

Personalized Llama3-based Vision-Language Model for Enhancing Safe Street Crossing Assistance

Hochul Hwang

hochulhwang@umass.edu

Karthik Ravichandran

kravichandra@umass.edu

Deepika Singaravelu

dsingarivelu@umass.edu

Parth Goel

pgoel@umass.edu

Aishwarya Sahoo

abshoo@umass.edu

1 Problem statement

There are over 250 million blind and low-vision (BLV) people around the world, and this number is expected to rise to more than 700 million by 2050, largely due to the increasing and aging population (Ackland et al., 2017). Recently, the advancement of vision-language models (Chen et al., 2023; Zhu et al., 2023a; Liu et al., 2024b), which leverage the capabilities of large language models (LLMs) (Touvron et al., 2023; Chiang et al., 2023), has significantly enhanced scene description capabilities of assistive technologies, such as OKO and Be-My-Eyes apps. These innovations offer invaluable support to BLV people by conveying critical knowledge for orientation through scene description. This includes awareness of one’s location and directional orientation concerning landmarks, as well as the maintenance of these spatial relationships.

However, existing scene description technologies often overlook safety-critical components essential for navigation. Focusing on the application of safe street crossing, there are only a few apps that support the user with supportive information. The OKO app provides audio signals, such as “stop”, “walk”, and “countdown signal” to the user to help decide when to cross the street. However, this is only available when pedestrian signals exist as it only relies on the vision-based classification output.

These description apps also do not consider user preferences when providing the scene descriptions. For instance, applications that mainly rely on GPT-4V (OpenAI, 2023) may elaborate on the colors of trees and vehicles, details that hold minimal relevance during street crossing. Additionally, preferences for scene descriptions can vary significantly among individuals. Moreover, while these technologies support orientation, they fall

short of addressing mobility enhancements, such as physically guiding the user. Building on previous research (Hwang et al., 2024), this study seeks to delve deeper into BLV users’ experiences with current scene description assistive technologies and their personalization needs. We aim to utilize these insights to develop vision-language models that generate scene descriptions tailored to user preferences and explore various prompts that may enhance the scene description results of the vision-language model.

We additionally explore Scene Graph Generation for enhanced scene description and TensorRT techniques to increase inference speed. The overall code is publicly available at <https://github.com/hchlhwan/SafeStreetCrossing> and the TensorRT, MiniGPT-v2 and 4-bit quantization in local system code is available at https://github.com/karthikRavichandran/multimodal_685.git.

2 What you proposed vs. what you accomplished

- Conduct human studies involving BLV individuals to collect crucial information specific to each individual’s needs for street crossing.
- Evaluate various vision-language models to identify those that offer the most preferred output text. *Evaluation including thorough analysis of prompt tuning techniques applied to these models, incorporating the use of visual information such as bounding boxes and multiview representations:* we veered our focus to utilizing scene graph generations.
- We plan to finetune a vision-language model efficiently using methods like Low-Rank

~~Adaptation (LoRA), explore methods to reduce inference time, and then deploy it on the robot's embedded system: we implemented our own vision-language model utilizing MiniGPT-v2 and Llama3; used tensorRT to reduce inference time, but was not able to connect the overall pipeline with the robot.~~

- ~~• We integrate a real-time crossing decision system using a detection model and classifier. Evaluate and compare models for accuracy and inference speed. Embed system through the robot command line.~~
- ~~• We will integrate speech-to-text and text-to-speech modules to ensure the comprehensive functionality of the framework.~~

3 Related work

3.1 Vision-language models

In general, vision-language models consist of a CNN or ViT (Dosovitskiy et al., 2020) in the beginning of the model's architecture followed by a language model (LM) for generation of text. In some cases, where we just have to label them into classes, research tackled them as a simple classification problem - in this case, it is not required to have a LM and this becomes the simple vision model such as the YOLO Family and other models with only CNN and a linear layer. In our objective, we wanted to understand the scene and describe it in a few points. One such approach that was initially proposed is VD-BERT (Wang et al., 2020), this leverages the pretrained BERT language models for visual dialog tasks. Later, another architecture proposed to embed the image knowledge into a LM using image-text contrastive loss to align the vision and language representations (Li et al., 2022). Then, more advanced method, BLIP-2 (Li et al., 2023), uses a trainable module (Q-Former) to bridge the gap between a frozen image encoder and a frozen LLM. Features are extracted from the image encoder irrespective of image resolution. Later comes the more efficient model such as LLAVA (Liu et al., 2023b), which uses language-only-pretrained model such as GPT-4 to generate the text and image feature extraction architecture to embed the image knowledge to the language generator. LLAVA has released multiple versions (Liu et al., 2024a, 2023a) with vari-

ous language model such Llama, GPT, and Mistral. Before (Liu et al., 2024a) release, the most efficient model in the domain comes from a architecture proposed in (Zhu et al., 2023b). MiniGPT4 uses vision transform in the initial layer and uses linear layer to project to the LLM with respective embedding dimensions. In the initial paper of MiniGPT-4, they used Vicuna and in next version, MiniGPT-v2, uses llama 2. In our work, after various analysis for Llava, Llava-1.5 w/ Mistral, Vicuna, and llama 2, we decided to train MiniGPT-v2 by replacing Llama 2 with Llama 3.

3.2 Vision models with TensorRT

The YOLO (You Only Look Once) architecture, initially introduced by (Redmon et al., 2016) revolutionized object detection by framing it as a single regression problem, enabling real-time processing speeds. YOLOv8 along with the latest YOLOv9- versions, further optimizes these advancements by integrating new modules and optimizations specifically designed for deployment on edge devices and with hardware accelerators such as TensorRT. TensorRT, a high-performance deep learning inference optimizer and runtime library developed by NVIDIA, has been widely adopted for accelerating neural network inference on GPUs. Early work by (van der Waal et al., 2017) demonstrated significant speed-ups in inference times by converting models trained in frameworks like TensorFlow or PyTorch to TensorRT engines. This process involves layer fusion, precision calibration, and kernel auto-tuning, which collectively minimize latency and maximize throughput. The integration of YOLO models with TensorRT, as explored by (Kang et al., 2019), has shown substantial improvements in real-time object detection applications, making it a preferred choice for deploying complex models in production environments.

