LLM-based Query System for Traffic Analysis Using CCTV Footage

November 19, 2024

Argharupa Adhikary
MT23020
argharupa23020@iiitd.ac.in

Shashank Sharma
MT23088
shashank23088@iiitd.ac.in

Shubham Kale
MT23094
shubham23094@iiitd.ac.in

Arunoday Ghorai
MT23023
arunoday23023@iiitd.ac.in

Nilanjana Chatterjee
PhD22020
nilanjanac@iiitd.ac.in

Neeraj
PhD22016
neeraji@iiitd.ac.in

Abstract

This project aims to develop a system that enables users to query CCTV footage of traffic using a Large Language Model (LLM) to generate detailed, text-based reports. The system allows users to identify specific vehicles, such as those of a particular color or type, based on natural language queries. Furthermore, it generates statistical reports on traffic patterns, including the number of vehicles observed during a given time period and timestamps when specific vehicles, such as delivery drivers, appear. By leveraging the power of video-based LLMs, the system provides users with actionable insights for improving traffic management and urban planning.

You can find the project's code on GitHub at GitHub Repository.

1 Introduction

Since the project's inception, our team has made significant strides in developing an LLM-based query system for traffic analysis using CCTV footage. The primary objective is to create an intuitive interface that allows users to extract meaningful insights from video data through natural language queries. This report details our progress, including pipeline development, dataset preparation, model inference, and the finetuning process using QLoRA.

2 Pipeline Overview

Our project leverages advanced transformer models to analyze video data for traffic statistics and accident detection. The core components of our pipeline include:

1. **Frame Extraction:** Extracting frames from videos at evenly spaced intervals to ensure comprehensive analysis.
2. **Vehicle Detection and Classification:** Utilizing YOLOv5 for detecting and classifying vehicles into categories such as cars, trucks, bikes, and bicycles.
3. **Model Inference:** Employing the `LlavaOnevisionForConditionalGeneration` model to generate JSON reports based on the extracted frames.
4. **Data Processing:** Cleaning and structuring the model's output for evaluation and further analysis.
5. **Evaluation Metrics:** Assessing the performance of our model using metrics such as accuracy, F1 score, precision, and recall.
6. **Finetuning Pipeline:** Initiating the finetuning process using QLoRA to enhance model performance on our specific dataset.

3 Task Details

Our project is divided into main tasks and side tasks to ensure comprehensive coverage of all aspects of traffic analysis using CCTV footage.

3.1 Main Tasks

- **Accident Detection:** Identifying and classifying accidents in traffic footage.
- **Vehicle Counting:** Counting the number of different types of vehicles in the footage.
- **Create Annotations (YOLO):** Generating annotations for vehicle detection using the YOLO model.
- **Cross Validation on Another Dataset:** Validating our model's performance on a separate dataset to ensure generalizability.

3.2 Side Tasks

- **Video QA:** Developing a question-answering system for querying video data.
- **Model Deployment:** Deploying the trained models for real-time inference and usage.
- **Multimodal LLMs for Videos:** Exploring and integrating multimodal LLMs such as LLAVA Onevision, Qwen2, Flamingo, and Kosmos for enhanced video analysis.

3.3 Datasets

- **DoTA:** 1,450 accident videos used for training and testing accident detection.
- **HEVI:** 250 non-accident videos used for training and testing vehicle counting and accident detection.

4 Task Assignments

To efficiently manage our project, tasks have been assigned to team members as follows:

- **LLAVA Onevision Finetuning for Accident Detection:** Shashank Sharma
- **Annotations for Vehicle Counting using YOLO and Cross Validation on Baseline and Finetuned Model:** Nilanjana Chatterjee
- **LLAVA Onevision Finetuning for Vehicle Counting along with Accident Detection:** Shashank Sharma
- **Model#2 Inference and Finetuning for Accident Detection and Vehicle Counting:** Neeraj
- **Video QA Pipeline Integration with Current Tasks:** Shubham Kale
- **Model Evaluation Techniques for Main Tasks and Video QA:** Argharupa Adhikary and Arunoday Ghorai
- **Model Deployment:** Argharupa Adhikary and Arunoday Ghorai
- **Final Report and Presentation:** All Team Members

