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

*Kai Huang, Xiangyu Yin, Heng Huang, Wei Gao* ✉

*December 2024*

**MobiCom’ 25**

**汇报人：刘佳伟 2025年1月10日**

# 提纲

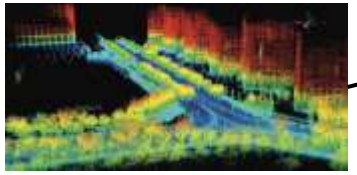
# 提纲

# Introduction

自然语言文本


音频

激光雷达点云



RGB图像

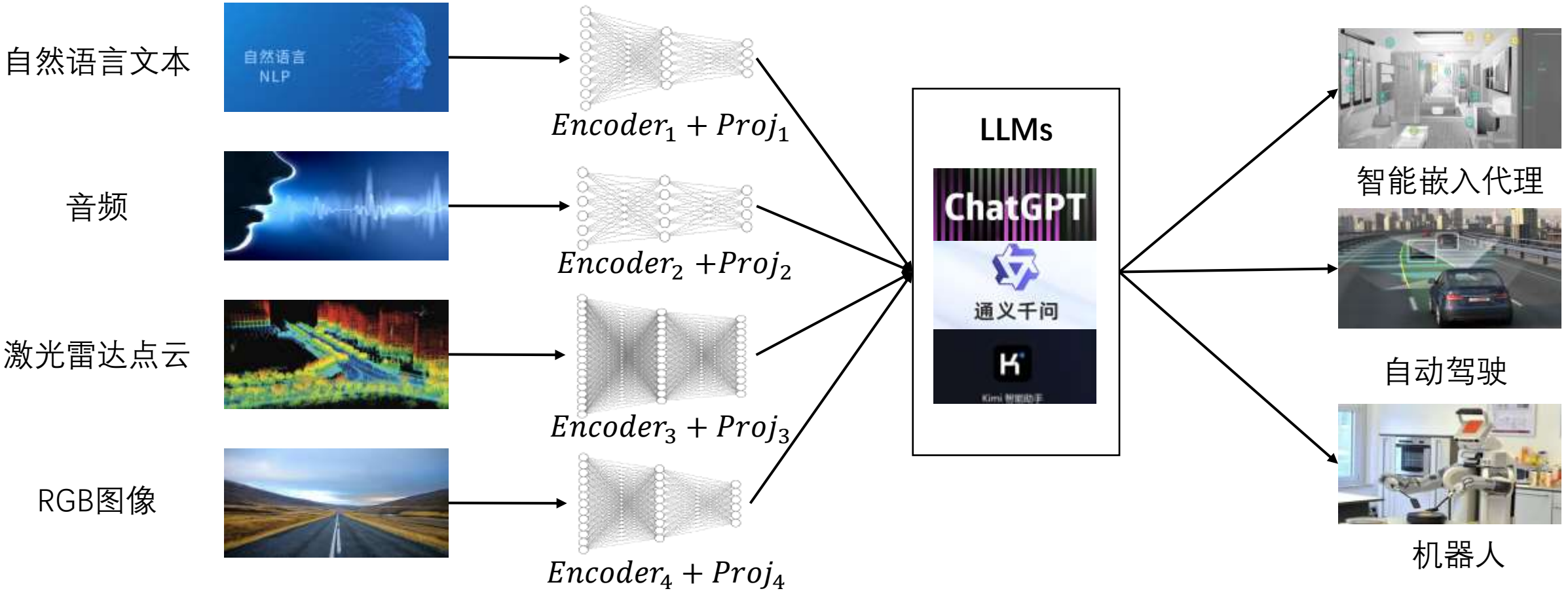
MLLM

智能嵌入代理

自动驾驶

机器人

# Introduction




- 基于Transformer的编码器（如ViT）通常**尺寸很大**
- 输入数据模态**日益多样化**

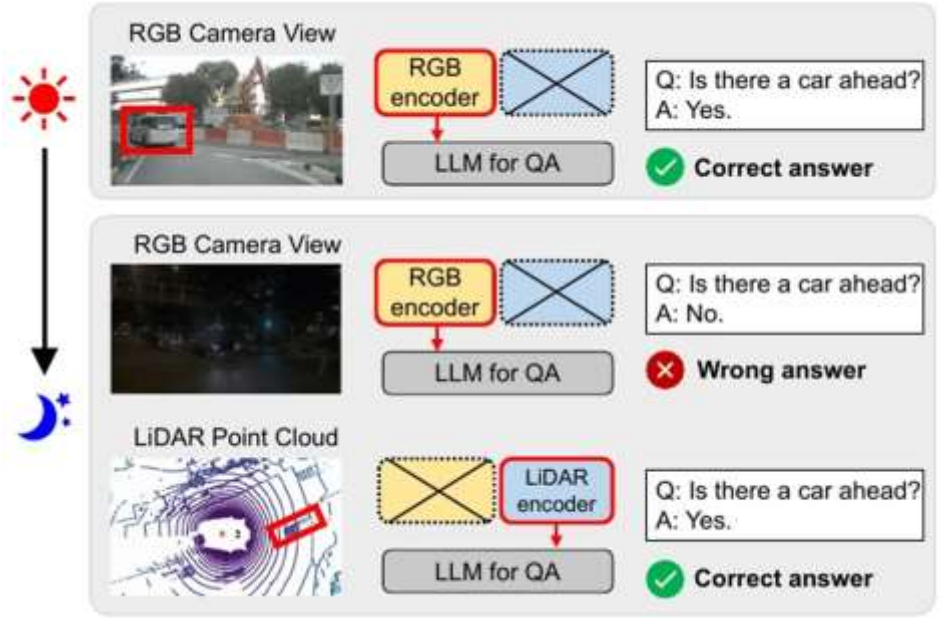
→ 将所有模态合并到LLM中的**训练与运行成本都很高。**

# Introduction

环境上下文信息变化

↓ 影响

输入模态的有用性

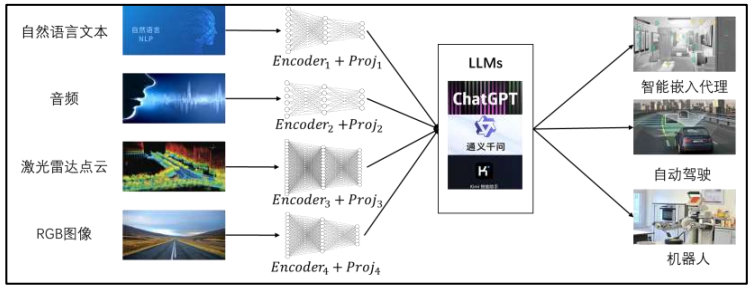



希望运行时**自适应地只选取有用的模态**

直观方法

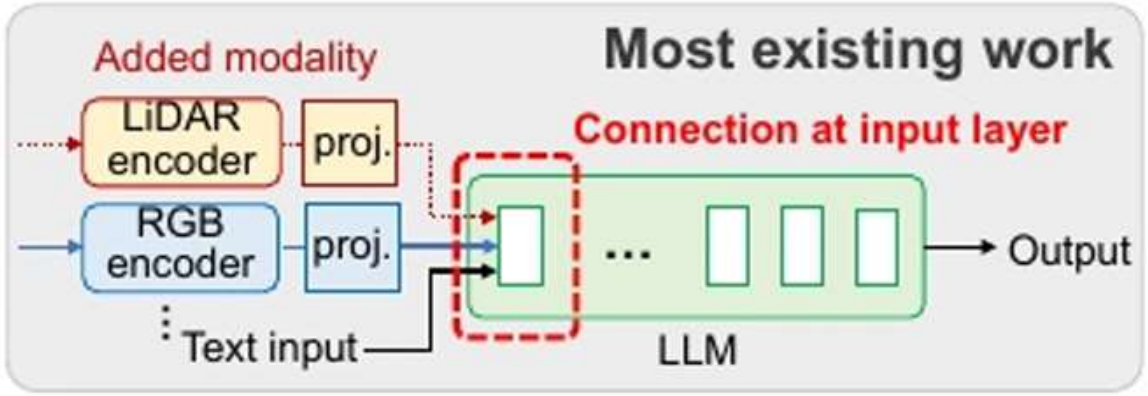
所有相关模态的编码器都参与LLM的联合训练，使每个模态与自然语言域对齐

**成本高！**



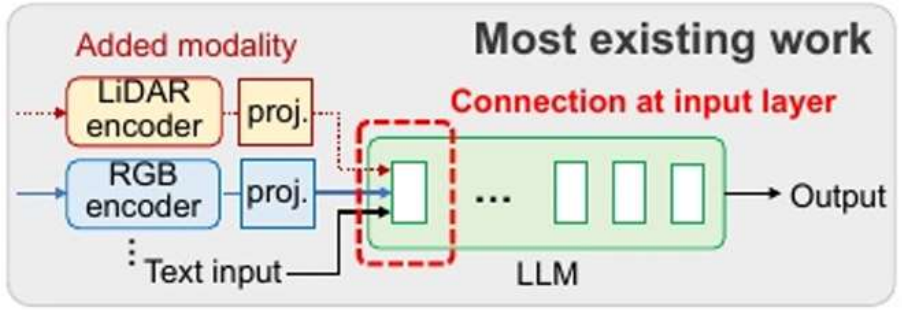

优化方案

投射层全接入LLM输入层，冻结编码器和 LLM，只训练中间插入的投射层，以此通过微调来提高精度




仍然需要在**整个LLM中使用反向传播，成本高！**

# Introduction




那就用更轻量的投射层?

