2024 Design, Automation & Test in Europe Conference (DATE 2024)
Special Initiative "Autonomous Systems Design"

Automated Traffic Scenario Description Extraction Using Video Transformers

Aron Harder
Computer Science, University of Virginia
Email: ah2ph@virginia.edu

Madhur Behl
Computer Science, University of Virginia
Email: madhur.behl@virginia.edu

***Abstract*—Scenario Description Languages (SDLs) serve as high-level encodings, offering an interpretable representation of traffic situations encountered by autonomous vehicles (AVs). Their utility extends to critical safety analyses, such as identifying analogous traffic scenarios within vast AV datasets, and aiding in real-to-simulation transfers. This paper addresses the challenging task of autonomously deriving SDL embeddings from AV data. We introduce the Scenario2Vector method, leveraging video transformers to automatically detect spatio-temporal actions of the ego AV through front-camera video footage. Our methodology draws upon the Berkeley Deep Drive - eXplanations (BDD-X) dataset. To determine ground truth actions of the ego AV, we employ BERT combined and dependency grammar-based trees, utilizing the resulting labels for Scenario2Vector training. Our approach is benchmarked against a 3D convolution (C3D)-based method and a transfer-learned video transformer (ViViT) model, evaluating both extraction accuracy and scenario retrieval capabilities. The results reveal that Scenario2Vector is highly effective in detecting ego vehicle actions from video input, adeptly handling traffic scenarios with multiple ego vehicle maneuvers.**

I. Introduction

Highly automated cyber-physical systems such as self-driving cars and delivery robots have the potential for significant societal benefits. Autonomous Vehicles (AVs) could prevent crashes and reduce traffic fatalities, as well as increase mobility independence for millions of people. However, despite years of development, the widespread deployment of fully autonomous vehicles that can operate safely remains an ongoing challenge. Public testing of AVs has revealed their failures [1], sometimes leading to tragic fatal outcomes [2]. Without a methodical approach for measuring and improving safety for AVs, we can anticipate the same issues to continue.

Defining safety for AVs is a complex issue within the industry and its definition varies considerably, from reducing road fatalities to imagining a future without driver's licenses. Safety assessments rely on self-reported data from AV companies, each with their own interpretations of safe driving behavior. Metrics such as autonomous miles driven and miles per disengagements, provide a high-level assessment of AV safety. However, it's important to note that low disengagement rates do not necessarily equate to higher safety.

Safety evaluations for AVs need to move beyond voluntary self-reporting. The transition from distance-based to scenario-based assessments of AV performance is evident in the rise of Scenario Description Languages (SDLs). These SDLs serve as semantic embeddings, capturing the complexity of driving scenarios composed of static, dynamic, and temporary elements in a structured format. SDLs and systems proposed by researchers such as [3]–[5] offer an interpretable framework for humans and support essential AV safety tasks, including large-scale virtual testing scenario generation, real-to-sim transfer, and post-deployment operational design domain monitoring. For instance, suppose we want to compare the safety behavior of two AVs. We can analyze one AV's performance in a specific traffic scenario, then utilize the corresponding SDL to find similar traffic situations for the other AV. Here, SDLs facilitate the observation of both AVs' behaviors under analogous traffic conditions, providing a deeper insight into their safety. It is essential to highlight that extracting SDLs from AV scenario data requires intensive manual labeling and oversight.

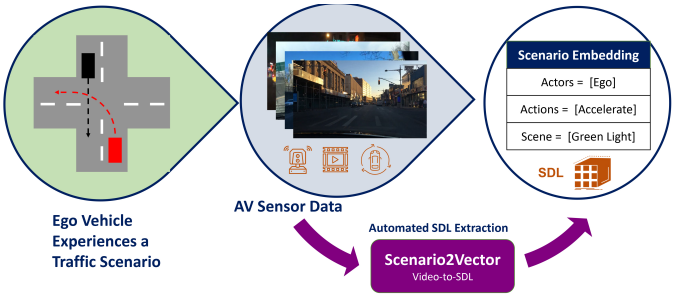

Fig. 1: Scenario2Vector Automated Scenario Description Language (SDL) Extraction Pipeline.

