Event Retrieval for Driving Scenarios Highlighting

Bingqing Zu
Stanford University
zubq@stanford.com

John Ren
Stanford University
zren16@stanford.edu

Abstract

Retrieval of events from driving scenarios videos is essential to improve the efficiency of developing autonomous driving systems. In this project, we experiment with object detection with tracking method, as well as a vision-language retrieval model to extract the clip of interested driving scenarios from long test drive video. The object detection and tracking model effectively identifies object-centric scenarios by localizing and maintaining object identities across frames. In contrast, the vision-language retrieval model is capable of retrieving both object-level and scene-level scenarios based on natural language queries, leveraging cross-modal semantic alignment between visual content and textual descriptions.

1. Introduction

Modern autonomous driving systems generate a large amount of video data from test drives, while efficiently navigating and analyzing the recorded videos has remained challenging due to involved efforts to label events like "crossing pedestrians" in long drives. Manual review has proven to be time-consuming and error-prone. This project aims to automate event retrievals by developing a trained deep neural network to extract the most relevant video clips from a long-drive test video. The challenge lies in video scenario understanding and classification, and we believe this feature would facilitate developers' scenario-specific debugging in self-driving systems.

We develop and evaluate two distinct models: an object detection with tracking model and a vision-language retrieval model.

For the object detection and tracking model, the input consists of sequential frames extracted from video clips. We employ a pipeline that integrates YOLO [6] for object detection with Deep SORT [9] for multi-object tracking, outputting the classification of each clip based on the detected and tracked object patterns over time.

In contrast, the vision-language retrieval model is designed to align video clips with natural language descriptions in a shared embedding space through contrastive learning. This model combines VideoMAEForVideoClassification [7], a transformer-based video encoder pretrained via masked autoencoding, with the CLIP [5] text encoder, pretrained for image-text alignment. Given a video and a textual query, the model retrieves the most semantically relevant clips by computing similarity scores between video and text embeddings.

2. Related Work

Video scenario understanding starts with the detection of objects in the image frame. There are many state-of-the-art models for objects detection, such as YOLO [6] and transformer based model DETR [4]. However, object detection is not enough to understand the scenarios. By adding object tracking after detection, the model can capture scenarios requiring dynamic information of objects, such as a pedestrian is walking by in front of our vehicle. Popular object tracking models include real-time tracker SORT [2] and ByteTrack [12]. Some other scenarios, such as intersection, are not purely determined by objects and require deeper scene understanding. Some recent video understanding models offer state-of-the-art performance, such as VideoMAE [7] and ViViT [1]. In order to enable text query, CLIP [5] presents a framework to align visual and textual representations. Below we list several literature reviewed in details.

2.1. Transformer based integrated model

VideoMAEForVideoClassification [7] proposes a self-supervised learning framework for video representation learning based on masked autoencoding. The model extends the Masked AutoEncoder (MAE) paradigm from images to videos by randomly masking a high proportion of video tokens and training a transformer to reconstruct the missing content. This approach efficiently captures both spatial and temporal dependencies with reduced computational cost. VideoMAE achieves state-of-the-art performance on multiple video classification benchmarks, demonstrating strong generalization and scalability in learning spatiotemporal features without relying on labeled data.

CLIP [5] presents a contrastive learning framework that jointly trains an image encoder and a text encoder to align visual and textual representations in a shared latent space. Trained on large-scale image-text pairs, CLIP enables zero-shot transfer to a wide variety of visual recognition tasks by leveraging natural language supervision. Its success demonstrates the effectiveness of vision-language pretraining in bridging semantic gaps between modalities.

Combining these approaches, the VideoMAE and CLIP integrated model leverages VideoMAE's powerful spatiotemporal video embeddings alongside CLIP's language-grounded textual embeddings. This fusion facilitates cross-modal video retrieval and scene understanding by aligning video clips with natural language descriptions in a shared embedding space.

