Modality Plug-and-Play: Runtime Modality Adaptation in LLM-Driven Autonomous Mobile Systems

Kai Huang*
University of Pittsburgh
USA
k.huang@pitt.edu

Xiangyu Yin*
University of Pittsburgh
USA
eric.yin@pitt.edu

Heng Huang
University of Maryland
USA
heng@umd.edu

Wei Gao
University of Pittsburgh
USA
weigao@pitt.edu

Abstract

Multimodal reasoning by LLMs is critical to autonomous mobile systems, but the growing diversity of input data modalities prevents incorporating all modalities into LLMs. Instead, only the useful modalities should be adaptively involved at runtime, based on the current environmental contexts and task requirements. Existing work on runtime modality adaptation uses fixed connections between data encoders and LLM's input layer, but results in high training costs and ineffective cross-modal interaction. In this paper, we present MPnP, a new modality adaptation technique that connects data encoders to a flexible set of last LLM blocks and makes such latent connections fully trainable at runtime. Evaluation results show that MPnP has high compute and data efficiency, with 3.7× FLOPs reduction and 30% memory usage reduction compared to best baselines. It requires only few hundreds of training samples at runtime, and completes modality adaptation within few minutes on weak devices.

CCS Concepts

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Computing methodologies** → **Artificial intelligence**.

Keywords

Autonomous Systems, Large Language Models, Multimodal Reasoning, Modality Adaptation, Cross-Modal Interaction

*Both authors contributed equally to the paper.

ACM MOBICOM '25, Hong Kong, China

ACM ISBN 979-8-4007-1129-9/2025/11
https://doi.org/10.1145/3680207.3723491

ACM Reference Format:
Kai Huang, Xiangyu Yin, Heng Huang, and Wei Gao. 2025. Modality Plug-and-Play: Runtime Modality Adaptation in LLM-Driven Autonomous Mobile Systems. In *The 31st Annual International Conference on Mobile Computing and Networking (ACM MOBICOM '25), November 4–8, 2025, Hong Kong, China.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3680207.3723491

1 Introduction

Multimodal Large Language Models (LLMs) have been widely applied in autonomous mobile systems, including self-driving vehicles, drones and robots. In these systems, LLMs are used to perceive the physical world from input data and allow intelligent embodied agents' multimodal reasoning based on the environmental contexts and task needs [33, 63]. Such capability of multimodal reasoning has significantly enhanced the capability of autonomous navigation and actions in multiple industry systems, including Google's PaLM-E [19], Meta's Habitat [57] and Qualcomm's EDGI [7].

The major challenge of democratizing such multimodal reasoning in practical systems, however, is the rapidly growing diversity of input data modalities available on these systems, which has recently expanded from texts, RGB images [24, 32] and audios [76] to LiDAR point clouds [26], mmWave radar images [36, 77], ultrasound signals [48] and motion sensory data [59]. For each modality, a pre-trained data encoder is used first to extract task-relevant features, and these features from multiple modalities are fused into LLM for perception and reasoning. However, since the volumes of input data in many new modalities are very large[1], incorporating all modalities on resource-constrained autonomous mobile systems is computationally expensive or even infeasible.

To address this challenge and minimize the on-device computing cost, only the useful modalities should be adaptively involved at runtime, because the usefulness of different

[1]For example, LiDAR point cloud [58] and mmWave radar signal [37] are high-dimensional data in gigabytes per sec when continuously generated.

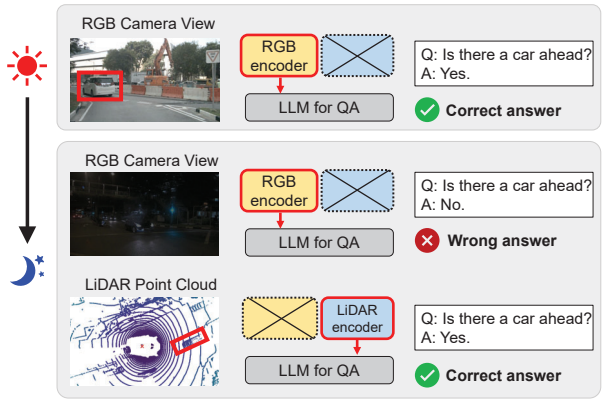


Figure 1: Runtime modality adaptation with environmental context changes: In daytime, using RGB images can sufficiently enable visual perception for LLM. However, it cannot provide useful information in nighttime with low visibility, and we switch to LiDAR to provide distance perception for accurate QA.

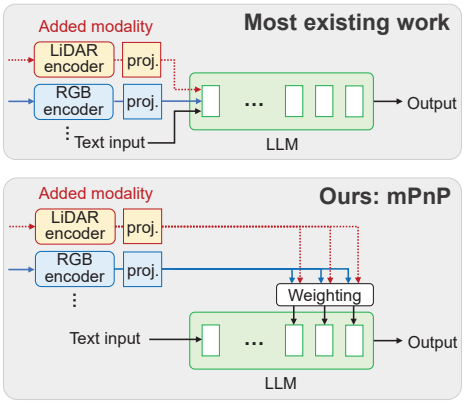


Figure 2: Existing work vs. MPnP for modality adaptation, where encoders are connected to LLM via trainable projectors.

modalities could greatly vary when the environmental contexts change, even on the same system. An example of such runtime modality adaptation for multimodal question answering (QA) tasks in autonomous driving is shown in Figure 1. Recent autonomous system efforts, such as DeepMind's AutoRT [2], have allowed real-time collection of training data, which makes runtime modality adaptation the key enabler for using multimodal LLM to enhance the capability of autonomous mobile systems in the future.

A straightforward approach to such runtime modality adaptation is to pre-train a set of single-modal encoders and projectors offline and selectively deploy them at runtime. However, adding or removing encoders without runtime retraining can introduce alignment issues across different modalities and lead to significant performance degradation. Alternatively, one can jointly retrain all the data encoders with LLM when a new modality is inserted or an existing modality is removed, to align input modalities with the natural language domain [8, 19, 76], but doing so is too expensive for runtime. Instead, we can freeze encoders and LLM, but only train the inserted projectors in between. As shown in Figure 2 - Top, existing work connects encoders to LLM's *input layer* through a trainable projector [43, 85], and then applies parameter-efficient fine-tuning [27, 30, 31, 49, 65] to improve model accuracy. However, they still require backpropagation throughout the entire LLM and incur large training costs. The connection also requires the projected multimodal features to be aligned with LLM input layer's text embedding, but such alignment can be inefficient in bridging the semantic gap across modalities, because text representations in early LLM blocks are too superficial to match input details [62]. Recent work [61] explored connections to LLM's *intermediate layers*, but such connections are arbitrarily decided offline and could hence impair the model accuracy in dynamic runtime conditions.

To address these limitations, in this paper we present *Modality Plug-and-Play (MPnP)*, a new technique that aims to maximize the compute and data efficiency of runtime modality adaptation, i.e., to retain the LLM's task performance with the minimum computing costs and requirement of training data. As shown in Figure 2 - Bottom, our basic idea is to connect unimodal encoders to a flexible set of last LLM blocks and make such latent connections fully trainable at runtime. In this way, MPnP can flexibly shorten the backpropagation depth and hence reduce the gradient propagation cost, by shifting the modality insertion points closer to LLM's output layer. We can also adjust the amount of LLM blocks being connected to balance between accuracy and training cost, and optimize the efficiency of cross-modal interaction by controlling the amount of information injected in each connection with a trainable weighting module at runtime.

In this paper, we assume that the need for modality adaptation and the modalities to be used can be decided by either the human user or direct interpretation of sensory data, and we focus on how to enable the needed modality adaptation with high compute and data efficiency. In practice, such decisions in most autonomous mobile systems can be made by applying simple and pre-defined criteria. For example in the autonomous driving scenario in Figure 1, the need for adapting from RGB camera to LiDAR modality can be decided from the ambient lighting conditions. The LiDAR modality, similarly, is not applicable in rainy or foggy weather.

The major challenge of MPnP design, however, is how to ensure efficient injection of multimodal information to LLM, so that sufficient information can be available for multimodal reasoning even if only some LLM blocks are connected. To address this challenge, we enforce full non-linearity in key &

value aligners that project encoder outputs to K-V pairs, to ensure expressivity of feature representations. Afterwards, we project multimodal tokens via plug-in projectors into MHA modules of LLM blocks, so that trainable weights of such projection can be directly applied to multimodal tokens. In this way, we can effectively minimize the semantic gap across different modalities when being connected to LLM.

