CAS2LLM: Efficient, Lightweight and Multi-frame Models for QA in Autonomous Driving

Jiabao Wang, Zepeng Wu, Yukuan Yang†, Yu Xia, Qing Tan

Abstract

Vision-Language Models (VLMs) and Multi-Modal Language Models (MMLMs) have emerged as pivotal components in the domain of autonomous driving research. Despite their significance, existing systems struggle to provide precise responses to scene-related inquiries. To surmount this challenge, we have developed CAS2LLM, an efficient, lightweight, multi-frame vision-language model tailored for visual question-answering tasks within the autonomous driving context. CAS2LLM outperforms its predecessors on the nuScenes dataset across multiple metrics, including Accuracy, GPT-4, BLEU1-4, ROUGE-L, CIDEr, and Match, achieving respective improvements. Furthermore, CAS2LLM exhibits an adeptness at distilling pertinent information from traffic scenes and adeptly addressing a diverse range of subtask questions pertinent to autonomous driving. Group id: Course Assignment. Submission ID: 0a1bd0d2-5925-47a7-bd5f-295769d99077.

1. Introduction

The main goal of this competition is to furnish multi-view images as input, tasking the model with answering questions that span various driving scenarios. We refrain from discussing competition specifics here; for further information, please visit https://huggingface.co/spaces/AGC2024/driving-with-language-2024. The structure of this paper is as follows: Section 2 presents a detailed exposition of the proposed CAS2LLM method, while Section 3 outlines the experimental setup and results.

2. Methodology

The CAS2LLM network, enhancing upon the *llama adapter v2* [2], is presented. We succinctly delineate the architecture and encompassing processes: image segmentation and feature extraction, sub-image embedding, view embedding, and the fine-tuning of CAS2LLM. Our main contributions are as follows:

- By integrating view embeddings, we enhance the model's ability to interpret multi-view images, leading to more accurate recognition of feature origins and an improvement over previous suboptimal performances.
- We address distortion issues with a novel approach to view segmentation and sub-image embedding, allowing for richer feature extraction while maintaining the original aspect ratio.
- We implement visual LoRA fine-tuning and an Agent Fusion framework to efficiently enhance features and achieve superior fusion outcomes.
- Building on these advancements, we introduce CAS2LLM, a streamlined multi-frame vision-language model specifically designed for visual question-answering in autonomous driving. CAS2LLM shows significant performance improvements over existing models on the DriveLM-nuScenes dataset across key metrics such as accuracy, GPT-4, BLEU1-4, ROUGE-L, CIDEr, and Match scores, marking a substantial advancement in autonomous driving technology.

2.1. Network architecture

The network's comprehensive architecture is illustrated in Figure 1. It parallels the *llama adapter v2*, encompassing a visual encoder, a multimodal alignment module, and an expansive language model. Specifically, the visual encoder employs *Clip Vit-L/14* [6], the multimodal alignment is facilitated by a pure transformer encoder, and the language model is the *llama2-7b* variant [8]. Each module is initialized with the competition-prescribed baseline pre-trained weights.

2.2. Image segmentation and visual features extraction

This approach to segmenting images was developed after attempting view embedding. The original image resolution is 1600×900. During the preprocessing stage, the image transform scales the original image resolution down

Jiabao Wang, Zepeng Wu, and Yukuan Yang are with the Institute of Software, Chinese Academy of Sciences (ISCAS), Beijing 100190, China. Yu Xia is with Central South University, Changsha 410083, China. Tan Qing is with Nanjing University of Science and Technology, Nanjing 210094, China.

†Corresponding author. Email: yangyukuan@iscas.ac.cn.