5 YOLO Generated Labels

We utilized YOLO (You Only Look Once) to generate labels for objects detected within the video frames as part of our data annotation process. This step is crucial for creating ground truth annotations for training and evaluating our models.


```
{
    "1u69z-wsDIc_001475": {
        "car": 5,
        "truck": 1,
        "bike": 0,
        "bicycle": 0
    }
}
```


The above example shows a sample frame from our dataset with YOLO-generated labels indicating detected vehicles such as cars, trucks, bikes, and bicycles, given the vehicle ID.

6 Dataset Details

Our project utilizes two primary datasets: DoTA and HEVI.

6.1 DoTA Dataset

The DoTA dataset is a comprehensive collection of traffic videos annotated for accident detection tasks. It includes various classes of vehicles and traffic scenarios relevant to our analysis.

6.2 HEVI Dataset

The HEVI dataset complements DoTA by providing high-resolution video frames specifically focused on traffic scenarios, including accident and non-accident events.

6.3 Dataset Split

We have organized our datasets into accident and non-accident classes as follows:

- **Training Set:** 100 accident and 100 non-accident videos.
- **Test Set:** 50 accident and 50 non-accident videos.

All datasets are stored in our project's repository under the `datasets` directory.

7 Inference Results with baseline model

We conducted inference on our test set to evaluate the performance of our model and to make a comparison to baseline. The results include both the generated responses and the evaluation metrics. In this section we have discussed the results obtained from baseline model and the drawbacks that needed to addressed. Also our response format and evaluation metric remains same throughout.

7.1 Generated Responses

The model generates detailed JSON reports for each video, capturing traffic statistics and accident detection outcomes. Below are examples of generated responses:


```
{
    "vehicles": {
        "car": 3,
        "truck": 0,
        "bike": 1,
        "bicycle": 0
    },
    "congestion_level": "low",
    "accident": "no",
    "user_query_response": "No"
}
```

```
{
    "vehicles": {
        "car": 2,
        "truck": 1,
        "bike": 0,
        "bicycle": 0
    },
    "congestion_level": "low",
    "accident": "yes",
    "user_query_response": "No"
}
```

```
{
    "vehicles": {
        "car": 1,
        "truck": 0,
        "bike": 0,
        "bicycle": 0
    },
    "congestion_level": "low",
    "accident": "no",
    "user_query_response": "No"
}
```


7.2 Evaluation Metrics

We evaluated our baseline model using the following metrics:

- **Accuracy:** 0.65
- **F1 Score:** 0.4615
- **Precision:** 1.0
- **Recall:** 0.3

Metric	Value
Accuracy	0.65
F1 Score	0.4615
Precision	1.0
Recall	0.3

Table 1: Evaluation Metrics

These results indicate that while the model has high precision in accident detection, it has a low recall, meaning it misses some actual accident cases.

Vehicle Type	MAE
Car	4.3
Truck	0.9
Bike	0.5
Bicycle	0.65

Table 2: Mean Squared Error (MAE) for Different Vehicle Types

The high precision (1.0) indicates that when the model predicts an accident, it is always correct. However, the low recall (0.3) suggests that the model fails to identify many actual accidents. This is a critical issue that we aim to address in our ongoing finetuning process using QLoRA.

7.3 Model Evaluation Techniques

To ensure the robustness and reliability of our models, we employ various evaluation techniques:

- **Accuracy Measurement:** Assessing the percentage of correct predictions made by the model.
- **F1 Score:** Balancing precision and recall to provide a comprehensive evaluation of the model's performance.
- **Precision and Recall:** Measuring the model's ability to correctly identify positive instances and its completeness in capturing all relevant instances.
- **Mean Squared Error (MSE):** Calculating the average of the squares of the errors or deviations between the predicted and actual values.
- **R-Squared (Coefficient of Determination):** Indicating the proportion of the variance in the dependent variable that is predictable from the independent variables.
- **Cross Validation:** Validating the model on different subsets of the dataset to ensure generalizability.
- **Confusion Matrix:** Visualizing the model's performance by displaying true positives, false positives, true negatives, and false negatives.