并没有减少LLM反向传播的代价，**杯水车薪！**

**总结一下不足之处：**

投射的多模态特征**全部**与LLM输入层的文本嵌入对齐

⬇

弥合不同模态之间的语义差距方面可能效率低下

在**整个LLM中**进行反向传播

⬇

训练成本比较高

**希望做到的效果：**

在LLM中实现弹性、自动化和快速模态自适应的新技术

自适应地调整连接的LLM块的数量，以在精度和训练成本之间进行权衡

# 提纲

# Background

➢ **Transformer Blocks的接入点**

**方案1**

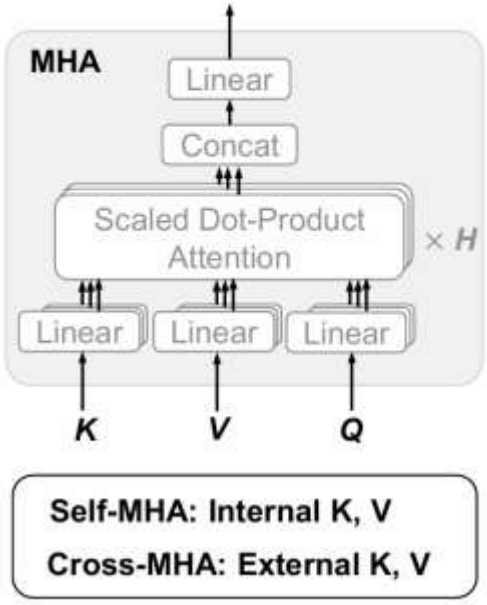
作为cross-MHA中的K-V对进行接入

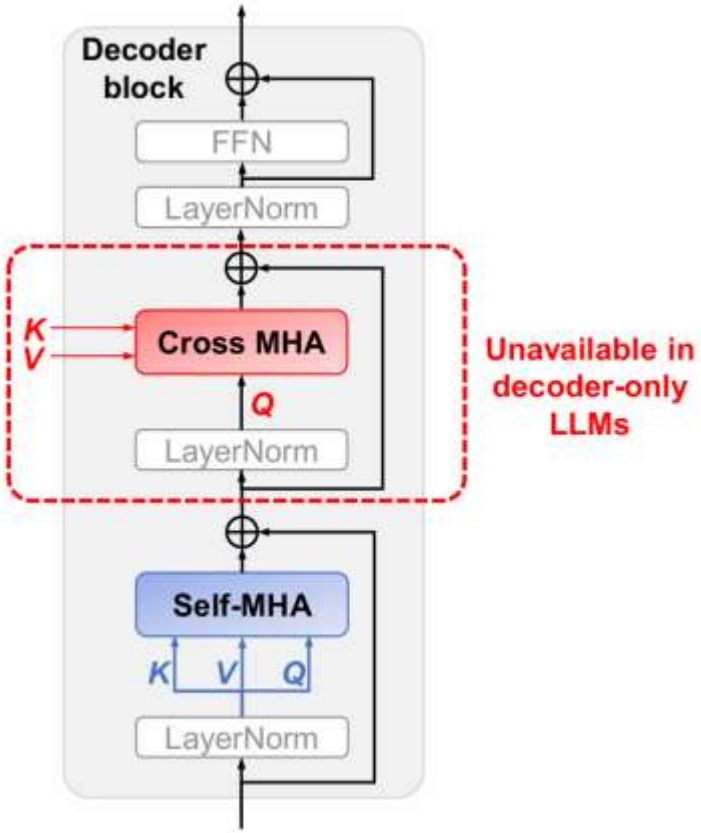
仅在编码器-解码器架构的
大型语言模型中可用

**方案2**

提示调优：在输入文本或中间K-V对前，
添加可训练的token（拼接到K、V）