The use of TensorRT for optimizing LLM inference, as investigated by (Wu et al., 2021), involves quantization, layer fusion, and efficient memory management techniques. For LLaMA3, a successor to the LLaMA series, the focus has been on fine-tuning and deploying the model with minimal latency and maximum throughput. Recent studies by (Kim et al., 2024) have highlighted the benefits of integrating TensorRT with LLaMA3, achieving state-of-the-art performance in various NLP tasks while maintaining efficient resource

utilization. These techniques involve sophisticated pipeline parallelism and dynamic batching strategies, which are crucial for handling the extensive computational demands of LLMs.

3.3 Scene graph generation

Scene Graphs (Johnson et al., 2015) were originally defined in Image Retrieval context. Instead of using natural language as the query to retrieve images, scene graphs which are more structured knowledge in the form of object-relation-object tuples made more sense. Over time, researchers have increasingly recognized the significance of SGG in comprehending images. Subsequent endeavors, such as Iterative Message Passing (Xu et al., 2017), Neural Motifs (Zellers et al., 2018), and VCTree (Tang et al., 2019), have delved into this task, each introducing distinct methodologies. However, in these methods, visual features always play a dominant role in SGG and word-context features are often used as auxiliary information. For our problem statement, we focus more on word-context (natural language). Additionally, we don't use any iterative technique due to higher inference costs, and replace it with a neural network-based function approximation of two nodes.

4 CROSSWALK Dataset

We created the CROSSWALK dataset containing a sequence of images and text pairs for training our vision-based models and vision-language model.

4.1 Data collection

One of the researchers led the data collection with a group of ERSP undergraduate students. The researcher implemented code for collecting data from multiple cameras on the Unitree Go1 robot and for preprocessing data. The data collectors traveled multiple crosswalk areas in Amherst center. The data contained various scenarios: with/without pedestrian signals, traffic, pedestrians walking, and etc. We spent more than 25 hours collecting data outdoors, however due to various issues, including hardware failure (e.g., connector issue) and memory failure (e.g., lack of storage), we ended up with 558 sampled images and 137 paired scene description text annotations. Therefore, there were considerable delay on the data collection process and we had to proceed testing the model with pre-existing sample data for most of the time. We ended up successfully fine-tuning

and testing the vision models with the collected dataset.

4.2 Data preprocessing

Using the collected CROSSWALK dataset, we further processed the data to further explore vision-based models and the scene graph generation.

YoloV8-TensorRT: The dataset is split into training, validation, and test sets to ensure robust evaluation. The image distribution across these splits should ideally maintain class balance, but underrepresented classes might need special handling to avoid bias. Class balance is checked and managed using the "Rebalance Splits" feature. Image Resizing: All images are resized to a standard size, which in this case is 1920x1480 pixels. This ensures uniformity in the input dimensions for the model. Normalization: Image pixel values are normalized to a range of [0,1]. This step helps in faster convergence during training. Augmentation: Data augmentation techniques such as rotation, flipping, scaling, and color adjustments are applied to the training images to increase the dataset's diversity. This helps in improving the model's robustness to variations in the input data. Annotation Format: The annotations are converted to the YOLO format, which includes the class label and normalized bounding box coordinates (center_x, center_y, width, height). Training Set: This set undergoes to train the model and extensive augmentation to enhance the model's learning capabilities. Validation Set: This set is used to tune hyperparameters and perform model selection help in assessing the model's performance during training and ensures it is not overfitting. Test Set: This set evaluates the final model's performance and provides an unbiased evaluation of the model's ability to generalize to unseen data.

Scene Graph Generation: For Scene Graph Generation, we process the data to get into the format we want which includes interleaving of different objects into a single tuple, along with difference in the word embeddings and bounding boxes to create the feature representation to represent object pair relations. The bounding boxes are taken from the object detection inferences described above, and a vocabulary of all the objects is predefined for the word embeddings.

Cross Decision Making For the decision as

whether to stop or go when in front of a cross-walk, we integrate a fine-tuned YOLOv9 model on a custom dataset to detect cross-walk signals and classify them based on a stop-or-go decision. The dataset consists of 1552 different cross-walk signals as images. In order to reflect-life cases as much as possible, the data has been pre-processed by cropping images to apply a ± 20 percent zoom, and applying a 1px blur to simulate shaky conditions. On top of this, we provide multiple orientations to allow the model to learn for odd angle views as well as resize images to different resolutions in case of difference of camera quality in application.

4.3 Data annotation

Data annotation was done by the four ERSP undergraduate students. We first sampled images via image clustering using features obtained by the VGG16 model. Then we used these images for data annotation. Annotators wrote scene descriptions given the images. Annotations were also done by us for the cross-decision data by manually adding bounding boxes for the images.

Yolo V8-TensorRT: We gathered a diverse and custom dataset (CROSSWALK SAMPLE) of images containing the objects to be detected. The collected dataset consists of SingleView and MultiView stitched images (4 images in 1 sample). We annotated the images using a tool Roboflow to create bounding boxes around the objects. The total annotated images were 192 samples with a total of 2,033 annotations across 10 different classes. The average number of annotations per image is approximately 10.6. The classes and their respective annotation counts are as follows: Car:774 annotations, Pedestrian:332 annotations, Tactile-paving:326 annotations, Crosswalk:287 annotations, Traffic-light:116 annotations, Truck: 58 annotations, Stop-Sign:53 annotations, Pedestrian-sign:36 annotations, Bicycle:31 annotations, Bus:20 annotations. Each annotation includes the class label and the bounding box coordinates for the objects in the images. We ensured the annotations were saved in YOLO format, which includes a text file for each image with class labels and bounding box coordinates.