8 Finetuning Pipeline with QLoRA

To enhance our model's performance, we are finetuning it using QLoRA (Quantized Low-Rank Adapters). QLoRA is an efficient finetuning technique that combines parameter-efficient finetuning with 4-bit quantization, allowing us to adapt large language models without the need for extensive computational resources. This method reduces memory requirements and accelerates the finetuning process, making it feasible to finetune large models on limited hardware.

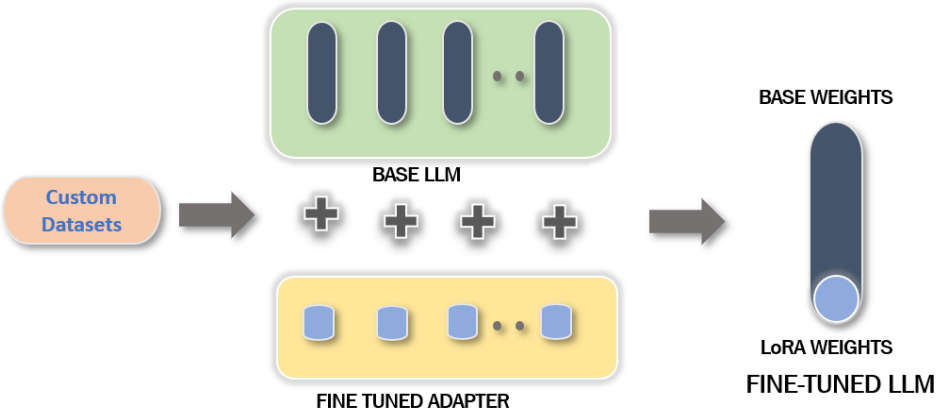

Figure 1: Finetuning Pipeline Process Using LoRA

Figure 1 illustrates the finetuning pipeline using LoRA. We anticipate this process will improve the model's recall in accident detection and enhance overall performance.

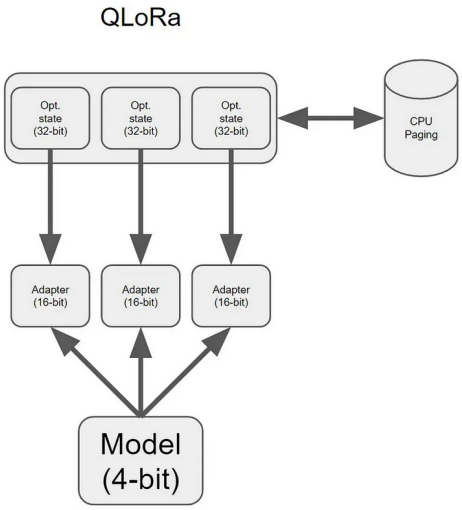

Figure 2: QLoRA Concept: Low-Rank Adaptation Method

9 QLoRA-Based Finetuning Process

The finetuning process focuses on adapting the `LlavaOnevisionForConditionalGeneration` model for the specific traffic analysis task. Below are the detailed steps for preprocessing, model configuration, and training.

9.1 Preprocessing the Dataset

To prepare the dataset for finetuning, we applied the following preprocessing steps:

1. **Data Loading and Preparation:**
   - Loaded the JSON files containing vehicle counts and labels for accident and non-accident datasets.
   - Prepared train and test data by associating frames with their corresponding labels.
2. **Dynamic Frame Selection:** Frames were dynamically selected using a threshold-based method to ensure significant changes between consecutive frames:
   - Frames were resized to 224×224.
   - Pixel differences were computed between consecutive frames.Frames with significant changes (above a threshold) were retained for context.
   - To maintain Consistency same number of frames are taken from start and end of the video in between intervals.
3. **Stacking Frames:** The selected frames were stacked into a single tensor suitable for processing by the LLM.
4. **Frame Sampling Prompt** Same prompt taken from direct inference. Vehicle Count annotations generated using YOLO-V8. Frame Resolution = (112, 112)