2.2. Object detection and tracking integrated model

A different way of performing video classification is to utilize the benefits of advanced object detection model and fuse it into an efficient tracking model. YOLOv8, explained by Muhammad et al. (2024) [10] and Varghese et al. (2024) [8] has been proved to be an effective and efficient online object detection method, which introduces architectural enhancements such as anchor-free detection head and improved backbone networks compared to its earlier versions. While object detection identifies objects in individual frames and it is critical in the pipeline, tracking objects across frames is crucial for understanding temporal dynamics. Tracking model such as SORT (Simple Online and Realtime Tracking) [2] and Deep SORT [9] use the detected objects information and assign consistent identities and tracks their movements across frames. With the captured movement of objects over frames, we can integrate features such as object states and counts and stack them into feature vector and feed into fully-connected layers for scene classification tasks.

3. Methods

We develop and evaluate two methodologies in parallel. The first method is a transformer-based vision-language model that integrates a video understanding encoder with an image-text alignment framework. The second method is an integrated pipeline combining object detection and multi-object tracking. Detailed descriptions of both approaches are provided below.

3.1. Vision Language Model

We propose a combined model integrating a Video Vision Transformer (ViViT)-based Variational Autoencoder (VAE) [7] with a CLIP [5] text encoder to perform video-text retrieval tasks. The model jointly learns aligned video and text embeddings for natural language video retrieval.

- Input:
  - Video clips $V = \{v_1, v_2, ..., v_B\}$, each clip containing T frames, B is batch size.
  - Corresponding text queries $Q = \{q_1, q_2, ..., q_B\}$.
- Output:
  - Embeddings in a shared latent space: video embedding $z_v \in R^d$ and text embedding $z_q \in R^d$

The framework includes three components, including video encoder, text encoder and projection head. The architecture can be viewed as below:

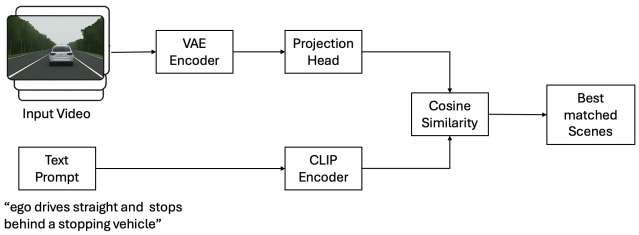

Figure 1. Vision language scene retrieval pipeline

The video encoder is based on a ViViT architecture wrapped inside a Variational Autoencoder framework. It encodes input frames V into latent features z_v.

$$z_v = VAE_{video}(V)$$

We use a pretrained CLIP text encoder to embed natural language queries q into the same d-dimensional latent space.

$$z_q = CLIP_{text}(q)$$

Lastly, both z_v and z_q are projected into a common embedding space via learnable projection heads f_v and f_q.

$$h_v = f_v(z_v), h_q = f_q(z_1)$$

The outputs h_v, h_q are normalized to unit vectors for cosine similarity computation.

We then train the model to maximize the similarity between matching video-text pairs and minimize similarity for mismatched pairs via a contrastive learning objective through contrastive loss. Given a batch of size B, the similarity matrix is computed

$$S_{ij} = h_{v_i}^T h_{q_i}, i, j \in \{1, ..., B\}$$

The loss is symmetrized cross-entropy loss:

$$L_{contrast} = \frac{1}{2B} \sum_{i=1}^{B} (l_{CE}(S_{i,:}, i) + l_{CE}(S_{:,i}, i))$$

where $l_{CE}(.,i)$ is the cross-entropy loss treating the i-th pair as the positive sample.

The model is fine-tuned on a custom dataset (refer to Section 4) by unfreezing only the final two transformer layers of the ViViT backbone (pretrained via VideoMAE[7]) and the CLIP ViT-B/32 [5]text encoder, along with the projection head. The projection head is implemented as a one linear layer with 512 units, projecting the [CLS] token output from ViViT (of dimension 768) to a shared 512-dimensional embedding space, which is aligned with the CLIP text embedding space. All other parameters are frozen to retain pretrained representations.