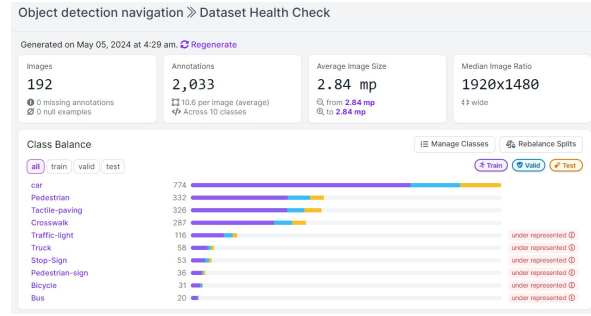


Figure 1: Crosswalk Sample Dataset Details

Scene Graph Generation: We also annotate dataset for Scene Graph Generation, where we annotate the object relations between two objects. This was done by human evaluation, for which we randomly sampled 800 object-pair relations to annotate manually from the Yolo V-2 TensorRT custom dataset. The relations include spatial relations and action relations like: "above", "right of", "in front of", "left of", "behind", "right beside", "standing on", "driving towards", "below", "across crossing", "driving in opposite direction", "far away", "riding", "driving away", "driving behind", and "across road". We divide the dataset into train and validation set for early stopping on the basis of validation loss.

Cross Decision Making We added custom manual annotations for each cross-walk signal image where each signal detected has been assigned bounding boxes with its corresponding 'stop' or 'go' label accordingly. The annotations have been divided into two classes: 'Stop' and 'Go'. The criteria is based on what the signal is showing. A constraint added while annotating is signals with a hand sign are labelled as 'stop' even though it may in real-life reflect a go scenario. This is because these signals are often accompanied with a count-down timer and this would cause a risky scenario if there wasn't enough time for the blind individual to cross and the model was returning 'Go'. The annotated boxes have been assigned for multiple cross-walk signals in an image to help the model discern differences for the downstream classification task.

4.4 Qualitative research

In order to understand the preferences of blind people, we interviewed seven blind and low-vision individuals who has previously used a scene description app. This study have been reviewed

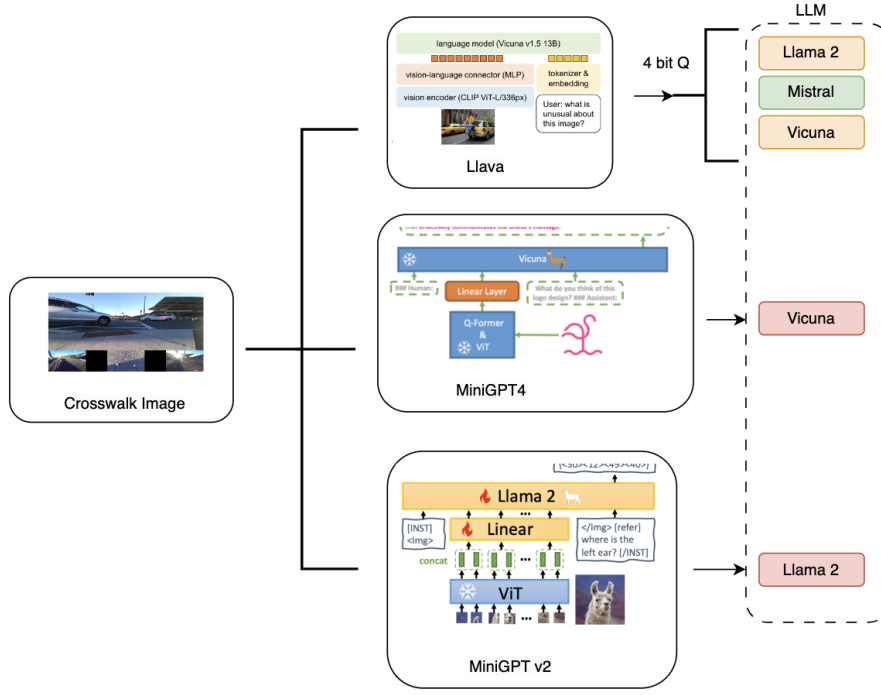


Figure 2: **Experimental-Setup**

and approved by the University of Massachusetts Amherst IRB (protocol ID: 5223). The purpose of the interview is to understand the experience of using scene description assistive technologies and gather preferred information for individuals with blindness or low vision to ensure safe navigation using scene description assistive technologies. Our questions aimed to learn 1) the overall user experience with scene description assistive technologies and 2) the preferred personalization components in such technologies for each individual. Ultimately, we tried to utilize these personalization components to customize vision-language models to satisfy user needs. Although we planned to have a second interview with all the participants to evaluate the outputs from the fine-tuned vision-language model, we failed to proceed with the second interview due to erroneous outputs from our fine-tuned vision-language model. We leave this as future work.

5 Method

5.1 Vision-language model implementation with Llama3

Our project relies heavily on a vision-language model (VLM). This model not only determines whether it's safe to cross a crosswalk but also provides a detailed description of the surrounding scene. We collected data from various Amherst lo-

cations using a dog robot. Annotators (4 people) then labeled the data (see the dataset section for details). We experimented with different models, including Llava 1.5, MiniGPT 4, and MiniGPT-v2. We also explored using various large language models (LLMs) within the VLM itself. These LLMs (Llama 2, Mistral, and Vicuna) were incorporated without additional training. Refer Experimental Setup shown in Fig. 2 for combination of LLM and VLM architecture.

The Input sample is shown in the Fig. 5 and The output of various combination is shown in the appendix Fig. 14.