We implemented MPnP on NVidia Jetson AGX Orin devices, and evaluated its performance with multiple LLMs and vision-QA (VQA) datasets, because VQA is the most representative LLM-driven task in autonomous mobile systems [19] and more complicated uses of multimodal generative AI (e.g., behavior planning on humanoid robots [21]) are derived from VQA. Based on experiment results, our major contributions are as follows:

- To our best knowledge, MPnP is the first work that enables runtime modality adaptation in LLM-driven autonomous mobile systems.
- MPnP has high compute efficiency. Compared to the existing schemes, its runtime training reduces the FLOPs by 3.7× and GPU memory usage by 30% while retaining on-par task accuracy, or it can improve the task accuracy by up to 4% with the same compute budget. Its compute efficiency becomes higher when larger LLMs or more complicated system tasks are involved.
- MPnP is data efficient. It achieves high compute efficiency with only few hundreds of training samples, well satisfying the timing and resource constraints of runtime modality adaptation.
- With such high compute and data efficiency, MPnP completes modality adaptation within few minutes on weak edge devices, up to 3.75× faster than the existing schemes when measured in wall-clock time.

2 Background & Motivation

We first motivate the need for runtime modality adaptation with quantitative studies. Then, we discuss the possibilities of connecting data encoders to LLM, and motivate our design of reducing the runtime training costs by showing the necessity of shortening the backpropagation depth.

In the rest of this paper, we focus on decoder-only LLMs, which is the dominant LLM architecture due to stronger generative power [13] and has been widely adopted by most existing LLMs, ranging from BLOOMZ-1.1B [53] and OPT-1.3B [84] to GPT3-175B [10] and Llama3-405B [20].

2.1 Need for Runtime Modality Adaptation

To tackle the possible environmental context changes in autonomous mobile systems (e.g., shown in Figure 1), one intuitive solution is to include all the available input data modalities, so that sufficient information can be provided for LLM's reasoning in any case. However, as shown in Table 1, this method incurs high amounts of GPU memory usage and computing cost, mainly due to large volume of input data sizes. In particular, data encoders, especially when taking high-volume data as input (e.g., for LiDAR), incur higher GPU memory usage and computing costs in both inference and training, compared to those incurred by LLM. Such high costs make it difficult to simultaneously involving all data modalities, especially those with large input data, on resource-constrained mobile devices used in autonomous mobile systems (e.g., autonomous drones and robots) [29, 71].

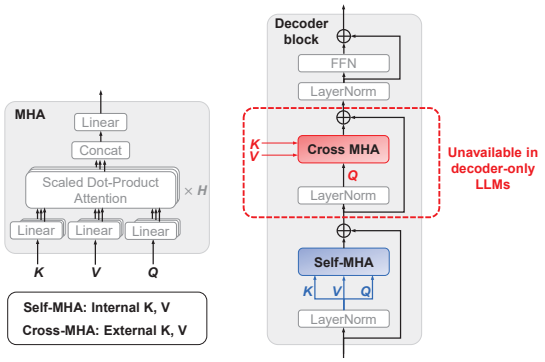


Figure 3: LLM architecture. Left: In MHA, K and V sequences can be provided either internally (self-MHA) or externally (cross-MHA). Right: Cross-MHA is unavailable in decoder-only LLMs.

Another practical difficulty of involving all data modalities is the lack of multimodal training data. Such training data is necessary for retraining the encoders with LLM to align input modalities, but is usually unavailable due to the increasing number of data modalities and difficulty for data labeling. Instead, a better option is to adaptively involve only the useful modalities at runtime, so that only input data of selected modalities is needed. A key research issue, then, is how to minimize the runtime computing costs of such modality adaptation, which require runtime training efforts to connect the corresponding data encoders to LLM.

2.2 Connecting Data Encoders to LLM

To better explain how modality adaptation can be done in transformer-based LLMs, we will introduce the structure of transformer blocks and then discuss the possibilities of connecting new modalities to LLM. LLMs [14, 60, 69, 84] are stacked by transformer blocks [70], each of which contains Multi-Head Attention (MHA) modules, LayerNorms, and a Feed-Forward Network (FFN). As shown in Figure 3 - Left, MHA receives H segments of Key (K), Value (V), and Query (Q) token sequences and performs attention through H heads. The scaled dot-product attention mimics database search by weighted-summing Values using Keys' similarity to Query.

Model (# of parameters)	Input size (input type)	Memory usage	Inference cost	Training cost with LoRA [27]
Encoders				
ViT-Base [18] (86M)	3x224x224 (RGB image)	2.09 GB	53.938	87.171
RangeViT [3] (22M)	5x32x2048 (point cloud)	7.02 GB	641.201	1440.161
Wav2Vec2-Base [5] (95M)	93,680 (audio samples)	2.22 GB	79.955	129.911
LLMs				
OPT-350M [84]	13 (text tokens)	1.96 GB	8.566	17.273
BLOOMZ-1.1B [60]	13 (text tokens)	4.76 GB	25.652	51.211
OPT-1.3B	13 (text tokens)	5.64 GB	34.204	68.521
OPT-2.7B	13 (text tokens)	11.89 GB	58.248	102.309
OPT-6.7B	13 (text tokens)	25.55 GB	172.951	345.342
Llama-3.1-8B [20]	13 (text tokens)	30.67 GB	195.322	390.179

Table 1: Computational complexity of different data encoders and LLMs. The model's inference and training costs are both measured for one single forward/backword pass in GFLOPs.

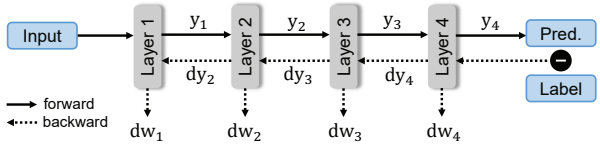


Figure 4: Backpropagation of a 4-layer dense NN

An intuitive solution to connecting an encoder to the LLM is to use the K-V pairs in cross-MHA. However, cross-MHA is only available in encoder-decoder LLMs [14]. Decoder-only LLMs do not contain pre-trained MHA modules for cross-attention purposes[2]. Alternatively, prompt tuning methods prepend trainable tokens to the input text [40] or intermediate K-V pairs [46, 50], but they only apply to the text domain. Inspired by these existing efforts, we further expand the scope of adaptation to multiple modalities, by projecting multimodal tokens via plug-in projectors and then inserting them as new K-V pairs into MHA modules of LLM blocks.

2.3 Reducing the Backpropagation Cost

Our design of MPnP is inspired by the backpropagation cost in LLM training. Typically, the FLOPs of backpropagation can be decomposed into two parts using the chain rule. As shown in Figure 4, when training a 4-layer dense neural network, each layer computes *i*) the activation gradient dy_i and passes it backward, and *ii*) computes the weight update dw_i using dy_{i+1} from the upstream layer. Freezing some layers can eliminate the FLOPs of computing these layers' weight updates, but activation gradients will still be computed. For example, when freezing layer 2 to 4, the activation gradients from dy_2 to dy_4 still need to be computed for layer 1 to compute its weight update. Due to the generality of the chain rule, this mechanism applies to any other types of models, including transformer blocks in LLMs.

Existing work connects new modalities into LLM's input layer through trainable projectors [43, 85], and the inserted projectors can be considered as the model's first layer. Even when all the LLM layers are frozen, the projector still needs activation gradients to be passed through the entire LLM and incurs backpropagation costs for all transformer blocks in LLM. Instead, our design connects input modalities into the last layers of LLM, to minimize the training cost at runtime.

[2]Although it is possible to re-purpose the self-MHA for cross-MHA by using external K-V pairs, it requires expensive retraining at runtime.

3 MPnP Design

Our design of MPnP aims to maximize the compute and data efficiency of runtime modality adaptation, while retaining the LLM's task performance. Our design is generic and applicable to different types of decoder-only LLMs and generative tasks. In this paper, we use the multimodal QA task in autonomous driving [58] as an illustrative example to explain our design. We assume that the need for modality adaptation and the modalities needed are decided in advance by the human user or direct interpretation of sensory data, and instead focus on systematic execution of such modality adaptation.

As shown in Figure 5, MPnP adapts new input modalities and connects the corresponding unimodal encoders to LLM blocks by retraining two major components at runtime:

(1) **Key & Value Aligners**, which project encoder outputs (i.e., multimodal tokens) to K-V pairs.
(2) **Trainable Latent Connections**, which decide to which LLM blocks and to what extent should the multimodal information in the generated K-V pairs be injected.

Such retraining is done using the training data of newly inserted input modalities, which can be collected at runtime by existing robotic system efforts such as AutoRT [2].