3.2. Detection and Tracking Model

Alternatively, we could obtain the desired scene by classifying the provided video clip. We propose an integrated method that takes advantage of YOLO[11] as the detection model. The detected per-object information including the class, bounding box, and confidence are then passed to the DeepSORT[9] model that performs multi-object tracking, which maintains consistent object IDs over frames. In this step, each object is tracked using a Kalman filter assuming a constant velocity model for motion prediction. From the tracked object data, we crafted a feature extractor that summarizes each object's motion state over frames and concatenate into a fixed-size feature vector, which is then passed to a MLP head to predict the scene label:

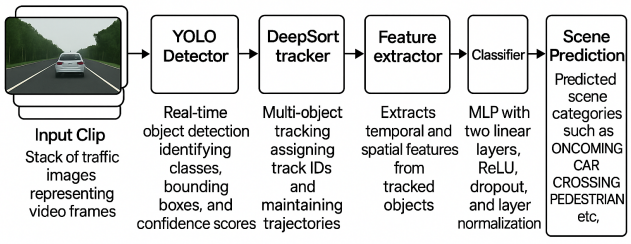

Figure 2. Detector-tracker scene classification pipeline

3.2.1 Detection

The pipeline begins with object detection and we used YOLOv12 in this experiment as it has been proven to be efficient and fast. Each video frame is processed sequentially in a single forward pass, generating class labels (e.g. car, pedestrian), bounding boxes, and confidence scores for detected objects. By filtering detections below a confidence threshold (e.g. 0.5), the system retains only reliable object candidates for subsequent racking. For simplicity, we have limited our interests of object classes to be ["person", "bicycle", "car", "motorcycle", "bus", "truck"] so other types are dropped in the detection process.

3.2.2 Tracking

DeepSORT extends the detection results by associating objects across frames, enabling consistent tracking in dynamic scenes. It integrates a Kalman filter for object motion estimation which assumes a constant velocity model for frame-to-frame prediction. Meanwhile, DeepSORT leverages deep appearance descriptors to identify objects, where a feature vector of size 128 is generated for each tracked object. The Kalman filter estimates an object's future position, while a Hungarian algorithm matches detections to existing tracks using both motion and appearance metrics hence assigns a consistent ID to the same object. In this experiment, we have limited the maximum number of tracked objects to be 5 and a minimum number of 10 frames for an object to appear to be considered relevant in a clip.

3.2.3 Feature Extraction

A custom feature extraction module aggregates spatio-temporal patterns from all tracked objects. For each confirmed track, following state is reported:

$$v_i = (u_i, v_i, \gamma_i, h_i, \dot{x}_i, \dot{y}_i, \dot{\gamma}_i, \dot{h}_i)$$

which contains bounding box center (u, v), aspect ratio γ_i, height h, velocities $(\dot{x}_i, \dot{y}_i)$. We defined the following feature vector for each object:

$$f_i = (class_i, \bar{p}_{x_i}, \bar{p}_{y_i}, \bar{v}_{x_i}, \bar{v}_{y_i}, \bar{d}_i, age_i)$$

where $class_i$ is its class ID, $\bar{p}_{x_i}, \bar{p}_{y_i}$ and $(\bar{v}_{x_i}, \bar{v}_{y_i})$ are its mean velocity and position across frames, respectively. $\bar{d}_i$ is the mean distance to the origin. Since all inputs are collected from a front camera, the origin is defined to be at the middle of bottom edge of each image. age_i indicates the number of frames an object appears in a clip:

$$\bar{p}_x = \frac{1}{T}\sum_t \frac{x_1 + x_2}{2}, \bar{p}_y = \frac{1}{T}\sum_t y_2$$

$$\bar{v}_x = \frac{1}{T-1}\sum_t \frac{p_{x_t} - p_{x_{t-1}}}{frame\ rate}, \bar{v}_y = \frac{1}{T-1}\sum_t \frac{p_{y_t} - p_{y_{t-1}}}{frame\ rate}$$

These features are expected to capture an object's behaviors such as direction of motions and proximity to the ego vehicle. To handle variable object counts per scene, we choose the top-k closest tracks by distance, and zero-padding ensures fixed-dimensional feature vector. This yields a 35-dimensional vector (5 tracks x 7 features), encapsulating the scene's dynamic context while remaining robust to transient detections.