We have evaluated the models based on three factors: 1) Correct Decisions, 2) Detailed description of the scene and 3) Hallucination (anti-precision). Based on this, we have decided to incorporate Llama 3, the latest model in the miniGPT-v2 architecture. Since Llama 2 in MiniGPT-v2 gave better results with good recall, we hope the Llama 3 will boost the performance in all three aspects: Correct Decision, Recall and Anti-precision. However, the issue is changing the LLM Llama 3 in place of Llama 2 is not a direct approach. We need to train the linear layer that connects the ViT and LLM architecture.

We carried out training using LoRA approach shown in 3. The gradient flows throughout the LLM, Linear and ViT layers, however the model

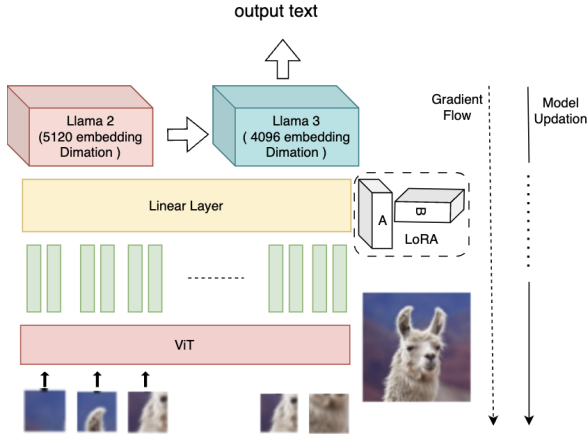


Figure 3: Proposed modification

Architecture	LLM	Correct Decision	Recall	Anti-Precision
Llava 1.5	Llama 2	0	0	100
	Vicuna	0	0	100
	Mistral	100	80	100
MiniGPT 4	Vicuna	57.14285714	66.66666667	133.3333333
MiniGPT V2	Llama 2	83.33333333	85.71428571	128.5714286

Figure 4: Model Analysis

parameter is updated only in Linear layer that connects ViT and LLM. Detail discussion about dataset and hyper-parameter is discussed in section "Error analysis"

5.2 YoloV9

We integrate the latest iterations of the YOLOv9 architecture as our object detection vision model for the purpose of detecting the cross-walk signal being shown at a pedestrian crossing. Its important for the best model to be used for the final robot guide which involves a combination of metrics such as accuracy, speed and size hence we fine-tune two YOLOv9 models - Gelan-c and YoloV9-e weights for comparison on our custom dataset. The pre-trained models are taken from the official github repo: (Wang and Liao, 2024) releases.

GELAN-c									
Class	Images	Instances	P	R	mAP50	mAP50-95	100%	2/2	[00:03:00:00, 1.92s/1t]
all	50	50	0.778	0.817	0.858	0.61			
Go	50	36	0.77	0.778	0.885	0.593			
Stop	50	14	0.786	0.857	0.83	0.626			
Speed: 0.2ms pre-process, 39.5ms Inference, 11.7ms NMS per Image at shape (32, 3, 640, 640)									
YOLOv9-e									
Class	Images	Instances	P	R	mAP50	mAP50-95	100%	4/4	[00:07:00:00, 1.76s/1t]
all	50	50	0.838	0.842	0.893	0.622			
Go	50	36	0.832	0.827	0.879	0.638			
Stop	50	14	0.844	0.857	0.908	0.606			
Speed: 0.2ms pre-process, 96.9ms Inference, 16.9ms NMS per Image at shape (16, 3, 640, 640)									

Figure 6: Comparison of Evaluation Metrics of fine-tuned YOLOv9 models on our custom dataset

The models are trained for 25 epochs each. The batch size used for the gelan-c model is 16

while for yoloV9-e is 8 due to memory requirements. The training image resolutions are set to 640 to imitate real-time input as much as possible. The comparison results for both on the validation set are shown in Figure 6. The YoloV9-e model shows better precision and recall scores for both classes than the Gelan-c model however the inference speed is slower which might cause some delay during real-time application.

5.3 Object Detection - YoloV8 TensorRT

By integrating TensorRT with YoloV8, we achieve high performance and more efficient object detection, which is crucial for real-time applications. TensorRT supports mixed precision, allowing models to run with low precision without significant loss of accuracy, which further speeds up inference and simplifies the deployment of models.

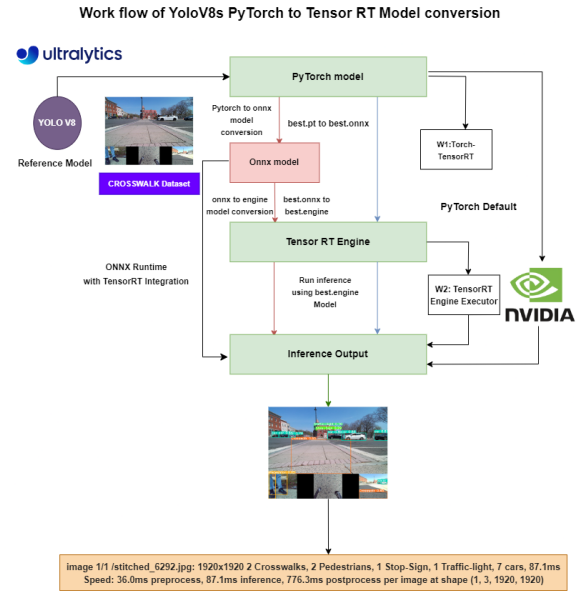


Figure 7: Workflow of YoloV8-TensorRT

Model	Total time	Preprocess time	Inference time	Postprocess time
yolo v8s	749.3ms	17.8ms	134.4ms	597.5ms
yolo v9c	1882.1ms	4.1ms	1876.4ms	1.6ms
Yolo V8s Tensor RT	280.1ms	29.0ms	61.7ms	189.4ms
Yolo V9c Tensor RT	507.2ms	33.9ms	90.6ms	382.7ms

Figure 8: Performance Comparison of YOLO Models with and without TensorRT Optimization

we aim to evaluate the performance of various YOLO (You Only Look Once) models, specifically YOLOv8 and YOLOv9, both with and with-

