferring Budget from Platform State

The last missing piece for realizing SWAN on a realistic system is inferring the available budget from the current system state. Due to factors such as thermal throttling or multi-tenancy, the hardware platform exhibits variable computational resource availability. Assuming that the input sample must be processed within a set amount of time (e.g., 30 ms), SWAN must infer the best budget from underlying system statistics to meet the latency threshold. The budget prediction module is realized in two stages: device profiling, and model training.

First, during the device profiling stage, various background GPU loads are simulated on the edge device while the SWAN network is run at various different layer configurations. The current number of active layers, inference time required per sample, and underlying hardware statistics are recorded. We record the *GR3D_FREQ* (akin to GPU usage on Nvidia Orin), *GPU Counter* (represents number of active GPU cycles), and *Power Consumed*. Background loads are simulated by running matrix multiplications of different sizes in the background

every 1 millisecond. These metrics provide an understanding of how underlying system metrics are related to the main SWAN network's inference speed.

With a large amount of data collected during device profiling, the next stage is to train a small predictor network, implemented as an MLP. From the GR3D_FREQ, GPU Counter, Power Consumed, and number of active layers, the predictor network outputs an estimated latency. After training, the predictor network achieves a validation error of 0.8196 ms, demonstrating its ability to accuracy estimate latency from hardware metrics. This predictor network is then integrated into the SWAN system by reading the current hardware metrics, running a forward pass with all the available budgets, and outputting a tensor of estimated latencies for each possible budget. The system then simply select the maximum layer budget that satisfies a user specified latency bound.

We profile the system on the GTDM dataset with a background process alternating between high-GPU usage and low-GPU usage, showcased in Figure 4.8 as GPU Level 1900 and 2. The latency bound is set to 30 ms. The SWAN-C variant can adapt its budget to meet a fixed latency despite the influence of a background process, scaling the layer usage up when the process goes idle, and sharply dropping the budget when the background process dominates the GPU.

4.6 Discussion and Limitations

Compiling SWAN for nuScenes. While we successfully showcased deployment of SWAN on an Orin Jetson while running TensorRT, we could only do it on the GDTM dataset and were unable to showcase it on the nuScenes dataset. This is due to the increased complexity of the models used for nuScenes detection – both the FlatFormer LiDAR backbone as well as the Swin-Transformer posed several problems when converting to TensorRT. We hope future work can explore how to export these backbones to TensorRT, enabling a more realistic set of evaluations.

Full Systems Realization of SWAN. While Section 4.5 addresses the majority of the real-world issues associated with practical deployment of SWAN, it stopped short of realizing the full system. A full systems implementation of SWAN would run on an edge device with TensorRT, contain all modules (i.e., SkipGate, Controller, Token Pruner), and be able to appropriately detect and respond to a wide variety of background processes. We currently only evaluate the SWAN-C variant under two GPU background workloads, and intend to explore how to detect and adapt to other forms of workloads that may impact overall model latency. There also exist several unexplored extensions, ranging from accounting for temporal consistency, to exploring how well the latency prediction system generalizes to real-world workloads (e.g., ResNet classification) running as a background process.

4.7 Conclusion

We present SWAN, the first multimodal neural network jointly addressing runtime variations across modality QoI, platform dynamics, and input complexity. Its QoI-aware controller assigns resources to modality backbones under a strict compute budget, while the input-aware SkipGate and token pruning modules enable further efficiency. Evaluations on a corrupted variant of the nuScenes dataset demonstrates that SWAN achieves comparable accuracy to a fully-provisioned network despite utilizing only a fraction of the resources. SWAN also retains performance on real-world data with exciting results on edge deployments.

CHAPTER 5

Decoupling Vision and Language: Codebook Anchored Visual Adaptation

In this chapter, we go beyond adapting single-task models to performing adaptation of general purpose multimodal foundation models, highlighting and addressing some challenges unique to these foundation models. Specifically, we consider Large Vision-Language Models (LVLMs). These models use their vision encoders to translate images into representations for downstream reasoning, but the encoders often underperform in domain-specific visual tasks such as medical image diagnosis or fine-grained classification, where representation errors can cascade through the language model, leading to incorrect responses. Existing adaptation methods modify the continuous feature interface between encoder and language model through projector tuning or other parameter-efficient updates, which still couples the two components and requires re-alignment whenever the encoder changes. We introduce CRAFT (Codebook RegulAted Fine-Tuning), a lightweight method that fine-tunes the encoder using a discrete codebook that anchors visual representations to a stable token space, achieving domain adaptation without modifying other parts of the model. This decoupled design allows the adapted encoder to seamlessly boost the performance of LVLMs with different language architectures, as long as they share the same codebook. Empirically, CRAFT achieves an average gain of 13.51% across 10 domain-specific benchmarks such as VQARAD and PlantVillage, while preserving the LLM's linguistic capabilities and outperforming peer methods that operate on continuous tokens.

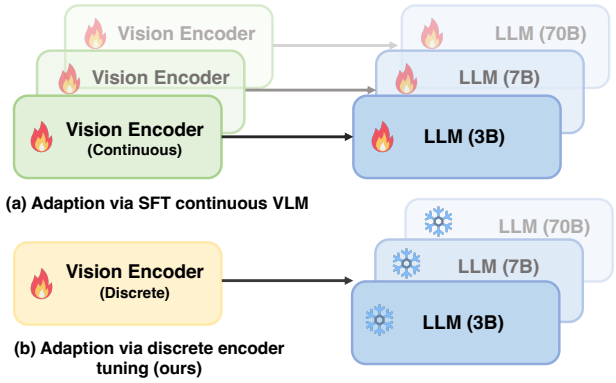

Figure 5.1: Continuous *vs.* Discrete Adaptation

5.1 Introduction

The vision encoders of LVLMs often struggle on tail domains or tasks that were underrepresented during pretraining [71–73]. They may misinterpret important visual cues or overlook domain-specific details required for the task, and these errors can propagate through the alignment layers into the language model, misleading the entire LVLM towards incorrect answers, *e.g.*, underperforming human accuracy by more than 50% on curated evaluations that stress vision-encoder weaknesses [74]. The common strategy is to perform fine-tuning with domain-specific data. However, updating the encoder alone is rarely sufficient: once its feature distribution shifts, the language model must also be recalibrated to interpret the new visual embeddings. This implies that each time a new domain is introduced or a stronger language backbone is adopted, the alignment process has to be repeated from the beginning.

The more fundamental problems arise when both the vision encoder and the language model are adapted together. On the data side, training under narrowly scoped, domain-specific supervision makes the model prone to memorizing superficial text patterns and can thus forget its linguistic capabilities; as noted in DeepSeek-VL [142], this necessitates a carefully curated mixture of multimodal data, which is challenging to obtain in practice. On the compute side, adapting the full multimodal stack quickly becomes impractical because even small changes in the vision encoder force the entire language model to be retrained – a

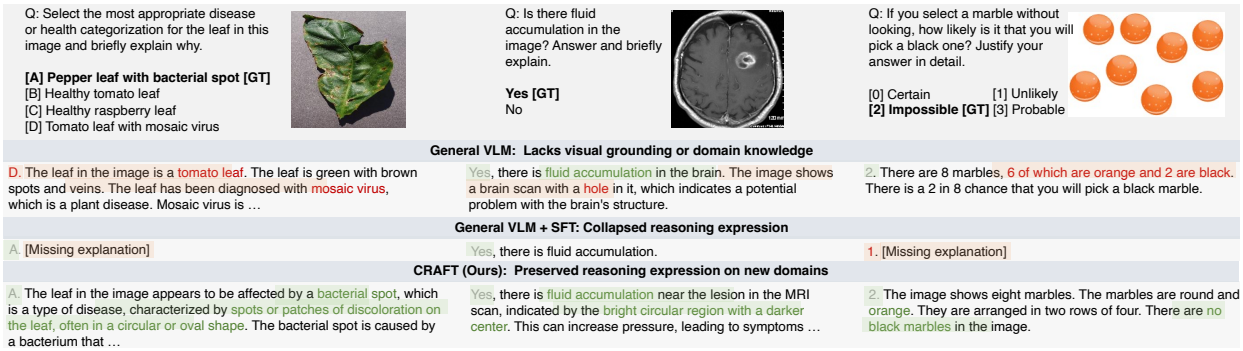

Figure 5.2: Examples from plant pathology [5], medical imaging [6], and abstract diagram understanding [7] compared across a general continuous LVLM [8], its PEFT-tuned variant, and our CRAFT model built on the discrete LVLM [9]. Correct and incorrect answers or explanations are marked in green and red, respectively.

process that can involve billions of parameters and is seldom necessary for the target task. These constraints motivate a fundamental question: *Can we adapt a large vision–language model without ever touching the original LLM?*

Achieving this goal requires bridging the representational gap between a frozen language model and an evolving vision encoder. The key challenge is maintaining consistent LLM comprehension of the new visual embedding space, while ensuring that the encoder's updates genuinely enhance multimodal reasoning rather than distort cross-modal alignment. Inspired by recent advances in discretized LVLMs [9, 143] which demonstrate on-par or even superior performance against their continuous counterparts, we propose **CRAFT** (**C**odebook **R**egul**A**ted **F**ine-**T**uning), a lightweight framework for vision encoder adaptation in discrete LVLMs that builds upon a shared "world language" between visual and textual modalities. Specifically, CRAFT discretizes continuous visual embeddings according to a shared, frozen codebook before feeding them into the language model. Conceptually, adapting the discrete vision encoder to a new domain involves learning how to select and arrange existing codebook entries so that they carry the visual evidence the language model needs to solve the target tasks. Once trained, the encoder speaks a stable "visual vocabulary" defined by the

codebook and can be plugged into any LVLM that uses the same codebook, effectively improving in-domain performance without retraining the language component. As illustrated in Figure 5.1, conventional fine-tuning methods shift the feature distribution, requiring costly re-alignment with the language model. In contrast, CRAFT introduces a discrete interface that anchors visual features to a shared codebook, allowing a single adapted encoder to work seamlessly across language models of different architectures without additional re-training or alignment.

To ensure these tokens are both faithful to the image and concise for reasoning, CRAFT adopts a composite training objective with a test-time regularizer. During training, a surrogate language model scores the joint image-text sequence and backpropagates the loss to the visual encoder, guiding it to select discrete tokens that are genuinely useful for the target domain. At test time, a discrete token pruning scheme removes redundant, background-like tokens and preserves the informative ones, so the LLM receives a compact visual summary instead of a cluttered grid of patches. This design is lightweight in both computation and data: training cost is mainly determined by the size of the surrogate model, which can be much smaller than the inference backbone, and the method requires only domain-specific supervision without curating extra instruction data to prevent forgetting. Yet, CRAFT outperforms both continuous-feature baselines and their PEFT counterparts, offering sharper domain-specific understanding without sacrificing the model's ability to follow instructions. Figure 5.2 reveals that general LVLMs often lack visual grounding or domain-specific knowledge in under-represented domains (*e.g.*, misidentifying plant diseases or misinterpreting a brain scan). Applying PEFT can improve task accuracy, but the language output collapses into rigid responses, and the model loses its linguistic capabilities. In contrast, CRAFT captures domain-specific visual cues (*e.g.*, identifying lesions in medical images) while keeping alignment stable through the shared discrete token interface, allowing the model to produce both accurate decisions and coherent explanations.