3.2.4 Classification

The final scene classification is performed by a MLP with one hidden layer of 128 units. ReLU is used as the activation function and dropout regularization is used. It maps the feature vector to a probability distribution over predefined scene classes. During training, cross-entropy loss with Adam optimizer are used. Its simplicity ensures efficient inference and makes it suitable for real-time deployment.

4. Dataset

We used front-facing camera video frames from 1,000 driving scenes from the nuScenes dataset [3]. Each image frame has size 1600 $\times$ 900. We manually extracted and labeled 200 clips in total. The data is partitioned into training, validation, and test sets. To fine-tune both models, we extract short video clips consisting of 16 consecutive frames that correspond to scenarios of interest. Each clip is annotated with a categorical label from one of nine target classes and a natural-language description summarizing the scene. These categories represent the key driving scenarios most relevant to our study, listed in 1.

class	description
car_following	Ego is following a vehicle
car_merging	Vehicle merging into ego lane
car_oncoming	Oncoming direction has vehicle
stationary_object_ahead	Static object exists in front of ego
pedestrian_crossing	People cross in front of the ego
pedestrian_nearby	Pedestrian is nearby ego vehicle
free_traffic	No traffic in the driving scenario
complex	Complex scenario

Table 1. Dataset Classification

An example of one clip with label is shown in 3 and 4.

Figure 3. Example of one clip containing 16 frames

For the vision-language model, the input data is preprocessed using the VAEPreprocessor API provided by the VAE library. This includes resizing and cropping transformations to produce fixed-size inputs of 224×224 pixels, as required by the VAE backbone. For the detector-tracker model, the original size of each image is kept and no additional pixel-level normalization is performed. Each clip is paired with a class label as the ground truth for training.


```
{
  "clip_id": "clip_110",
  "text": [
    "ego is driving straight while pedestrians are crossing from right to left"
  ],
  "class": "pedestrian_crossing"
}
```


Figure 4. Classification and text label of the example clip

5. Experiments

We separated this section into two parts based on 2 different experimental models, since they present different results and use different evaluation metrics.

5.1. Vision-Language Model

Our model combines a pretrained VideoMAE-based ViViT encoder for video and a CLIP ViT-B/32 encoder for text, followed by a one linear layer projection head. To fine-tune the model on our custom dataset of 200 video-text pairs, we unfreeze the last two transformer layers of both the video and text encoders, as well as the entire projection head. The rest of the weights remain frozen to leverage pretrained knowledge and reduce overfitting.

We chose the AdamW optimizer due to its effectiveness in handling weight decay regularization in transformer-based models. We set the learning rate to 1e-4, with weight decay of 1e-5 to encourage generalization. The mini-batch size was set to 4, balancing GPU memory limitations and dataset size. We monitored training using cosine similarity between projected video and text embeddings, and the contrastive loss function used was the symmetric InfoNCE loss, refer to Section 3.

We evaluate our model under the natural language video retrieval setup using Recall@K.

$$Recall@K = \frac{1}{N} \sum_{i=1}^{N} 1\{rank(i) <= K\}$$

where $rank(i)$ denotes the rank of the correct video for the i-th text query. We report Recall@1, Recall@5, and Recall@10 on the test set. Additionally, we analyze the mean cosine similarity between video and text embeddings to assess embedding space alignment.

The model reaches the following results with 20 testing clips.

Recall@1	6.1%
Recall@5	45.4%
Recall@10	62.5%
Mean Cosine Similarity	0.42

Since the training data is limited, the model demonstrates moderate retrieval ability. The relatively low Recall@1 and

mean cosine similarity suggest that further fine-tuning or larger training dataset may be necessary for better performance.

To further demonstrate the performance, the model is also tested on a video with 4800 frames. We use some text queries to extract several top score relevant events. 5 shows couple good results of relevant clips extracted based on the corresponding text query.