In this paper, we introduce Scenario2Vector - a video transformer-based architecture capable of automating SDL extraction (Figure 1). We address the challenging task of identifying spatio-temporal actions within an AV scenario and encoding them into an interpretable, and searchable, SDL embedding. This paper has the following research contributions:

1) We introduce Scenario2Vector, a video transformer-based architecture for automatic extraction of SDLs using real-world traffic scenario data from the BDD-X dataset.
2) We explore the effectiveness of parse tree and dependency graph-based grammars in extracting ground truth actions from human-specified text captions in the BDD-X dataset.
3) We evaluate our SDL extraction approach against established video understanding methods, including C3D and ViViT [6], [7], assessing both detection accuracy and its capability in scenario retrieval tasks.

979-8-3503-4859-0/DATE24/© 2024 EDAA

II. Related Work

As detailed in Section IV, we will work with traffic scenarios that are based on video data. Therefore, extracting SDLs is comparable to extracting video embeddings. Thus, our related work spans video transformer models, video action recognition, and traffic scenario descriptions, which we will combine to form a semantic embedding extraction pipeline.

Transformer-based models have been used to achieve impressive results on video analysis tasks. Building off of the Vision Transformer, the Video Vision Transformer [7] (ViViT) aims to extend the model to process temporal data. Despite ViViT's success, it was not designed for videos that are highly variable in length or videos containing multiple actions, both of which are common in our dataset. The transformer model in [8] deals with videos of differing lengths by including a recurrent model. The Cooperative Hierarchical Transformer [9] and the Bi-Modal Transformer [10] addresses videos with multiple actions by performing caption generation. All of these transformer models rely on large datasets such as Kinetics-400 for their training [11]. Compared to these benchmark datasets, our dataset is smaller and the data is not as visually distinct as most traffic videos look alike.

Approaches to video action recognition that do not use transformers typically involve using a 3D Convolutional (C3D) Network [6] to tokenize the video input for video understanding. Often the C3D Vector is used with a deep learning head to produce an action prediction [12]. This approach requires kernels of different size in the temporal dimension to accurately represent actions that last for different lengths of time. Another popular approach uses optical flow to encode temporal information [13], [14]. We use a C3D network as a traditional, non-transformer approach in this work.

Video action embeddings from these networks can be used to perform video retrieval. In video retrieval, the model learns the actions of the video to create an embedding space that groups similar videos together. Many models [15] use CNNs to compare videos in their embedding space. Multi-modal transformers [16] combine video, audio, and text to match video embeddings with a text embedding space, to perform video retrieval using text queries.

Efforts have been made to provide a detailed definition of traffic scenarios using Scenario Description Languages (SDLs). One traffic scenario description is provided by Fortellix's Measurable Scenario Description Language (M-SDL) [4]. M-SDL presents scenarios with information such as traffic lanes, pedestrians, vehicles, and motion parameters. Scenic [5] provides a traffic scenario description designed for use with the CARLA simulator. Both M-SDL and Scenic were designed with the intention of being used to make computer-generated simulations. However, they both have schemas which require manual labeling, a time consuming and potentially error prone process. Therefore, it is of interest to explore methodologies that can automatically extract SDLs.

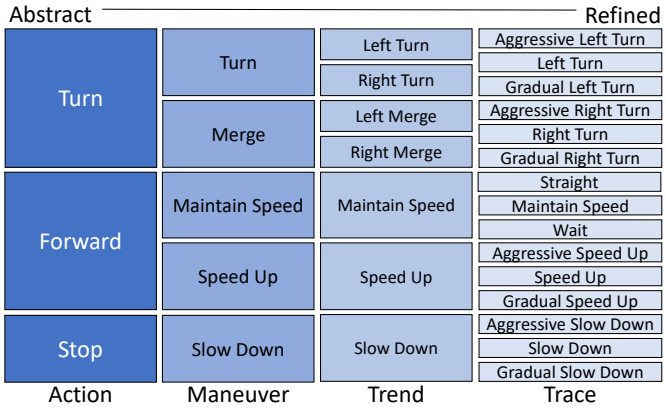

Fig. 2: The spectrum of actions from abstract to refined. Refined actions represent low level traces of lateral and longitudinal behavior. These traces can be combined into more abstract representations that define more deliberate actions.