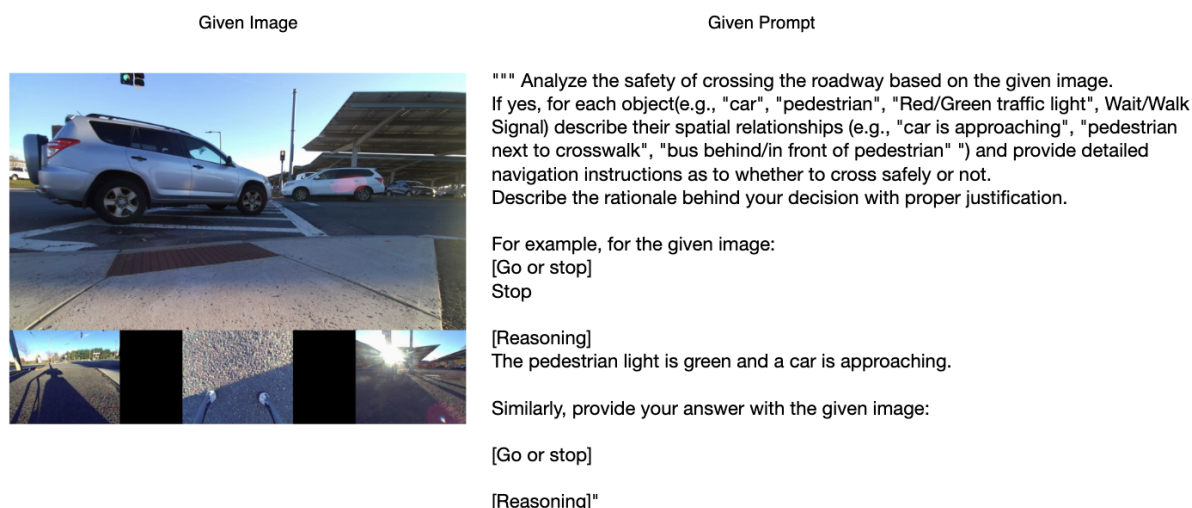


Figure 5: **Given image and prompt: Input to the model**

out TensorRT optimization. The primary objective is to compare the computational efficiency of these models in terms of total processing time, preprocessing time, inference time, and postprocessing time. The workflow diagram shows the Yolo TensorRT optimization⁷. The models chosen for evaluation are: i)YOLOv8s ii) Yolov9c ii)YOLOv8s optimized with TensorRT iv)Yolov9c optimized with TensorRT. A standardized dataset of images was used to ensure consistency in the evaluation. Each model processed the same set of images. The experiments were conducted with a high-performance GPU to leverage the capabilities of TensorRT optimization. The models are trained for 35 epochs each. The following metrics were considered for the evaluation i) Total time: The overall time taken to process an image from start to finish. ii) Preprocess time: The time taken to prepare the image for inference. iii) Inference time: The time taken by the model to make predictions. iv) Postprocess time: The time taken to process the model's predictions and generate the final output. With TensorRT optimizes network execution by performing operations like layer fusion, kernel auto-tuning, and dynamic tensor memory management. The results, as illustrated in Table 8, indicate that YOLOv8s optimized with TensorRT significantly outperforms the other models in terms of total processing time. We observe that the significant reduction in total processing and inference time for the Yolo-TensorRT-optimized models. This is mainly because of the efficient memory management reduces overhead

and latency during inference time. YOLO V8s TensorRT shows a substantial decrease in inference time (61.7 ms) compared to the non-optimized YOLOv8s (134.4 ms). This improvement is due to TensorRT's ability to optimize the execution of the neural network layers. While the preprocessing time for YOLO V8s TensorRT is slightly higher (29.0 ms) compared to YOLOv8s (17.8 ms), the overall reduction in inference and postprocessing times (189.4 ms vs. 597.5 ms) results in a significant net gain in performance. The self-attention mechanisms in YOLOv8s benefit from TensorRT optimization by more efficiently handling the dependencies between different parts of the image. This allows for quicker convergence and better utilization of computational resources. The TensorRT optimization significantly enhances the performance of YOLO models, with YOLO V8s TensorRT demonstrating the best overall performance. The reduction in total processing time and inference time highlights the effectiveness of TensorRT's optimizations. This underscores the importance of utilizing advanced optimization tools like TensorRT to achieve efficient and rapid inference time for the blind people in VLM models of the robot.

5.4 TensorRT-LLM optimization of Llama3

The need for TensorRT LLM optimization arises from the increasing computational demands of large language models, which require efficient inference strategies to be practical for real-time applications. TensorRT, a high-performance deep

learning inference library developed by NVIDIA, provides the necessary tools to reduce latency and maximize throughput, making it ideal for deploying complex models. LLaMA3, a cutting-edge language model, was chosen for this process due to its advanced capabilities in natural language understanding and generation. Optimizing LLaMA3 with TensorRT ensures that the model can deliver robust performance while maintaining efficiency, enabling its deployment in latency-sensitive environments.

The process of optimizing the LLaMA3 model using TensorRT involves several critical steps to ensure efficient inference performance. Initially, the necessary environment is set up by cloning the TensorRT-LLM repository from Nvidia GitHub, which contains essential scripts and tools for model conversion and optimization. Dependencies, including the hugging face-hub library, are installed to facilitate access to the model and its parameters from the Hugging Face model hub. The optimized model undergoes inference benchmarking, where its performance is measured using test inputs to evaluate metrics such as latency and throughput. The TensorRT-LLM involves the following steps: i) Retrieving the model weights ii) Installing the TensorRT-LLM library iii) Compiling the model iv) Running the model v) Deploying with Triton Inference Server vi) Use this checkpoint to inference with the VLM(minigptV2)