5.2. Detection and Tracking Model

We trained the scene classification model using the Adam optimizer with a learning rate of $1e-3$ and weight decay rate of $1e-4$, selected through grid search on a validation set. The mini-batch size was set to 4 due to GPU memory constraints. We finetune the model by training the MLP classifier with input dimension = 35 and hidden dimesion = 128 while freezing the YOLOv12n and DeepSORT weights. We used confidence threshold of 0.4 for YOLO. nms_max_overlap = 0.7 and max_cosine_distance=0.4 are chosen for DeepSORT. Training is run for 50 epochs, with early stopping if validation loss plateaued for 3 consecutive epochs. We partitioned the dataset into 85% training, 10% validation, and 5% test splits.

We used Precision(P) to assess the accuracy of the classified scenes:

$$P = \frac{TP}{TP + FP}$$

The ability to classify all scenes is evaluated by Recall(R):

$$R = \frac{TP}{TP + FN}$$

The model achieved the following results with 20 test clips:

Metric	**Training**	**Validation**	**Test**
Accuracy (%)	49.1	43.2	35.3
Recall (%)	48.5	41.7	36.9

We experimented with a variant model that instead of collapsing tracker output into a single feature vector for each object, the feature extractor and MLP are replaced with a transformer-based classifier, which can be described by the following architecture: This way per-object and per-frame tracking information are stacked and fed into a multi-headed transformer encoder with 2 hidden layers with the hope that transformer can learn temporal dependencies and relationship between objects. However, we are not obtaining better results and we hypothesized this is due to the limited amount of training data.

As an example of results, Fig. 7 is a scene involving multiple car objects: In this scene, there is a mixture of intersection and leading cars while the model successfully predicts it to be the car_following in the presence

