Universal Embeddings for Spatio-Temporal Tagging of Self-Driving Logs

Sean Segal[12], **Eric Kee**[1], **Wenjie Luo**[12], **Abbas Sadat**[1], **Ersin Yumer**[1], **Raquel Urtasun**[12]
Uber Advanced Technologies Group[1], University of Toronto[2]
{ssegal,asadat,yumer,urtasun}@uber.com

Abstract: In this paper, we tackle the problem of spatio-temporal tagging of self-driving scenes from raw sensor data. Our approach learns a universal embedding for all tags, enabling efficient tagging of many attributes and faster learning of new attributes with limited data. Importantly, the embedding is spatio-temporally aware, allowing the model to naturally output spatio-temporal tag values. Values can then be pooled over arbitrary regions, in order to, for example, compute the pedestrian density *in front of the SDV*, or determine if *a car is blocking another car at a 4-way intersection*. We demonstrate the effectiveness of our approach on a new large scale self-driving dataset, **SDVScenes**, containing 15 attributes relating to vehicle and pedestrian density, the actions of each actor, the speed of each actor, interactions between actors, and the topology of the road map.

Keywords: Self-Driving Cars, Tagging, Deep Learning

1 Introduction

In order to be deployed at scale, self-driving vehicles (SDVs) need to be extensively analyzed and tested in various challenging scenarios to ensure proper handling of safety critical situations. Augmenting previous recordings of self-driving trips, or data logs, with rich metadata has many applications. For example, it enables efficient retrieval of scenarios for simulation, insightful failure analysis through visualization of tags most correlated with system failures, and rapid curation of better datasets for training the learned components of the system. Therefore, the ability to tag self-driving logs with useful metadata has become increasingly important for the development of SDVs.

Human experts are often used to label scenes with a variety of attributes to analyze failure modes, particularly in cases where the self-driving system disengaged or the safety driver took over. In the industry, this is known as *triage*. While this approach produces useful data, it does not scale. Alternatively, the outputs from the onboard perception system could be reused to heuristically reason about different scenarios based on the detections over time. Unfortunately, this approach requires extensive engineering, often returns noisy results, and cannot generalize to many scenarios which require reasoning beyond more than the detected actors.

Motivated by the shortcomings of these methods, in this paper we introduce a novel approach for spatio-temporal tagging of self-driving scenes, which requires only raw sensor data and HD maps as input and generalizes to tagging a diverse set of attributes. Our approach learns a universal embedding for all tags, enabling efficient tagging of many attributes in a given scene. Given a particular tag attribute, we combine a learned attribute embedding with the universal data log embeddings to obtain a spatio-temporal tensor of attribute values. This spatio-temporal tensor can be pooled over arbitrary regions, producing interpretable scene tags. Our approach is trained end-to-end to minimize a multi-task tagging loss, with techniques to ensure learning is balanced across tags.

Existing self-driving datasets either lack rich scene metadata [1, 2, 3, 4], use only camera images [5], or only provide annotations for the SDV itself rather than all actors in the scene [6]. Therefore, we introduce **SDVScenes**, a novel large-scale dataset with over 40 hours of driving, containing LiDAR observations, HD maps and spatio-temporal annotations for 15 important scene attributes. Attributes are both discrete and continuous valued and cover actor density, vehicle actions, interactions, map topology information and vehicle speed for all actors.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

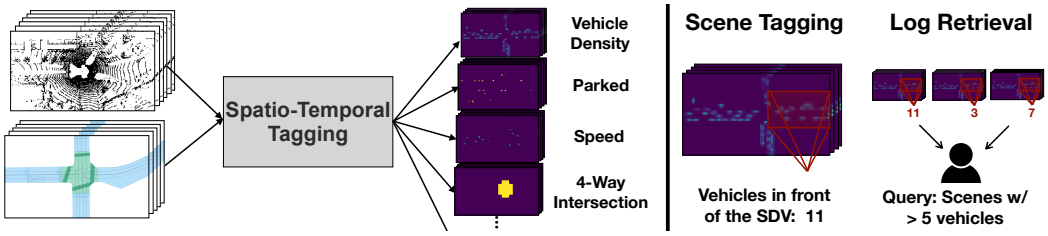


Figure 1: (Left) **Our task:** Tagging diverse attributes of a self-driving scene. (Right) **Applications:** Spatio-temporal tags used downstream to generate interpretable scene tags and retrieve relevant logs.