3.1 Unimodal Encoders and Feature Representations

Today's unimodal encoders for most input data modalities are pre-trained transformer-based encoders, such as ViT [18] for RGB images, AST [23] for audios, Point-BERT [81] for 3D point cloud data and Autoformer [75] for time-series data, which extract modality-specific features into token representations. With such token representations, when doing

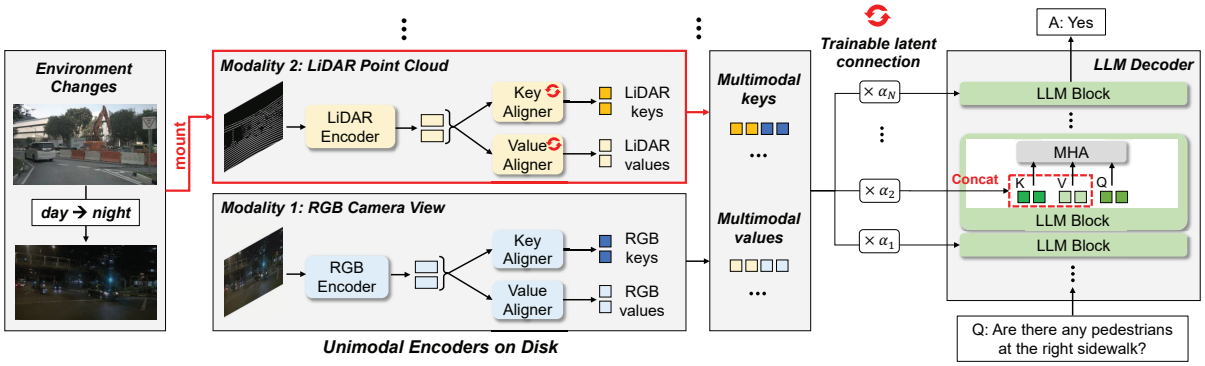


Figure 5: An example of MPnP's modality adaptation for multimodal QA task between two input modalities: RGB camera view and LiDAR point cloud. The text tokenizer, detokenizer, input and output embedding layers of the LLM are omitted for simplicity.

runtime modality adaptation, we only need to reshape the dimensions of token embedding when designing projectors for inserting new modalities into LLM. Other legacy encoders based on MLP, CNN and LSTM networks can also be adopted by MPnP, with extra efforts on reshaping their feature vectors or feature maps to token representations for LLM.

Most transformer-based encoders extract features into the [CLS] token sequences, and we further follow recent studies [61] to extract [CLS] tokens from multiple intermediate layers of encoders, to form a multi-level feature representation for more opportunities to align with the text features. For other transformers that do not contain usable [CLS] tokens, such as those trained using Masked Auto-Encoder (MAE) [24, 26] or other non-classification tasks [3, 47], we will use the averaged-pooled tokens for feature representation.

3.2 Key & Value Aligners

Whenever we switch input modalities at runtime, the aligners of the newly inserted modality (e.g., LiDAR in Figure 5) will need to be retrained, to ensure alignment of the new modality's key (K) & value (V) representations in the LLM's embedding space. The most straightforward design choice of the key & value aligners is to use linear projections [61, 85]. However, since our design of MPnP injects the multimodal information to last LLM blocks as shown in Figure 5, such injection requires higher expressivity for accurate alignment, which cannot be achieved by linear projection. Instead, we introduce non-linearity in the aligner's projection to efficiently bridging the semantic gap between input modalities and the LLM's natural language domain, while incurring negligible computing costs.

As shown in Figure 6, both key and value aligners in MPnP are point-wise multilayer perceptron (MLP) networks, with two linear transformations (namely weights and biases) and a nonlinear activation function in between. The two aligners' parameters, however, are independent and separately trained. Each aligners' operation can then be written as:

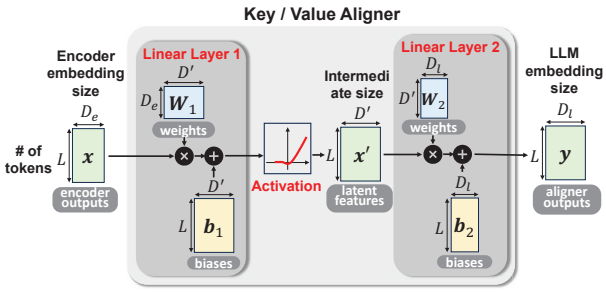


Figure 6: The design of key & value aligners

$$\text{Aligner}(\mathbf{x}) = \mathbf{Act}(\mathbf{x}W_1 + b_1)W_2 + b_2, \quad (1)$$

where W_1, W_2, b_1, b_2 are trainable parameters and $\mathbf{Act}(\cdot)$ is the activation function. This architecture is similar to the FFN in most transformer blocks, but we aim to align the token representation and match its shape to the LLM. Specifically, the aligner converts the embedding size of token representations $\mathbf{x}$ from data encoder's dimension (D_e) to LLM's dimension (D_l), while keeping the sequence length L unchanged.

Besides, being different from the existing plug-and-play approaches [9, 16, 61, 66] that require re-implementation of LLM's source codes, MPnP inserts the projected multimodal tokens as new key-value pairs into the multi-head attention (MHA) module of LLM block. Since such interfacing is well supported in popular LLM frameworks (e.g., HuggingFace Transformers [74]), we can avoid any manual programming and reconfiguration efforts at runtime. MPnP, hence, fully satisfies the plug-and-play requirement for LLMs [15].

To further justify our design choices made above, we conducted an detailed ablation study in Section 6.3, which systematically evaluates the impact of different design modifications on the overall system performance.

3.3 Trainable Latent Connection

As shown in Figure 5, we inject the aligned K-V pairs to a flexible set of last LLM blocks to reduce the depth of backpropagation and hence save the runtime training cost, and

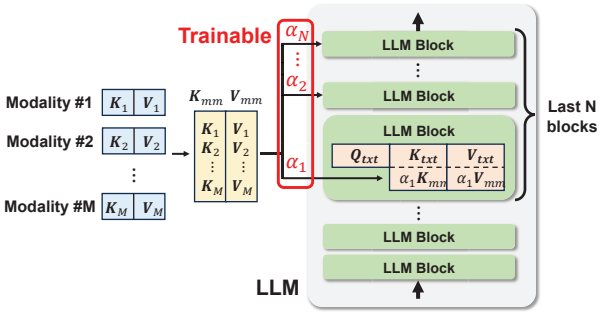


Figure 7: Trainable connections from K-V pairs to LLM

use a lightweight post-weighting module to optimize the amount of multimodal information that flows to each LLM block. More specifically, as shown in Figure 7, the aligners' outputs of unimodal key and value tokens are concatenated into multimodal keys and values:

$$\mathbf{K}_{\text{mm}} = [K_1; K_2; ..., K_M] + \mathbf{T}_{\text{k}}^{\text{pos}}$$
$$\mathbf{V}_{\text{mm}} = [V_1; V_2; ..., V_M] + \mathbf{T}_{\text{v}}^{\text{pos}},$$

where M is the number of new modalities and $\{K_j\}$, $\{V_j\}$ are unimodal key-value pairs. Adding the learnable positional embedding $\mathbf{T}_{\text{k,v}}^{\text{pos}}$ is optional and depends on the specific task. For example, when there are multiple camera views around an autonomous vehicle and each view is mapped to an individual K-V pair, the order of K-V pairs is then critical to answer a question like "Is there any pedestrian on my *left* side?". Only with positional embedding, the model can identify which K-V pair corresponds to the view of "*left*".

Afterwards, to reduce the backpropagation depth and hence computing costs in runtime training, we only inject multimodal information into the last N LLM blocks. To do so, $\mathbf{K}_{\text{mm}}$ and $\mathbf{V}_{\text{mm}}$ are duplicated N times, and each duplication is multiplied by a trainable weight to tune its intensity. Such weighting procedure can be represented as:

$$\mathbf{K'}_{\text{mm}} = [\alpha_1\mathbf{K}_{\text{mm}}, \alpha_2\mathbf{K}_{\text{mm}}, ..., \alpha_N\mathbf{K}_{\text{mm}}]$$
$$\mathbf{V'}_{\text{mm}} = [\alpha_1\mathbf{V}_{\text{mm}}, \alpha_2\mathbf{V}_{\text{mm}}, ..., \alpha_N\mathbf{V}_{\text{mm}}],$$

where N is the selected number of LLM blocks to inject multimodal information, and each α_j is valued between 0 and 1 to decide the amount of information being injected to the j-th LLM block. More specifically, each $\alpha_j = \text{sigmoid}(w_j/T)$ is parameterized by a trainable weight w_j, and we use the temperature T as a hyper-parameter to control the sharpness of α_j. Existing work [22, 35] opts to learn an exact binary representation of α_j using the straight-through estimator [6], but may cause unstable training. Instead, we aim to enable continuous tuning of the intensity of injected multimodal information, so that each LLM block can flexibly retrieve different amounts of multimodal information as needed.

The weighted multimodal K-V pairs are then appended to the projected text K-V pairs in the corresponding LLM block. For LLM block j, the K-V pairs for MHA are extended as:

$$\mathbf{K}_{\text{ext}} = [\alpha_j\mathbf{K}_{\text{mm}}; \mathbf{K}_{\text{txt}}]$$
$$\mathbf{V}_{\text{ext}} = [\alpha_j\mathbf{V}_{\text{mm}}; \mathbf{V}_{\text{txt}}].$$

When $\alpha_j = 1$, information in $\mathbf{K}_{\text{mm}}$ and $\mathbf{V}_{\text{mm}}$ are fed without any decay to LLM. Otherwise, $\alpha_j = 0$ degrades the impact of such information to zero, due to MHA's attention calculation:

$$\text{Out} = \text{softmax}(\frac{\mathbf{Q}[\alpha_j\mathbf{K}_{\text{mm}}; \mathbf{K}_{\text{txt}}]^\top}{\sqrt{d}})[\alpha_j\mathbf{V}_{\text{mm}}; \mathbf{V}_{\text{txt}}]. \quad (2)$$

As long as $\alpha_j\mathbf{V}_{\text{mm}} = 0$, the involvement of $\mathbf{K}_{\text{mm}}$ and $\mathbf{V}_{\text{mm}}$ is zeroed in MHA calculation. This property is important because it allows eliminating the impact of some modality's information if it impairs accuracy. Weighting on both keys and values also avoids complex designs of attention masks.