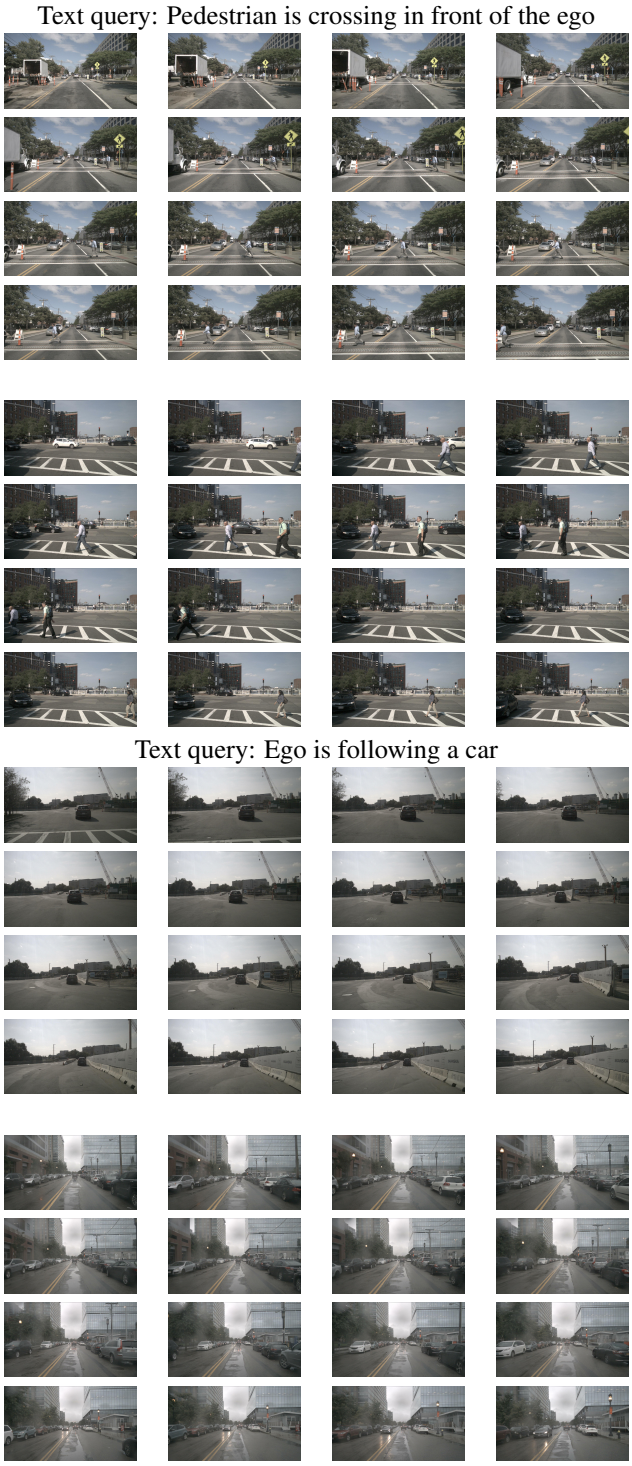

Figure 5. Event clip retrieval examples with text query

of complex background. We think the model learned the dominant feature due to the proximity criteria mentioned above. In other scenarios, motions from persons are captured and relied for scene classification. In the example in Fig. 8, even with a rich categories of objects, a person in

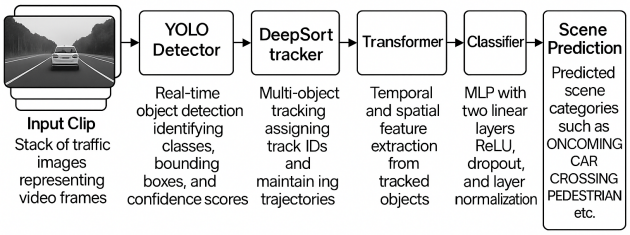

Figure 6. Detector-tracker scene classification pipeline variant

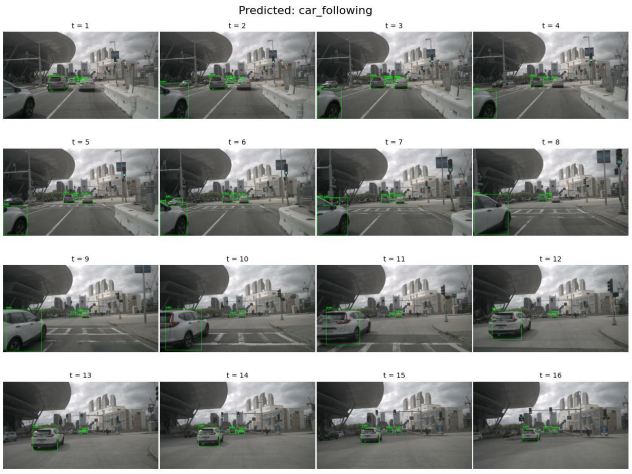

Figure 7. Ego following a white car at an intersection

the front walking across the street is correctly classified as $pedestrian\_crossing$:

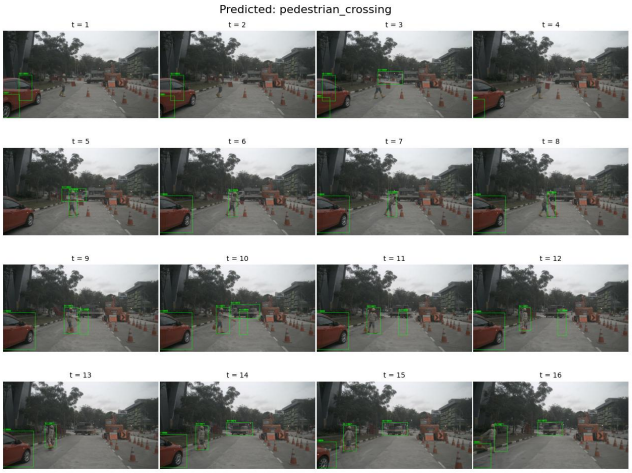

Figure 8. A crossing pedestrian in front of ego at a construction zone

Meanwhile, we noticed that the model tends to fail in the scene with combination of object behaviors that seem to be equally important. For example in the case Fig. 9, there is a stopping car in the front while there are oncoming cars on the adjacent lane at an intersection. The leading car is moving slowly but the model is not able to differentiate it from a truly stationary object. We hypothesize this type of