III. Scenario Description Language

To create an SDL representation of a scenario, we first define a scenario $\mathcal{S}$ as the sequence of interactions between the autonomous *ego* vehicle and the other actors (vehicles and pedestrians) and scene elements (traffic lights and signs) that affect the ego's driving. The AV has a suite of sensors that provide data $\mathcal{D}_{ego}$ of the vehicle's experiences for each scenario, including cameras, lidar, and radar. Given a set Ω containing many traffic scenarios $\mathcal{S}$ as the recorded data from AV sensors, our goal is to extract from each scenario $\mathcal{S}$ an interpretable embedding that adequately represents this data. To do this, the embedding must be defined to encapsulate the elements of the traffic scenario that are relevant to the operation of the vehicle - the actors on the road, the motion of each actor, and environmental factors that may affect traffic.

We have defined our high-level embedding based on the SDL described by [3]. For scenario $\mathcal{S}_i$, the SDL is defined as

$$SDL(\mathcal{S}_i) \equiv \langle \{\mathcal{AC}_i, \mathcal{AT}_i\}, \mathcal{E}_i \rangle \quad (1)$$

where $\mathcal{AC}_i$, $\mathcal{AT}_i$, and $\mathcal{E}_i$ are lists of the actors, the actions performed by the actors, and the scene elements respectively. Each action also has a label associating it with a specific actor.

The contents of the SDL lists are modular to adapt to the data provided. To capture this behavior, we define the elements on an SDL spectrum from abstract to refined descriptions. Figure 2 shows the spectrum for the actions. The most refined descriptions represent low level Traces of lateral and longitudinal behavior. The next abstraction are Trends which combine similar Traces, such as "Aggressive Left Turn" represented by "Left Turn". Further up are Maneuvers which further combine Trends. The most abstract description has three Actions - "Turn" for lateral motion, "Forward" for speeding up or maintaining speed, and "Stop" for slowing down or sitting still. Determining which level of refinement to use depends on the quality of the data provided - datasets with descriptive labels can generate refined Traces, but smaller datasets may necessitate abstract Actions.

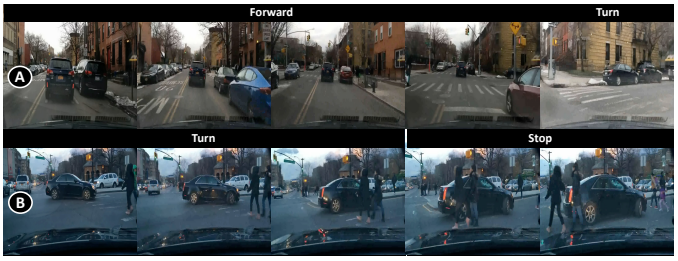

Fig. 3: Samples from the BDD-X Dataset containing multiple actions. (A): The ego drives forward for some time, before making a turn. (B): The ego starts to turn, but stops to avoid hitting pedestrians.

IV. Problem Statement

A. BDD-X Dataset

For extracting SDL embeddings from AV data, we use the publicly available Berkeley Deep Drive-eXplanations (BDD-X) dataset [17]. This dataset contains 6,970 videos lasting from 20 to 60 seconds in length. Each video $\mathcal{V}$ comprises of k clips, $\mathcal{V}_1$ to $\mathcal{V}_k$, totaling up to 26,539 clips. Each clip $\mathcal{V}_i$ also contains a human annotated caption $\mathcal{T}_i$ describing the traffic scenario. These captions are used to create the ground truth SDL representation of the scenario.

As shown in Figure 3, many BDD-X clips contain multiple visually distinct actions. Therefore, we label every clip $\mathcal{V}_i$ with both a primary action $\mathcal{P}_i$ and a secondary action $\mathcal{Q}_i$. Even with multiple actions in every clip, the BDD-X captions are not rich enough for us to operate at the refined trace-level actions of the SDL spectrum. Hence, we will operate at the abstract end, containing the three actions "Turn", "Forward", and "Stop". These actions are visually distinct for automatic detection.

B. Automatic SDL Extraction

Our goal is to find the mapping f that will take as input sensor data $\mathcal{D}_{ego,i}$ and create an SDL tuple as output. Since the BDD-X dataset only provides forward-facing camera $\mathcal{V}_{ego,i}$ data, we can state the problem as

$$f(\mathcal{V}_{ego,i}) \rightarrow \langle \{\mathcal{AC}_i, \mathcal{AT}_i\}, \mathcal{E}_i \rangle \quad (2)$$

Of the three classes in our SDL, extracting the actors $\mathcal{AC}$ and scene elements $\mathcal{E}$ is relatively easy using off-the-shelf method such as Segment Anything [18] or YOLO [19]. Identifying the actions $\mathcal{AT}$ is more difficult as they represent spatio-temporal interactions. The BDD-X dataset lacks actuation data to easily identify actions, so they need be inferred from the video clip alone. We focus on action detection for the ego to develop our approach, but the methodology can be used to detect actions for other actors as well. Therefore, our goal can be stated as

$$f(\mathcal{V}_{ego,i}) \rightarrow \mathcal{AT}_{ego,i} \quad (3)$$

The mapping function f can be realized through the models described in Section VI.