Such concatenation does not modify LLM's computing graph. Current transformer frameworks (e.g., HuggingFace [74]) implemented interfaces to pass $\alpha_j\mathbf{K}_{\text{mm}}$ and $\alpha_j\mathbf{V}_{\text{mm}}$ for internal concatenation. Users only need to extend the attention mask, and LLM can attend to inserted keys and values.

3.4 Deciding the Number of Connections

In our design, the multimodal information is injected to the last N LLM blocks, and the training cost hence depends on the value of N. Specifically, the backpropagation FLOPs can be calculated as:

$$T_{\text{backprop}}(N) = T_{\text{dw}}^{\text{Aligners}} + T_{\text{dy}}^{\text{Emb}} + N/L \cdot T_{\text{dy}}^{\text{LLM}}, \quad (3)$$

where $T_{\text{dw}}^{\text{Aligners}}$ is the FLOPs to compute weight updates of aligners, $T_{\text{dy}}^{\text{Emb}}$ is the FLOPs to pass activation gradients through LLM's output embedding layer, and $T_{\text{dy}}^{\text{LLM}}$ is the FLOPs to pass activation gradients through all the LLM blocks. In practice, due to much higher complexity of the LLM blocks, we have $T_{\text{dw}}^{\text{Aligners}} + T_{\text{dy}}^{\text{Emb}} \ll T_{\text{dy}}^{\text{LLM}}$. The ratio of training cost with respect to connecting to all L LLM blocks can be approximated as $r = T_{\text{backprop}}(L)/T_{\text{backprop}}(N) \approx L/N$.

In this way, the user can flexibly control the training cost at runtime, by tuning N based on different application requirements and the local system resource conditions. More specifically, to reduce N at runtime, we can simply discard the excessive connections and their corresponding trainable weights. Reversely, to increase N and add new connections at runtime, we can append new trainable weights between input modalities' encoders and the corresponding LLM blocks. In either way, no extra computing cost will be incurred.

4 Practical Workflow

As summarized in Algorithm 1, in practical systems, the process of modality adaption consists of two phases: offline preparation and online adaptation. Both phases involve efforts of model training, and we elaborate each of them in detail below.

4.1 Offline Preparation

In practical autonomous mobile systems, before actual system deployment, the mobile devices should have unimodal encoders and aligners for all input modalities pre-loaded locally, depending on the specific task being assigned (e.g., multimodal QA shown in Figure 5) and the available sensors on the device. Afterwards, the initial modalities to be connected to the LLM can be decided by either the human user or direct interpretation of sensory data. For example, in the scenario of autonomous driving in Figure 5, we can solely enable the RGB cameras at daytime, and connect the pre-trained RGB encoder to LLM accordingly.

Algorithm 1 Modality Adaptation in MPnP

Input: : A set of pre-trained encoders $\mathbf{E} = \{E_1, E_2, ...\}$ and the corresponding aligners $\mathbf{A} = \{A_1, A_2, ...\}$ stored on local external storage; a pre-trained LLM loaded in memory; trainable latent connections $\alpha_{1,...,N}$

/* Offline preparation */
$\mathbf{E}_0, \mathbf{A}_0 \leftarrow \texttt{Select}(\mathbf{E}, \mathbf{A})$ //Load initial encoders
$\mathbf{LLM} \leftarrow \texttt{Reconnect}(\mathbf{E}_0, \mathbf{A}_0)$ //Connect to LLM
$\texttt{Train}(\text{LLM}_{k,v}, \mathbf{A}_0, \alpha_{1,...,N})$ //Offline training
/* Online adaptation */
while $t < T_{\text{end}}$ **do**
  $\mathbf{E}_t, \mathbf{A}_t \leftarrow \texttt{Select}(\mathbf{E}, \mathbf{A})$ //Reload encoders
  $\mathbf{LLM} \leftarrow \texttt{Reconnect}(\mathbf{E}_t, \mathbf{A}_t)$ //Reconnect
  $\texttt{Train}(\text{LLM}_{k_{1,...,N}, v_{1,...,N}}, \mathbf{A}_t, \alpha_{1,...,N})$ //Adapt
end while

To initialize such connection and the adaptation to prompt structures, in MPnP we conduct an offline pre-training phase using a relevant base dataset. For example, for domain-specific QA tasks as shown in Figure 5, pre-training can be done with a generic dataset (e.g., OKVQA [52]) to equip the model with basic QA capabilities. By doing so, we ensure that in runtime modality adaptation later, any newly adapted modalities can be correctly recognized by LLM. In particular, all the K & V projectors in the LLM are also trained to quickly adapt to input and output prompt structures, such as "Given the question: Q, the answer is:".

During the offline pre-training, only the encoder is frozen and all other components, including the aligners, connections, and LLM, are jointly trained. To improve such training efficiency, we allow the LLM to generate the entire sequence of output tokens in a single forward pass by adopting the teacher-forcing method [38] Causal masks are applied to MHA's attention scores, so that each output token can be predicted from the label tokens at previous positions. In this way, when being fine-tuned, LLMs can be trained in a standard way like any feed-forward models, which only need one forward-backward pass to update the model weights.

4.2 Online Adaptation

Afterwards at runtime, whenever a new modality is needed due to environmental changes, we correspondingly load the data encoder and aligners into memory and connect them to the LLM, by retraining the aligner and latent connections as described in Section 3. The LLM itself also needs to be updated to distinguish newly inserted tokens from existing ones, but we only need to update the K & V projectors of those connected LLM blocks, hence involving a very small portion of trainable parameters.

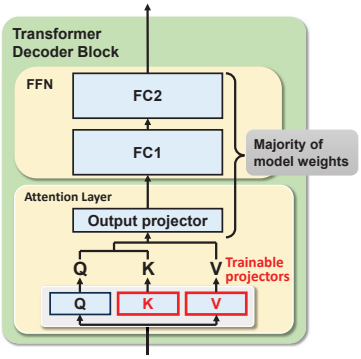


Figure 8: Composition of model weights in an transformer decoder block

As shown in Figure 8, compared to the large output projector and fully connected (FC) layers, the matrices in K & V projectors are much smaller. For example, in OPT-1.3B, the K & V projectors only contribute 15.3% of model parameters. In larger models like Llama-8B which have more fully connected layers in FFN, this percentage is as low as 3.3%.

To further reduce the runtime training cost, we adopted LoRA [27], a technique used to fine-tune large models in a more memory-efficient and computationally efficient manner, to decompose the weight matrices in K & V projectors into much smaller ones. In this way, the amount of trainable parameters in the LLM can be reduced by up to another 100×.

In practice, repetitive adaptation may be needed among the same set of modalities (e.g., day → night → day). MPnP does not repeat the same runtime training, but caches and reuses the modules that have been previously trained at runtime. The small parameter sizes of these modules ensure that they incur little storage costs on mobile devices.

During online adaptation, MPnP requires labeled and aligned training data corresponding to the new modalities being inserted. Nevertheless, MPnP is highly data efficient by design, because the model has sufficiently acquired general knowledge through offline pre-training. In Section 6.5, we will further show experiment results on MPnP's data efficiency, such that it only requires few hundreds of training samples for online adaptation, hence minimizing the runtime memory and computing costs.

Question: How many parked trucks are to the back of the construction vehicle? ***Answer:*** 4

Figure 9: A data sample from the nuScenes-QA dataset

5 Dataset Preparation

To evaluate MPnP in autonomous mobile systems with multiple input modalities, we adopt the nuScenes-QA dataset built for multimodal visual QA in autonomous driving [58] . As shown in Figure 9, each data sample contains 6-view RGB camera captures, a 5D LiDAR point cloud, and a corresponding text QA pair. Each LiDAR point cloud includes 5 dimensions of data about distance, intensity, X/Y/Z axes.

Since the original nuScenes-QA dataset is too large (~460k QA pairs) for runtime training, we build and use a smaller dataset, namely nuScenes-QA-mini, on top of the nuScenes-QA benchmark, as the overlapping part between nuScenes-QA dataset and mini-split of original nuScenes database [11].