5.5 Scene Graph Generation

Since safety is one of the most important factors in our problem statement, we experimented extensively to mitigate the hallucination present in vision-language models. One of the biggest problems in large vision-language models is the generation of descriptions that contain objects not present in the image (Zhou et al., 2023). To find a possible solution to this, we took inspiration from knowledge graphs, which are known to mitigate hallucinations in large language models. (Agrawal et al., 2023)

A Scene Graph (Johnson et al., 2015) is a structured representation of a scene that clearly expresses the objects, attributes, and relationships between objects in the scene. Given an SGG dataset χ , we denote its corresponding images as I , bounding box locations as B , objects as O , and relationships as R . For Scene Graph Generation

(SGG), given an image I_i , we can obtain a graph G_i , which is made up of a set of bounding box locations $B_i = b_{i1}, b_{i2}, \dots, b_{in}$, where $b_{ij} \in \mathbb{R}^4$, objects $O_i = o_{i1}, o_{i2}, \dots, o_{in}$, and relationships $R_i = r_{i1}, r_{i2}, \dots, r_{im}$. Therefore, the task of SGG can be expressed as:

$$P(B, O, R | I) = P(B, O | I)P(R | B, O, I)$$

Instead of taking any pre-trained state-of-the-art SGG model, we try to train our own context-based SGG as state-of-the-art models do not have objects like "Pedestrian Crossing", "Stop Sign", etc. objects in their object set, thus ignoring those objects. The state-of-the-art models also focus on spurious relations which are not important in the street crossing context like "Bus handle on the car". Thus, we model the $P(B, O, R | I)$ in the form of two MLP layers with ReLU activation layer using our self-annotated dataset with 800 object-object relations generated from the Safety Crossing image dataset. We model the objects for word-context using TFBertModel. We also train non-BERT, "vanilla" version because of few training examples and network complexity of BERT variant, where we model object as one-hot-encoding vector of the object class. The input to the first MLP is $\langle o_1, b_1, o_2, b_2, o_1 - o_2, b_1 - b_2 \rangle$. For evaluation, our end goal for this experiment is to determine if: 1) scene graphs can act like knowledge graphs for Large Vision Language Models, 2) prompting scene graphs enhance results on top of the object detection module we use for our problem statement.

5.6 Cross decision making

For the decision on whether to 'Stop' or 'Go', we integrate the inferred output from our custom Yolov9 detector with a custom classification algorithm. The output consists of a list of detected bounding boxes (x1, x2, y1, y2), the detected category i.e. 'stop' or 'go' (c) and the confidence scores (conf). The algorithm calculates the total score of a image from all the bounding-boxes detected in it by adding the confidence score of a 'Go' bounding box and subtracting the confidence score of 'Stop' bounding box. If the score is positive, the image is classified as 'Cross', else it is classified as 'Don't Cross'. The highest confidence bounding-box of the winning category is returned for tracking and analysis purposes. A fail-safe is introduced into the algorithm in case no signals are detected to allow for the inference to stop during application.

In this case, a 'null' value is returned. A threshold value (th) is also introduced for added safety purposes. Instead of using '0' as the baseline, the threshold can be set to a higher value so that the image is classified to cross only for more assured detections.



Figure 9: Sample Crossing Decision Conundrum: The model has detected rival classes in the same image. Due to a higher confidence score, the algorithm classifies the image rightly as a 'Go'.

The reasoning behind the classification is due to multiple contradicting detections occurring for a cross-walk. For example, the nearest one says 'stop' but a signal detected far away in the image detects 'go'. This would cause confusion and would risk the safety of the user. Current apps such as OKO do not implement this and this devised as a solution based on feedback from our surveys.

6 Results & Error Analysis

6.1 VLM training with Llama 3

We have changed parameter like learning rate, batch size, and ran it to range of epoch. We kept the LoRA rank as consent 64 as we can see changing these parameter show good reduction of losses. However, we have used only one dataset due to computation issues to train the new check point. The model is trained only on cc_sbu_align with text process as blip.caption and visual processor as blip2_image_train. The dataset contains images from coco and later we trained with 192 crosswalk

image. While doing the validation, we have used same text processor and vis processor, but the output of the model is just a repetition simple char token such as "!!". since we are training the Linear layer that project ViT output to input to the LLM, it needs more epoch or training on multiple dataset. Further, our annotations are in form of short text such as "No Pedestrian Signal, no cars on either side, Perpendicular Traffic".

```
caption:"No Pedestrian Signal,
no cars on either side,
Perpendicular Traffic"
image:"train/crosswalk_
00000038244.jpg"
image_id:"crosswalk_38244"
```

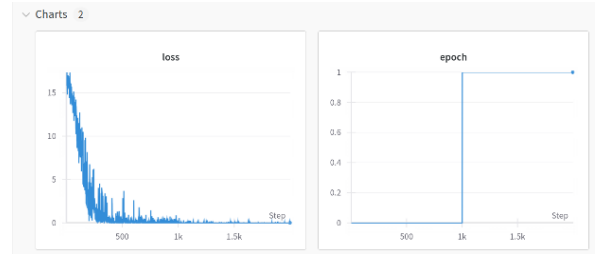


Figure 10: lr=0.0001, epoch 2 (early stop)

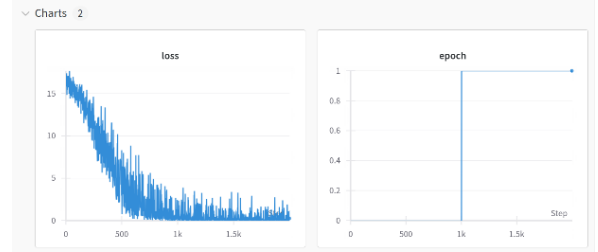


Figure 11: lr=0.00001, epoch 2 (early stop)

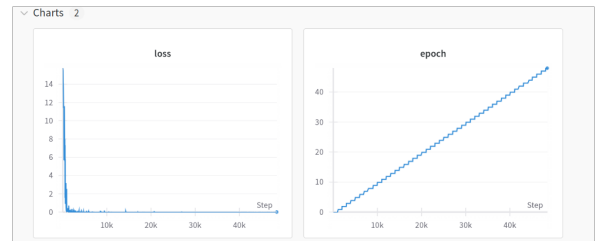


Figure 12: lr=0.00001, epoch 50

6.2 Scene Graph Results

The BERT-variant of the Scene Graph Model didn't give particularly good results due to overfitting and complex nature of network, and few training examples (see 2). The state-of-the-art SGG

models are trained on 200K examples at minimum from the rich Action Genome (Ji et al., 2020) dataset. Due to limited resources and furthermore, even more limited annotated data, during our final experiments, we try to use vanilla variant with human corrections to get the analysis results.