C. Scenario Retrieval

Given a query scenario $\mathcal{S}_i$, the scenario retrieval task is to find scenarios most similar to $\mathcal{S}_i$ within Ω. We define a function $d(\mathcal{S}_i, \mathcal{S}_j)$ that calculates the distance between two SDLs based on the actors, actions, and scene elements that differ between the two SDLs. A smaller distance value indicates similar SDLs. As an example, let S_A from Figure 3 be the query, and S_B be one of the scenarios in Ω. S_A has actions $\mathcal{AT}_A = \{Forward, Turn\}$ and S_B has actions $\mathcal{AT}_B = \{Turn, Stop\}$. We define $\mathcal{AT}_{AB}$ as the actions in the query $\mathcal{AT}_A$ that are missing from S_B: $\mathcal{AT}_{AB} = \{Forward\}$. Similarly, $\mathcal{AT}_{BA} = \{Stop\}$ represents the extra actions in S_B that are not in the query. These set differences measure of how many elements in S_B belong in the query S_A. We compute these set differences for actors $\mathcal{AC}$ and scene elements $\mathcal{E}$ as well. We then compose a distance vector by counting the number of elements in each of these sets:

$$D_{ij} = \langle |\mathcal{AC}_{ij}|, |\mathcal{AC}_{ji}|, |\mathcal{AT}_{ij}|, |\mathcal{AT}_{ji}|, |\mathcal{E}_{ij}|, |\mathcal{E}_{ji}| \rangle \quad (4)$$

Not all of the sets in the SDL are equally important in determining similarity. Instead, we want to amplify similarity in actions more than actors, and actors more than scene elements, so that two SDLs are not considered similar just because they have stop signs. This is achieved by scaling the effects of actions by 9 and actors by 3. This penalizes missing actions resulting in less similar SDLs. In addition, an SDL that is missing an element from the query should be less similar than one with all the elements in the query and an extra term. We capture this behavior by further penalizing missing elements by a factor of 2. This results in a weighted distance vector $wD_{ij} = \langle 6\times|\mathcal{AC}_{ij}|, 3\times|\mathcal{AC}_{ji}|, 18\times|\mathcal{AT}_{ij}|, 9\times|\mathcal{AT}_{ji}|, 2\times|\mathcal{E}_{ij}|, 1\times|\mathcal{E}_{ji}| \rangle$. The distance function is defined as the magnitude of this weighted distance vector i.e. $d(\mathcal{S}_i, \mathcal{S}_j) = \|wD_{ij}\|$ and can be used to calculate the distance between query $\mathcal{S}_i$ and each scenario in Ω. If two comparison scenarios have the same distance, we use a "tie-breaking" scheme that considers scenarios with the same actions as the query to be more similar than those with different actions.

V. BDD-X Ground Truth Extraction

Given the video clips $\mathcal{V}_i$, and captions $\mathcal{T}_i$, the ground truth actions $\mathcal{AT}_i$ are extracted by organizing $\mathcal{T}_i$ into a tree structure, shown in Figure 4. This gives us the benefit of having part-of-speech tags for each word, allowing us to identify actions by focusing on the verbs in the sentence. Furthermore, the tree structure provides knowledge about the relationship held between two words. For example, in each caption we expect a close relationship between an action described and the actor performing that action.

A. BERT Parse Tree Extraction Method

A pre-trained BERT model [20] is used to turn the caption into a context-free parse tree (Figure 4-Left). This parse tree splits a sentence into phrases, and each phrase is further split into parts of speech. Under a parse tree, the relationship strength between an action and an actor is based on the distance to a shared parent node. Nouns are converted into corresponding actors and scene elements. Each verb is converted into an action, which is then associated with the nearest actor that precedes it.