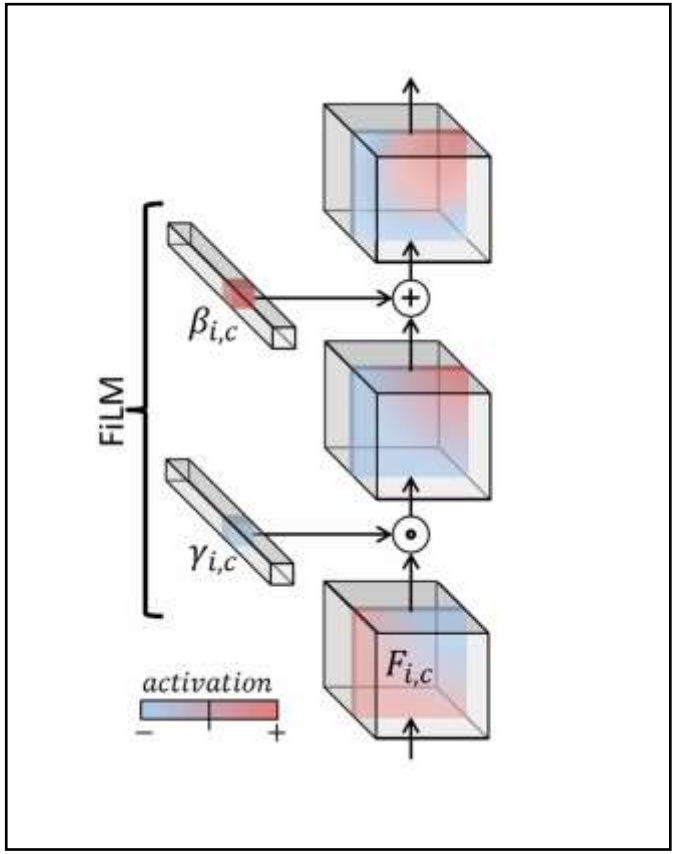
目前只适用于文本域

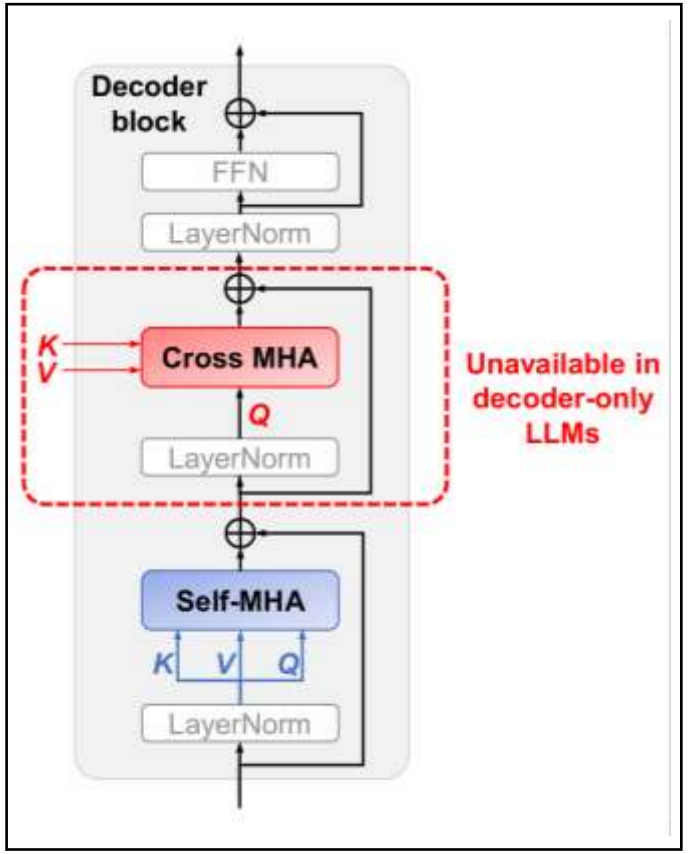


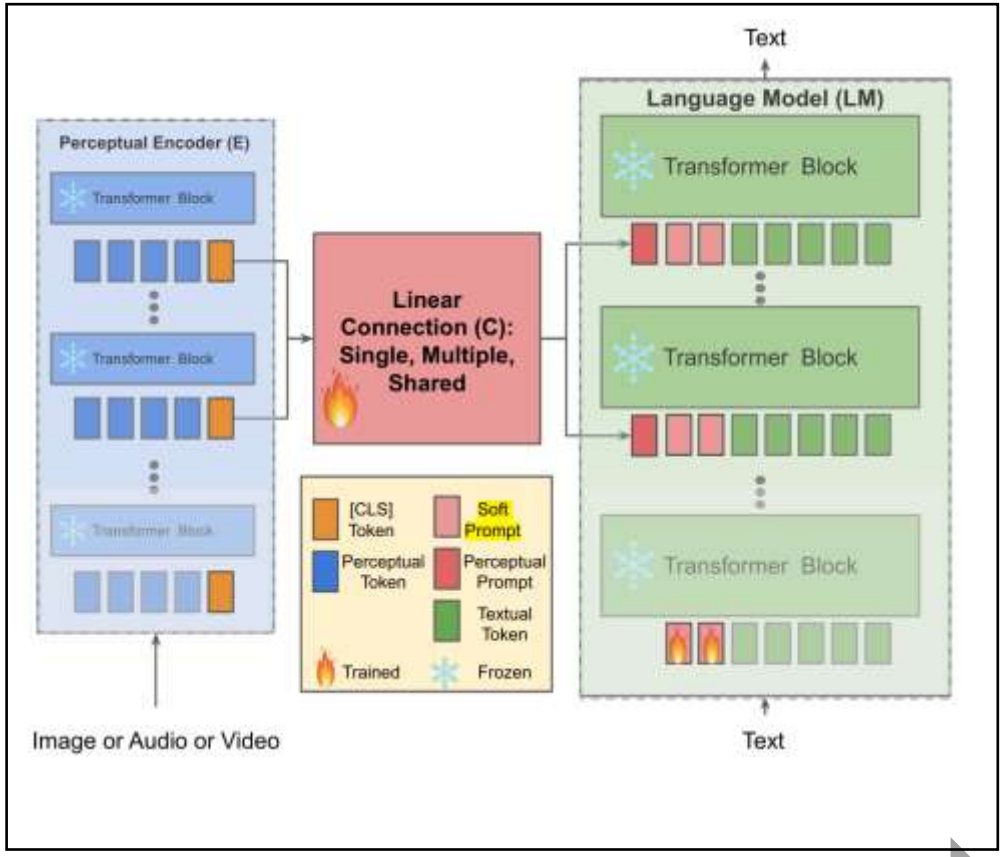



# Background

➢ 有效插入外部模态




FiLM-Like Weighting Schemes

在中间层中将一个模态与其他模态相结合，但只硬编码两个模态，并且需要从头开始进行特定于领域的结构设计。

Cross-MHA Mechanisms

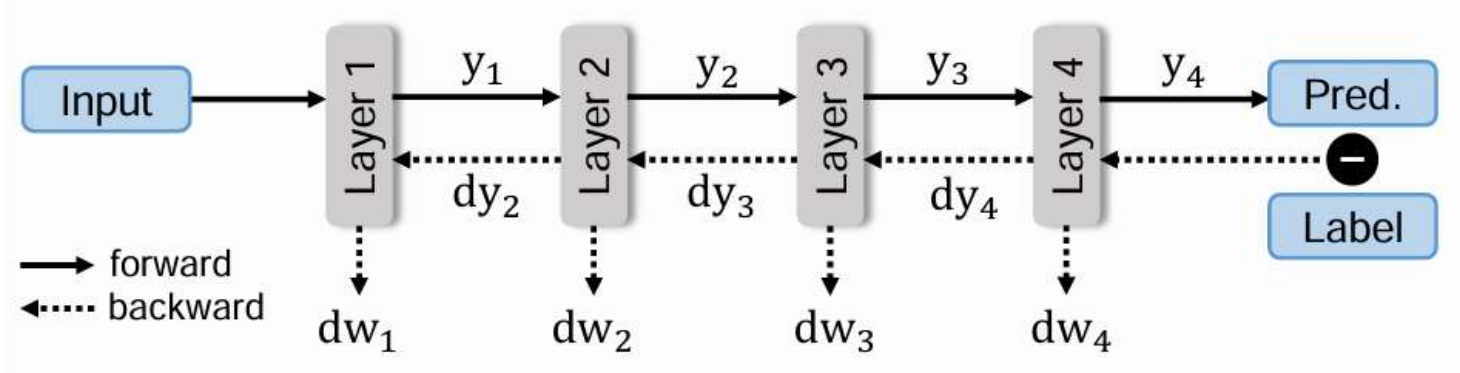
不能应用于仅解码器的大型语言模型。

Multimodal Tokens Concatenation

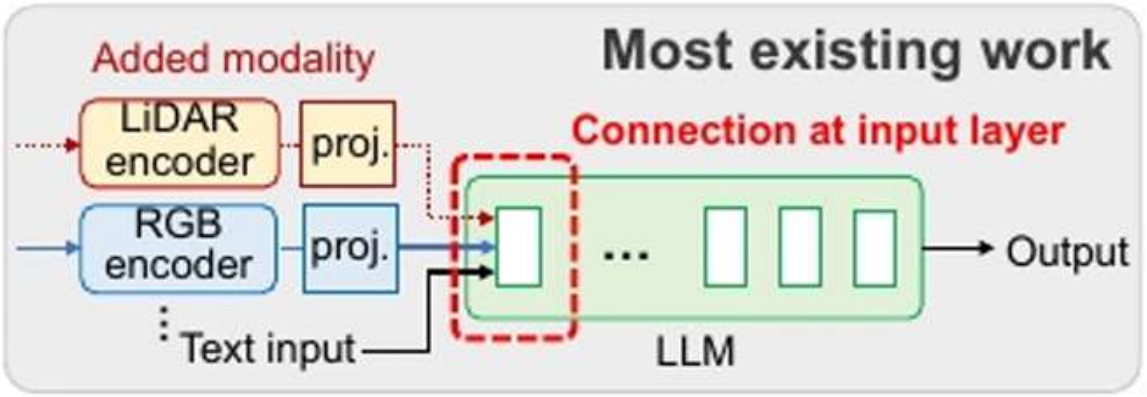
多模态tokens以硬编码方式接入大型语言模型块，并没有针对训练精度或速度进行优化。