Our contributions are summarized as follows:

- We introduce CRAFT, a lightweight framework for adapting LVLMs that fine-tunes only the discrete vision encoder while keeping the language model frozen, enabling the adapted encoder to transfer across LLM backbones that share the same codebook.
- We develop a training and inference scheme that combines surrogate-based supervision and a test-time token pruning strategy, which together improve visual inputs to the LLM with domain-specific priors.
- Across ten benchmarks, CRAFT improves domain-specific performance by an average of 13.51% points while preserving strong instruction-following and explanatory abilities, outperforming continuous-feature and PEFT-based baselines.

5.2 Related Work

Domain Adaptation. Improving general-purpose LVLMs for specific domains through fine-tuning is an active research area [66,71,75,144–147], mostly performing supervised instruction fine-tuning [148] on domain-specific text–image datasets. Current methods usually update only the projector [66, 71], a small number of LLM parameters [75], or use LoRA-based fine-tuning [145–147, 149]. However, these methods can cause the model to forget its original instruction-following ability [150], even when using parameter-efficient approaches [151], if the new data are not well balanced. In addition, tuning the LLM or projector cannot fix perceptual errors from the vision encoder. For example, [71] found that errors from the visual backbone can lead to unrecoverable mistakes in the LVLM during fine-grained classification. In contrast, we propose a simple and lightweight method to adapt domains using expert discrete vision encoders, which can be directly plugged into LLMs that share the same visual codebook, improving in-domain performance without inducing catastrophic forgetting.

Adapted Vision Encoders. LVLMs usually rely on CLIP-style vision encoders to produce

visual embeddings that align with text to serve as input to the LLM [38, 58]. Although domain-specific fine-tuning of vision encoders is still underexplored, some studies have used extra vision encoders [152–154] or scaled existing ones [155,156] to boost general performance. Other works have fine-tuned CLIP encoders outside of LVLMs [157–161]. For example, [158] applied contrastive fine-tuning with an ℓ_2 regularization term to mitigate forgetting. These studies show that visual embeddings play a crucial role in LVLM quality, motivating us to focus on fine-tuning the vision encoder for domain-specific improvement.

Discrete LVLMs. Discrete LVLMs have gained popularity by unifying text and image generation in a single framework [9,143,162–164]. A major challenge is balancing the vision encoder's ability to both *understand* and *generate* visual content [163, 165–171], as these tasks rely on different levels of visual abstraction. In this work, we demonstrate another advantage of discrete vision tokens over continuous ones: they allow seamless interchange between expert vision encoders and LLM backbones that share a common visual codebook.

5.3 Methodology

Adapting the discrete vision encoder of an LVLM to a target domain is essentially learning how to select a subset of visual tokens from a shared codebook that best supports the language model in solving domain-specific tasks. The goal is to make the selected tokens both faithful to the image and concise in representation, so that they convey the most relevant visual information for downstream multimodal reasoning. To achieve this, CRAFT introduces a training objective and a test-time regularization scheme. During training, we fine-tune the vision encoder with a composite loss that guides it to produce tokens that are most helpful for the target domain when interpreted by the language model. At test time, we apply discrete vision token pruning to remove redundant tokens, improving the clarity and focus of the visual input. The two stages form a lightweight and modular approach to domain adaptation through a shared discrete interface, transferring domain knowledge to

the LVLM without changing the language model component.

5.3.1 Preliminaries

We denote the visual encoder as E_θ and a frozen codebook as $\mathcal{C} = \{c_k \in \mathbb{R}^d\}_{k=1}^K$. Given an image x, the encoder produces a grid of continuous features $z = E_\theta(x) \in \mathbb{R}^{N \times d}$, where each vector z_i corresponds to one of the N local patches. In a *continuous* latent model, these feature vectors are directly projected into the language model's embedding space. In a *discrete* latent model, each z_i is replaced by its nearest entry in the codebook:

$$\tilde{z}_i = \mathrm{q}(z_i) = c_{i^*}, \quad i^* = \arg\min_k \|z_i - c_k\|_2, \tag{5.1}$$

where $\mathrm{q}(\cdot)$ denotes quantization. This operation maps the continuous feature grid z to a sequence of discrete visual tokens $\tilde{z} = (\tilde{z}_1, \ldots, \tilde{z}_N)$, which can also be represented by their corresponding codebook indices $(i_1, i_2, \ldots, i_N)$, where each i_n identifies a visual token $c_{i_n} \in \mathcal{C}$. This idea was first introduced by VQVAE [172] and later refined in discrete vision–language models such as VQGAN [173] and VILA-U [9], with the latter extending the codebook into a hierarchical form that captures both coarse and fine visual semantics. The token embeddings $\tilde{z}$ are then mapped into the LLM's embedding space by a projector $g_\phi : \mathbb{R}^d \to \mathbb{R}^{d_{\mathrm{LM}}}$ and concatenated with text tokens for autoregressive processing by the language model.

5.3.2 Training Process

As shown in Figure 5.3, CRAFT adapts to a target domain by fine-tuning only the vision encoder, and discretizes the output visual embeddings with a frozen, shared discrete codebook. At inference time, the adapted encoder can be used with any LLM that shares the same codebook. Training is guided by commitment and contrastive losses, with an additional surrogate LLM providing multimodal supervision. The *surrogate alignment loss* updates the encoder through next-token prediction using a surrogate language model, providing end-to-end supervision from text. The *commitment loss* keeps the encoder's outputs close to

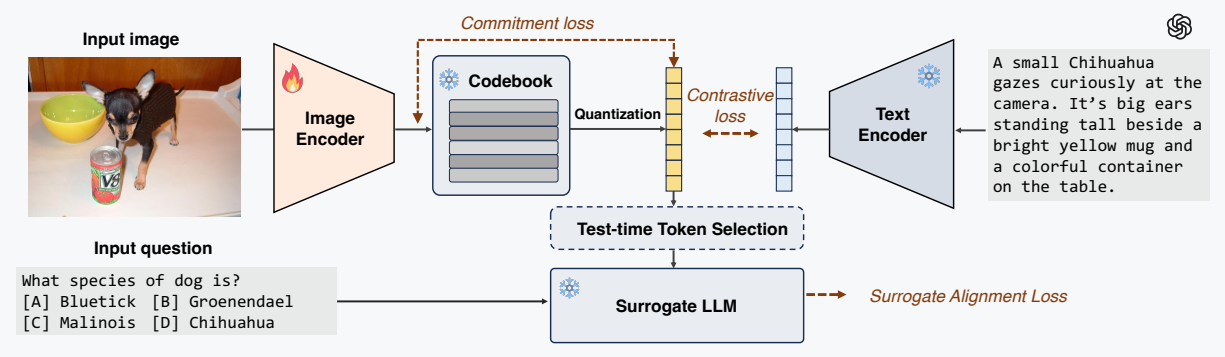

Figure 5.3: Overview of the CRAFT framework. A surrogate LLM coupled with a frozen codebook enables lightweight adaptation

their assigned codebook entries. The *contrastive loss* regularizes the encoder to preserve the semantic structure learned during pretraining, maintaining the overall quality of visual representations. Each objective is introduced in more detail below.

Surrogate Alignment Loss. The goal of adapting a vision encoder to a new domain is not just to make it "see" the right pixels, but to to generate discrete tokens that capture domain-specific cues interpretable by a language model. Standard visual objectives can refine image features, but they do not teach the encoder which details are actually useful for question answering. As illustrated in Figure 5.2, the vanilla model mistakes a bright region in a brain scan for a "hole," an interpretation that might seem plausible in general settings but is clearly invalid in medical imaging. This happens because the vision encoder passes low-level signals to the LLM that do not convey meaningful context. The surrogate alignment loss addresses this issue by letting a surrogate language model evaluate the joint image-text sequence and propagate gradients back through the encoder. It guides the encoder to select and arrange codebook tokens that the language model can naturally reason over for the target task. Note that the surrogate language model serves only as a bridge during training and is not used at test time, as our goal is to train the encoder to produce tokens that capture domain-relevant visual cues for reasoning, regardless of which LLM later interprets them.

Specifically, the surrogate alignment loss $\mathcal{L}_{\text{SAL}}$ is defined as an autoregressive loss computed using the surrogate language model $\mathcal{M}$:

$$\mathcal{L}_{\text{SAL}} = -\sum_{i=1}^{|y|} \log p_{\mathcal{M}}(y_i \mid y_{<i},\, t,\, \tilde{z})\,, \tag{5.2}$$

where $\tilde{z} = \mathrm{q}(E_\theta(x))$, and x, t, and y represent the input image, input prompt, and target output, respectively.

Commitment Loss. During training, the encoder produces continuous features that are mapped to their nearest codebook entries. Without regularization, these features can drift away from the assigned entries, weakening the connection between the encoder output and the codebook. As a result, the quantized tokens may no longer reflect the visual input faithfully. To address this, we apply a *commitment loss* that keeps the encoder outputs close to the codebook entries:

$$\mathcal{L}_{\text{commit}} = \quad E_\theta(x) - \mathrm{sg}\big[\mathrm{q}\big(E_\theta(x)\big)\big] \quad {}_2^2. \tag{5.3}$$

Here, $\mathrm{q}(\cdot)$ denotes quantization and $\mathrm{sg}(\cdot)$ is the stop-gradient operator. Note that unlike standard vector quantization methods [172], which also update the codebook during training, our codebook remains fixed, and the encoder is trained with the commitment loss that keeps its features well represented after quantization.

Contrastive Loss. Following common practice in the pre-training of discrete LVLMs, we further adopt a contrastive objective that leverages both image captions produced by captioning models and ground-truth text refined by an LLM. More specifically, for classification tasks, each caption is expended with: *"An image of a {domain}, specifically a {label}"*. For VQA tasks, each question–answer pair is rewritten as a declarative statement, *e.g.*, *"Q: What color is the car? A: Red"* becomes *"The car is red"*. These augmented captions help connect the image to both its visual content and the intended task. Each image x_i is paired with a set of text samples: $\mathcal{T}_i = \{t_i^{\text{GT}}\} \cup \{t_i^{(1)}, t_i^{(2)}, \ldots, t_i^{(L_i)}\}$, where t_i^{GT} is the ground-truth text and $t_i^{(l)}$ are generated captions [174, 175]. At each training step, we sample one sentence $\tilde{t}_i \in \mathcal{T}_i$

and encode it using a frozen text encoder to obtain $\mathbf{t}_i \in \mathbb{R}^{d_{\text{LM}}}$. The contrastive loss [176] is formulated as:

$$\mathcal{L}_{\text{con}}^{i} = -\log \sigma(s_{ii}) - \sum_{j:y_j \neq y_i} \log\big(1 - \sigma(s_{ij})\big), \tag{5.4}$$

where $\sigma(\cdot)$ is the sigmoid function, $s_{ij} = \tau \cos(\mathbf{v}_i, \mathbf{t}_j)$ is the cosine similarity between the image embedding $\mathbf{v}_i$ and text embedding $\mathbf{t}_j$, τ is a learnable temperature parameter, and y_i is the class label of image x_i.

CRAFT combines three complementary components into a single training objective:

$$\mathcal{L}_{\text{CRAFT}} = \lambda_{\text{con}}\mathcal{L}_{\text{con}} + \lambda_{\text{commit}}\mathcal{L}_{\text{commit}} + \mathcal{L}_{\text{SAL}}. \tag{5.5}$$

During optimization, only the vision encoder E_θ is updated with this loss; the surrogate language model and the projector remain frozen.

5.3.3 Test-Time Vision Token Pruning

As shown in Figure 5.4, discrete vision encoders transform an image into a grid of codebook token indices, each corresponding to a local patch. We see that many white background patches are mapped to the same codebook entry (ID 11745), whereas semantically meaningful objects such as the Chihuahua are represented by rarer token IDs (ID-5825). Given the redudancy across tokens and the fact that not all tokens are equally informative for a downstream task, tokens carrying redundant information can be pruned with little semantic loss. We introduce a simple test-time discrete visual token pruning method that adaptively removes redundant tokens, allowing the model to focus on salient concepts of the image, thereby improving its performance and efficiency.