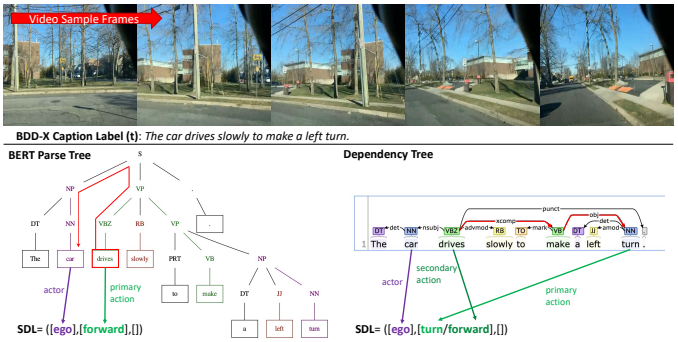

Fig. 4: The ground truth action extraction methods. The BERT-based method produces a parse tree that associates actions with actors based on sentence proximity. The Dependency Tree method associates words regardless of their distance in the sentence. The BERT-based method relies on word proximity, so it cannot easily produce secondary action ground truth.

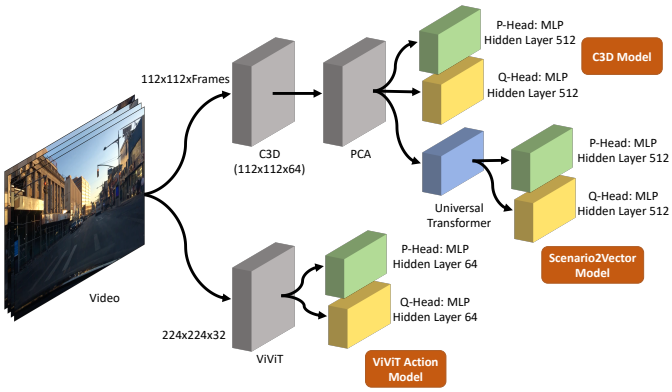

Fig. 5: Model architecture for each of the models. Pre-trained components are shown in gray. Single-headed models only use the primary heads (green) and two-headed use both primary and secondary heads (yellow).

B. Dependency Tree Extraction Method

Using Stanford Core NLP [21] to generate a dependency tree, related words are connected by labeled edges that define their relationship. For example, in Figure 4 "car" and "drives" are connected by an edge labeled *nsubj*, meaning that "car" is the *nominal subject* of the verb "drives". That is, the edge tag indicates "car" performs the action "drives". By analyzing edge tags we can also connect "car" with the action "turn".

Edge tags help relate words that are spatially distance in the sentence. The extraction process for the dependency tree starts by finding nouns corresponding to actors and scene elements. For each actor, we follow its edges to find its associated actions. When an actor has multiple actions, it is difficult to obtain their temporal order from the caption. Instead, we use a hierarchy of importance $\{turn, stop, forward\}$ to determine the primary action $\mathcal{P}_i$. "Turn" is given the most importance as it is the most visually distinct action.

VI. SDL Extraction Methods

We train three models for automatic SDL extraction (Section IV-B). The C3D Model uses a pre-trained C3D backbone. The ViViT Model uses a pre-trained Transformer backbone. The Scenario2Vector Model uses a universal transformer on top of the C3D backbone. The overview of the models is shown in Figure 5.

A. C3D Model

The C3D Model uses a C3D network [6] backbone as a feature extractor for BDDX videos. The C3D backbone has been pre-trained on the Sports-1M dataset. The BDD-X video frames are downsampled to 16 Hz and encoded into vectors by the C3D network. Since our videos are variable length, the vectors are padded with zeros to the maximum video length. These resultant vectors are then reduced from 4096 values per second to 512 using PCA. The head of the C3D model is a MLP with a hidden layer of size 512, which takes the PCA-reduced vectors as inputs and detects the action as output.

B. ViViT Action Model

The ViViT Action Model uses a ViViT encoder model [7] backbone pre-trained on the Kinetics-400 dataset and fine-tuned on the BDD-X data. The inputs for the encoder are 32 video frames uniformly sampled across the sample. The encoder then performs its own tokenization and produces a 1x768 vector. This MLP head takes this 1x768 vector as input, uses a hidden layer of size 64, and detects the action as output.