Figure 10: An example of darkening and blurring the RGB view in the original nuScenes-QA dataset

In most RGB images in the nuScenes-QA dataset, as shown in Figure 10 - Left, the lighting conditions in night scenes are still abundant (e.g., with street lights). Hence, to create a challenging evaluation scenario in which modality adaptation from RGB camera view to liDAR point cloud is necessary in night scenes, we further reduce the brightness of RGB camera captures in night scenes by 80% and apply Gaussian blur with a radius of 7, as shown in Figure 10 - Right. By applying such preprocessing to the RGB views in night-scene samples of our nuScenes-QA-mini dataset, we obtained the training and validation splits of night scenes, with 659 samples for each split. On the other hand, the RGB views in daytime scenes remain unchanged: the daytime split in the dataset contains 2,229 for training and 2,229 for validation, respectively. These amounts of data samples align with those in other popular language understanding datasets (e.g., 285 for Winograd [41], 976 for MBPP [4], and 164 for penAI HumanEval [12]) for large generative models.

6 Performance Evaluation

Our evaluations use a workstation with a Nvidia RTX A6000 48GB GPU for offline preparation, and a Nvidia AGX Orin edge device with a Nvidia Ampere GPU for online adaptation. In our evaluations, we further cast the QA task as an open-ended text generation task, which more challenging than the commonly used classification task over a predefined answer set [11]. We use the *exact match ratio* as the accuracy metric, which measures how many generated answers by LLM exactly match the ground truth. Note that, achieving high in such sentence-level correctness is difficult, and existing work showed that accuracy at around 50% is satisfiable to ensure semantically correct text answers [34].

Datasets. We primarily use the nuScenes-QA-mini dataset described above to evaluate the performance of MPnP's adaptation between modalities of RGB camera captures and LiDAR point cloud. In addition, since most of today's multimodal LLMs are designed mainly for text and RGB image modalities, we also evaluate MPnP's generic performance in bridging these two modalities on multiple VQA datasets in other domains, including VQA-Rad [39] and Path-VQA [25] for QA on medical images and Instructional GQA [44] for generic QA on real-world image captures.

Encoders and LLMs. For the RGB camera modality, we use a ViT-Base [18] encoder pre-trained on ImageNet-1k [17] to extract RGB tokens. The RGB images are resized to 224×224 to match ViT-Base's input requirement. For each of the 6 RGB camera views in a data sample, we extract the [CLS] tokens from the last 4 transformer blocks of the encoder. For the LiDAR point cloud modality, we use a RangeViT [3] encoder pre-trained on the nuScenes segmentation tasks to extract rich object-level information from the LiDAR point cloud. Each LiDAR point cloud is projected to a 5×32×2048 tensor internally by RangeViT, and we average pool the RangeViT output into 4 tokens to represent features of LiDAR modality.

To perform QA tasks on multimodal data, we adopt pre-trained decoder-only LLMs, including OPT [84] and BLOOMZ [53]. These LLMs offer different variants of parameter sizes (125M-176B), and we select models with parameter sizes from 350M to 2.7B, to take different resource constraints of mobile devices into consideration. For answer generation, we adopt the prompt structure "question:{Q}</s>answer:{A}</s>", where "</s>" is the EOS token that has been already included in the pre-trained embeddings of OPT and BLOOMZ models.

Baseline schemes. Existing modular and parameter-efficient multimodal learning schemes are the most competitive baselines with MPnP. They adopt trainable projectors and adapters (e.g., prompt tuning [40]) for plug-and-play encoders. We compare MPnP with the following baselines[3]:

- **Full LLM** [9, 19] connects multimodal encoders with the LLM input layer, and trains the projector and fine-tunes the entire LLM for modality adaptation. It is

[3]We did not include those methods that hard-code model structures for adapting to a specific downstream task [56, 72], because their designs lack generality but require redesigning and training a new model from scratch.

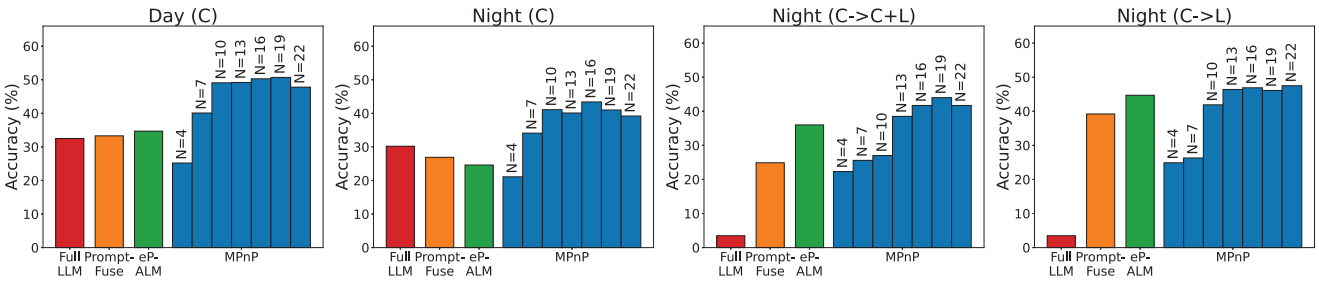


Figure 11: Performance of MPnP vs. baseline schemes w.r.t different scenes (Day/Night) and modalities (C: 6 RGB camera views, L: LiDAR point cloud) using the OPT-1.3B model, by connecting with different numbers of LLM blocks (N). The OPT-1.3B model has 24 LLM blocks in total.

the most expensive and data-hungry method, but can potentially lead to the highest task accuracy.

- **PromptFuse** [49] connects encoders with LLM's input layer. To reduce the training cost, instead of fine-tuning all parameters, it uses prompt tuning to adapt the LLM.
- **eP-ALM** [61] applies hard-coded connections between encoders' [CLS] tokens and intermediate LLM blocks, and uses prompt tuning to adapt LLM. Since LiDAR encoder does not provide well-trained [CLS] tokens, we feed its average-pooled output tokens to LLM.

6.1 Modality Adaptation Cost & Accuracy

We evaluate MPnP on OPT-1.3B using nuScenes-QA-mini. The model is offline trained on the day-train split to connect RGB modality to LLM, and tested on both day-test and night-test splits. Test accuracies in these two cases are indicated as **Day (C)** and **Night (C)**. Then, two modality adaptations are evaluated on the night-test split: 1) **Night (C → C + L)** mounts LiDAR modality and keeps RGB modality; 2) **Night (C → L)** mounts LiDAR modality but detaches RGB modality.

Results in Figure 11 show that in offline training - Day (C), MPnP achieves 5%-15% higher test accuracy. When being zero-shot evaluated on Night (C), the accuracy drops and demonstrates the need of modality adaptation. After having adapted to the LiDAR modality, MPnP can resume accuracy by up to 8%, but uses 20%-45% less training FLOPs and 25% less GPU memory, as shown in Figure 12.

Compared to the significant performance improvements over baseline methods on RGB images in daytime, the observed smaller performance gap on LiDAR data during nighttime transitions could be due to the model's limited representation power, as we are using a backbone LLM with a small parameter size. The LLM itself struggles to understand the complicated modalities like LiDAR, especially when N is small, i.e., only little LiDAR information is injected.

Furthermore, including both RGB and LiDAR modalities in nighttime (C → C + L) surprisingly impairs accuracy, especially when fully training the LLM. The possible reason is that keeping the uninformative RGB modality confuses the LLM inference. Note that the task accuracy with different

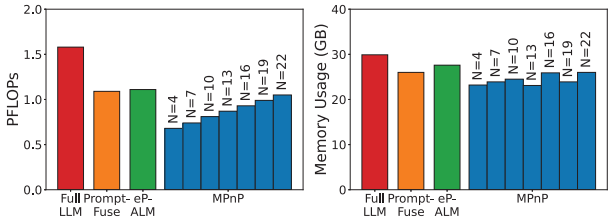
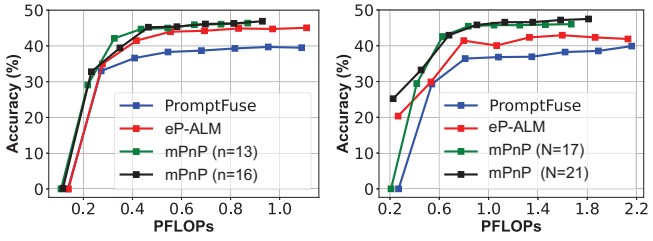


Figure 12: Computational cost and GPU memory usage of MPnP vs. baseline schemes, with different numbers of LLM blocks (N) on OPT-1.3B model

input modalities could be inherently different, depending on the knowledge being provided. For example, RGB images contain fine-grained object information (e.g., color and texture), but LiDAR point clouds are sparse and only provide information about distances and 3D object structures. Hence, the accuracy with LiDAR modality at nighttime is lower than that with RGB modality at daytime.

In particular, Full LLM has low accuracy when undertaking LiDAR modality, due to the large semantic gap between LiDAR modality and the text domain. Directly feeding LiDAR modality to LLM's input layer makes it hard for LLM to correctly understand the context. This observation highlights the necessity of our latent trainable connections, to efficiently bridge the gap between distant modalities.