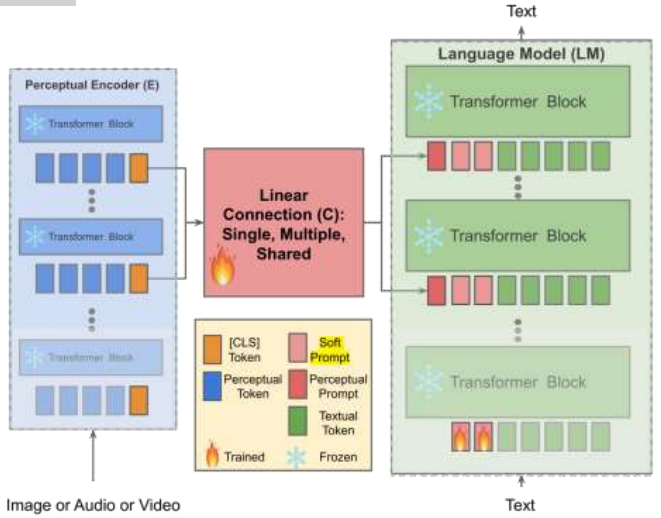
# Background

➢ 模态适应的FLOPs模型




冻结的层，仍会计算loss并反向传播




接入LLM输入层

接入LLM后面的层

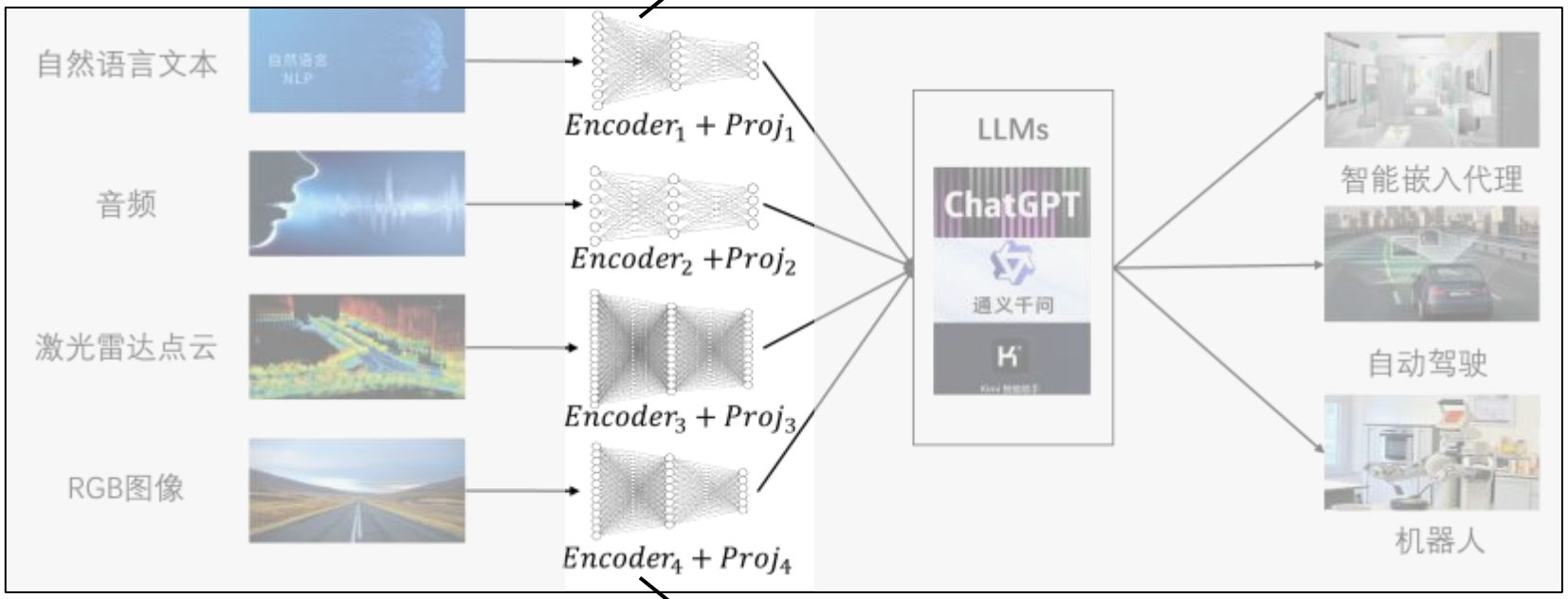
# 提纲

# Design

- 单模态编码器的选择

预训练的Transformer（ViT、AST、Autoformer、PointBERT）+ Projector

很方便!




其他编码器（MLP、CNN、LSTM）

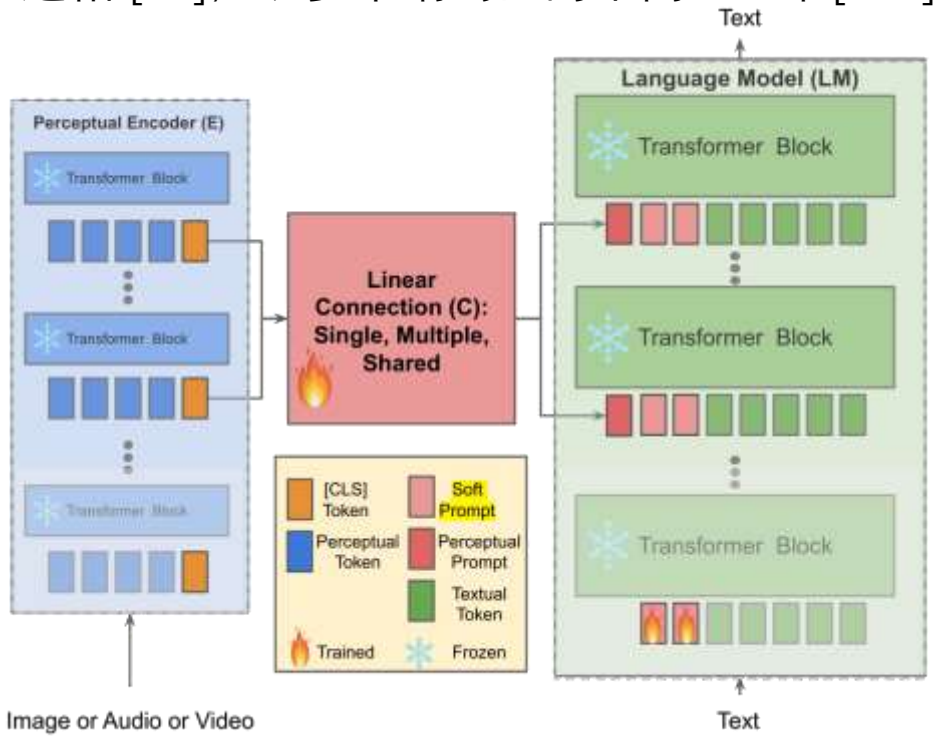
需要额外的工作，来将编码的结果与文本域的token对齐

# Design

➢ **特征表示**

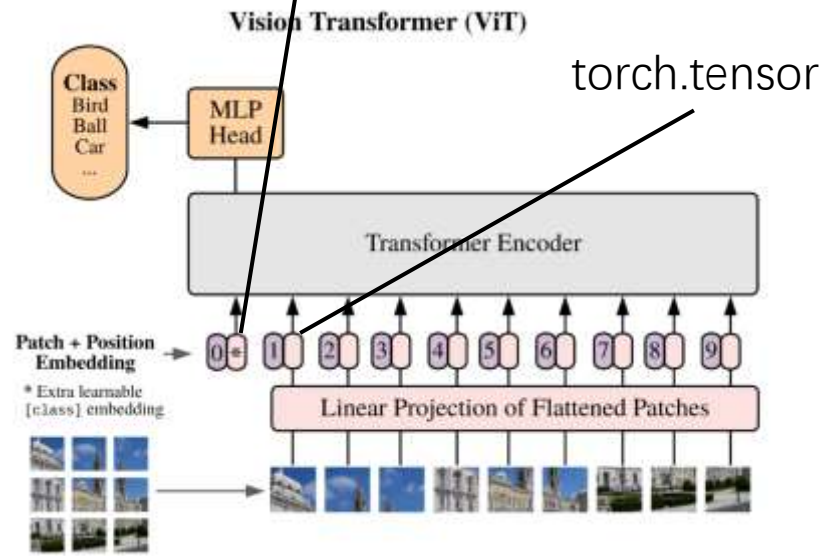
大多数基于Transformer的编码器将特征提取到[CLS] token序列中。

遵循 [51]，从多个编码器的中间层提取[CLS]




非分类任务的Transformer模型，使用“averaged-pooled tokens”作为特征表示。

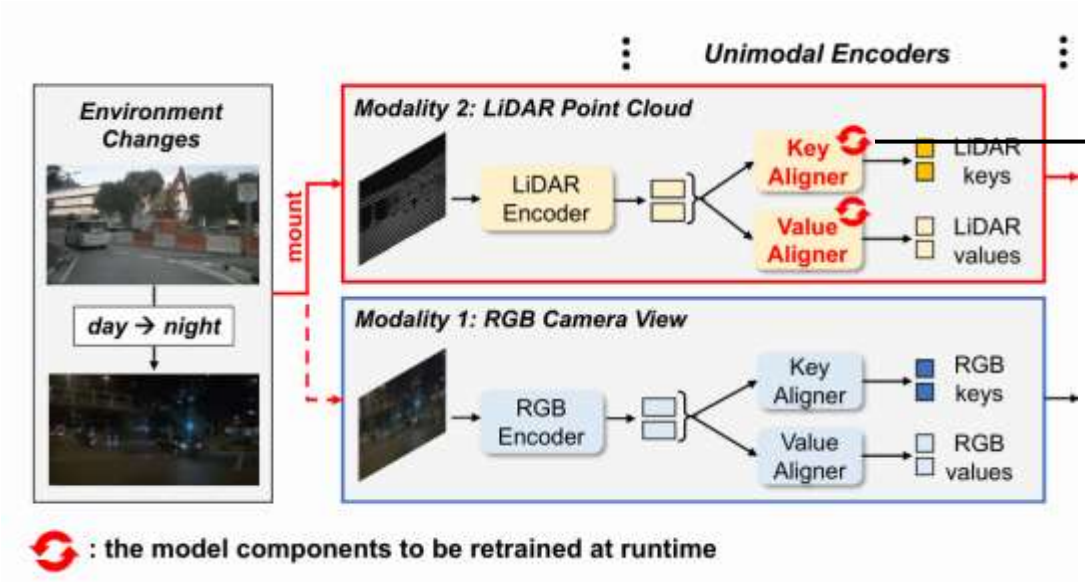
nn.Parameter([torch.tensor])