Rarity-weighted allocation. Let n_k denote the number of tokens in the current image that are assigned to codebook entry k, with $\sum_k n_k = N$. From the training set, we estimate the global frequency $p_{\text{dom}}(k)$ for each entry, and define a rarity weight $\rho_k = 1/p_{\text{dom}}(k)$. We allocate a per-ID keep quota m_k according to $m_k = \lceil n_k \, \rho_k^{\gamma} \rceil$, where γ controls the overall

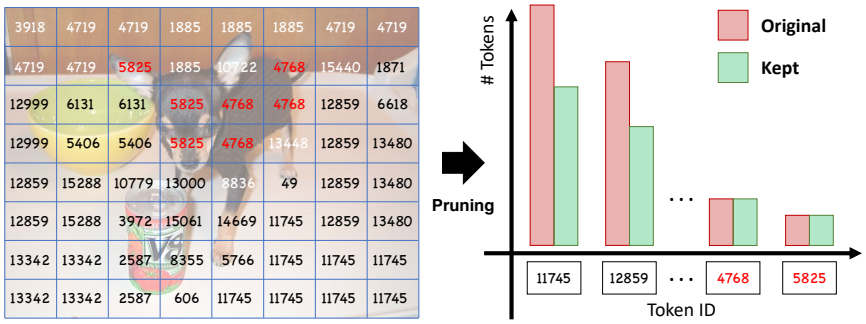

Figure 5.4: Illustration of the Token Pruning Process, where tokens are assigned rarity weights according to frequency during training

pruning strength. To satisfy a target budget $M < N$, we solve γ via a one-dimensional search so that the total number of kept tokens matches the budget: $\sum_k \lceil n_k \rho_k^\gamma \rceil \approx M$. Here, setting $\gamma = 0$ keeps all tokens, and increasing γ gradually prunes more redundant ones until the target budget is satisfied.

Within-ID selection. For each codebook entry k, exactly m_k of its n_k tokens are kept based on two cues. First, we favor tokens with large quantization residuals $e_j = \|z_j - c_k\|_2^2$ because they are harder to quantize and typically correspond to regions that are underrepresented during training [177]. Second, we prefer spatially isolated tokens by computing each token's average distance to its nearest neighbors of the same codebook ID and keeping those with the largest values. This encourages diverse spatial coverage and prunes densely clustered background patches.

Inference-time pruning. Collecting all selected tokens gives the final subset $S^\star$ with $|S^\star| \approx M$. We sort these tokens in their original raster order and feed them into the projector and language model. We define the *keep ratio* as M/N, meaning that roughly $N - M$ tokens are pruned. This pruning strategy produces a compact and semantically faithful visual representation that improves efficiency and robustness of LVLM.

Method	Train Surro.	Inference BB.	Vision Token	IconQA	OCRVQA	ScienceQA	VQARAD	EuroSAT	Flowers	Kvasir	PlantVillage	Cars	Dogs	Avg.
Zero-shot	-	VILA-U-7B	Discrete	10.17	66.40	65.59	35.67	69.15	75.80	29.20	43.83	72.50	82.40	55.07
Zero-shot	-	VILA-7B	Continuous	31.27	59.20	68.81	41.67	79.48	76.96	39.63	47.20	76.30	78.33	59.89
Vision FT	VILA-7B	VILA-7B	Continuous	39.97	62.27	67.42	43.67	67.35	79.02	37.54	62.13	86.80	71.43	61.76
Projector [71]	VILA-7B	VILA-7B	Continuous	34.47	57.37	67.91	42.67	65.61	63.03	41.36	49.00	75.57	69.03	56.60
LDIFS [158]	VILA-7B	VILA-7B	Continuous	28.97	59.31	67.18	41.92	63.20	63.70	36.93	48.54	74.79	65.30	54.98
	Qwen2-0.5B	VILA-U-7B	Discrete	30.50	66.77	68.86	39.00	78.87	72.31	32.55	70.70	78.80	74.11	61.25
	Qwen2.5-0.5B	VILA-U-7B	Discrete	31.90	65.90	68.81	42.33	75.76	72.18	29.65	57.53	76.7	72.01	59.28
Ours	Qwen2-1.5B	VILA-U-7B	Discrete	33.30	66.80	**69.15**	42.33	**83.11**	74.10	29.19	83.43	77.54	77.94	63.69
	Qwen2.5-3B	VILA-U-7B	Discrete	33.73	67.97	69.06	**46.33**	79.70	77.92	33.71	66.70	87.34	83.81	64.63
	VILA-U-7B	VILA-U-7B	Discrete	**48.50**	**68.13**	68.71	45.67	77.80	**80.26**	**41.93**	**77.27**	**92.74**	**84.77**	**68.58**

Table 5.1: Comparison of zero-shot and fine-tuned models across 10 classification and VQA datasets. We report accuracy (%) using the exact match criterion. Discrete LVLMs are trained with different surrogates sharing the same codebook and evaluated on the VILA-U-7B backbone

5.4 Experiments

5.4.1 Experimental Setup

Tasks and Metrics. We evaluate VQA accuracy and multimodal reasoning using a unified protocol. We consider the following benchmarks: *IconQA* [7], *OCRVQA* [64], *ScienceQA* [178], *VQARAD* [6], *EuroSAT* [179], *Flowers* [180], *Kvasir* [181], *PlantVillage* [5], *Cars* [182], and *Dogs* [183]. For classification tasks (*e.g.*, *Dogs*, *Cars*, *Flowers*), we cast them to multiple choice question answering format with four options. In addition, we randomly remove 20% of label categories during training to measure out-of-domain generalization. All datasets use short-answer formats. For accuracy evaluation, we cap decoding at 5 tokens and compute exact-match accuracy after normalizing case and whitespace. For reasoning evaluation, the model may generate up to 1024 tokens. To ensure reproducibility, we fix random seeds across all runs and report the average over 3 seeds. Unless mentioned otherwise, we use a keep ratio of 0.8.

Language Model Backbones. We conduct experiments using two LLM families: Llama2-7B (used in VILA-U-7B and VILA-7B) and Qwen2/2.5 (0.5B, 1.5B, 3B), each paired with

a SigLIP-Large (ViT-L/16@256) visual encoder. To build surrogates for our experimental setting, we follow the training procedure outlined in the VILA-U paper and train four Qwen-based LLM backbones to be compatible with the VILA-U vision codebook.

Training Details. Unless stated otherwise, we train the model with a global batch size of 24 using the AdamW optimizer. We apply a cosine learning rate schedule with a 3% warmup ratio, no weight decay. During fine-tuning, only the vision tower is updated, the projector and the language model remain frozen. We set λ_{commit}=0.1 and λ_{con} to be 0.1 for VQA datasets and 1.0 for visual grounding datasets like *Cats* and *Dogs*, *etc.*

5.4.2 Performance Comparison

We compare CRAFT with the following vision encoder tuning methods. In the *Zero-shot* setting, the model is tested without any fine-tuning. In the *Vision FT* setting, we update only the continuous image encoder and keep both the projector and the language model frozen, using an autoregressive loss. In the *Projector FT* setting, following Zhang et al. [71], we adjust only the projector weights, a method mainly studied for classification tasks. We also consider LDIFS [158], which reduces CLIP feature drift by adding an ℓ_2 regularization term. In later experiments, we compare against LLM LoRA [149], where the language model is fine-tuned using LoRA adapters.

The results are shown in Table 5.1. We first compare CRAFT with the zero-shot baselines with VILA and VILA-U, and find that CRAFT delivers large improvements across five surrogate models and ten datasets. For example, fine-tuning the vision encoder with only a 0.5B surrogate LLM boosts *PlantVillage* and *IconQA* accuracy by 26.97% and 20.33%, respectively. In the strongest setting, where both the surrogate and the inference backbone are 7B models, CRAFT surpasses existing continuous-feature fine-tuning approaches by 6.82% on average, even even though it is built on discrete models, which are often regarded as less competitive on standard benchmarks.

However, the gains are not consistent across datasets. One possible reason is that some domains are limited by the reasoning ability of the language model, such as *ScienceQA*, which reduces the benefit of having stronger visual features. Another factor is the capacity gap between the surrogate model and the inference backbone. On fine-grained datasets like *Flowers* and *Dogs*, using a small surrogate such as Qwen2-0.5B lowers accuracy to 72.31% and 74.11%, compared with the zero-shot accuracy of 75.80% and 82.40%. As shown in Table 5.3, the surrogate itself performs poorly (42.84% and 35.17%), suggesting that a weak surrogate may not provide the encoder with useful signals to learn features beyond what a strong backbone already supports. In contrast, using a stronger surrogate consistently improves performance, giving gains of 4.46% on *Flowers* and 2.37% on *Dogs*.

5.4.3 Evaluation of Instruction-Following

CRAFT does not need to re-align the LLM to accommodate the distribution shift of vision features. This not only reduces training cost but also improves data efficiency. For example, fine-tuning an LLM on datasets with short answers can cause it to forget how to follow instructions. As a result, the model may respond with short answers even when explicitly instructed to provide explanations.

To illustrate this phenomenon, we evaluate the reasoning capability of LVLMs by systematically modifying existing evaluation datasets. Specifically, we transform answer directives into justification requests by replacing phrases such as *"Answer in one letter only"* or *"Answer as succinctly as possible"* with *"Justify your answer."* We use *Claude Sonnet 4.5* as an automated evaluator, following the *scoring evaluation* protocol of [184]. Each model response is evaluated along five dimensions: **Correctness**, a binary measure of whether the chosen answer matches the ground truth; **Presence**, whether the response includes a genuine justification beyond a bare answer; **Relevance** (0–5), how directly the explanation addresses the question and chosen answer; **Faithfulness** (0–5), how well the reasoning is grounded in the given image and prompt while penalizing hallucinations; and **Verbosity** (0–5), penalizing

Model	Corr. ↑	Pres. ↑	Rel. ↑	Faith. ↑	Verb. ↓	Overall ↑
VQA-RAD						
VILA-U-7B	33.34	38.31	1.44	1.08	3.03	1.01
VILA-7B	42.99	**82.34**	**3.08**	1.79	3.41	3.17
VILA-LDIFS [158]	46.31	24.35	0.69	0.40	2.66	-0.24
VILA-Projector FT [71]	44.89	4.01	0.28	0.22	**2.21**	-0.61
VILA-LLM-LoRA [149]	44.65	6.34	0.26	0.25	2.97	-0.98
Ours	**47.34**	75.98	2.95	**1.99**	3.47	**3.21**
Flowers						
VILA-U-7B	78.48	22.68	0.74	0.68	2.50	0.17
VILA-7B	80.16	44.03	1.92	**1.81**	2.33	2.36
VILA-LDIFS [158]	81.31	18.01	0.82	0.62	2.56	0.16
VILA-Projector FT [71]	69.98	20.51	0.74	0.66	2.97	-0.09
VILA-LLM-LoRA [149]	**84.69**	2.18	0.08	0.04	**1.92**	-0.84
Ours	81.65	**48.65**	**2.09**	1.75	2.79	**2.45**
Cars						
VILA-U-7B	74.85	38.34	1.71	1.54	2.60	1.95
VILA-7B	80.15	**46.18**	1.55	1.29	2.88	1.40
VILA-LDIFS [158]	86.89	10.32	0.01	0.01	**1.85**	-0.91
VILA-Projector FT [71]	76.68	21.98	0.62	0.51	2.12	0.07
VILA-LLM-LoRA [149]	**94.18**	0.01	0.02	0.02	**1.85**	-0.89
Ours	92.68	41.98	**1.99**	**1.60**	2.81	**2.19**
Dogs						
VILA-U-7B	83.13	22.33	1.03	0.88	3.08	0.37
VILA-7B	78.34	26.66	1.24	1.03	2.93	0.81
VILA-LDIFS [158]	73.66	4.37	0.26	0.19	2.55	-0.83
VILA-Projector FT [71]	71.48	7.02	0.64	0.35	2.09	-0.05
VILA-LLM-LoRA [149]	**88.02**	2.37	0.09	0.05	**1.77**	-0.75
Ours	85.13	**38.65**	**1.39**	**1.13**	2.66	**1.19**