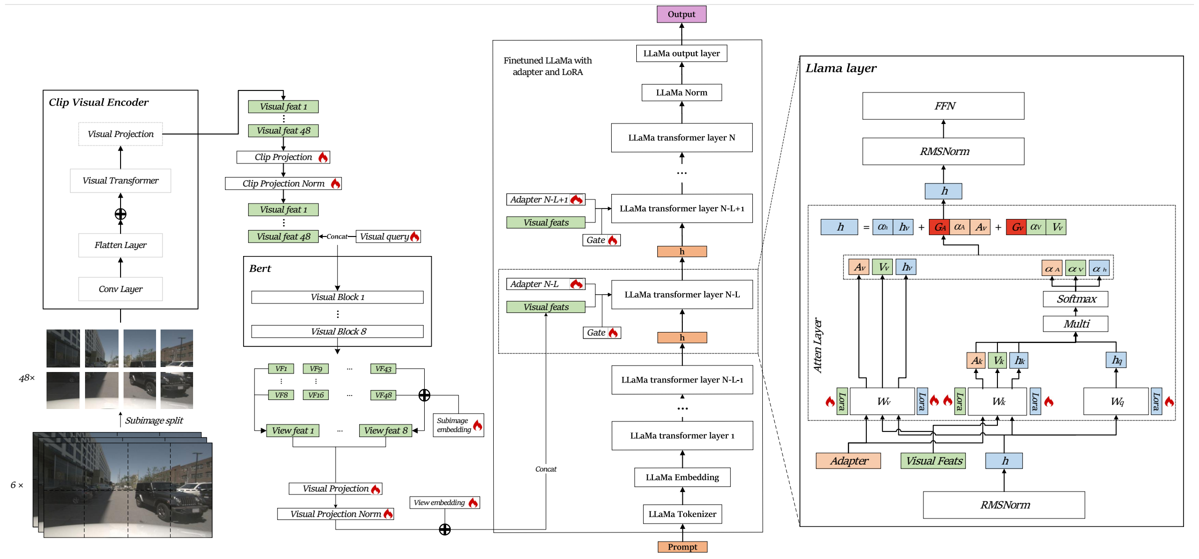

Figure 1. **The overall framework of CAS2LLM.** The regions indicated by a '*fire*' symbol here denote the parameters that are subject to training. Each view is first segmented into subimages, and the features output by Bert are embedded into the subimages. The visual features are further embedded with respect to view embedding. Visual LoRA module (depicted in green) is introduced to transform the visual features, and gate coefficient is utilized to control the degree of fusion of the visual features.

to 224×224. It is evident that this results in a significant change in the aspect ratio of the objects in the image, which may lead to distortion of the objects. Moreover, after scaling, the reduced resolution leads to a decrease in image clarity, potentially affecting the model's understanding of the image. This is similar to the improvements made in Fast RCNN [3], so we adopted a relatively straightforward approach. The segmentation approach is shown in Figure 1.

An image is evenly divided into 8 parts:

$$V_i \xrightarrow{\text{segment}(\cdot)} \{V_{i,1}, V_{i,2}, .., V_{i,8}\}. \tag{1}$$

Each part is with a size of 400×500, and an aspect ratio close to $1:1$. Subsequent processing is then performed.

A single sample consists of 6 images. After sub-image segmentation, there are a total of 48 sub-images. Each of these 48 sub-images is then individually processed by the visual encoder for feature extraction:

$$\{V_{1,1}, V_{1,2}, .., V_{6,8}\} \xrightarrow{\text{visual encoder}(\cdot)} \{V^{'}_{1,1}, V^{'}_{1,2}, .., V^{'}_{6,8}\} \tag{2}$$

where $V^{'}_{i,j} \in R^{l \times v_d}$, l represents the number of features in a single sub-image.

After extracting visual features, they are first projected to the input dimension of the multimodal alignment module through a projection layer and a regularization layer:

$$V^{'}_{i,j} \xrightarrow{\text{norm}(\text{proj}(\cdot))} V^{''}_{i,j} \tag{3}$$

where $V^{''}_{i,j} \in R^{l \times a_d}$, a_d represents the input dimension of the multimodal alignment module.

Then, the features need to be concatenated with the Visual-Query and together pass through a multimodal alignment module for feature alignment, compressing the subgraph information into the Visual-Query:

$$[V^{''}_{i,j} : VQ] \xrightarrow{\text{alignment module}(\cdot)} VQ_{i,j} \tag{4}$$

where $VQ_{i,j} \in R^{q_l \times a_d}$, q_l is the length of Visual-Query.

Finally, all the extracted Visual Queries need to be projected to the same dimension as the embedding of the large model:

$$VQ_{i,j} \xrightarrow{\text{norm}(\text{proj}(\cdot))} VQ^{'}_{i,j} \tag{5}$$

where $VQ_{i,j} \in R^{q_l \times t_d}$, t_d represents the input dimension of the LLM.

The Visual Query used here, as per our understanding, is a component used to specify the information to be extracted. It tells the multimodal alignment module which features to obtain, such as buildings, vehicles, pedestrians, etc. Based on this idea, we have also separately designed a method for constructing the Visual Query, which will be introduced in Section 2.6.

2.3. Subimage embedding

In this section, we will introduce how to perform subgraph embedding, which is used to identify which parts come from which sub-images of a single view.

For subgraph embedding, we designed two embedding methods and conducted experiments separately. Firstly, the first method creates a new trainable token embedding for each feature token of each sub-image:

$$SE^{(1)} = \{SE_1^{(1)}, ..., SE_8^{(1)}\} \tag{6}$$

where, the length of $SE_i^{(1)}$ is q_l.

The second method involves creating a shared trainable token embedding for the features of each sub-image:

$$SE^{(2)} = \{SE_1^{(2)}, ..., SE_8^{(2)}\} \tag{7}$$

where, the length of $SE_i^{(2)}$ is 1.