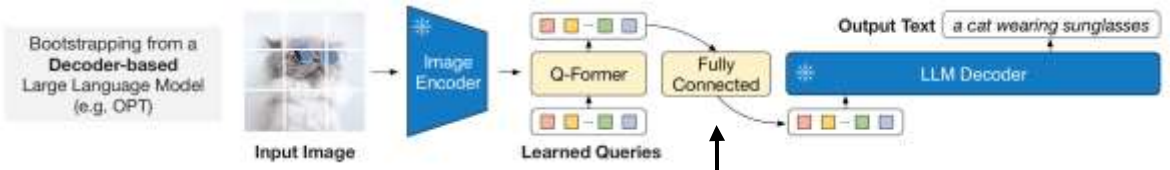



- 特殊的标记
- 位于输入序列的开始位置
- 用作整个序列的表示，进而用于分类

# Design

➢ 键与值对齐器



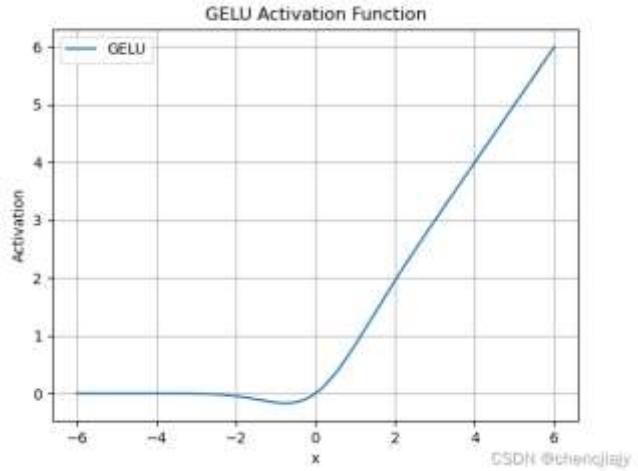




As shown in Figure 3, we use a fully-connected (FC) layer to linearly project the output query embeddings $Z$ into the same dimension as the text embedding of the LLM. The

BLIP-2[35]

将来自单模编码器的多模态token投射到K-V对中，这些token应该与LLM块中的文本K-V对对齐。

$$\text{Aligner}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2,$$

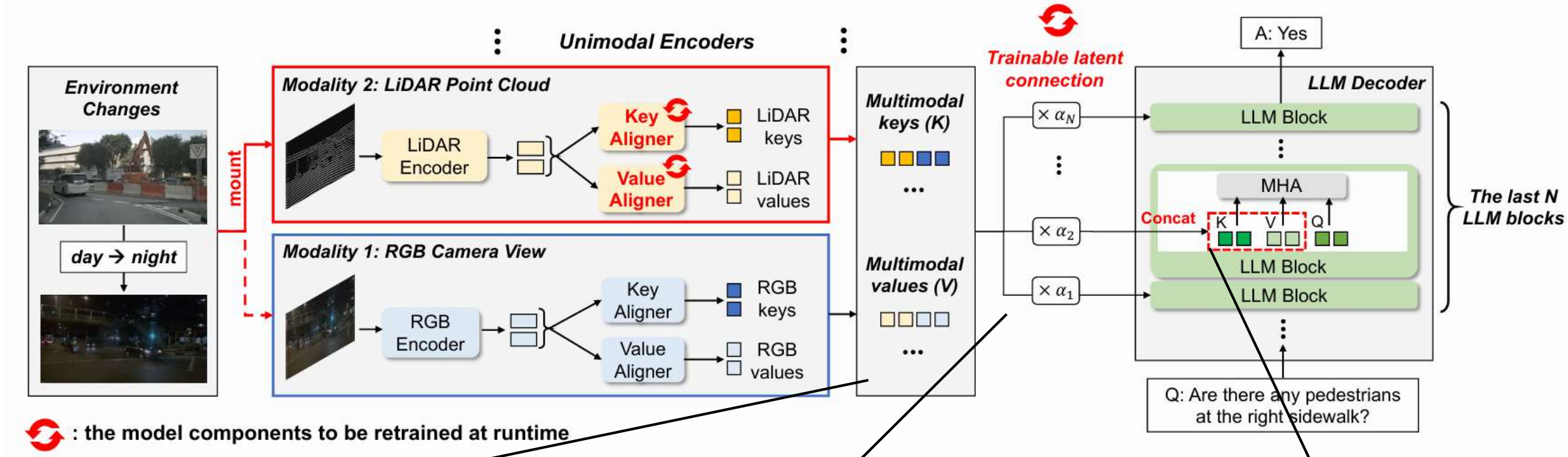



Gaussian Error Linear Unit

Transformer架构中的首选激活函数

# Design

➢ 可训练的潜在连接



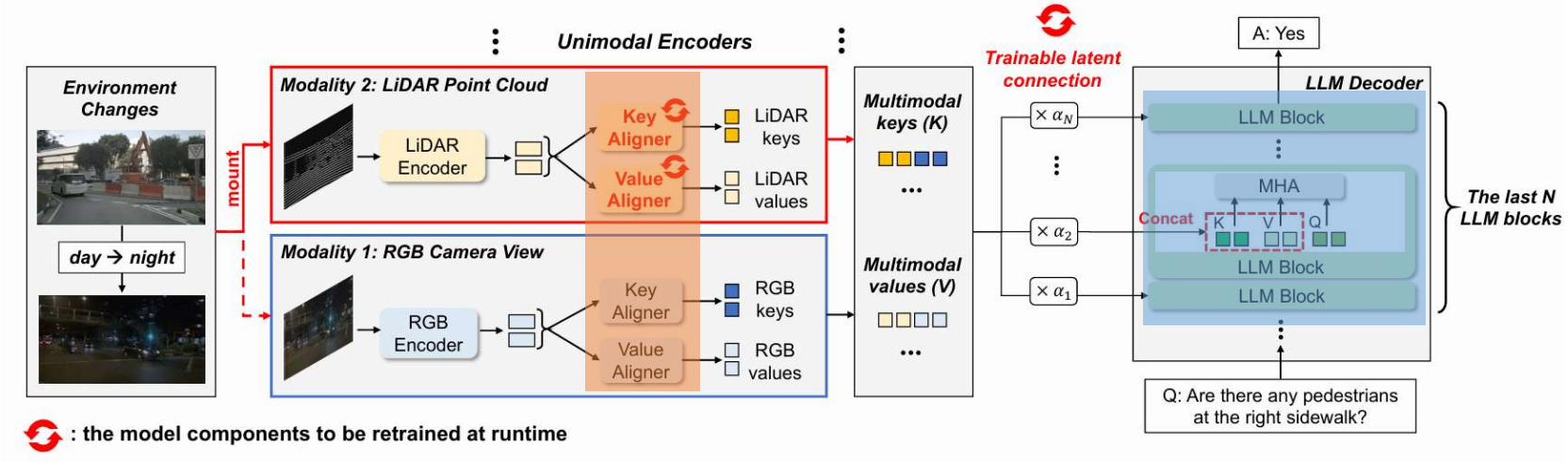

$$\mathbf{K}_{\mathrm{mm}} = [K_1; K_2; ..., K_M] + \mathbf{T}_{\mathbf{k}}^{\mathrm{pos}} \qquad \mathbf{K'}_{\mathrm{mm}} = [\alpha_1 \mathbf{K}_{\mathrm{mm}}, \alpha_2 \mathbf{K}_{\mathrm{mm}}, ..., \alpha_N \mathbf{K}_{\mathrm{mm}}] \qquad \mathbf{K}_{\mathrm{ext}} = [\alpha_j \mathbf{K}_{\mathrm{mm}}; \mathbf{K}_{\mathrm{txt}}]$$