Table 5.2: Comparison of reasoning performance across four datasets across correctness (Corr), explanation presence (Pres), relevance (Rel), faithfulness (Faith), verbosity (Verb), and the combined Overall score.

unnecessary length. The judge is provided with a system prompt and receives the original task instruction, image, ground-truth answer, and candidate model output for each evaluation sample. Due to limited token budget, up to 350 samples per dataset are randomly selected for evaluation. The overall quality score is computed as

$$\text{Overall} = \text{Faithfulness} + \text{Relevance} - \text{Verbosity}/2.$$

Results are summarized in Table 5.2, comparing our method with the VILA-U and

Method	Train Surrogate	Inference Backbone	IconQA	OCRVQA	ScienceQA	VQARAD	EuroSAT	Flowers	Kvasir	PlantVillage	Cars	Dogs	Avg.
Zero-shot	-	Qwen2-0.5B	2.40	55.43	43.43	36.67	61.70	42.84	28.74	28.23	35.47	35.17	37.01
Ours	Qwen2-0.5B	Qwen2-0.5B	**30.97**	54.33	47.24	**39.00**	**80.68**	**52.29**	**38.11**	**76.43**	57.94	**67.17**	**54.42**
Ours	VILA-U-7B	Qwen2-0.5B	17.73	**60.27**	**47.44**	37.33	70.70	51.44	31.85	69.67	**57.70**	54.77	49.89
Zero-shot	-	Qwen2.5-0.5B	1.57	41.93	55.63	36.67	67.46	50.89	32.68	27.63	31.37	34.60	38.04
Ours	Qwen2-0.5B	Qwen2.5-0.5B	**26.13**	43.70	**56.91**	**42.67**	**81.46**	**61.88**	**46.68**	**70.27**	**50.10**	**65.54**	**54.53**
Ours	VILA-U-7B	Qwen2.5-0.5B	19.27	**51.73**	**56.91**	41.00	70.26	57.10	43.67	65.97	43.97	57.97	50.79
Zero-shot	-	Qwen2-1.5B	7.07	59.33	68.27	37.00	68.20	55.15	33.26	51.17	55.27	55.90	49.06
Ours	Qwen2-0.5B	Qwen2-1.5B	**22.93**	59.47	74.11	**42.67**	**82.87**	**69.48**	**44.37**	**82.53**	73.54	**80.54**	63.25
Ours	VILA-U-7B	Qwen2-1.5B	22.90	**68.67**	**74.41**	41.33	80.68	68.44	34.29	81.37	**85.87**	78.24	**63.62**
Zero-shot	-	Qwen2.5-3B	3.80	58.13	63.26	33.67	66.46	59.16	22.25	46.73	63.07	50.90	46.74
Ours	Qwen2-0.5B	Qwen2.5-3B	19.80	59.63	63.60	36.33	**84.11**	65.76	**47.15**	**81.97**	70.60	**70.84**	59.98
Ours	VILA-U-7B	Qwen2.5-3B	**25.03**	**69.17**	**63.70**	**39.67**	80.50	**66.02**	38.69	81.37	**77.64**	67.41	**60.92**
Zero-shot	-	VILA-U-7B	10.17	66.40	65.59	35.67	69.15	75.80	29.20	43.83	72.50	82.40	55.07
Ours	Qwen2-0.5B	VILA-U-7B	30.50	66.77	**68.86**	39.00	**78.87**	72.31	32.55	70.70	78.80	74.11	61.25
Ours	VILA-U-7B	VILA-U-7B	**48.50**	**68.13**	68.71	**45.67**	77.80	**80.26**	**41.93**	**77.27**	**92.74**	**84.77**	**68.58**

Table 5.3: Cross-LLM transfer with a shared discrete codebook. A vision encoder trained with one surrogate language model during training can be directly paired with different LLM backbones at test time.

VILA baselines, as well as continuous fine-tuning methods, including LDIFS [158], Projector FT [71], and LLM LoRA [149]. Zero-shot VILA-U-7B and VILA-7B achieve moderate correctness but provide weaker explanations. Continuous fine-tuning methods, especially LLM-LoRA, often raise correctness but severely degrading instruction-following and explanation quality (catastrophic forgetting), leading to low Pres, Rel, Faith, and Overall scores. CRAFT offers a better balance, matching or improving both correctness and explanation presence and quality over both VILA and its continuously fine-tuned variants.

5.4.4 Decoupling Vision and Language

CRAFT genuinely improves the visual representations learned by the adapted encoder, allowing LVLMs with very different language backbones to perform better in their target domains. Table 5.3 further supports this through a detailed study of CRAFT's modularity. We fine-tune the encoder using two surrogate models at opposite ends of the size spectrum, Qwen2-0.5B and VILA-U-7B, and run inference on five backbones that vary in size and

Training Efficiency			
Surrogate	**Method**	**VRAM (MiB)**	**Train time (min)**
VILA-7B	LoRA FT	27,725	5.09
VILA-7B	Vision FT	27,671	5.19
VILA-7B	Projector FT	27,383	4.46
Qwen2-0.5B	**Ours**	10,678	1.35
VILA-U-7B	**Ours**	20,754	2.27
Inference Efficiency (Pruning)			
Backbone	**Method**	**TFLOPs**	**Runtime (ms)**
VILA-U-7B	Zero-shot	2.56	52.40 ± 1.95
VILA-U-7B	**Ours**	2.15	48.92 ± 2.02

Table 5.4: Training and inference compute statistics. CRAFT training is lightweight when using a small surrogate, requiring substantially less memory and runtime compared with VILA-7B fine-tuning. With a keep ratio of 0.8, CRAFT's pruning achieves meaningful reductions in both FLOPs and inference latency.

architecture. Across these backbones, CRAFT delivers consistent and often substantial improvements. For example, when the Qwen2.5-3B backbone uses an encoder trained with the Qwen2-0.5B surrogate, its average accuracy increases from 44.26% in the zero-shot setting to 58.55%. Another interesting observation is that training with a stronger surrogate does not always yield the best results. For example, on the Qwen2.5-0.5B backbone, the encoder trained with the Qwen2-0.5B surrogate outperforms the one trained with the VILA-U-7B surrogate. We suspect this is because Qwen2 and Qwen2.5 share closer architectural designs than VILA-U, so they interpret visual tokens in more similar ways, making the adapted encoder transfer more smoothly.

Setting	VQARAD	Dogs	PlantVillage	IconQA	All-Avg
w.o. $\mathcal{L}_{\text{con}}$	45.13	71.57	45.69	47.24	52.41
w.o. $\mathcal{L}_{\text{commit}}$	10.33	16.53	25.97	3.31	14.54
w.o. $\mathcal{L}_{\text{SAL}}$	37.87	83.66	75.03	15.49	52.16
Ours (All)	**45.67**	**84.77**	**77.27**	**48.50**	**64.55**

Table 5.5: Loss ablation on the proposed training objectives on the VILA-U-7B backbone. Dropping any component degrades performance to varying degrees, highlighting the contribution of each part of the objective.

5.4.5 Runtime and Efficiency

Table 5.4 reports the training and inference efficiency of models under the same experimental settings. Training efficiency is measured on the *Dog* dataset using eight 40GB A100 GPUs with an effective batch size of 32. Inference efficiency is averaged over 1000 runs with a batch size of 1 across ten datasets. CRAFT training becomes especially lightweight when the surrogate model is small. For example, using Qwen2-0.5B as the surrogate cuts memory usage by 61.6% and reduces training time by 73.5% compared with other fine-tuning methods on VILA-7B, while still delivering competitive performance as shown in Table 5.2. At inference time, CRAFT benefits from the discrete structure of visual tokens by removing redundant tokens to reduce computation. With a keep ratio of about 0.8, token pruning lowers FLOPs by 16% and reduces runtime by 7% on average across datasets.

5.4.6 Ablation Studies

Ablation on Loss Components. Table 5.5 shows the results of removing each of the three loss components. When the commitment loss is removed, performance drops sharply, likely because the visual features are no longer aligned with the codebook, causing the quantization step to damage the learned representations. The contrastive and SAL losses complement each

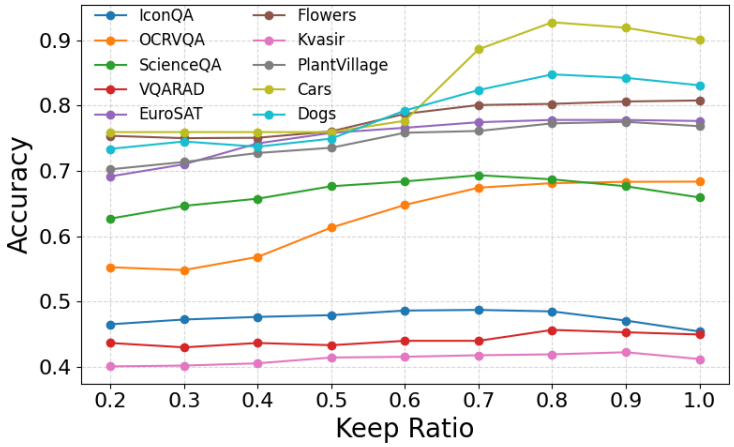

Figure 5.5: Accuracy of CRAFT encoder with VILA-U-7B backbone on various datasets versus the keep ratio. The keep ratio is the target budget M divided by the image token number N where 1.0 indicates no pruning.

other depending on the task. Removing the SAL loss lowers accuracy by 7.8% and 33.0% on *VQARAD* and *IconQA*, and removing the contrastive loss decreases accuracy by 13.2% and 31.6% on *Dogs* and *PlantVillage*. These results suggest that the contrastive loss helps more on tasks that rely on direct visual evidence, such as recognition and classification, and the SAL loss helps more on tasks that need deeper text understanding and reasoning.

Ablation on Keep Ratio. By removing redundant and task-irrelevant tokens at inference, the model can focus on the most informative regions of the image, which improves performance. Figure 5.5 shows how accuracy changes with different keep ratios across datasets. The effect of pruning varies a lot: datasets such as *Kvasir* and *VQARAD* show little accuracy drop even when only 20% of tokens are kept, while *Dogs* and *Cars* degrade more noticeably. This difference reflects how information is distributed: medical VQA often depends on a few localized patches (as in the middle example of Figure 5.2), whereas fine-grained classification relies on broader spatial details, making more tokens important. Empirically, light pruning with a keep ratio above 0.6 is consistently reliable, and we use 0.8 throughout this paper.