For the first embedding method, we directly perform an addition operation between the corresponding embeddings and the $VQ_{i,j}^{'}$:

$$VQ_{i,j}^{'} = VQ_{i,j}^{'} + SE_j^{(1)} \tag{8}$$

For the second embedding method, we need to repeat the corresponding embeddings q_l times before performing an addition operation with the $VQ_{i,j}^{'}$:

$$VQ_{i,j}^{'} = VQ_{i,j}^{'} + repeat(SE_j^{(2)}, q_l) \tag{9}$$

We first attempted the second embedding method and found that the effect of performing subgraph embedding after sub-image segmentation was better than not performing sub-image segmentation. Subsequently, we wanted to further improve the matching score, hoping that subgraph embedding could directly represent the source information of the features. Therefore, we tried the first embedding method. However, after testing, we found that the first embedding method did not achieve a higher score than the second embedding method.

2.4. View embedding

In this section, we will introduce how to perform view embedding, which is used to represent which features come from which view.

Originally, the pre-training task of *llama adapter v2* was a single-view task used to describe image information. In the source code provided in the competition's GitHub repository, all view features were simply concatenated, and no positional embedding was applied during subsequent cross-attention with text features. This led to the model's inability to distinguish which features came from which view, resulting in poor model performance. Based on this idea, we added a simple view embedding method.

We created a shared trainable token embedding (VE) for the features of each view:

$$VE = \{VE_1, .., VE_6\} \tag{10}$$

For VQs from the same view, we repeated the corresponding VE $8 \times q_l$ times and then added the two together:

$$VQ_{i,j}^{'} = VQ_{i,j}^{'} + repeat(VE_i, 8 \times q_l) \tag{11}$$

Finally, all VQs are concatenated to obtain the visual feature sequence (VF), which is used for subsequent feature fusion:

$$VF = concat(\{VQ_{i,j}^{'} | i = 1, ..., 6, j = 1, ..., 8\}) \tag{12}$$

2.5. Fine-tuning LLM

When attempting fine-tuning of the LLM, we experimented with using LoRA [4], BitFit [1], and P-tuning [5] separately. After conducting ablation experiments, we found that fine-tuning with a combination of LoRA and BitFit resulted in faster convergence and the best performance.

We added a text LoRA block to the parameter matrices of all layers of the LLM, except for the kv matrix, where we added bias and normalization layer parameters. We also continued fine-tuning with the adapter from *llama adapter v2*, although we believe its role may be more in guiding the fusion of multimodal features rather than organizing textual information. Additionally, for llama layers where feature fusion is required, we added a visual LoRA block to the attention layer's kv matrix. This was because, even though feature alignment was performed by the multimodal alignment module, differences may still exist between features from different modalities. Therefore, we hoped to address feature variations through additional LoRA blocks.

In addition, to prevent large differences in attention coefficients between text features and adapter, as well as visual features, which may lead to a decrease in attention to their own features, we set multiple heads for both adapter and visual features and initialized them to 0 to prevent significant fluctuations during the early stages of training.

Next, we will detail how to reuse llama parameters and use fine-tuning parameters to achieve modal fusion.The main task lies in the attention layer of each llama layer, where we denote the latent variable sequence of the text features in the current layer as t_h.

After introducing LoRA fine-tuning, the result of text feature after calculating the qkv matrix and the corresponding LoRA block is:

$$\begin{aligned} t_h^q &= W_q t_h + lora_q^t(t_h) \\ t_h^k &= W_k t_h + lora_k^t(t_h) \\ t_h^v &= W_v t_h + lora_v^t(t_h) \end{aligned} \tag{13}$$

The resulting visual feature after calculating the kv matrix and the corresponding visual LoRA block is:

$$\begin{aligned} v_h^k &= W_k VF + lora_k^t(VF) \\ v_h^v &= W_v VF + lora_v^t(VF) \end{aligned} \tag{14}$$

The resulting adapter feature after calculating the kv matrix is:

$$\begin{aligned} a_h^k &= W_k adapter + lora_k^t(adapter) \\ a_h^v &= W_v adapter + lora_v^t(adapter) \end{aligned} \tag{15}$$

It is worth noting that we did not specifically set a corresponding LoRA block for the adapter. The reason is that the adapter itself is trainable and can adapt to the kv matrix on its own.