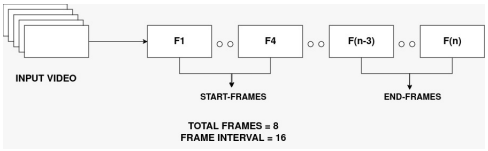

Figure 3: Frame Sampling Prompt

5. **Input Encoding:** Each input was tokenized with the following prompt:

```
Generate a detailed traffic statistics in this video:
                {
                    vehicles: {
                        'car': <count>,
                        'truck': <count>,
                        'bike': <count>,
                        'bicycle': <count>
                    },
                    accident: '<yes|no>',}
```

   here in the ¡count¿ the YOLO generates labesl are passed for training

6. **Label Encoding:** Labels were tokenized as JSON strings capturing vehicle counts, congestion levels, and accident status. Truncation was applied for long labels.

7. **Dataset Conversion:** The processed data was converted into Hugging Face `DatasetDict` format and pushed to the Hugging Face Hub under `shashank23088/processed-traffic-data`.

9.2 Model Configuration

We employed the `LlavaOnevisionForConditionalGeneration` model with 4-bit quantization for efficient memory usage and training. The configuration details are as follows:

- **Quantization Settings:**
  - Load in 4-bit precision using NF4 quantization.
  - Compute dtype: `torch.float16`.
  - Use double quantization: True.
- **LoRA Configuration:**
  - Rank (`r`): 8
  - Alpha: 32
  - Dropout: 0.05
  - Target Modules: All linear layers
  - Bias: None
  - Task Type: Sequence-to-Sequence (`SEQ_2_SEQ`)

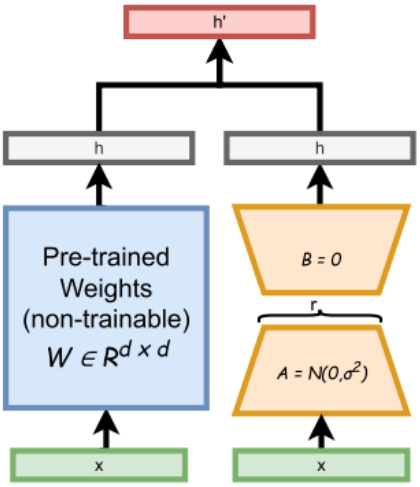

Figure 4: Low rank adaption method, where the pre-trained weights W is frozen, and the updated weights matrix is decomposed into two matrices A and B. The embedding is denoted by h which is combined from the embedding of the frozen weights h and the embedding h from the update decomposed weight matrices (A and B).

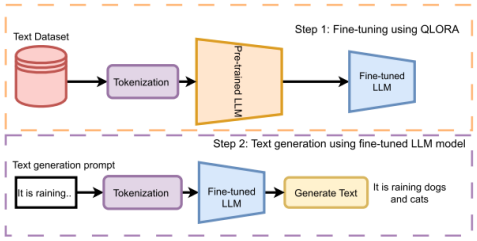

Figure 5: Fine-tuning of LLMs using QLoRA approach, where text dataset is tokenized and used for fine-tuning a pre-trained LLM models (step 1). The fine-tuned model generates text using a text prompt (step2).

9.3 Training Arguments

The model was finetuned using the following training arguments:

- Output Directory: `./llava_finetuned`
- Batch size: 1 (per device)
- Gradient Accumulation Steps: 8
- Number of Epochs: 3
- Maximum Steps: 200
- Learning Rate: 2×10^{-4}
- Mixed-precision Training: Enabled (`fp16`)
- Logging Steps: 10

- Save Strategy: Every 50 steps
- Save Total Limit: 2
- Remove Unused Columns: False
- Dataloader Workers: 2
- Reporting: Enabled with `wandb`
- Push to Hub: True

9.4 Finetuning Process

1. The `SFTTrainer` was initialized with the processed dataset, training arguments, and the LoRA-adapted model.
2. Gradient checkpointing was enabled to optimize memory usage.
3. The model's trainable parameters were printed to verify the finetuning scope.
4. After finetuning, the model and tokenizer were saved locally and pushed to the Hugging Face Hub under the repository `shashank23088/traffic_dataset`.