Ablation on Codebook Size. A primary limitation of discrete codebooks is their limited

Codebook Size	Dogs	Cars	Flowers	PlantVillage	IconQA	Avg
10%	28.00	30.93	29.04	43.83	29.59	30.60
25%	40.33	29.54	39.75	49.02	33.74	37.99
50%	61.30	64.58	61.74	62.12	40.66	55.11
100%	84.77	92.74	80.26	77.27	48.50	71.87

Table 5.6: Effect of codebook size on performance across benchmarks. Larger codebook size consistently improves accuracy, emphasizing the importance of both sufficient codebook capacity and a capable vision encoder for robust LVLM performance.

expressiveness in representing image features. In VILA-U, the default vision-token codebook contains 16384 entries. To assess the effect of codebook capacity, we uniformly downsample the codebook to 50%, 25%, and 10% of its original size and apply CRAFT using the VILA-U-7B backbone for both training and evaluation. As shown in Table 5.6, larger codebooks provide stronger representational capacity, leading to higher overall accuracy. Specifically, the average accuracy increases from 30.60% at 10% capacity to 71.87% at full size, clearly demonstrating the significance of expressiveness of encoder in LVLM performance.

5.5 Future Work

For future work, we would like to understand how changes to the shared codebook affect backward compatibility. While the codebook is assumed to be fixed in this paper, in practice, future models may introduce expanded or more detailed visual vocabularies [143, 185, 186]. If new entries are added while retaining the old ones, an open question is whether existing vision encoders will continue to function properly without additional retraining.

5.6 Conclusion

In this chapter, we introduced CRAFT, a lightweight and modular framework for LVLM specialization without altering the language model. CRAFT trains only the vision encoder while keeping a fixed discrete codebook, so the encoder learns to produce visual tokens that any LVLM sharing the same codebook can understand. This avoids modifying the language model and keeps its reasoning abilities intact. Through comprehensive experiments across a diverse set of visual and reasoning benchmarks, we demonstrated that CRAFT significantly improves domain-specific classification and VQA accuracy by good margins, while maintaining faithful reasoning capability. The shared codebook interface allows encoders fine-tuned with small surrogate models to transfer seamlessly to larger LLMs with different architectures, delivering strong results at a fraction of the training and inference cost and offering a practical solution for resource-constrained settings.

CHAPTER 6

Conclusions and Future Work

In conclusion, this dissertation adapts various multimodal neural networks to several inference time variations spanning both deployment and inference time. First, in Chapter 2 (FlexLoc), we address the challenge of deployment time sensor perspective shift in distributed multimodal localization systems. We discover that conditioning a portion of the multimodal network's weights on the world state (i.e., sensor pose) can drastically improve localization performance by almost 50%. Next, we move onto mitigating runtime variations while simultaneously considering an expanded set of models and tasks. Two commonly occurring runtime variations involve dynamic modality quality and variable computational resource availability. In Chapter 3 (ADMN), we demonstrate that LayerDrop training, coupled with a quality aware controller, can effectively mitigate these runtime variations. By allocating resources across a multimodal neural network according to the relative modality quality-of-information, ADMN matches the accuracy of state-of-the-art networks while reducing up to 75% of the floating-point operations. In Chapter 4 (SWAN), we broaden the scope of ADMN by considering another runtime variation – variable sample complexity – while extending the target task towards multi-object detection with LiDAR and camera. SWAN boasts an improved controller trained through NeuralSort gradient propagation, while introducing a layer-skipping module to address volatile sample complexity. Under the nuScenes dataset, SWAN outperforms related baselines by up to 9 NDS and 14.5 mAP while achieving up to a 49% reduction in FLOPs compared to a fully-provisioned network. SWAN demonstrates that navigating concurrent runtime variations requires careful consideration, as the interplay between data quality, sample complexity, and platform state demands highly

interlocked adaptive algorithms. Finally, in Chapter 5 (CRAFT), we extend the theme of model adaptation to the highest level of model complexity by adapting multimodal foundation models with billions of parameters. CRAFT adapts a vision-language foundation model to a variety of domain-specific visual oriented deployments ranging from satellite imagery classification to radiology question-answering. Through surrogate fine-tuning of a discrete vision encoder while maintaining a frozen codebook, CRAFT improves performance on visual-oriented tasks by an average of 13.51% points while requiring vastly fewer resources during model training.

The central, overarching theme throughout the thesis is how adaptation of the model itself can effectively address changes in the state of the world during inference time. Building off of this central theme, we propose the following future research directions.

Runtime Adaptation of Foundation Models: The transition from task-specific networks to multi-billion parameter foundation models amplifies the need for real-time computational scaling, as static architectures become deeply inefficient at such model sizes. The methodology presented in this dissertation addresses deployment-time alignment of multimodal foundation models and runtime adaptation of task-specific networks, lacking a comprehensive runtime framework for Large Vision-Language Models (LVLMs). These models have several distinct axes for adaptation, ranging from architectural depth [45], token length [187, 188], cross-modal density, and even quantization [189]. In addition, recent foundation models have integrated reasoning skills through "thinking tokens" [190, 191] as part of the autoregressive generation process, enabling the model to think through complex problems step-by-step.

Future work can develop methods of adaptation across each axis to discover those with the highest impact upon model accuracy and latency. Moreover, as discovered in SWAN, adapting a model along several axes often introduces greater complexity, as careful architectural design is required to convey information across adaptive modules (i.e., modules should be aware of the actions of others). Other meaningful optimizations can involve context-aware

modality token pruning or hierarchical token merging to eliminate cross-modal redundancy. Existing papers such as SparseVLM [187] and FastVLM [188] are representative works in this area. Realizing a fully adaptive foundation model across all of these avenues can greatly reduce their computational burden while still retaining similar accuracy.

Physical Sensor Adaptation: Building upon the overarching theme of performing adaptation to address inference-time variations, a profound paradigm shift involves moving from algorithmic resilience to closed-loop physical sensor adaptation. Traditional adaptive perception pipelines treat the physical world as an unalterable source of input data, focusing entirely on mutating the downstream network to ensure resilience. A more powerful yet complementary approach shifts some of the burden onto the sensors themselves – dynamically altering the internal capture parameters, position, or orientation of the hardware sensors to ensure the incoming data offers the maximum perceptual utility for the downstream model [192, 193]. This creates an interconnected framework where the physical sensors and the neural network components co-adapt in real time to navigate highly dynamic environments.

To realize this unified system, an upstream, low-latency control policy network can be trained to directly manipulate hardware sensor primitives – such as camera exposure time, analog ISO gain, lens aperture, or LiDAR vertical scanning frequencies. Instead of optimizing sensor configurations for human aesthetic or photographic preferences, the policy network is explicitly driven by task-specific validation metrics or proxy gradients to maximize the downstream model's confidence. For instance, in a low-light environment, the policy network can select an optimal set of hardware parameters (e.g., increasing ISO, aperture) to capture a sample with significantly higher signal-to-noise ratio. This high-quality sample can lead to more aggressive layer-skipping or budget reduction in the downstream neural network. Introducing co-adaptation of the sensor creates a additional control knob during runtime variations, where the sensor itself actively adjusts to capture the optimal input data according to the preferences of the downstream neural model.

When extending this closed-loop paradigm to multimodal architectures, there exist ad-

ditional avenues for future research. For instance, the unique attributes of one sensor can be leveraged to optimize the capture parameters of another. In a heterogeneous LiDAR-RGB configuration, LiDAR can instantly resolve a target's absolute position regardless of environmental lighting, allowing the system to compute a more precise focus point for the RGB camera, ensuring the downstream model receives the best image possible. Furthermore, these cross-modal dependencies can be exploited for power and latency optimization on resource-constrained platforms. For instance, data from an always-on, low-power radar can predetermine the optimal internal hardware primitives for a high-power camera. This eliminates the need to capture multiple exploratory camera samples, allowing the system to achieve the same predictive accuracy with a single, highly efficient capture.

Additional Inference Time Variations: While this dissertation addresses several forms of the most common inference-time variations, there remain several additional variations open for future research. First, in distributed networked systems, the communication infrastructure between edge devices can be highly volatile due to variable bandwidth at runtime [194]. Traditional multi-node deep learning infrastructures often assume a static channel state [26, 27], resulting in high latencies or rejected samples when network congestion occurs. This variation can be mitigated through the same concept of model adaptation. Instead of continually sharing static packets with constant size, the payload itself can be of dynamic size, enabling adaptation to a changing channel state. For example, when network bandwidth is plentiful, edge nodes can transmit raw data or dense, multi-scale feature embeddings to guarantee maximum performance. However, under severe channel degradation, an automated token-pruning or dimensionality-reduction algorithm can dynamically compress the payload to only the most salient embeddings. The ultimate predictor network that aggregates payloads across all nodes must also exhibit adaptive behavior, accepting and extracting crucial information from packets of drastically different sizes. By adapting according to the network congestion, the system prevents catastrophic pipeline freezes that can compromise latency-critical applications.

Another inference-time variation involves deployment on highly heterogeneous devices. Existing work such as LayerDrop [45] and Once-For-All [44] mainly focus on the issue of scaling *computation* – reducing the number of parameters on weaker platforms while utilizing the full network on more powerful hardware. Unfortunately, the staggeringly large diversity of possible deployment devices, from embedded boards with small AI accelerators (e.g., Raspberry Pi with AI Hat) to server grade Nvidia B200s, creates a much larger issue outside of naively scaling parameter counts. Owing to vastly different hardware specifications across cache size, RAM size, memory bandwidth, and GPU clock speed, certain neural operations that are highly efficient on one device may be highly inefficient on another. Thus, it is highly desirable to create a cross-platform profiling tool that identifies the hardware-specific bottlenecks in the main multimodal neural network, and automatically restructures the main network to maximize accuracy subject to some budget constraint on an arbitrary hardware device.

CHAPTER 7

Statement of Contributions

FlexLoc: Conditional Neural Networks for Zero-Shot Sensor Perspective Invariance in Object Localization with Distributed Multimodal Sensors [1]

I proposed the idea of leveraging conditional neural networks for the purpose of enabling perspective invariance and developed the Conditional Layer Normalization architecture, while Ziqi developed the Conditional Convolution method. Ziqi and I contributed equally to the indoor evaluations. I also want to thank Anvitha Mattapalli for her help in processing the GQ-Husky dataset and performing the outdoor evaluations. I am also thankful for the help of my collaborators Prof. Benjamin Marlin, Dr. Lance Kaplan, Dr. Jeffrey Twigg, and Dr. Colin Samplawski for their assistance on this project.

ADMN: A Layer-Wise Adaptive Multimodal Network for Dynamic Input Noise and Compute Resources [2]

I was solely responsible for the problem formulation, system design, and paper writing on ADMN as the lead author. I held a primary role in performing the evaluations, while Yuyang Yuan contributed towards implementing the MNS baseline and assisting in cross-dataset evaluation (e.g., MM-Fi, AVE). Additionally, I would like to thank Dr. Kang Yang and Dr. Lance Kaplan for providing feedback on the paper draft, as well as participating in discussions during the project's duration.

SWAN: World-Aware Adaptive Multimodal Networks for Runtime Variations [3]

As the lead author on this project, I was solely responsible for the problem formulation, system design, and paper writing. I held a primary role in performing the evaluations,

assisted by Shir-Kang Jin and Yuyang Yuan. Shir-Kang helped develop a more efficient version of the voxel feature encoder, and created splits of the nuScenes dataset to avoid training on rainy and dark nuScenes samples. Yuyang performed the Orin evaluations in Fig. 4.6.

Decoupling Vision and Language: Codebook Anchored Visual Adaptation [4]

This work was done during a summer internship at Amazon Web Services. I proposed the central idea of codebook aligned visual adaptation, while developing the surrogate fine-tuning approach supervised by contrastive loss, commitment loss, and surrogate cross-entropy loss. I performed evaluations on a subset of the datasets and models in Table 5.1 and Table 5.2. The rest of the authors took over the paper after my internship ended.