Using **SDVScenes**, we demonstrate that a single model can simultaneously tag a diverse set of attributes relating to the self-driving scene. Additionally, we show that new attributes can be added at later training stages and achieve better performance than an independently trained model. Finally, we analyze our system's performance with multiple ablation studies showcasing that our model leverages all inputs and is more efficient than separately trained models, while providing better performance. We plan to release a benchmark to encourage future work on this exciting new task.

2 Related Work

Self-Driving Log Understanding: Few works address data log tagging or retrieval in the context of self-driving. [7] solves the problem of video retrieval using natural language queries by first parsing descriptions into semantic graphs and then matching them to visual concepts using a bipartite matching algorithm. This model, however, was trained on only 21 videos (8008 frames), whereas our dataset is several orders of magnitude larger. More recently, [8] introduced a system to retrieve driving scenarios based on similarities in driver behavior using dash-cameras and IMU sensors. [9] introduces a library for composing outputs of pretrained computer vision models for video analysis and demonstrates applications in the context of self-driving data log mining.

Video Understanding: As the amount of video data continues to grow at a staggering rate, a rich literature has developed for tools to better understand and summarize this data. Many new network architectures (e.g., [10, 11, 12, 13]) have been introduced to best capture both spatial and temporal relationships in videos for better classification and tagging. Many works additionally consider fusing video representations with language [14, 15, 16, 17] to solve video question answering and captioning tasks. Work in video summarization introduced models which select either a subset of keyframes [18, 19] or keyshots [20, 21, 22, 23], which best preserve the information from the original video.

Action and Interaction Prediction: Self-driving scenes contain many actors, each performing one or more actions at any given time, such as turning, lane changing, or braking. As a result, many desired tags for self-driving scenes naturally relate to the current actions of agents in the scene. Previous work has studied predicting the future actions, or intentions, of each agent in a scene. For example, [24, 25] predict the intentions of vehicles at intersections. More recently, [26] introduced a model which jointly detects actors, classifies their intention and predicts their future trajectory from raw sensor data and HD maps. [5] predicts a set of hierarchically organized actions, used downstream as priors to better predict the future trajectories of actors and ego-motion. There has also been a growing interest in not only understanding each agent's current action, but also the interactions between agents in the scene [27, 28, 29, 30, 31, 32].

Multi-task Learning: Recent works in the self-driving domain have shown the ability to jointly learn multiple sub-tasks [33, 34, 35, 36, 37] for increased efficiency, and in certain cases better performance. More generally, there is increasing interest in techniques which balance conflicting multi-task objectives, including automatically reweighting losses [34, 38] and prioritizing specific tasks for efficient learning [39].

3 Universal Embeddings for Spatio-Temporal Log Tagging

Understanding raw data logs captured by self-driving vehicles is of crucial importance for applications such as simulation, triage analysis and dataset curation. In this paper, we tackle the problem of identifying precisely *when* and *where* complex events occur in raw data logs, represented by HD maps and LiDAR observations captured by a self-driving fleet. Towards this goal, we propose to

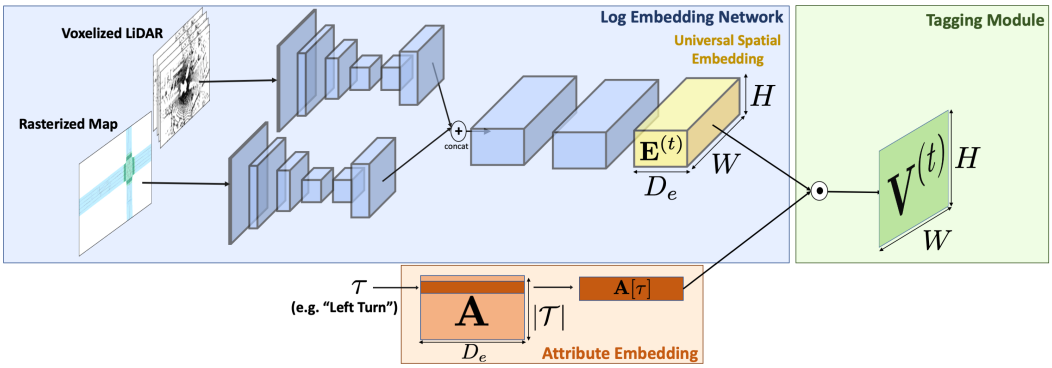


Figure 2: **Model design:** High level design of our model, shown for one timestep, t.