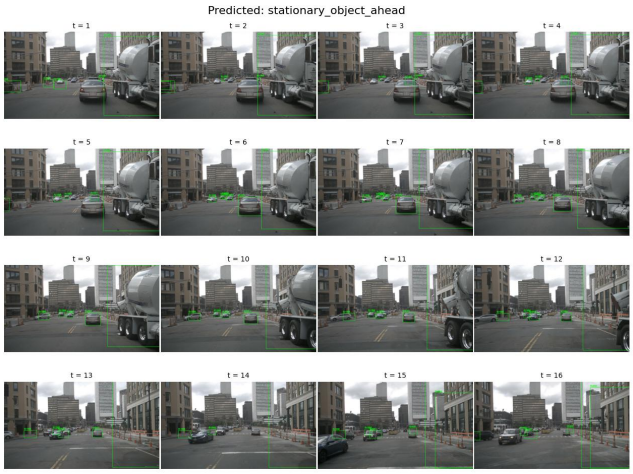

Figure 9. Ego coming to a stop behind a slow-moving car at an intersection

failure is mainly due to the missing information about the ego motions.

6. Conclusion & Future Work

To achieve event retrieval through classification or text query, we experimented two models with our own labeled dataset. We presented a vision-language retrieval framework combining VideoMAE (ViViT backbone) and CLIP (ViT-B/32) to align video clips with natural language queries. By fine-tuning only the final layers of both models and introducing a one linear layer projection head, we learned a shared embedding space using contrastive loss. Despite the small customized dataset we processed, vision-language model demonstrates ability to retrieve meaningful matching clips with text queries, and achieves promising retrieval performance on qualitative queries. On the other hand, detection and tracking model demonstrates moderate result in object-related scenario classification. Evaluation with Recall@K metrics confirms the model's capability to retrieve relevant clips. Vision-language model has more flexibility through text-image alignment and is capable of retrieving both object-related or scene-related scenarios. However it requires more testing data with detailed description for each clip. On the other hand, detection and tracking model performs better object-related classification task with limited training dataset, since the predefined class size is small. However, it limits the query

flexibility for complex and unseen scenarios. With the detection and tracking model, we presented an integrated pipeline for driving scene classification that leverages the state-of-the-art object detection and multi-object tracking to extract meaningful spatio-temporal features from video clips. By combining YOLO-based detection, DeepSORT tracking, and a compact motion feature extractor with a MLP classifier, our approach effectively captures the dynamic context of complex traffic scenes. Experimental results show that the method achieves reasonable accuracy across diverse scene categories with visualizations confirming robust understanding. While some gaps remains between similar scenarios, the modular design allows great flexibility for future improvement through enhanced feature engineering or further pursuit of transformer-based models. Based on the results, we noticed there is some level of overfit and worse generalization and we expect to improve this by expanding dataset size.

For future work, we plan to expand the labeled dataset for better performance either through manual labeling or exploring some semi-automated annotation techniques such as synthetic data generation. Additionally, we aim to incorporate auxiliary sensor modalities, such as ego vehilce position and velocity information, to enrich the model's contextual understanding and improve retrieval performance.

7. Contributions & Acknowledgments

Bingqing and John work closely together in this project from topic selection, model design and implementation, data processing, model training and testing, and report writing. Bingqing focus more on Vision Langue model while John focus more on Detection and Trakcing model.

References

[1] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lucic, and C. Schmid. Vivit: A video vision transformer. 2021.

[2] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. 2017.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. 2020.

[5] A. Radford, J. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. 2021.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. 2016.

[7] Z. Tong, Y. Song, J. Wang, and L. Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *36th Conference on Neural Information Processing Systems*, 2022.

[8] R. Varghese and S. M. Yolov8: A novel object detection algorithm with enhanced performance and robustness. *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 2024.

[9] N. Wojke, A. Bewley, and D. Paulus. Simple online and real time tracking with a deep association metric. 2017.

[10] M. Yaseen. What is yolov8: An in-depth exploration of the internal features of the next-generation object detector. 2024.

[11] Q. Y. Yunjie Tian and D. Doermann. Yolov12: Attention-centric real-time object detectors. 2025.

[12] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang. Bytetrack: Multi-object tracking by associating every detection box. 2022.