REFERENCES

[1] J. Wu, Z. Wang, X. Ouyang, H. L. Jeong, C. Samplawski, L. M. Kaplan, B. Marlin, and M. Srivastava, "Flexloc: Conditional neural networks for zero-shot sensor perspective invariance in object localization with distributed multimodal sensors," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 8563–8570.

[2] J. Wu, Y. Yuan, K. Yang, L. Kaplan, and M. Srivastava, "Admn: A layer-wise adaptive multimodal network for dynamic input noise and compute resources," 2025. [Online]. Available: https://arxiv.org/abs/2502.07862

[3] J. Wu, S.-K. S. Jin, Y. Yuan, M. Wigness, L. M. Kaplan, H. Qiu, and M. Srivastava, "Swan: World-aware adaptive multimodal networks for runtime variations," *arXiv preprint arXiv:2604.26181*, 2026.

[4] J. Wu, T. Zhao, C. Liu, J. Cai, Z. Zhang, Z. Li, A. Singh, X. Xu, M. Srivastava, and J. Wu, "Decoupling vision and language: Codebook anchored visual adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026, pp. 36 957–36 967.

[5] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, 2016.

[6] J. J. Lau, S. Gayen, A. Ben Abacha, and D. Demner-Fushman, "A dataset of clinically generated visual questions and answers about radiology images," *Scientific data*, 2018.

[7] P. Lu, L. Qiu, J. Chen, T. Xia, Y. Zhao, W. Zhang, Z. Yu, X. Liang, and S.-C. Zhu, "Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning," in *NeurIPS Track on Datasets and Benchmarks*, 2021.

[8] J. Lin, H. Yin, W. Ping, Y. Lu, P. Molchanov, A. Tao, H. Mao, J. Kautz, M. Shoeybi, and S. Han, "Vila: On pre-training for visual language models," 2023.

[9] Y. Wu, Z. Zhang, J. Chen, H. Tang, D. Li, Y. Fang, L. Zhu, E. Xie, H. Yin, L. Yi, *et al.*, "Vila-u: a unified foundation model integrating visual understanding and generation," *arXiv:2409.04429*, 2024.

[10] Z. Lu, "A theory of multimodal learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 57 244–57 255, 2023.

[11] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, "Imagebind: One embedding space to bind them all," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 15 180–15 190.

[12] T. Ophoff, K. Van Beeck, and T. Goedemé, "Exploring rgb+ depth fusion for real-time object detection," *Sensors*, vol. 19, no. 4, p. 866, 2019.

[13] T. Liang, H. Xie, K. Yu, Z. Xia, Z. Lin, Y. Wang, T. Tang, B. Wang, and Z. Tang, "Bevfusion: A simple and robust lidar-camera fusion framework," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10 421–10 434, 2022.

[14] A. Nagrani, S. Yang, A. Arnab, A. Jansen, C. Schmid, and C. Sun, "Attention bottlenecks for multimodal fusion," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 14 200–14 213.

[15] W. Kim, B. Son, and I. Kim, "Vilt: Vision-and-language transformer without convolution or region supervision," in *International conference on machine learning*. PMLR, 2021, pp. 5583–5594.

[16] M. Wang, J. Xing, B. Jiang, J. Chen, J. Mei, X. Zuo, G. Dai, J. Wang, and Y. Liu, "A multimodal, multi-task adapting framework for video action recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 6, 2024, pp. 5517–5525.

[17] L. Ma, Z. Lu, L. Shang, and H. Li, "Multimodal convolutional neural networks for matching image and sentence," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2623–2631.

[18] S. Liu, T. Kimura, D. Liu, R. Wang, J. Li, S. Diggavi, M. Srivastava, and T. Abdelzaher, "Focal: Contrastive learning for multimodal time-series sensing signals in factorized orthogonal latent space," *Advances in Neural Information Processing Systems*, vol. 36, pp. 47 309–47 338, 2023.

[19] D. Hazarika, Y. Li, B. Cheng, S. Zhao, R. Zimmermann, and S. Poria, "Analyzing modality robustness in multimodal sentiment analysis," *arXiv preprint arXiv:2205.15465*, 2022.

[20] C. Zhang, C. Zhang, Y. Guo, L. Chen, and M. Happold, "Motiontrack: End-to-end transformer-based multi-object tracking with lidar-camera fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 151–160.

[21] L. Solomon, W. Wang, and M. Hage, "Battlefield acoustic sensing, multimodal sensing, and networked sensing for intelligence, surveillance, and reconnaissance (isr) applications," Tech. Rep., 2015.

[22] T. Lewicki and K. Liu, "Multimodal wildfire surveillance with uav," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[23] Z. Liu, L. Cheng, A. Liu, L. Zhang, X. He, and R. Zimmermann, "Multiview and multimodal pervasive indoor localization," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 109–117.

[24] Y. Zhao, J. Xu, J. Wu, J. Hao, and H. Qian, "Enhancing camera-based multimodal indoor localization with device-free movement measurement using wifi," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1024–1038, 2019.

[25] M.-H. Amri, Y. Becis, D. Aubry, and N. Ramdani, "Indoor human/robot localization using robust multi-modal data fusion," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3456–3463.

[26] C. Samplawski, S. Fang, Z. Wang, D. Ganesan, M. Srivastava, and B. M. Marlin, "Heteroskedastic geospatial tracking with distributed camera networks," in *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, ser. UAI '23. JMLR.org, 2023.

[27] H. L. Jeong, Z. Wang, C. Samplawski, J. Wu, S. Fang, L. Kaplan, D. Ganesan, B. Marlin, and M. Srivastava, "Gtdm: An indoor geospatial tracking dataset with distributed multimodal sensors," in *arXiv preprint arXiv:2402.14136*, 2024.

[28] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[29] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[30] Y. Zhao, S. Kong, and C. Fowlkes, "Camera pose matters: Improving depth prediction by mitigating pose distribution bias," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 759–15 768.

[31] Y. Chen, T. Wang, Y. Lyu, Y. Hu, J. Li, T. Kimura, H. Zhao, Y. Hu, D. Kara, and T. Abdelzaher, "Spar: Self-supervised placement-aware representation learning for multi-node iot systems," *arXiv preprint arXiv:2505.16936*, 2025.

[32] H. Gao, R. Jiang, Z. Dong, J. Deng, Y. Ma, and X. Song, "Spatial-temporal-decoupled masked pre-training for spatiotemporal forecasting," *arXiv preprint arXiv:2312.00516*, 2023.

[33] S. Miao, L. Chen, and R. Hu, "Spatial-temporal masked autoencoder for multi-device wearable human activity recognition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 7, no. 4, pp. 1–25, 2024.

[34] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, "Modulating early visual processing by language," *Advances in neural information processing systems*, vol. 30, 2017.

[35] Z. Zhang, X. Li, Y. Li, Y. Dong, D. Wang, and S. Xiong, "Neural noise embedding for end-to-end speech enhancement with conditional layer normalization," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7113–7117.

[36] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[37] J. Yin, J. Shen, R. Chen, W. Li, R. Yang, P. Frossard, and W. Wang, "Is-fusion: Instance-scene collaborative fusion for multimodal 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 14 905–14 915.

[38] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, *et al.*, "Llava-onevision: Easy visual task transfer," *arXiv:2408.03326*, 2024.

[39] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *2023 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 2774–2781.

[40] J. Yan, Y. Liu, J. Sun, F. Jia, S. Li, T. Wang, and X. Zhang, "Cross modal transformer: Towards fast and robust 3d object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 18 268–18 278.

[41] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "DeeBERT: Dynamic early exiting for accelerating BERT inference," *arXiv preprint arXiv:2004.12993*, 2020.

[42] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "BERT Loses Patience: Fast and Robust Inference with Early Exit," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 18 330–18 341.

[43] L. Meng, H. Li, B.-C. Chen, S. Lan, Z. Wu, Y.-G. Jiang, and S.-N. Lim, "Adavit: Adaptive vision transformers for efficient image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 309–12 318.

[44] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791*, 2019.

[45] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," *arXiv preprint arXiv:1909.11556*, 2019.

[46] Z. Xue and R. Marculescu, "Dynamic multimodal fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2575–2584.

[47] R. Panda, C.-F. R. Chen, Q. Fan, X. Sun, K. Saenko, A. Oliva, and R. Feris, "Adamml: Adaptive multi-modal learning for efficient video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7576–7585.

[48] R. Gao, T.-H. Oh, K. Grauman, and L. Torresani, "Listen to look: Action recognition by previewing audio," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 457–10 467.

[49] Q. Cai, X. Liu, K. Zhang, X. Xie, X. Tong, and K. Li, "Acf: An adaptive compression framework for multimodal network in embedded devices," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5195–5211, 2024.

[50] R. T. Mullapudi, W. R. Mark, N. Shazeer, and K. Fatahalian, "Hydranets: Specialized dynamic architectures for efficient inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8080–8089.

[51] Y. Tian, J. Shi, B. Li, Z. Duan, and C. Xu, "Audio-visual event localization in unconstrained videos," in *ECCV*, 2018.

[52] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[53] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[54] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[55] NVIDIA Corporation, "NVIDIA TensorRT: High-performance deep learning inference sdk," 2024, accessed: 2026-03-04. [Online]. Available: https://developer.nvidia.com/tensorrt

[56] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[57] A. Grover, E. Wang, A. Zweig, and S. Ermon, "Stochastic optimization of sorting networks via continuous relaxations," *arXiv preprint arXiv:1903.08850*, 2019.

[58] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, *et al.*, "Qwen2. 5-vl technical report," *arXiv:2502.13923*, 2025.

[59] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, *et al.*, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," *arXiv preprint arXiv:2507.06261*, 2025.

[60] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[61] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *Advances in neural information processing systems*, vol. 35, pp. 25 278–25 294, 2022.

[62] S. Changpinyo, P. Sharma, N. Ding, and R. Soricut, "Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 3558–3568.

[63] X. Zhang, N. Rajabi, K. Duh, and P. Koehn, "Machine translation with large language models: Prompting, few-shot learning, and fine-tuning with QLoRA," in *Proceedings of the Eighth Conference on Machine Translation*, P. Koehn, B. Haddow, T. Kocmi, and C. Monz, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 468–481. [Online]. Available: https://aclanthology.org/2023.wmt-1.43/

[64] A. Mishra, S. Shekhar, A. K. Singh, and A. Chakraborty, "Ocr-vqa: Visual question answering by reading text in images," in *ICDAR*, 2019.

[65] D. A. Hudson and C. D. Manning, "Gqa: A new dataset for real-world visual reasoning and compositional question answering," in *CVPR*, 2019.

[66] G. Geigle, R. Timofte, and G. Glavaš, "African or european swallow? benchmarking large vision-language models for fine-grained object classification," *arXiv:2406.14496*, 2024.