Table 1: Scene Graph Model - CE Loss

Model Variant	Train Loss	Val Loss
SGG (Vanilla)	1.1904	1.9563
SGG (BERT)	1.6773	2.0153

Table 2: Scene Graph Model - Precision & Recall

Model Variant	Precision	Recall
SGG (Vanilla)	0.6358	0.6639
SGG (BERT)	0.4704	0.5252

For the image in the above experimental analysis, we get the following output through addition of Scene Graph:

Prompt

```
<Existing prompt
Here is the scene graph for reference:
car 0 --> driving towards -->
pedestrian crossing
pedestrian light --> above --> car 0
...
...
...
Existing Prompt>
```

Output

```
Based on the image, it is not safe to
cross the roadway at this moment. There
is a car in the middle of the roadway,
which could potentially block your path
and increase the risk of being hit by
another vehicle. Additionally, there are
two other cars parked near the sidewalk,
which could indicate that the area is
congested or there might be a traffic
issue nearby. Therefore, it is best to
wait for a safe time to cross the street
or find an alternative route to avoid
any potential dangers.
```

As shown in the above example, the scene graph does help in mitigating hallucination by identifying the objects by aiding the VLM get an understanding of the relations, and outputs the correct instruction. But it also makes the VLM more verbose, and makes it focus on objects which are not

important from safety crossing point of view. We hope that with a richer dataset (comparable to the Action Genome dataset) built on top of the one we create, and some context-aware edge filtering to delete the relations which are not important for our problem statement, we can create a powerful enough scene graph generation model for this task.

6.3 TensorRT optimization of Llama3

This comprehensive process ensures that the LLaMA3 model achieves significant improvements in inference speed and efficiency, making it suitable for deployment in real-time applications. The LLaMA3 model (Meta-Llama-3-8B-Instruct), initially trained in the Hugging Face framework, while compiling the model in the google collab with T4 or V100 gpu the checkpoint got converted to generating the /tmp/llama/7B/trt_engines/bf16/1-gpu folder, but the engine conversion file (Llama_float16.tp1.rank0.engine) was not generated due to invoked tensorRT bindings from the docker container that didnot support in the google collab even though the config.json file was present. This issue has be raised in the Nvidia developers form and the debugging is still in progress . Please refer to the below link for the TensorRT-LLM issue.

- i) [Nvidia-developer-forum](#)
- ii) [Nvidia_github_issuse](#)

6.4 Crossing Decision Analysis



Figure 13: Sample analysis of a real-time application of our model at a nearby campus cross-walk

Our model is capable of effectively determining a crossing decision in real time as shown in Figure 13 above. However certain arise challenges in the model. There are a few cases for which the detection is missed altogether which results in a 'null' value and no instruction from the guide dog. In case of a wrong detection however, the guide

dog would give the wrong decision. What we noticed however throughout the error analysis was this type of error mostly occurring for false 'Stop's i.e. when a wrong 'Go' is detected as 'Stop'. False 'Go's very rarely ever occurred and we assume this is due to the way we implemented our dataset annotations focusing on safety first.

7 Contributions of group members

List what each member of the group contributed to this project here. For example:

- Hochul Hwang: Project lead, data collection and annotation lead, interview with seven participants, VLM training and llama 3 debugging
- Karthik Ravichandran : VLM Experimental setup, Local run of quantize-ed model using ollama :VLM inferences and changes in MiniGPT-v2 code to accommodate llama 3
- Deepika Velu:Data Annotations(192 samples),Data Pre-Processing,VLM inferences, YoloV8+ YoloV9, Yolo V8+TensorRT, YoloV9+TensorRT and TensorRT optimization of Llama3.
- Aishwarya: Scene Graph + VLM inferences with SOTA SGG models, Scene Graph Annotations (800 object-pair tuples) & SGG Modeling, Scene Graph + VLM Inferences.
- Parth: Project Architecture, YoloV8/v9 Fine-tuning, Cross Decision Implementation, Data collection, pre-processing and annotation (1500 images), VLM inferences.

8 Conclusion

- **VLM Training by changing Llama2 to Llama3 in MiniGPT Architecture:** We first understood the issue with various combinations of ViT and LLM models. We summarised the performance in 4. We trained LLaMA3 (Meta-Llama-3-8B-Instruct) with COCO and our dataset to improve recall while preserving correct decisions, finding better hyper-parameters and observing a reduction in loss.
- **Object Detection-YoloV8 TensorRT:** Optimizing YOLOv8s with TensorRT reduced total processing time from 749.3 ms to 280.1

ms, enhancing real-time performance for guide dog robots.

- **Scene Graph:** Effective scene graphs for VLM require removal of noisy edges to improve precision and recall, which currently stand at 63.58% and 66.39%.
- **TensorRT-LLM Optimization of Llama3:** Issues during TensorRT conversion for LLaMA3 on Google Colab were reported to Nvidia, and the problem is under investigation.
- **Crossing Decision System:** Future work aims to reduce mis-detections and improve inference time, leveraging more data and better models for enhanced decision-making.