$$\mathbf{V}_{\mathrm{mm}} = [V_1; V_2; ..., V_M] + \mathbf{T}_{\mathbf{v}}^{\mathrm{pos}} \qquad \mathbf{V'}_{\mathrm{mm}} = [\alpha_1 \mathbf{V}_{\mathrm{mm}}, \alpha_2 \mathbf{V}_{\mathrm{mm}}, ..., \alpha_N \mathbf{V}_{\mathrm{mm}}] \qquad \mathbf{V}_{\mathrm{ext}} = [\alpha_j \mathbf{V}_{\mathrm{mm}}; \mathbf{V}_{\mathrm{txt}}].$$

$$\alpha_j = \mathrm{sigmoid}(w_j / T)$$

# Design

➢ 确定连接数量




反向传播FLOPs可以计算为：

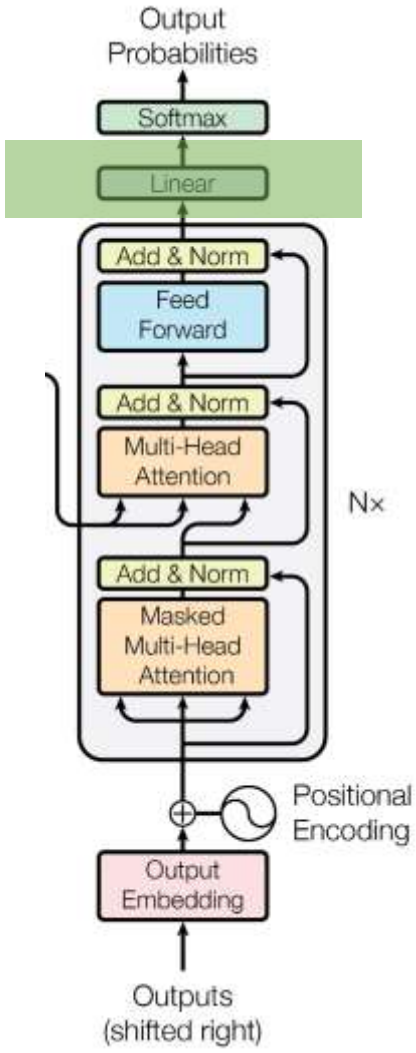
$$T_{\text{backprop}}(N) = T_{\text{dw}}^{\text{Aligners}} + T_{\text{dy}}^{\text{Emb}} + \frac{N}{L} T_{\text{dy}}^{\text{LLM}},$$

- 用于计算Aligner权重更新的FLOPs
- 通过LLM的输出嵌入层传递激活梯度的FLOPs
- 通过所有LLM块传递激活梯度的FLOPs

观察到：

$$T_{\text{dw}}^{\text{Aligners}} + T_{\text{dy}}^{\text{Emb}} \ll T_{\text{dy}}^{\text{LLM}}$$

**训练成本取决于N**




Transformer Decoder Layer

# Design

➢ **模态自适应完整流程**

**Algorithm 1** Modality Adaptation in mPnP-LLM

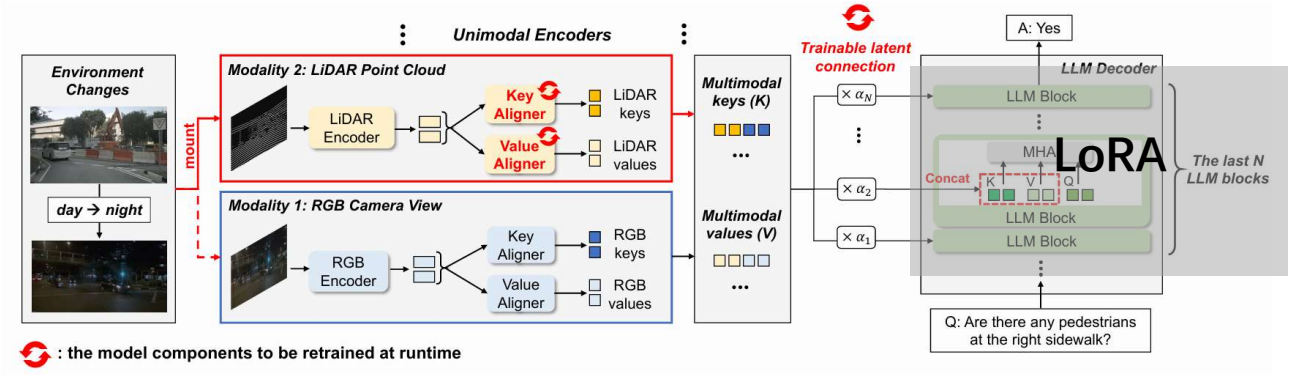
**Require:** : A set of pre-trained encoders $\mathbf{E} = \{E_1, E_2, ...\}$ and the corresponding aligners $\mathbf{A} = \{A_1, A_2, ...\}$ stored on local external storage; a pre-trained LLM loaded in memory; trainable latent connections $\alpha_{1,...,N}$

```
/* Offline preparation */
E_0, A_0 ← Select(E, A)                          ▷ Load initial encoders
LLM ← Reconnect(E_0, A_0)                        ▷ Connect to LLM
Train(LLM_{k,v}, A_0, α_{1,...,N})               ▷ Offline training
/* Runtime modality adaptation */
while t < T_end do
    E_t, A_t ← Select(E, A)                      ▷ Reload encoders
    LLM ← Reconnect(E_t, A_t)                    ▷ Reconnect
    Train(LLM_{k_{1,...,N}, v_{1,...,N}}, A_t, α_{1,...,N})   ▷ Adapt
end while
```

离线小数据集初训练，
获得基本的QA能力

在线大数据集模态自适应训练

- 系统默认从RGB相机视图的模态开始
- 用户手动将对应模态的编码器和对齐器连接到LLM
- 系统最终能够在夜间自动适应LiDAR点云的形态



# 提纲

# Evaluation

## ➢ 数据集准备

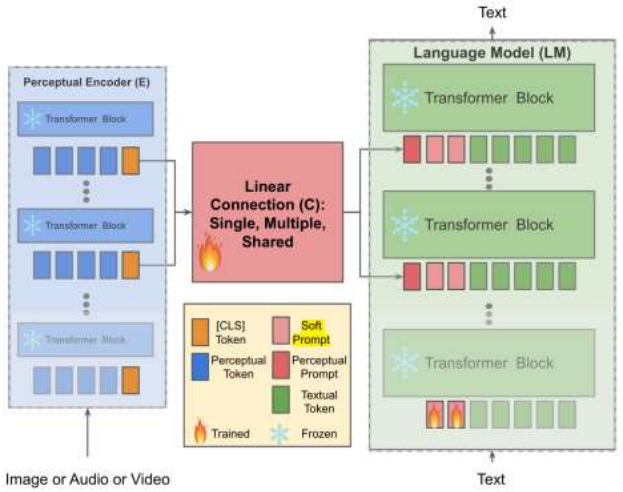

nuScenes-QA数据集太大(~ 460k QA对)

- 以nuScenes-mini数据集为参考，在nuScenes-QA数据集中选出nuScenes-QA-mini数据集
- 4458 day samples & 1138 night samples
- Train : Test = 1 : 1

运行时模态适应评估场景：模型从白天的RGB相机视图切换到夜间的激光雷达点云。

## ➢ 基线方案

- **Full LLM**：将**多模态编码器**与LLM**输入层**连接， 训练投射层并微调整个LLM进行模态适应。
- **PromptFuse**：将**多模态编码器**与LLM**输入层**连接，采用**提示调优**实现模态自适应。
- **eP-ALM**：将**编码器的[CLS] token**与LLM**中间块**硬编码连接，使用**提示调优**实现模态自适应。