C. Scenario2Vector Model

Our method uses a transformer encoder model on top of the C3D backbone to perform attention on the inputs. We use a universal transformer architecture [22] for the Scenario2Vector Model because its recurrent nature and attention mechanism allows us to better capture temporal data. The Scenario2Vector Model uses the same PCA-reduced C3D vectors as the C3D Model for the inputs to the transformer encoder model. The transformer model uses 6 heads, 2 attention layers, a total key depth of 16, a total query depth of 16, and an output depth of 16. These values were chosen to keep the model's resource overhead low. The vectors produced by the encoder model are then passed to an MLP head with a hidden layer of size 512. The Scenario2Vector Model is designed to improve upon the C3D Model by including a transformer to apply self-attention to the C3D vectors, while being more lightweight than the ViViT Action Model.

D. PQ-Head Model Variants

For each of the three models we also trained a variant of the model with two MLP heads, the $\mathcal{P}$-head which detects the primary action of the video and the $\mathcal{Q}$-head which detects the secondary action of the video. This two-headed variation allows us to detect two actions for samples containing both a primary and secondary action, rather than detecting only a single action. We are interested in evaluating if training two action heads separately will improve action detection accuracy.

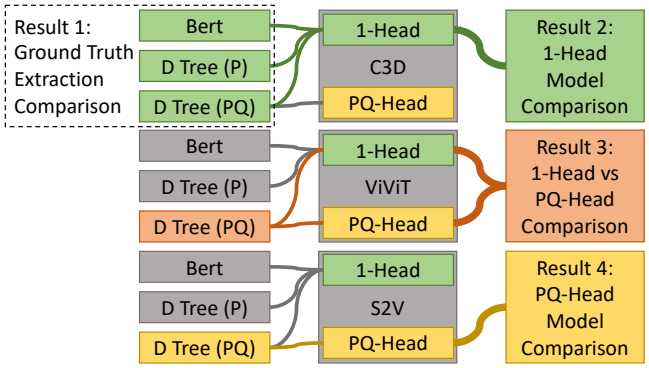

Fig. 6: Single-headed models were trained on each extraction method, while two-headed models were only trained on D. Tree ($\mathcal{PQ}$). The results compare each model based on the ground truth they were trained on.

VII. Experiments

We trained the three models on different ground truth extraction methods (Figure 6): 1) BERT Models were trained on the primary action extracted using the BERT parse tree. 2) Dependency Tree ($\mathcal{P}$) Models were trained on the primary action extracted using the Dependency Tree. 3) Dependency Tree ($\mathcal{PQ}$) Models were trained on both the primary and the secondary actions extracted using the Dependency Tree. The single-headed variants were trained on all three ground truth extraction methods, while the two-headed variants were only trained on the Dependency Tree ($\mathcal{PQ}$) method.

A. Dataset and Metrics

Each model was trained on the BDD-X Dataset to detect the action of the ego vehicle $\mathcal{AT}_{ego,i}$. We split the BDD-X Dataset into a training set containing 90% of the data (23885 samples) and a validation set comprised of the remaining 10% of samples. For the scenario retrieval task, we use the validation set as the dataset Ω from which we are finding similar scenarios as defined in section IV-C.

Model Accuracy Metrics We use the following methods for comparing the accuracy of the models:

1) $\mathcal{A}_{\mathcal{P}}$ is the percent of samples where the model correctly detects the primary action $\mathcal{P}_{ego,i}$. This metric is used for models that were trained on BERT or Dependency Tree ($\mathcal{P}$) ground truth data which only has a primary action.
2) $\mathcal{A}_{\mathcal{PQ}}$ is the percent of samples where the model correctly detects either $\mathcal{P}_{ego,i}$ or $\mathcal{Q}_{ego,i}$. For single-headed models, this represents whether the single detection matches either label. For two-headed models, this represents whether the detected primary action matches $\mathcal{P}_{ego,i}$ or the secondary action matches $\mathcal{Q}_{ego,i}$. This metric is used for models that were trained on Dependency Tree ($\mathcal{PQ}$) ground truth data which has both labels.
3) $\mathcal{A}^2_{Both}$ is the percent of samples where the model correctly detects both the primary action and the secondary action. This metric is only used for two-headed models, since single-headed models cannot detect both actions.