(a) OPT-1.3B (b) OPT-2.7B

Figure 13: Accuracy-compute tradeoff w.r.t the modality adaptation of Night (C → L)

Figure 13 further shows that MPnP achieves better accuracy-compute efficiency, which measures how the task accuracy evolves with respect to the training FLOPs when adapting C → C + L. MPnP reduces the FLOPs by 3.6× to achieve 40%

accuracy compared to PromptFuse, and reduces the FLOPs by 1.7× to achieve 45% accuracy compared to eP-ALM. On OPT-2.7B, the FLOPs reduction is further enlarged to 3.7×.

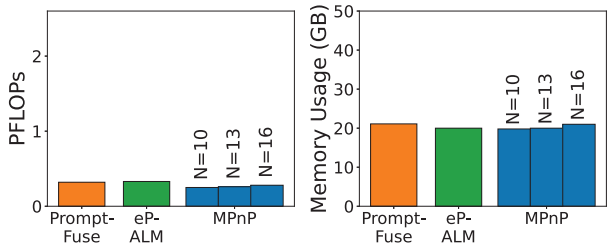


Figure 14: Computational cost and GPU memory usage of MPnP vs. baseline schemes, with different numbers of LLM blocks (N) on OPT-350M model

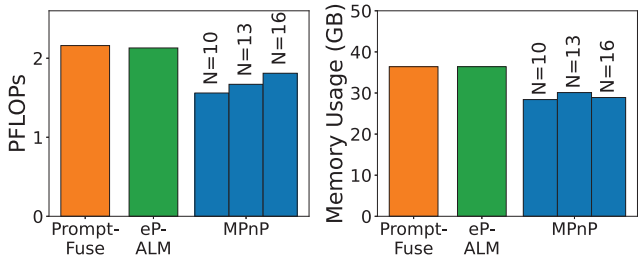


Figure 15: Computational cost and GPU memory usage of MPnP vs. baseline schemes, with different numbers of LLM blocks (N) on OPT-2.7B model

6.2 Impact of LLM Size

When being applied to LLMs in different sizes, as shown in Figures 16 and 17, MPnP results in higher accuracy improvements on bigger LLMs. With the OPT-2.7B model, eP-ALM suffers a significant accuracy drop on Night (C → L), while MPnP can retain performance compared to the OPT-1.3B cases due to the trainable connections. Essentially, as shown in Figure 14 and 15 with different values of N, the training costs of MPnP have been very close to the backpropagation costs of encoders and LLM blocks. Hence, training the aligners and latent connections in MPnP is very lightweight.

The peak GPU memory usage is mainly decided by the large activation size in the LiDAR encoder, which can be 16× larger than the LLM's activation size when using a small context window (64 tokens). Since MPnP freezes the encoders in runtime training, its memory savings only stem from the smaller backpropagation cost and are hence limited to 20% when the LLM size is small (350M and 1.3B). However, as shown in Figure 15, the memory savings increase with larger models and larger context windows. On OPT-2.7B, the memory savings can increase by another 10-15%.

6.3 Ablation Study

We conduct an ablation study to verify the effectiveness of our different design choices in MPnP. Table 2 shows that disabling the training of the K,V projector causes the highest accuracy drop, due to the required updates of K, V projectors to correctly distinguish newly inserted tokens from existing tokens. The same phenomena are observed in the existing work [61] when adopting LLM adapters. Accuracy drop due to ablation is lower when connecting to more LLM blocks, because higher representation power is present when the multimodal tokens go through more LLM blocks. Similarly, when the latent connections between data encoders and LLM are not trainable, the model accuracy also drops by 5%, demonstrating the necessity of using trainable latent connections in modality adaptation.

Ablated module	Acc. (N = 13)	Acc. (N = 17)
Base		
None	42.2%	46.0%
Aligner Structure		
MLP→Linear	37.8%	42.9%
Aligner Activation Function		
GeLU→ReLU	41.8%	46.7%
GeLU→SiLU	43.0%	44.1%
Latent Connection		
trained → fixed	36.5%	41.7%
K,V projector		
trained → fixed	33.8%	35.4%

Table 2: Ablation study of MPnP with OPT-2.7B model

When replacing the MLP aligner with a single linear projector, the accuracy also drops significantly for 3-5%. Such results verify the necessity of introducing full non-linearity in the design of key & value pairs. On the other hand, we also evaluated the performance of MPnP using different activation functions in the key & value pairs, and results in Table 2 show that the choice of activation function leads to minor changes in the task performance, but generally using GeLU activation function provides the most reliable performance.

Overall, our approach is able to strike a flexible balance between computational complexity and system performance, making it suitable for practical deployment in diverse scenarios with different conditions of system resources and application requirements.

6.4 Data Efficiency

Data efficiency is important to minimize the training latency in runtime modality adaptation, and is also crucial for autonomous mobile systems where only a very limited amount of training data may be available on mobile devices. We demonstrated that MPnP can outperform baseline schemes by using a small amount (659) of runtime training samples, and Figure 18 further shows MPnP's advantage when such amount of samples reduces by another 40%. When using only 527 training samples, MPnP can still achieve on-par accuracy

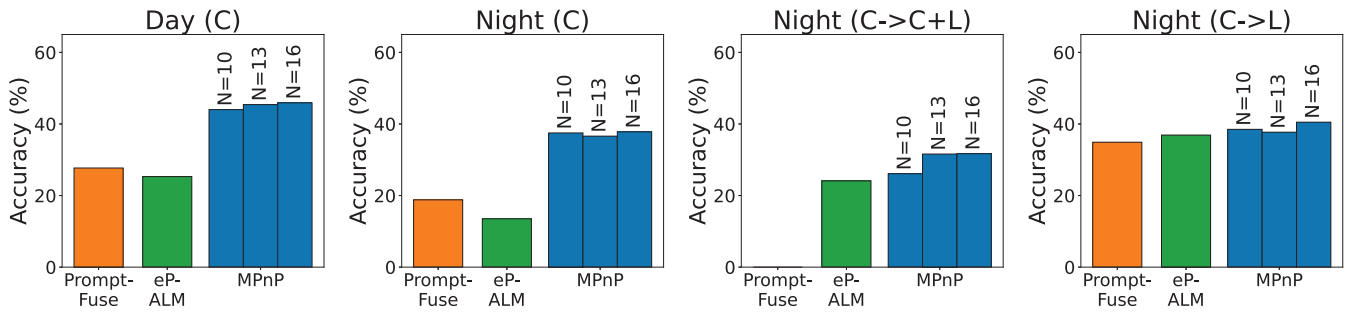


Figure 16: Performance of MPnP vs. baseline schemes w.r.t scenes (Day/Night) and modalities (C: 6 RGB camera views, L: LiDAR point cloud) using OPT-350M by connecting with different numbers of LLM blocks (N). The OPT-350M model has 16 LLM blocks in total.

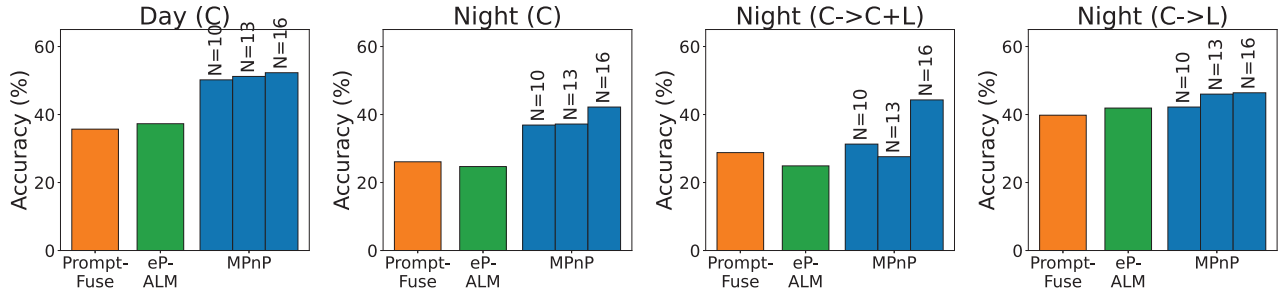


Figure 17: Performance of MPnP vs. baseline schemes w.r.t scenes (Day/Night) and modalities (C: 6 RGB camera views, L: LiDAR point cloud) using OPT-2.7B by connecting with different numbers of LLM blocks (N). The OPT-2.7B model has 32 LLM blocks in total.

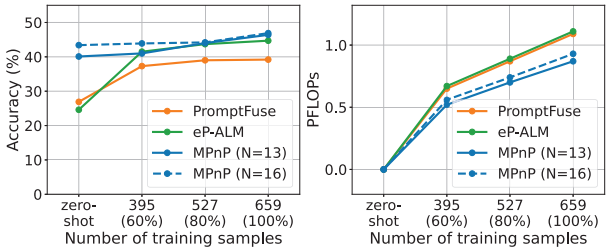


Figure 18: Data efficiency in runtime modality adaptation, using the OPT-1.3B model

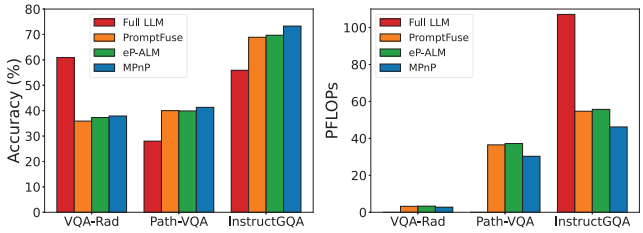


Figure 19: Model accuracy and computing costs of fine-tuning the OPT-1.3B model on text-image VQA tasks

with ePALM using 659 training samples, but further reduces training FLOPs by 20%.