[67] J. Fang, C.-T. Liu, J. Kim, Y. Bhedaru, E. Liu, N. Singh, N. Lipka, P. Mathur, N. K. Ahmed, F. Dernoncourt, R. Rossi, and H. Deilamsalehy, "Multi-LLM text summarization," in *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing - Natural Language Processing in the Generative AI Era*, G. Angelova, M. Kunilovskaya, M. Escribe, and R. Mitkov, Eds. Varna, Bulgaria: INCOMA Ltd., Shoumen, Bulgaria, Sept. 2025, pp. 352–362. [Online]. Available: https://aclanthology.org/2025.ranlp-1.43/

[68] H. Yuan and H. Zhang, "Understanding LLM reasoning for abstractive summarization," in *Findings of the Association for Computational Linguistics: ACL 2026*, M. Liakata, V. P. Moreira, J. Zhang, and D. Jurgens, Eds. San Diego, California, United States: Association for Computational Linguistics, July 2026, pp. 17 360–17 386. [Online]. Available: https://aclanthology.org/2026.findings-acl.859/

[69] W. Zhang, Y. Deng, B. Liu, S. Pan, and L. Bing, "Sentiment analysis in the era of large language models: A reality check," in *Findings of the Association for Computational Linguistics: NAACL 2024*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 3881–3906. [Online]. Available: https://aclanthology.org/2024.findings-naacl.246/

[70] Y. Feng, Y. Liu, S. Yang, W. Cai, J. Zhang, Q. Zhan, Z. Huang, H. Yan, Q. Wan, C. Liu, *et al.*, "Vision-language model for object detection and segmentation: A review and evaluation," *arXiv preprint arXiv:2504.09480*, 2025.

[71] Y. Zhang, A. Unell, X. Wang, D. Ghosh, Y. Su, L. Schmidt, and S. Yeung-Levy, "Why are visually-grounded language models bad at image classification?" *NeurIPS*, 2024.

[72] M. Yuksekgonul, F. Bianchi, P. Kalluri, D. Jurafsky, and J. Zou, "When and why vision-language models behave like bags-of-words, and what to do about it?" *arXiv:2210.01936*, 2022.

[73] J. Liu, W. Zeng, X. Zhang, Y. Wang, Z. Shan, and J. He, "On the perception bottleneck of vlms for chart understanding," *arXiv:2503.18435*, 2025.

[74] S. Tong, Z. Liu, Y. Zhai, Y. Ma, Y. LeCun, and S. Xie, "Eyes wide shut? exploring the visual shortcomings of multimodal llms," in *CVPR*, 2024, pp. 9568–9578.

[75] J. Chen, D. Yang, Y. Jiang, M. Li, J. Wei, X. Hou, and L. Zhang, "Efficiency in focus: Layernorm as a catalyst for fine-tuning medical visual language models," in *ACM International Conference on Multimedia*, 2024.

[76] B. Wu, W. Shi, J. Wang, and M. Ye, "Synthetic data is an elegant gift for continual vision-language models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2025, pp. 2813–2823.

[77] "Warehouse automation market worldwide," https://www.statista.com/topics/8741/warehouse-automation-market-worldwide/#topicOverview.

[78] C. Rohrig and S. Spieker, "Tracking of transport vehicles for warehouse management using a wireless sensor network," in *IEEE/RSJ IROS 2008*. IEEE, 2008, pp. 3260–3265.

[79] K. Lin, M. Chen, J. Deng, M. M. Hassan, and G. Fortino, "Enhanced fingerprinting and trajectory prediction for iot localization in smart buildings," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 3, pp. 1294–1307, 2016.

[80] X. Ouyang, X. Shuai, Y. Li, L. Pan, X. Zhang, H. Fu, X. Wang, S. Cao, J. Xin, H. Mok, *et al.*, "Admarker: A multi-modal federated learning system for monitoring digital biomarkers of alzheimer's disease," *arXiv preprint arXiv:2310.15301*, 2023.

[81] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[82] Y. Chen, J. Tang, C. Jiang, L. Zhu, M. Lehtomäki, H. Kaartinen, R. Kaijaluoto, Y. Wang, J. Hyyppä, *et al.*, "The accuracy comparison of three simultaneous localization and mapping (slam)-based indoor mapping technologies," *Sensors*, vol. 18, no. 10, p. 3228, 2018.

[83] C. Huang, Y. Tian, A. Kumar, and C. Xu, "Egocentric audio-visual object localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 22 910–22 921.

[84] J. Qiu, L. Chen, X. Gu, F. P.-W. Lo, Y.-Y. Tsai, J. Sun, J. Liu, and B. Lo, "Egocentric human trajectory forecasting with a wearable camera and multi-modal fusion," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8799–8806, 2022.

[85] M. J. Bocus and R. J. Piechocki, "Passive unsupervised localization and tracking using a multi-static uwb radar network," in *2021 IEEE Global Communications Conference*. IEEE, 2021, pp. 01–06.

[86] T. De Groot, "Localization and classification using an acoustic sensor network," Ph.D. dissertation, Delft University of Technology Delft, The Netherlands, 2010.

[87] A. Vakil, J. Liu, P. Zulch, E. Blasch, R. Ewing, and J. Li, "A survey of multimodal sensor fusion for passive rf and eo information integration," *IEEE Aerospace and Electronic Systems Magazine*, vol. 36, no. 7, pp. 44–61, 2021.

[88] K. Gadzicki, R. Khamsehashari, and C. Zetzsche, "Early vs late fusion in multimodal convolutional neural networks," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 2020, pp. 1–6.

[89] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler, "Vse++: Improving visual-semantic embeddings with hard negatives," *arXiv preprint arXiv:1707.05612*, 2017.

[90] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.

[91] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "inerf: Inverting neural radiance fields for pose estimation," in *IEEE/RSJ IROS 2021*, 2021, pp. 1323–1330.

[92] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407.

[93] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8844–8854.

[94] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[95] V. Pérez-Rosas, R. Mihalcea, and L.-P. Morency, "Utterance-level multimodal sentiment analysis," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 973–982.

[96] L. Zhao, H. Zhou, X. Zhu, X. Song, H. Li, and W. Tao, "Lif-seg: Lidar and camera image fusion for 3d lidar semantic segmentation," *IEEE Transactions on Multimedia*, vol. 26, pp. 1158–1168, 2024.

[97] Q. Tang, J. Liang, and F. Zhu, "A comparative review on multi-modal sensors fusion based on deep learning," *Signal Processing*, vol. 213, p. 109165, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165168423002396

[98] M. Salimibeni, Z. Hajiakhondi-Meybodi, P. Malekzadeh, M. Atashi, K. N. Plataniotis, and A. Mohammadi, "Iot-td: Iot dataset for multiple model ble-based indoor localization/tracking," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1697–1701.

[99] K. Ngamakeur, S. Yongchareon, J. Yu, and S. Islam, "Passive infrared sensor dataset and deep learning models for device-free indoor localization and tracking," *Pervasive and Mobile Computing*, vol. 88, p. 101721, 2023.

[100] Z. Kandylakis, K. Vasili, and K. Karantzalos, "Fusing multimodal video data for detecting moving objects/targets in challenging indoor and outdoor scenes," *Remote Sensing*, vol. 11, no. 4, p. 446, 2019.

[101] K. Nakamura, K. Nakadai, F. Asano, and G. Ince, "Intelligent sound source localization and its application to multimodal human tracking," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 143–148.

[102] M. J. Bocus, W. Li, S. Vishwakarma, R. Kou, C. Tang, K. Woodbridge, I. Craddock, K. McConville, *et al.*, "Operanet, a multimodal activity recognition dataset acquired from radio frequency and vision-based sensors," *Scientific data*, vol. 9, no. 1, p. 474, 2022.

[103] C. Torres, J. C. Fried, K. Rose, and B. S. Manjunath, "A multiview multimodal system for monitoring patient sleep," *IEEE Transactions on Multimedia*, vol. 20, no. 11, pp. 3057–3068, 2018.

[104] C. Robotics, "Husky a300 unmanned ground vehicle," https://clearpathrobotics.com/husky-a300-unmanned-ground-vehicle-robot/, n.d., accessed: 2025-06-02.

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[106] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[107] K. Lyu, Z. Li, and S. Arora, "Understanding the generalization benefit of normalization layers: Sharpness reduction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 689–34 708, 2022.

[108] S. Yang, L. Hou, H. Huang, C. Ma, P. Wan, D. Zhang, X. Chen, and J. Liao, "Direct-a-video: Customized video generation with user-directed camera movement and object motion," *arXiv preprint arXiv:2402.03162*, 2024.

[109] M. Ridolfi, A. Kaya, R. Berkvens, M. Weyn, W. Joseph, and E. D. Poorter, "Self-calibration and collaborative localization for uwb positioning systems: A survey and future research directions," *ACM Comput. Surv.*, vol. 54, no. 4, May 2021. [Online]. Available: https://doi.org/10.1145/3448303

[110] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, 2020.

[111] Y.-T. Chen, J. Shi, Z. Ye, C. Mertz, D. Ramanan, and S. Kong, "Multimodal object detection via probabilistic ensembling," in *European Conference on Computer Vision*. Springer, 2022, pp. 139–158.

[112] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 681–687.

[113] H. Alikhani, A. Kanduri, P. Liljeberg, A. M. Rahmani, and N. Dutt, "Dynafuse: dynamic fusion for resource efficient multimodal machine learning inference," *IEEE Embedded Systems Letters*, vol. 15, no. 4, pp. 222–225, 2023.

[114] M. A. Manzoor, S. Albarri, Z. Xian, Z. Meng, P. Nakov, and S. Liang, "Multimodality representation learning: A survey on evolution, pretraining and its applications," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 20, no. 3, Oct. 2023. [Online]. Available: https://doi.org/10.1145/3617833

[115] H. L. Jeong, Z. Wang, C. Samplawski, J. Wu, S. Fang, L. M. Kaplan, D. Ganesan, B. Marlin, and M. Srivastava, "Gdtm: An indoor geospatial tracking dataset with distributed multimodal sensors," *arXiv preprint arXiv:2402.14136*, 2024.

[116] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[117] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[118] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[119] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[120] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, F. Metze, and C. Feichtenhofer, "Masked autoencoders that listen," in *NeurIPS*, 2022.

[121] T. Luhman and E. Luhman, "High fidelity image synthesis with deep vaes in latent space," *arXiv preprint arXiv:2303.13714*, 2023.

[122] D. Zimmerer, S. A. Kohl, J. Petersen, F. Isensee, and K. H. Maier-Hein, "Context-encoding variational autoencoder for unsupervised anomaly detection," *arXiv preprint arXiv:1812.05941*, 2018.

[123] J. Yang, H. Huang, Y. Zhou, X. Chen, Y. Xu, S. Yuan, H. Zou, C. X. Lu, and L. Xie, "Mm-fi: Multi-modal non-intrusive 4d human dataset for versatile wireless sensing," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[124] R. Li, L.-F. Cheong, and R. T. Tan, "Heavy rain image restoration: Integrating physics model and conditional adversarial learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1633–1642.

[125] K. G. Lore, A. Akintayo, and S. Sarkar, "Llnet: A deep autoencoder approach to natural low-light image enhancement," *Pattern Recognition*, vol. 61, pp. 650–662, 2017.

[126] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," 2022. [Online]. Available: https://arxiv.org/abs/2203.11496

[127] G. Huang, D. Chen, T. Li, F. Wu, L. Van Der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," *arXiv preprint arXiv:1703.09844*, 2017.

[128] H. Li, H. Zhang, X. Qi, R. Yang, and G. Huang, "Improved techniques for training adaptive deep networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1891–1900.

[129] T. Beemelmanns, Q. Zhang, C. Geller, and L. Eckstein, "Multicorrupt: A multi-modal robustness dataset and benchmark of lidar-camera fusion for 3d object detection," in *2024 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2024, pp. 3255–3261.

[130] M. Ji, S. Zhang, and J. Yang, "Depthfusion: Depth-aware hybrid feature fusion for lidar-camera 3d object detection," *IEEE Transactions on Multimedia*, 2026.

[131] X. Chen, T. Zhang, Y. Wang, Y. Wang, and H. Zhao, "Futr3d: A unified sensor fusion framework for 3d detection," in *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 172–181.