Calculating the attention coefficients of text features, visual features, and adapter features with respect to the text features separately, using the same approach as the llama adapter [9]:

$$\begin{aligned} \alpha_{t2t} &= Softmax(t_h^q (t_h^k)^T / \sqrt{d_{head}}) \\ \alpha_{v2t} &= Softmax(t_h^q (v_h^k)^T / \sqrt{d_{head}}) \\ \alpha_{a2t} &= Softmax(t_h^q (a_h^k)^T / \sqrt{d_{head}}) \end{aligned} \tag{16}$$

where d_head represents the feature dimension of a single head. For α_{v2t} and α_{a2t}, the attention coefficients of each head are multiplied by the corresponding gating coefficient and also the corresponding global gating coefficient:

$$\begin{aligned} \alpha_{v2t} &= v_G * \alpha_{v2t} * v_gate \\ \alpha_{a2t} &= a_G * \alpha_{a2t} * a_gate \end{aligned} \tag{17}$$

Finally, calculate the updated value of t_h:

$$t_h = \alpha_{t2t} t_h^v + \alpha_{v2t} v_h^v + \alpha_{a2t} a_h^v \tag{18}$$

2.6. Agent Fusion

In this subsection, we present a previously attempted approach that, though not optimal, retains significant potential. We aim to document its key concepts and applications.

When humans tackle visual question answering tasks, they typically adopt one of two workflows: either memorizing image details before addressing the question or analyzing the question first and then seeking pertinent image information. Logically, the latter approach is more effective as it leverages question-informed targeting.

Currently, Visual Queries, serving as memory regions, treat all image information uniformly. For instance, in response to the question ”How many pedestrians are in the front view?”, a generic Visual Query might detect pedestrians but fail to quantify them.

To address this, we explored the possibility of an Agent that extracts question-specific information before analyzing image features. This Agent would replace the traditional Visual Query.

We implemented this idea by constructing a trainable agent and conducted two experiments. In the first experiment, we passed the agent and the question into llama, reusing the weights of the initial llama layers to build the agent. In the second experiment, we constructed a small Bert model, concatenated the agent and the question, and then passed them into this small Bert model for information extraction (due to space constraints, the structure diagram will not be shown).

In both experiments, we found that the scores for accuracy were the highest. We speculate that this may be because choice and true/false questions contain more specific and directed information, such as view positions and coordinates. Other types of questions are broader, which may cause the agent to be uncertain about what information to extract, leading to a performance drop. In future work, we will continue to refine the agent and select appropriate datasets for further experimentation.

3. Experiment setup and results

Here we will present the details of our experiments, including the fine-tuning processes, LLM templates, and results.

Fine-tuning processes: We conducted fine-tuning on the complete official DriveLM-nuScenes [7] training set, encompassing 4,072 unique multi-view frames and 377,983 individual QA pairs. The model parameter configurations detailed in this paper are as follows: the Lora rank is set to 16, q_l is specified at 10, the adapter layer is positioned at layer 31, the base learning rate (blr) is 10e-4, the weight decay is 0.02, and the efficient batch size is 96. The fine-tuning process spanned 3 epochs, utilizing 4 NVIDIA A100 80GB GPUs, and included a single warm-up epoch. The entire fine-tuning regimen was completed in approximately 36 hours.

LLM templates: For the input questions and answers, we used the default template provided by the competition's official *llama adapter v2*, without any Prompt engineering.

Experiment results: Table 1 shows the results of various versions of the solutions we attempted. The test results were obtained from the official test server provided by the competition.

Table 1. Results scores table

Scheme	Acc	B-4	Rouge-L	CIDEr	GPT	match	FS
Baseline	0.65	0.54	0.70	0.07	48.3	32.2	0.47
View Embedding	0.7	0.57	0.73	0.12	63.8	36.5	0.56
Agent Fusion	**0.72**	0.59	0.73	0.15	58.6	34.1	0.54
Subimage&View Embedding	0.71	**0.61**	**0.74**	**0.18**	**64.4**	**38.0**	**0.57**

As evident from the table, each attempted modification brought about varying degrees of performance enhancement to the model, indicating the significance of visual feature embedding in improving the model's capabilities. Furthermore, Agent Fusion was implemented on view embedding, and due to time constraints, we were unable to experiment with Agent Fusion on subimage embedding. However, it is noteworthy that Agent Fusion resulted in improvements in certain metrics, which may be attributed to the presence of explicit directional information in the questions corresponding to the acc metric. The validity of this assumption will be verified in our future work.

References

[1] Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Annual Meeting of the Association for Computational Linguistics*, page 1–9, 2022. 3

[2] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, Hongsheng Li, and Yu Qiao. Llama-adapter v2: Parameter-efficient visual instruction model. In *arXiv preprint arXiv:2304.15010*, 2023. 1

[3] Ross Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 2

[4] Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *arXiv preprint arXiv:2106.09685*, 2021. 3

[5] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI Open*, 2023. 3

[6] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 1

[7] Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. In *arXiv preprint arXiv:2312.14150*, 2023. 4

[8] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. In *arXiv preprint arXiv:2302.13971*, 2023. 1

[9] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Lu Pan, Hongsheng Li, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. In *arXiv preprint arXiv:2303.16199*, 2023. 4