In addition, even without runtime modality adaptation, MPnP can retain a moderate level of zero-shot accuracy, which is at least 15% higher than that of baselines and only 3% lower than the task accuracy achieved using 100% of training samples. Similarly, such accuracy in MPnP approximates to the peak even before it has iterated over all training samples.

6.5 Generic Cross-Modal Performance

We evaluate MPnP's generic performance of bridging between the text and RGB image modalities in LLMs, by using it in LLM fine-tuning with multiple VQA datasets in different problem domains, including VQA-Rad [39] and Path-VQA [25] for QA on medical images and Instructional GQA [44] for generic QA on real-world image captures[4]. Since these VQA datasets only contain one extra data modality other than texts, we cannot conduct evaluation on modality adaptation as we did on the nuScenes-QA-mini dataset. Instead, in our experiments, we freeze both the encoders and the LLM but train the latent connections in between, and then evaluate the VQA task accuracy with such trained connections. Standard evaluation frameworks and benchmarks are used as instructed in these datasets.

As shown in Figure 19, MPnP improves accuracy by up to 4% and reduces the training FLOPs by 20%. Note that, although existing Full LLM methods achieve superior accuracy on VQA-Rad, their models are intensively trained on >180M

[4]We measure accuracy using F1-score on the Instrucional GQA dataset.

data samples. Due to MPnP's good accuracy-compute efficiency shown in Figure 13, we expect that it can achieve higher accuracy when applied to similarly trained models.

LLM & Method	Day (C) Acc. (%)	Night (C → L) Acc. (%) / PFLOPs / Mem.
OPT-1.3B		
Full LLM	32.5	3.5 / 1.58 / 29.9
PromptFuse	33.3	39.2 / 1.09 / 26.0
eP-ALM	34.7	44.7 / 1.11 / 27.6
MPnP (N = 10)	49.1	41.9 / 0.81 / 24.5
MPnP (N = 13)	49.2	46.4 / 0.87 / 23.1
BLOOMZ-1.1B		
Full LLM	34.0	0.0 / 1.16 / 32.2
PromptFuse	35.4	26.6 / 0.90 / 25.7
eP-ALM	27.1	26.0 / 0.91 / 28.7
MPnP (N = 10)	39.4	25.8 / 0.73 / 22.7
MPnP (N = 13)	44.0	27.0 / 0.76 / 25.8

Table 3: Task accuracy & runtime training cost on different types of LLMs

6.6 Performance with Different Types of LLMs

Table 3 shows that MPnP achieves higher task accuracy than baseline schemes on both OPT and BLOOMZ models. Note that, since the cross-lingual pre-training in BLOOMZ could reduce the LLM's performance on a specific language when the model complexity is low [51], the overall task accuracy on BLOOMZ is lower than OPT.

6.7 On-Device Performance

The number of floating point operations (FLOPs) should be generally proportional to the wall-clock time. However, FLOPs cannot reflect the memory access overhead, which depends on specific hardware performance (e.g., GPU memory and its bandwidth). Today's GPUs are becoming more powerful and the discrepancy between FLOPs and wall-clock time is smaller. Especially for the existing mobile AI platforms (e.g., Nvidia Jetson series), they adopt shared memory architectures for on-board CPU and GPU, hence reducing the data transmission cost for memory access.

To evaluate the on-device performance of MPnP, we conduct experiments on the Nvidia Jetson AGX Orin device, by running the device at medium power mode, with an average power consumption of 26W. As shown in Figure 20 and 21, on a weak mobile computing device (Nvidia Jetson AGX Orin), MPnP iterates through 60% of training samples in the nuScenes-QA-mini dataset within few minutes. Compared to the existing baselines, it achieves on-par or better accuracy with 3.1×-3.75× wall-clock time speedup. Since MPnP can retain a good level of zero-shot accuracy as shown in Figure 18, such speedup ensures runtime modality adaptation to be smoothly operated in real autonomous mobile systems, which can still use the RGB modality during the process of adaptation to retain the task performance.

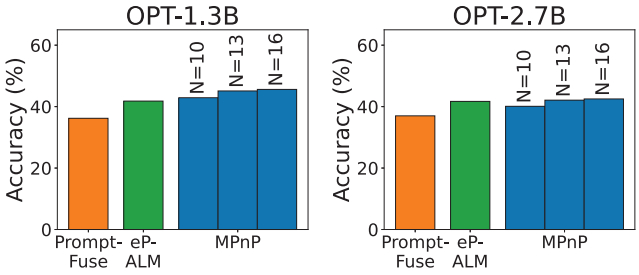


Figure 20: Task accuracy on a Nvidia Jetson AGX Orin device

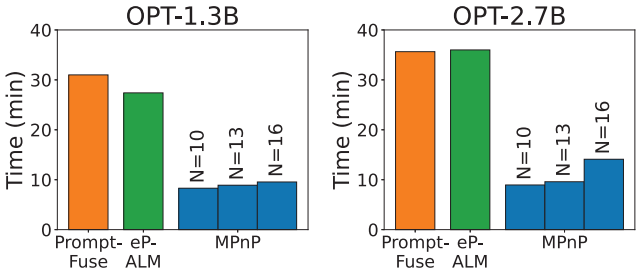


Figure 21: Computing latency of runtime modality adaptation on a Nvidia Jetson AGX Orin device

7 Related Work

Inserting multimodal information into LLM. Having connected encoders to LLM's transformer blocks, there are multiple choices of inserting multimodal information from encoders to LLM. Insertion into LLM's latent space enables more efficient cross-modal interaction [54, 61], by interacting with different levels of feature representations. Early work adopts FiLM-like weighting schemes [9, 16, 28, 56] to condition one modality on others in intermediate layers, but require structure designs from scratch. Later schemes used cross-MHA mechanisms [45, 66, 67, 78], but cannot be applied on decoder-only LLMs. To avoid changing pre-trained LLM structures, recent work [61] concatenates projected tokens with output sequences from multiple LLM blocks. However, these LLM blocks are arbitrarily selected. Extending the output sequence length for one block also affects such lengths in other blocks and reduces accuracy.

These limitations motivate our design of MPnP, which is a more flexible and generic method to insert multimodal information into pre-trained LLMs, by allowing adaptive weighting of multimodal tokens being inserted into the K-V set in LLM blocks. In this way, we can establish optimal connections between encoders and LLM at runtime, for efficient cross-modal interaction.

Few-shot learning. Our approach to runtime modality adaptation also relates to the existing work on few-shot learning [55, 73, 79], which allows training a model without using a large dataset. However, few-shot learning is very sensitive to the few data samples being used in training and hence prone to overfitting. In contrast, MPnP by design can achieve good generality over different types of LLMs and downstream tasks, while still achieving high data efficiency.

8 Discussions

Extending to more modalities. In this paper, we present the design of MPnP using an example of adapting between two input modalities in VQA, namely RGB images and LiDAR point clouds, on which comprehensive and representative datasets are present. By design, MPnP can support adaptation among an arbitrary number of input modalities, since there is no constraint on the K-V set's size in LLM, but simultaneously mounting more modalities could also be more expensive. In extreme cases, it would be necessary to meticulously rank the modalities' importance and only mount the most important modalities to fit the mobile devices' resource constraints [29]. Quantifying such importances of different modalities at runtime will be our future work.

Using larger LLMs. In our evaluations, we restrict the size of LLMs being used at 2.7B, to fit to the local resource constraints of devices in autonomous mobile systems. Being aware of the possible performance improvement with larger LLMs being used, we believe that one possibility is to adopt the recently emerging small language models (SLMs) [64, 83], such as Phi-3 [1] and Gemma [68] that have higher parameter efficiency by focusing on specific downstream tasks. Adopting these models to MPnP will be our future work.

Using a shared set vs. separate sets of connections. Our design of MPnP uses a shared set of trainable connections and weights for all the involved input modalities. In this way, we can minimize the amount of connections and further the runtime training costs. On the other hand, MPnP by design also allows using a separate set of connections for each modality's encoder to the LLM, with extra runtime training costs, so that each modality has extra flexibility to interact with different levels of text representations. For example, in some practical cases, an input modality may become less informative but not completely useless, and such extra flexibility will then allow partial involvement of this modality, e.g., only to a selected subset of LLM blocks or with reduced weights in cross-modal interaction.

Generality on other generative models. In this paper, we use the OPT and BLOOMZ model families as the backbone LLM in MPnP for multimodal reasoning. However, the design rationale of MPnP is generic and can be applied to other transformer-based generative backbones, e.g., the Llama-3 models, which share the similar architectural designs in their core attention mechanisms. MPnP can also be applied to pre-trained time-series transformer decoders [82], for simpler AI tasks such as motion planning.