Model	BERT	D. Tree ($\mathcal{P}$)	D. Tree ($\mathcal{PQ}$)
C3D	0.743	0.726	0.767
ViViT	0.755	0.728	**0.839**
Scenario2Vector	**0.790**	**0.769**	**0.839**

TABLE I: The accuracy for each single-headed model when trained on each ground truth extraction variation. The BERT and D. Tree ($\mathcal{P}$) variations use the $\mathcal{A}_{\mathcal{P}}$ accuracy, while the D. Tree ($\mathcal{PQ}$) uses the $\mathcal{A}_{\mathcal{PQ}}$ accuracy.

Scenario Retrieval Metrics For the scenario retrieval task, we know the action labels a prior, but in a real-world application we may not have the labels. Therefore, we use the model's detected actions to calculate distance. To evaluate the scenario retrieval results, we use mean Average Precision (mAP), which is commonly used for this task [15]. In this evaluation, a scenario is considered relevant if it has the same ground-truth $\mathcal{P}_{ego,i}$ and $\mathcal{Q}_{ego,i}$ labels as the query.

B. Implementation Details and Ablation

During training, the learning rate was ablated to 1e-4, 1e-5, and 1e-6, and 1e-5 had the best tradeoff between convergence and training. We observed that Adam optimizer converges faster than SGD. Xavier weight initialization helped reduce overfitting compared to random weight initialization. The code and pre-trained models are publicly available at https://github.com/linklab-uva/Scenario2Vector.

Both BERT and Dependency Tree ($\mathcal{P}$) use a weighted Categorical Cross Entropy loss, defined as

$$\mathcal{L} = -\frac{1}{M}\sum_{k=1}^{K}\sum_{m=1}^{M}\frac{n_k}{M} \times y_m^k \times log(\hat{y}_m^k) \tag{5}$$

where M is the number of training samples, K is the number of classes, and n_k is the number of training samples with ground truth label k. With Dependency Tree ($\mathcal{PQ}$), the primary and secondary actions are combined into a single weighted loss function:

$$\mathcal{L}_{constant} = \frac{2}{3}\mathcal{L}_{\mathcal{P}} + \frac{1}{3}\mathcal{L}_{\mathcal{Q}} \tag{6}$$

where $\mathcal{L}_{\mathcal{P}}$ and $\mathcal{L}_{\mathcal{Q}}$ are derived based on Eq. 5 for each head.

C. Result 1: Performance of Ground Truth Extraction Methods

The results of training each single-headed model on each experiment setup are shown in Table I. The models trained on the BERT ground truth achieved a higher accuracy than those trained on the Dependency Tree ($\mathcal{P}$) actions. This shows that the BERT method of associating actions and actors together by their proximity in the captions is more accurate to the videos than the Dependency Tree method of using contextual connections. However, the models trained on the Dependency Tree ($\mathcal{PQ}$) actions outperform the BERT method which has only a primary action. This validates our hypothesis that many videos have multiple actions, and that both actions should be considered in the detection.

D. Result 2: Performance of Single-Headed Architectures

In addition to comparing ground truth extraction methods, we can also infer the accuracy of the different model architectures from the data in Table I. Between the single-headed

Model	1-Head $\mathcal{A}_{PQ}$	PQ-Head $\mathcal{A}_{\mathcal{PQ}}$	$\mathcal{A}^2_{Both}$
C3D	0.767	**0.854**	0.596
ViViT	0.839	0.810	0.643
Scenario2Vector	0.839	**0.851**	**0.696**

TABLE II: The $\mathcal{A}_{\mathcal{PQ}}$ accuracy for single-headed and two-headed models when trained on the D. Tree $(\mathcal{PQ})$ ground truth, and the $\mathcal{A}^2_{Both}$ accuracy for two-headed models.

variants, Scenario2Vector outperforms the other models. For the Dependency Tree $(\mathcal{PQ})$ variation, the ViViT Model achieves the same performance as Scenario2Vector indicating that despite being lightweight, Scenario2Vector manages to perform better than the pre-trained ViViT Model.

E. Result 3: Single-Headed vs. Two-Headed Architectures

The accuracy of the single-headed models can be compared against the accuracy for the two-headed models using the $\mathcal{A}_{\mathcal{PQ}}$ metric. Table II shows the accuracy for each two-headed model. These results show C3D and Scenario2Vector outperform with their two-headed variants. This trend does not hold for ViViT, indicating that the ViViT Model does not generalize well to traffic videos containing multiple actions.