# Evaluation



➢ **模态适应成本与准确性**

在day-train-split（C+L）训练，其他情况测试

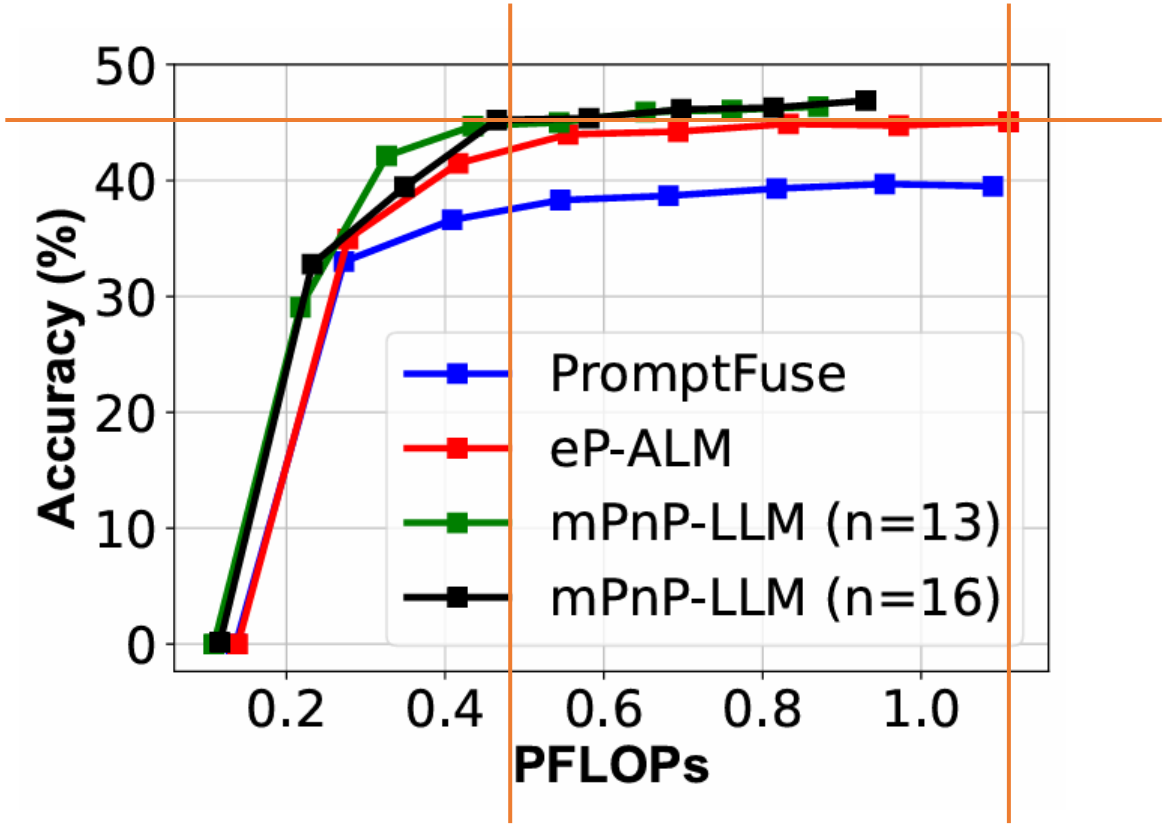
| Method | Accuracy (%) w.r.t Scene & Modality | | | | Cost w.r.t Night (C → L) | |
|---|---|---|---|---|---|---|
| | Day (C) | Night (C) | Night (C → C + L) | Night (C → L) | PFLOPs | Memory (GB) |
| Full LLM | 32.5 | 30.2 | 3.5 | 3.5 | 1.58 | 29.9 |
| PromptFuse | 33.3 | 26.9 | 24.9 | 39.2 | 1.09 | 26.0 |
| eP-ALM | 34.7 | 24.6 | 36.0 | 44.7 | 1.11 | 27.6 |
| mPnP-LLM ($N = 4$) | 25.2 | 21.1 | 22.3 | 24.9 | 0.68 | 23.2 |
| mPnP-LLM ($N = 7$) | 40.1 | 34.1 | 25.6 | 26.3 | 0.74 | 23.9 |
| mPnP-LLM ($N = 10$) | 49.1 | 41.1 | 27.0 | 41.9 | 0.81 | 24.5 |
| mPnP-LLM ($N = 13$) | 49.2 | 40.1 | 38.5 | 46.4 | 0.87 | 23.1 |
| mPnP-LLM ($N = 16$) | 50.3 | 43.4 | 41.7 | 46.9 | 0.93 | 25.9 |
| mPnP-LLM ($N = 19$) | 50.7 | 41.0 | 44.0 | 46.1 | 0.99 | 23.9 |
| mPnP-LLM ($N = 22$) | 47.8 | 39.2 | 41.7 | 47.5 | 1.05 | 26.0 |

Table 1: Performance of mPnP-LLM vs. baseline schemes w.r.t scenes (Day/Night) and modalities (C: 6 RGB camera views, L: LiDAR point cloud) using OPT-1.3B by connecting with different numbers of LLM blocks ($N$). The OPT-1.3B model has 24 LLM blocks in total.

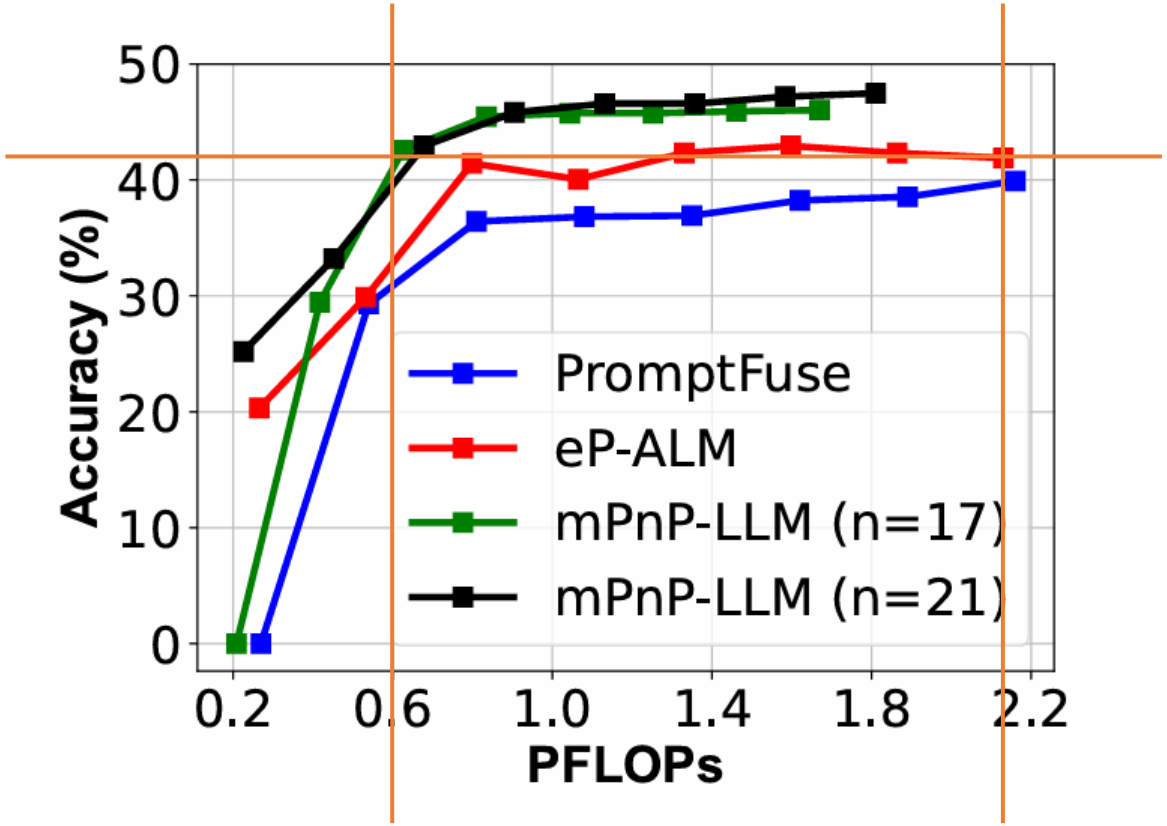
# Evaluation

➢ 模态适应成本与准确性

C⟶C + L




(a) OPT-1.3B




(b) OPT-2.7B

在更大的LLM中，eP-ALM想与mPnP-LLM达到同样的准确性，需要更大的FLOPs

# Evaluation



➢ 消融实验

| Ablated module | Accuracy ($N = 13$) | Accuracy ($N = 17$) |
|---|---|---|
| None | 42.2 | 46.0 |
| Offline train K,V proj. | 37.8 | 42.9 |
| Aligner: MLP→Linear | 37.3 | 43.8 |
| Conn.: trained → fixed | 39.5 | 44.7 |
| LoRA on K,V proj. | 33.8 | 35.4 |