Besides, although our design of MPnP focuses on decoder-only LLMs, it can also be applied to encoder-decoder LLMs (e.g., FLAN-T5 [14] and BART [42]), by only connecting the multimodal encoders to the model's decoder part, more specifically, to the K-V pairs in the decoder block's self-MHA.

Even for non-transformer-based models, such as diffusion models [80], the design rationale of MPnP still applies, as we can establish trainable latent connections with encoders. Efficient connections, in this case, will require specific model structure designs, which will be an interesting research direction for the future. Other types of generative models such as GANs and VAEs, however, are not inherently optimized for multi-modal generation: GANs focus on single-domain synthesis, and VAEs rely on probabilistic latent representations. They may need additional adjustments for modality alignment, which is out of the scope of this paper.

One-time adaptation vs. continuous learning. Our approach focuses on one-time adaptation, ensuring that the model parameters remain fixed after adaptation and prevent drift over time. The adaptation is triggered only when significant environmental changes occur (e.g., transitions between day and night), rather than continuously updating the model based on incremental changes. This is distinct from the scenarios of continuous learning, which frequently updates model parameters and may introduce challenges related to catastrophic forgetting, increased computational overhead, and potential performance degradation due to noisy or redundant updates. Nevertheless, we recognize that continuous learning may offer an alternative pathway and facilitate smoother transitions, and we will explore this pathway in our future work.

9 Conclusion

In this paper, we present MPnP, a new technique of runtime modality adaptation for LLM-driven autonomous mobile systems. It can achieve 3.7× FLOPs reduction while retaining on-par accuracy with the existing schemes, or improves the task accuracy by 4% with the same compute budget.

Acknowledgments

We thank the shepherd and reviewers for their comments and feedback. This work was supported in part by National Science Foundation (NSF) under grant number IIS-2205360, CCF-2217003, CCF-2215042, and National Institutes of Health (NIH) under grant number R01HL170368.

References

[1] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

[2] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. J. Joshi, R. C. Julian, S. Kirmani, I. Leal, E. Lee, S. Levine, Y. Lu, S. Maddineni, K. Rao, D. Sadigh, P. R. Sanketi, P. Sermanet, Q. H. Vuong, S. Welker, F. Xia, T. Xiao, P. Xu, S. Xu, and Z. Xu. Autort: Embodied foundation models for large scale orchestration of robotic agents. 2024.

[3] A. Ando, S. Gidaris, A. Bursuc, G. Puy, A. Boulch, and R. Marlet. Rangevit: Towards vision transformers for 3d semantic segmentation in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5240–5250, 2023.

[4] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[5] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

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

[7] J. Brehmer, J. Bose, P. De Haan, and T. Cohen. EDGI: Equivariant diffusion for planning with embodied agents. *arXiv preprint arXiv:2303.12410*, 2023.

[8] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[9] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] 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*, pages 11621–11631, 2020.

[12] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code, 2021.

[13] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[14] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[15] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[16] 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*, 30, 2017.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[19] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[20] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[21] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[22] S. Gao, F. Huang, W. Cai, and H. Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9270–9280, 2021.

[23] Y. Gong, Y.-A. Chung, and J. Glass. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021.

[24] 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*, pages 16000–16009, 2022.

[25] X. He, Z. Cai, W. Wei, Y. Zhang, L. Mou, E. Xing, and P. Xie. Pathological visual question answering. *arXiv preprint arXiv:2010.12435*, 2020.

[26] G. Hess, J. Jaxing, E. Svensson, D. Hagerman, C. Petersson, and L. Svensson. Masked autoencoder for self-supervised pre-training on lidar point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 350–359, 2023.

[27] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[28] R. Hu, J. Andreas, T. Darrell, and K. Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 53–69, 2018.

[29] K. Huang and W. Gao. Real-time neural network inference on extremely weak devices: agile offloading with explainable ai. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 200–213, 2022.

[30] K. Huang, B. Yang, and W. Gao. Elastictrainer: Speeding up on-device training with runtime elastic tensor selection. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, pages 56–69, 2023.

[31] K. Huang, H. Yin, H. Huang, and W. Gao. Towards green ai in fine-tuning large language models via adaptive backpropagation. *ICLR*, 2024.

[32] K. Huang, X. Yin, T. Gu, and W. Gao. Perceptual-centric image super-resolution using heterogeneous processors on mobile devices. In *ACM MobiCom*, 2024.

[33] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[34] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[35] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[36] H. Kong, X. Xu, J. Yu, Q. Chen, C. Ma, Y. Chen, Y.-C. Chen, and L. Kong. m3track: mmwave-based multi-user 3d posture tracking. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, pages 491–503, 2022.

[37] A. Kramer, K. Harlow, C. Williams, and C. Heckman. Coloradar: The direct 3d millimeter wave radar dataset. *The International Journal of Robotics Research*, 41(4):351–360, 2022.

[38] A. M. Lamb, A. G. ALIAS PARTH GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

[39] 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*, 5(1):1–10, 2018.

[40] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[41] H. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

[42] M. Lewis. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[43] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[44] L. Li, Y. Yin, S. Li, L. Chen, P. Wang, S. Ren, M. Li, Y. Yang, J. Xu, X. Sun, et al. M3IT: A large-scale dataset towards multi-modal multilingual instruction tuning. *arXiv preprint arXiv:2306.04387*, 2023.

[45] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13094–13102, 2023.

[46] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[47] Y. Li, H. Mao, R. Girshick, and K. He. Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision*, pages 280–296. Springer, 2022.

[48] Z. Li, Z. Rao, L. Pan, P. Wang, and Z. Xu. Ti-mae: Self-supervised masked time series autoencoders. *arXiv preprint arXiv:2301.08871*, 2023.

[49] S. Liang, M. Zhao, and H. Schütze. Modular and parameter-efficient multimodal fusion with prompting. *arXiv preprint arXiv:2203.08055*, 2022.

[50] X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, 2022.

[51] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

[52] K. Marino, M. Rastegari, A. Farhadi, and R. Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204, 2019.

[53] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf, et al. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*, 2022.

[54] A. Nagrani, S. Yang, A. Arnab, A. Jansen, C. Schmid, and C. Sun. Attention bottlenecks for multimodal fusion. *Advances in Neural Information Processing Systems*, 34:14200–14213, 2021.

[55] A. Parnami and M. Lee. Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint arXiv:2203.04291*, 2022.

[56] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[57] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlavac, S. Y. Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023.

[58] T. Qian, J. Chen, L. Zhuo, Y. Jiao, and Y.-G. Jiang. Nuscenes-qa: A multi-modal visual question answering benchmark for autonomous driving scenario. *arXiv preprint arXiv:2305.14836*, 2023.

[59] M. R. U. Saputra, A. Markham, and N. Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36, 2018.

[60] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[61] M. Shukor, C. Dancette, and M. Cord. ep-alm: Efficient perceptual augmentation of language models. *arXiv preprint arXiv:2303.11403*, 2023.

[62] C. G. Snoek, M. Worring, and A. W. Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402, 2005.

[63] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.

[64] J. Song, K. Huang, X. Yin, B. Yang, and W. Gao. Achieving sparse activation in small language models. *arXiv preprint arXiv:2406.06562*, 2024.

[65] Y.-L. Sung, J. Cho, and M. Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5227–5237, 2022.

[66] H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[67] Z. Tang, J. Cho, Y. Nie, and M. Bansal. Tvlt: Textless vision-language transformer. *Advances in Neural Information Processing Systems*, 35:9617–9632, 2022.

[68] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[69] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[71] H. Wang and W. Gao. When device delays meet data heterogeneity in federated aiot applications. *ACM MobiCom*, 2025.

[72] J. Wang, Z. Yang, X. Hu, L. Li, K. Lin, Z. Gan, Z. Liu, C. Liu, and L. Wang. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*, 2022.

[73] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[74] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[75] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

[76] S. Wu, H. Fei, L. Qu, W. Ji, and T.-S. Chua. Next-gpt: Any-to-any multimodal llm. *arXiv preprint arXiv:2309.05519*, 2023.

[77] W. Xu, W. Song, J. Liu, Y. Liu, X. Cui, Y. Zheng, J. Han, X. Wang, and K. Ren. Mask does not matter: Anti-spoofing face authentication using mmwave without on-site registration. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 310–323, 2022.

[78] X. Xu, C. Wu, S. Rosenman, V. Lal, W. Che, and N. Duan. Bridgetower: Building bridges between encoders in vision-language representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10637–10647, 2023.

[79] Q. Xue, X. Yin, B. Yang, and W. Gao. Phyt2v: Llm-guided iterative self-refinement for physics-grounded text-to-video generation. *Proc. CVPR*, 2025.

[80] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 2022.

[81] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022.

[82] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[83] Q. Zhang, Z. Liu, and S. Pan. The rise of small language models. *IEEE Intelligent Systems*, 40(1):30–37, 2025.

[84] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

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