F. Result 4: Performance of Two-Headed Architectures

In order to address the closeness in $\mathcal{A}_{\mathcal{PQ}}$ accuracy between the C3D and Scenario2Vector models, we further inspect their performance. Table II shows the $\mathcal{A}^2_{Both}$ accuracy for the three models. Scenario2Vector shows an improvement of 17% compared to C3D in terms of correctly detecting both the actions suggesting that Scenario2Vector is a more accurate model. Despite having the highest $\mathcal{A}_{\mathcal{PQ}}$ accuracy, C3D is the worst model when detecting both actions, highlighting the benefit of deeper analysis of these models.

G. Result 5: Performance on Scenario Retrieval

We use the two-headed models to perform scenario retrieval on the validation set Ω as described in Section IV-C. Scenario2Vector achieved a mAP of 0.628, 3% higher than C3D (0.607) and 10% higher than ViViT (0.570). These results show that Scenario2Vector ranks relevant videos more highly than the other two models. This suggests that Scenario2Vector is a suitable methodology to be used for scenario retrieval.

VIII. Conclusion

We presented Scenario2Vector, a transformer-based architecture capable of automatic SDL extraction (ego actions) from the BDD-X dataset. We compare Scenario2vector with C3D and ViVit and find that it has the best overall accuracy for detecting the primary and secondary actions, and well as on scenario retrieval. We also investigated the effectiveness of different methods for generating the ground truth labels from captions, and find that adding a secondary ego action to the SDL increases prediction accuracy across all models. Our ongoing work incorporating action detection for non-ego vehicles, preserving the actions' temporal order, and utilizing NuScenes data for more refined SDL extraction.

References

[1] Brad Templeton. Cruise cars crash into san francisco muni bus and tangle in fallen trolley wires. *Forbes*, Mar 2023.

[2] Jose Rodriguez. Tesla autopilot may be responsible for another fatal crash into a firetruck. *Jalopnik*, Mar 2023.

[3] Aron Harder, Jaspreet Ranjit, and Madhur Behl. Scenario2vector: scenario description language based embeddings for traffic situations. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, pages 167–176, 2021.

[4] Foretellix. Open m-sdl. https://www.foretellix.com, 2019.

[5] Eric Vin, Shun Kashiwa, Matthew Rhea, Daniel J. Fremont, Edward Kim, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. 3d environment modeling for falsification and beyond with scenic 3.0, 2023.

[6] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[7] Mostafa Dehghani, Alexey Gritsenko, Anurag Arnab, Matthias Minderer, and Yi Tay. Scenic: A JAX library for computer vision research and beyond. *arXiv preprint arXiv:2110.11403*, 2021.

[8] Jiewen Yang, Xingbo Dong, Liujun Liu, Chao Zhang, Jiajun Shen, and Dahai Yu. Recurring the transformer for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14063–14073, 2022.

[9] Simon Ging, Mohammadreza Zolfaghari, Hamed Pirsiavash, and Thomas Brox. COOT: cooperative hierarchical transformer for video-text representation learning. *CoRR*, abs/2011.00597, 2020.

[10] Vladimir Iashin and Esa Rahtu. A better use of audio-visual cues: Dense video captioning with bi-modal transformer. *CoRR*, abs/2005.08271, 2020.

[11] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

[12] Yang Jin, Linchao Zhu, and Yadong Mu. Complex video action reasoning via learnable markov logic network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3242–3251, 2022.

[13] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 568–576, Cambridge, MA, USA, 2014.

[14] Trent Weiss and Madhur Behl. Deepracing: A framework for autonomous racing. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1163–1168, 2020.

[15] Hao Cheng, Ping Wang, and Chun Qi. Cnn retrieval based unsupervised metric learning for near-duplicated video retrieval. *ArXiv*, abs/2105.14566, 2021.

[16] Nina Shvetsova, Brian Chen, Andrew Rouditchenko, Samuel Thomas, Brian Kingsbury, Rogerio S Feris, David Harwath, James Glass, and Hilde Kuehne. Everything at once-multi-modal fusion transformer for video retrieval. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 20020–20029, 2022.

[17] Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John Canny, and Zeynep Akata. Textual explanations for self-driving vehicles. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[18] Alexander Kirillov, Eric Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. Berg, W. Lo, P. Dollár, and R. Girshick. Segment anything, 2023.

[19] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[20] Nikita Kitaev, Steven Cao, and Dan Klein. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy, July 2019. Association for Computational Linguistics.

[21] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[22] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *CoRR*, abs/1807.03819, 2018.