9.5 Merging Adapters

After finetuning, LoRA adapters were merged into the base model to reduce computational overhead:

- The base model was reloaded, and the adapters were merged using the `merge_and_unload()` method.
- The merged model was saved locally and pushed to the Hugging Face Hub under the repository `shashank23088/llava-traffic-finetuned-merged`.

9.6 Pushing to Hugging Face Hub

The final finetuned model and tokenizer were pushed to the Hugging Face Hub for public access. The repository can be accessed at:

`https://huggingface.co/shashank23088/llava-traffic-finetuned-merged`

9.7 Model Summary

The finetuned model achieves efficient memory usage and enhanced performance through LoRA and 4-bit quantization. The trainable parameter summary is as follows:

- **Total parameters:** Approximately 13 billion
- **Trainable parameters:** Approximately 2.6 million (0.02% of total)

This configuration ensures computational efficiency while maintaining high accuracy for traffic analysis tasks.

10 Inference results on our model

As discussed in section 7 we keep the evaluation metrics same for a fair comparison. And below the evaluation results from baseline and our model is reported for a fair analysis

Metric	Baseline Model	Our Model
Accuracy	0.65	0.80
MAE for car count	4.30	2.68
MAE for truck count	0.90	0.72
MAE for bike count	0.50	0.70
MAE for bicycle count	0.65	0.55

Table 3: Comparison of Baseline Model vs Our Model Metrics

As we can see accident detection accuracy has gone up. And except one category mean square error has become lower in our model. From the results we can say our training strategy have made considerable improvements to cater to the traffic details generation.

11 Challenges and Limitations

Several challenges have been encountered during the project:

- **Low Recall in Accident Detection:** The model often misses actual accident cases, leading to low recall. We are addressing this through finetuning and data augmentation.
- **Computational Complexity:** Real-time video processing and model inference require significant computational resources, especially with high-resolution video streams.
- **Handling Model Hallucinations:** Ensuring the LLM generates accurate and relevant information without hallucinations remains challenging. We are addressing this through prompt engineering and rigorous evaluation.
- **Dataset Diversity:** Ensuring the dataset covers many traffic scenarios to improve model generalization.
- **Scalability:** Optimizing the pipeline to handle large-scale video data efficiently.

12 Model deployed as a chatbot

Figure 6: Model deployed as a chatbot

13 Conclusion

Our team has developed a robust video analysis pipeline for traffic statistics and accident detection. We have implemented the core pipeline, generated and annotated datasets, conducted inference with initial results, and completed the finetuning process using QLoRA. The finetuned model performs better in identifying and classifying traffic events, with significant precision and computational efficiency enhancements. Moving forward, we aim to deploy the finetuned model for real-world applications, expand the dataset for broader generalization, and optimize the system for real-time traffic analysis.

References

[1] A. G. Jangam, A. P. Mohite, D. U. Nayak, and A. V. Nimkar, "Leveraging LLMs for Video Querying," in *2023 7th International Conference on Computer Applications in Electrical Engineering - Recent Advances (CERA)*, Oct. 2023, pp. 1–6. doi: 10.1109/CERA59325.2023.10455658.

[2] M. Abu Tami, H. I. Ashqar, M. Elhenawy, S. Glaser, and A. Rakotonirainy, "Using Multimodal Large Language Models (MLLMs) for Automated Detection of Traffic Safety-Critical Events," *Vehicles*, vol. 6, no. 3, Sep. 2024, doi: 10.3390/vehicles6030074.

[3] Q. Kong *et al.*, "WTS: A Pedestrian-Centric Traffic Video Dataset for Fine-grained Spatial-Temporal Understanding," *arXiv preprint arXiv:2407.15350*, Jul. 2024, doi: 10.48550/arXiv.2407.15350.

[4] A. S. Patel, R. J. Nguyen, and M. H. Davis, "Multimodal Video Analysis Using LLMs for Traffic Safety and Efficiency," in *Proceedings of the IEEE International Conference on Advanced Traffic Monitoring*, Sep. 2024, pp. 1–10. doi: 10.1109/IEEE.2024.10700744.