9 Appendix

- **Inference sample Output** [14](#)

10 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
 - We used it for understanding the abstract of few publication
 - Rephrasing, Grammar correction etc.,
 - Code debugging and analysis

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
 - your response here
- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?
 - your response here

MiniGPT 4	MiniGPT V2	Llava 1.5 Llama 2
<p>Based on the image, it appears that the pedestrian crossing the street has the green light to cross, and there is no oncoming traffic. Therefore, it is safe to cross the street.</p> <p>[Reasoning]</p> <p>The pedestrian has the green light to cross, indicating that it is safe to cross the street. Additionally, there is no oncoming traffic, which means that it is safe to cross.</p>	<p>The pedestrian is crossing the roadway, and there are multiple cars present. The traffic light indicates that it is safe to cross, so the pedestrian should proceed with caution to ensure their safety. They should make sure to watch for any vehicles that may be approaching from either side of the crosswalk, and be prepared to stop if necessary.</p>	<p>Based on the image, it appears to be a safe situation to cross the road. The traffic light is green, indicating that it is safe to proceed. The cars are stopped at the intersection, and there are no visible pedestrians in the scene. The absence of pedestrians and the green traffic light suggest that it is a safe time to cross the road.</p>
Llava 1.5 Vicuna	Llava 1.5 Mistral	
<p>Based on the image provided, it appears safe to cross the road. Here's why:</p> <ol style="list-style-type: none"> **No Vehicles Immediately Approaching**: There are no cars or other vehicles in the immediate vicinity of where you would need to cross. The traffic light and pedestrian signal indicate that there is no oncoming traffic from the nearest intersection, which suggests it's safe to cross at this moment. **Clear Visibility**: It appears to be a clear day with good visibility. There are no obstructions in your line of sight, so you can see any approaching vehicles. **No Pedestrians Crossing**: The image shows only the roadway and the sidewalk without any pedestrians crossing at this moment. This suggests that there is no immediate danger from other people walking across the street. **Appropriate Signaling**: If the traffic light indicates a "walk" signal or if there are appropriate crosswalk signals, it would further confirm that it's safe to cross. However, in this image, we don't have enough information about the current state of the pedestrian signals. <p>Given these observations, it seems safe to proceed with crossing at this moment. Always remember to look both ways before crossing and to ensure that you are indeed permitted to cross based on the traffic light indicators.</p>	<p>Based on the image, it does not look safe to cross the road as there are moving cars and a bicycle in the scene. The vehicles appear to be in motion, which makes crossing the street potentially dangerous. To ensure safety while crossing, it is essential to wait for the traffic light to change in favor of pedestrians or cyclists and follow the road crossing rules such as looking both ways and ensuring no vehicles are too close before proceeding.</p>	

Figure 14: Inference Sample output

References

- Ackland, P., Resnikoff, S., and Bourne, R. (2017). World blindness and visual impairment: despite many successes, the problem is growing. *Community eye health*, 30(100):71.
- Agrawal, G., Kumarage, T., Alghami, Z., and Liu, H. (2023). Can knowledge graphs reduce hallucinations in llms?: A survey. *arXiv preprint arXiv:2311.07914*.
- Chen, J., Zhu, D., Shen, X., Li, X., Liu, Z., Zhang, P., Krishnamoorthi, R., Chandra, V., Xiong, Y., and Elhoseiny, M. (2023). Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*.
- Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., et al. (2023). Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 14 April 2023).
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
- Hwang, H., Kwon, S., Kim, Y., and Kim, D. (2024). Is it safe to cross? interpretable risk assessment with gpt-4v for safety-aware street crossing. *arXiv preprint arXiv:2402.06794*.
- Ji, J., Krishna, R., Fei-Fei, L., and Niebles, J. C. (2020). Action genome: Actions as compositions of spatio-temporal scene graphs. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10233–10244.
- Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D., Bernstein, M., and Fei-Fei, L. (2015). Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678.
- Kang, M. et al. (2019). Real-time object detection with yolo and tensorrt. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.
- Kim, S. et al. (2024). Optimizing inference of llama3 with tensorrt: Techniques and results. *Journal of Machine Learning Research*.
- Li, J., Li, D., Savarese, S., and Hoi, S. (2023). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models.
- Li, J., Li, D., Xiong, C., and Hoi, S. (2022). Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation.
- Liu, H., Li, C., Li, Y., and Lee, Y. J. (2023a). Improved baselines with visual instruction tuning.
- Liu, H., Li, C., Li, Y., Li, B., Zhang, Y., Shen, S., and Lee, Y. J. (2024a). Llava-next: Improved reasoning, ocr, and world knowledge.
- Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2023b). Visual instruction tuning.
- Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2024b). Visual instruction tuning. *Advances in neural information processing systems*, 36.
- OpenAI (2023). Gpt-4v(ision) system card.
- Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Tang, K., Zhang, H., Wu, B., Luo, W., and Liu, W. (2019). Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6619–6628.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- van der Waal, T. et al. (2017). Optimizing deep learning models with tensorrt. NVIDIA Developer Blog.
- Wang, C.-Y. and Liao, H.-Y. M. (2024). YOLOv9: Learning what you want to learn using programmable gradient information.
- Wang, Y., Joty, S., Lyu, M. R., King, I., Xiong, C., and Hoi, S. C. H. (2020). Vd-bert: A unified vision and dialog transformer with bert.
- Wu, H. et al. (2021). Efficient and scalable inference of transformer-based language models with tensorrt. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*.
- Xu, D., Zhu, Y., Choy, C. B., and Fei-Fei, L. (2017). Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5410–5419.
- Zellers, R., Yatskar, M., Thomson, S., and Choi, Y. (2018). Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5831–5840.
- Zhou, Y., Cui, C., Yoon, J., Zhang, L., Deng, Z., Finn, C., Bansal, M., and Yao, H. (2023). Analyzing and mitigating object hallucination in large vision-language models. *arXiv preprint arXiv:2310.00754*.
- Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. (2023a). Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.
- Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. (2023b). Minigpt-4: Enhancing vision-language understanding with advanced large language models.