禁用LoRA会导致最大的准确性下降，因为模式适应需要更新K、V投射层，以正确区分新插入的token和现有token。

# Evaluation



➢ LLM大小的影响

| LLM & Method | Accuracy (%) w.r.t Scene & Modality | | | | Cost w.r.t Night (C → L) | |
|---|---|---|---|---|---|---|
| | Day (C) | Night (C) | Night (C → C + L) | Night (C → L) | PFLOPs | Memory (GB) |
| **OPT-350M** | | | | | | |
| PromptFuse | 27.7 | 18.8 | 0.0 | 34.9 | 0.32 | 21.1 |
| eP-ALM | 25.3 | 13.5 | 24.1 | 36.9 | 0.33 | 20.0 |
| mPnP-LLM ($N = 10$) | 44.0 | 37.5 | 26.1 | 38.5 | 0.25 | 19.8 |
| mPnP-LLM ($N = 13$) | 45.4 | 36.6 | 31.6 | 37.7 | 0.26 | 20.0 |
| mPnP-LLM ($N = 16$) | 45.9 | 37.8 | 31.7 | 40.5 | 0.28 | 21.0 |
| **OPT-1.3B** | | | | | | |
| PromptFuse | 33.3 | 26.9 | 24.9 | 39.2 | 1.09 | 26.0 |
| eP-ALM | 34.7 | 24.6 | 36.0 | 44.7 | 1.11 | 27.6 |
| mPnP-LLM ($N = 10$) | 49.1 | 41.1 | 27.0 | 41.9 | 0.81 | 24.5 |
| mPnP-LLM ($N = 13$) | 49.2 | 40.1 | 38.5 | 46.4 | 0.87 | 23.1 |
| mPnP-LLM ($N = 16$) | 50.3 | 43.4 | 41.7 | 46.9 | 0.93 | 25.9 |
| **OPT-2.7B** | | | | | | |
| PromptFuse | 35.7 | 26.1 | 28.8 | 39.8 | 2.16 | 36.4 |
| eP-ALM | 37.3 | 24.7 | 24.9 | 41.9 | 2.13 | 36.4 |
| mPnP-LLM ($N = 13$) | 50.2 | 36.9 | 31.3 | 42.2 | 1.56 | 28.4 |
| mPnP-LLM ($N = 17$) | 51.2 | 37.2 | 27.6 | 46.0 | 1.67 | 30.1 |
| mPnP-LLM ($N = 21$) | 52.3 | 42.2 | 44.3 | 46.4 | 1.81 | 28.9 |

mPnP-LLM通过其可训练连接和轻量级训练成本，在不同大小的语言模型上表现出色，并且在处理大容量点云数据时，能够有效节省GPU内存。

# Evaluation

- 不同LLM下的表现

预训练更侧重自然语言处理任务

↓

**推理能力不同**

↑

预训练更侧重跨语言文本生成

| LLM & Method | Day (C) Acc. (%) | Night (C → L) Acc. (%) / PFLOPs / Mem. (GB) |
|---|---|---|
| **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-LLM ($n = 10$) | 49.1 | 41.9 / 0.81 / 24.5 |
| mPnP-LLM ($n = 13$) | 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-LLM ($n = 10$) | 39.4 | 25.8 / 0.73 / 22.7 |
| mPnP-LLM ($n = 13$) | 44.0 | 27.0 / 0.76 / 25.8 |

# Evaluation



➢ 训练样本量的影响

| Method | zero-shot | | # sample = 395 (60%) | |
|---|---|---|---|---|
| | Accuracy (%) | PFLOPs | Accuracy (%) | PFLOPs |
| PromptFuse | 26.9 | - | 37.3 | 0.65 |
| eP-ALM | 24.6 | - | 41.5 | 0.67 |
| mPnP-LLM ($N = 13$) | 40.1 | - | 41.0 | 0.52 |
| mPnP-LLM ($N = 16$) | 43.4 | - | 43.9 | 0.56 |
| **Method** | **# sample = 527 (80%)** | | **# sample = 659 (100%)** | |
| | **Accuracy (%)** | **PFLOPs** | **Accuracy (%)** | **PFLOPs** |
| PromptFuse | 39.0 | 0.87 | 39.2 | 1.09 |
| eP-ALM | 43.7 | 0.89 | 44.7 | 1.11 |
| mPnP-LLM ($N = 13$) | 44.0 | 0.70 | 46.4 | 0.87 |
| mPnP-LLM ($N = 16$) | 44.2 | 0.74 | 46.9 | 0.93 |

量变没有引发质变，因此可以在准确性与计算量中做权衡

# Evaluation

- 边缘设备上的模态自适应

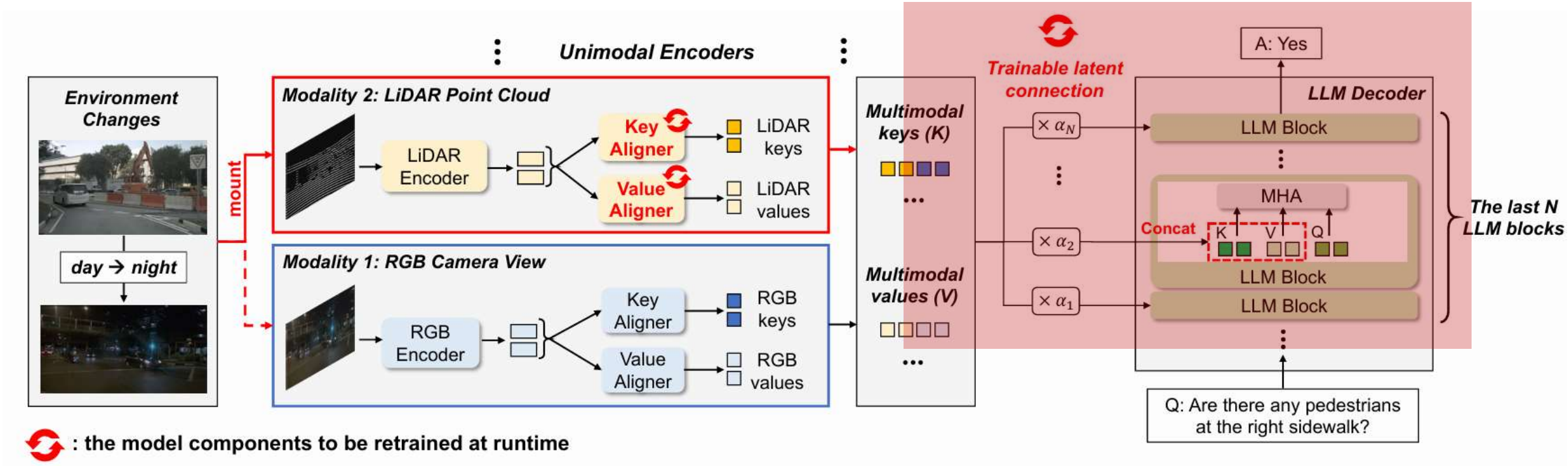
| Device setup | GPU Freq. | mPnP-LLM ($N = 13$) |
|---|---|---|
| RTX A6000 300W | 1.9GHz | 9.40 samples/s |
| AGX Orin 30W | 612MHz | 0.33 samples/s |
| AGX Orin 50W | 816MHz | 0.92 samples/s |
| AGX Orin MAXN | 1.3GHz | 1.41 samples/s |

边缘设备的性能虽不如工作站级GPU，但也能用

# 提纲

# Conclusion

在本文中，我们提出了mPnP-LLM，这是一种新的技术，允许在嵌入式AI中对LLM进行动态运行时模态自适应。mPnP-LLM在保持与现有方案相同精度的情况下，还实现了FLOPs的降低。在相同的计算预算下， mPnP-LLM的任务精度优于现有的最佳方案。



# Evaluation

- **能否用到我们的场景?**

  未来可能可以用上。目前，我们使用大模型，还停留在调用API的程度，没有涉及触动LLM模型本身的程度。但是，未来如果多模态大模型的能力更强后，我们可以尝试在本地部署大模型，然后可以借助这种方式，将感知等任务也交由LLM完成，而不是再让LLM帮我们调用外部模型。

- **能否进一步提高?**

  很显然，本文任务的准确性普遍不超过50%，仍有很大提升空间。第一个，就是他用的LLM不够好，而LLM本身能力的瓶颈，是可以使用更强的LLM来提升的。第二个，就是他的模态适应性不太“真适应”，可以试试复制N份，然后每一份中的每个模态提供一个可学习参数，用于决定“拼接程度”，如果不放心，在连接LLM块的时候，也可以再额外添加本文的这种参数（总体类似于输入门和输出门）。

- **能否泛化?**

  这种“加权”的方法，其实是比较常见的，比如GRU和LSTM的门控机制，也是一种加权。在后续研究中，对于输入存在“选择”或者“侧重”的场景，都可以尝试用这种添加可学习参数的方式。