[132] W. Jing, X. Chen, S. Nie, Z. Jiao, and J. Ma, "Hd-fusion: Hierarchical dynamic fusion of lidar-camera for robust 3-d object detection," *IEEE Transactions on Industrial Informatics*, 2026.

[133] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," *Advances in neural information processing systems*, vol. 34, pp. 13 937–13 949, 2021.

[134] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for fast test-time prediction," *arXiv preprint arXiv:1702.07811*, vol. 1, no. 3, 2017.

[135] Y. Li, C. Lyu, and H. Wang, "Freqexit: Enabling early-exit inference for visual autoregressive models via frequency-aware guidance," in *Advances in Neural Information Processing Systems*, D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, Eds., vol. 38. Curran Associates, Inc., 2025, pp. 103 763–103 788. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2025/file/95cb60b65d9f2c0cd78b567ffebde9e6-Paper-Conference.pdf

[136] L. Liu, B. Su, J. Jiang, G. Wu, C. Guo, C. Xu, and H. F. Yang, "Towards accurate and efficient 3d object detection for autonomous driving: A mixture of experts computing system on edge," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 25 903–25 913.

[137] B. Hu, L. Xu, J. Moon, N. J. Yadwadkar, and A. Akella, "Mosel: Inference serving using dynamic modality selection," 2023. [Online]. Available: https://arxiv.org/abs/2310.18481

[138] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[139] Z. Liu, X. Yang, H. Tang, S. Yang, and S. Han, "Flatformer: Flattened window attention for efficient point cloud transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 1200–1211.

[140] L. Fan, Z. Pang, T. Zhang, Y.-X. Wang, H. Zhao, F. Wang, N. Wang, and Z. Zhang, "Embracing single stride 3d object detector with sparse transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8458–8468.

[141] Y. Liu, T. Wang, X. Zhang, and J. Sun, "Petr: Position embedding transformation for multi-view 3d object detection," in *European conference on computer vision*. Springer, 2022, pp. 531–548.

[142] H. Lu, W. Liu, B. Zhang, B. Wang, K. Dong, B. Liu, J. Sun, T. Ren, Z. Li, H. Yang, *et al.*, "Deepseek-vl: towards real-world vision-language understanding," *arXiv:2403.05525*, 2024.

[143] L. Qu, H. Zhang, Y. Liu, X. Wang, Y. Jiang, Y. Gao, H. Ye, D. K. Du, Z. Yuan, and X. Wu, "Tokenflow: Unified image tokenizer for multimodal understanding and generation," in *CVPR*, 2025.

[144] H. He, G. Li, Z. Geng, J. Xu, and Y. Peng, "Analyzing and boosting the power of fine-grained visual recognition for multi-modal large language models," *arXiv:2501.15140*, 2025.

[145] Y. Li, Y. Tian, Y. Huang, W. Lu, S. Wang, W. Lin, and A. Rocha, "Fakescope: Large multimodal expert model for transparent ai-generated image forensics," *arXiv:2503.24267*, 2025.

[146] B. Li, W. Huang, Y. Shen, Y. Wang, S. Lin, J. Lin, L. You, Y. Zhang, K. Li, X. Sun, *et al.*, "Llava-radz: Can multimodal large language models effectively tackle zero-shot radiology recognition?" *arXiv:2503.07487*, 2025.

[147] G. Liu, J. He, P. Li, G. He, Z. Chen, and S. Zhong, "Pefomed: Parameter efficient fine-tuning of multimodal large language models for medical imaging," *arXiv:2401.02797*, 2024.

[148] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, *et al.*, "Scaling instruction-finetuned language models," *JMLR*, 2024.

[149] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, 2022.

[150] W. Ren, X. Li, L. Wang, T. Zhao, and W. Qin, "Analyzing and reducing catastrophic forgetting in parameter efficient tuning," *arXiv:2402.18865*, 2024.

[151] Y. Lin, X. Ma, X. Chu, Y. Jin, Z. Yang, Y. Wang, and H. Mei, "Lora dropout as a sparsity regularizer for overfitting control," *arXiv:2404.09610*, 2024.

[152] L. Shen, G. Chen, R. Shao, W. Guan, and L. Nie, "Mome: Mixture of multimodal experts for generalist multimodal large language models," *NeurIPS*, 2024.

[153] Z. Lin, C. Liu, R. Zhang, P. Gao, L. Qiu, H. Xiao, H. Qiu, C. Lin, W. Shao, K. Chen, *et al.*, "Sphinx: The joint mixing of weights, tasks, and visual embeddings for multi-modal large language models," *arXiv:2311.07575*, 2023.

[154] D. Jiang, Y. Liu, S. Liu, J. Zhao, H. Zhang, Z. Gao, X. Zhang, J. Li, and H. Xiong, "From clip to dino: Visual encoders shout in multi-modal large language models," *arXiv:2310.08825*, 2023.

[155] D. Fan, S. Tong, J. Zhu, K. Sinha, Z. Liu, X. Chen, M. Rabbat, N. Ballas, Y. LeCun, A. Bar, *et al.*, "Scaling language-free visual representation learning," *arXiv:2504.01017*, 2025.

[156] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, *et al.*, "Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks," in *CVPR*, 2024.

[157] J. Han, Z. Lin, Z. Sun, Y. Gao, K. Yan, S. Ding, Y. Gao, and G.-S. Xia, "Anchor-based robust finetuning of vision-language models," in *CVPR*, 2024, pp. 26 919–26 928.

[158] J. Mukhoti, Y. Gal, P. H. Torr, and P. K. Dokania, "Fine-tuning can cripple your foundation model; preserving features may be the solution," *arXiv:2308.13320*, 2023.

[159] N.-A. Ypsilantis, K. Chen, A. Araujo, and O. Chum, "Infusing fine-grained visual knowledge to vision-language models," in *ICCV*, 2025.

[160] S. Sanyal, H. Prairie, R. Das, A. Kavis, and S. Sanghavi, "Upweighting easy samples in fine-tuning mitigates forgetting," *arXiv:2502.02797*, 2025.

[161] T. Zhao, X. Chen, Z. Li, J. Fang, D. An, X. Xu, Z. Tu, and Y. Xing, "Salient concept-aware generative data augmentation," *arXiv:2510.15194*, 2025.

[162] Y. Jin, K. Xu, L. Chen, C. Liao, J. Tan, Q. Huang, B. Chen, C. Lei, A. Liu, C. Song, *et al.*, "Unified language-vision pretraining in llm with dynamic discrete visual tokenization," *arXiv:2309.04669*, 2023.

[163] H. Lin, T. Wang, Y. Ge, Y. Ge, Z. Lu, Y. Wei, Q. Zhang, Z. Sun, and Y. Shan, "Toklip: Marry visual tokens to clip for multimodal comprehension and generation," *arXiv:2505.05422*, 2025.

[164] R. Xie, C. Du, P. Song, and C. Liu, "Muse-vl: Modeling unified vlm through semantic discrete encoding," in *ICCV*, 2025.

[165] X. Wang, X. Zhang, Z. Luo, Q. Sun, Y. Cui, J. Wang, F. Zhang, Y. Wang, Z. Li, Q. Yu, *et al.*, "Emu3: Next-token prediction is all you need," *arXiv:2409.18869*, 2024.

[166] C. Team, "Chameleon: Mixed-modal early-fusion foundation models," *arXiv:2405.09818*, 2024.

[167] C. Zhou, L. Yu, A. Babu, K. Tirumala, M. Yasunaga, L. Shamis, J. Kahn, X. Ma, L. Zettlemoyer, and O. Levy, "Transfusion: Predict the next token and diffuse images with one multi-modal model," *arXiv:2408.11039*, 2024.

[168] Y. Zhao, F. Xue, S. Reed, L. Fan, Y. Zhu, J. Kautz, Z. Yu, P. Krähenbühl, and D.-A. Huang, "Qlip: Text-aligned visual tokenization unifies auto-regressive multimodal understanding and generation," *arXiv:2502.05178*, 2025.

[169] Z. Chen, C. Wang, X. Chen, H. Xu, R. Huang, J. Zhou, J. Han, H. Xu, and X. Liang, "Semhitok: A unified image tokenizer via semantic-guided hierarchical codebook for multimodal understanding and generation," *arXiv:2503.06764*, 2025.

[170] J. Xie, Z. Yang, and M. Z. Shou, "Show-o2: Improved native unified multimodal models," *arXiv:2506.15564*, 2025.

[171] W. Song, Y. Wang, Z. Song, Y. Li, H. Sun, W. Chen, Z. Zhou, J. Xu, J. Wang, and K. Yu, "Dualtoken: Towards unifying visual understanding and generation with dual visual vocabularies," *arXiv:2503.14324*, 2025.

[172] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *NeurIPS*, 2017.

[173] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *CVPR*, 2021.

[174] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *ICML*, 2023.

[175] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *arXiv:2304.10592*, 2023.

[176] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *ICCV*, 2023.

[177] A. A. Neloy and M. Turgeon, "A comprehensive study of auto-encoders for anomaly detection: Efficiency and trade-offs," *Machine Learning with Applications*, 2024.

[178] P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan, "Learn to explain: Multimodal reasoning via thought chains for science question answering," in *NeurIPS*, 2022.

[179] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE JSTARS*, 2019.

[180] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *CVPR*, 2006.

[181] K. Pogorelov, K. R. Randel, C. Griwodz, S. L. Eskeland, T. de Lange, D. Johansen, C. Spampinato, D.-T. Dang-Nguyen, M. Lux, P. T. Schmidt, M. Riegler, and P. Halvorsen, "Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection," in *MMSys*, 2017.

[182] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *ICCV Workshop*, 2013.

[183] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, "Novel dataset for fine-grained image categorization: Stanford dogs," in *CVPR Workshop*, 2011.

[184] D. Chen, R. Chen, S. Zhang, Y. Wang, Y. Liu, H. Zhou, Q. Zhang, Y. Wan, P. Zhou, and L. Sun, "Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark," in *ICML*, 2024.

[185] C. Ma, Y. Jiang, J. Wu, J. Yang, X. Yu, Z. Yuan, B. Peng, and X. Qi, "Unitok: A unified tokenizer for visual generation and understanding," *arXiv:2502.20321*, 2025.

[186] S. Long, Q. Zhou, X. Jiang, C. Ying, L. Ma, and Y. Luo, "Domain generalization via discrete codebook learning," *arXiv:2504.06572*, 2025.

[187] Y. Zhang, C.-K. Fan, J. Ma, W. Zheng, T. Huang, K. Cheng, D. Gudovskiy, T. Okuno, Y. Nakata, K. Keutzer, *et al.*, "Sparsevlm: Visual token sparsification for efficient vision-language model inference," *arXiv preprint arXiv:2410.04417*, 2024.

[188] P. K. A. Vasu, F. Faghri, C.-L. Li, C. Koc, N. True, A. Antony, G. Santhanam, J. Gabriel, P. Grasch, O. Tuzel, *et al.*, "Fastvlm: Efficient vision encoding for vision language models," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 19 769–19 780.

[189] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[190] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[191] Q. Team, "Qwen3. 5-omni technical report," *arXiv preprint arXiv:2604.15804*, 2026.

[192] E. Baek, K. Park, J. Kim, and H.-S. Kim, "Unexplored faces of robustness and out-of-distribution: Covariate shifts in environment and sensor domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 22 294–22 303.

[193] E. Baek, S. Han, T. Gong, and H.-S. Kim, "Adaptive camera sensor for vision models," *arXiv preprint arXiv:2503.02170*, 2025.

[194] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.