learn a single universal embedding from raw logs capturing all information that is relevant to any of the supported tags. Importantly, this embedding maintains spatio-temporal dimensions: each element in the embedding represents information at a corresponding spatial location and timestep in the scene. Given a tag attribute, we lookup its learned embedding representation. Then, our tagging module takes as input the universal embedding and the attribute embedding, and extracts the relevant information to produce spatio-temporal tag values. Importantly, our model is trained end-to-end to both learn the embeddings and compute all tags. An overview of our approach is outlined in Fig. 2.

We approach the problem of spatio-temporal tagging by learning a function, f, which takes a data log $\mathbf{L}$ and a tag attribute τ from a predefined set $\mathcal{T}$ and returns spatio-temporal tag values,

$$\mathbf{V} = f(\mathbf{L}, \tau) \ . \tag{1}$$

$\mathbf{V}$ maintains spatio-temporal dimensions, $T \times H \times W$. Each element $\mathbf{V}^{(t)}_{h,w}$ is a value, either discrete or continuous, corresponding to a specific timestep, t and spatial location in the observed scene, represented by the coordinates (h, w) in Bird's Eye View. Next, we describe our approach to representing f for all tags $\tau \in \mathcal{T}$ with a single neural network. We also demonstrate extensions to compute a tag's value for an arbitrary region and compose attributes for more complex tags.

3.1 Universal Embeddings

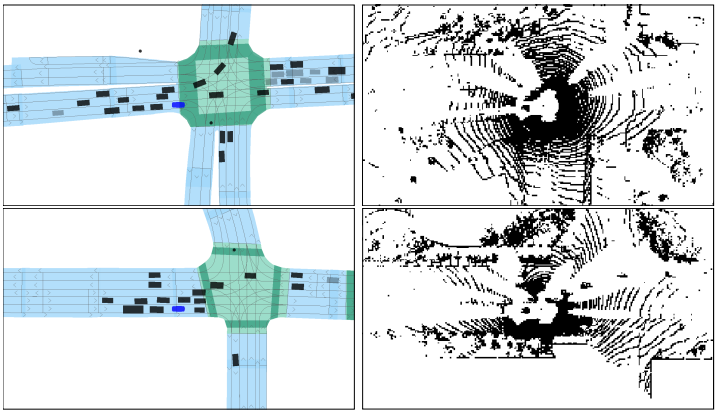
Log Embedding Network: Our fully convolutional embedding network, f^{θ}_{e} takes a raw log $\mathbf{L}$ as input and outputs a spatio-temporal universal embedding:

$$\mathbf{E} = f^{\theta}_{e}(\mathbf{L}) \ , \tag{2}$$

where $\mathbf{L}$ represents the recorded LiDAR and HD maps for the entire log. Note that f^{θ}_{e} does not depend on any particular tag attribute. This allows our model to efficiently share the computation of important intermediate features that may be relevant to multiple tags. Additionally, this enables our embedding to be precomputed and stored for fast on demand tagging.

Because data logs can have arbitrary length, T, our embeddings are fully convolutional across the time dimension with a receptive field of N timesteps. Having a sufficiently large temporal receptive field is important, as many attributes might require looking at several frames to be tagged accurately (e.g., braking, turning, vehicle speed). The N LiDAR sweeps are corrected for ego-motion to bring the point clouds into the same coordinate system, centered at the SDV. We follow [26] and rasterize the space into a 3D occupancy grid, where each voxel indicates whether it contains a LiDAR point.

Many tag attributes require reasoning about each actor's position with respect to the road map. For example, tagging lane changes requires understanding the position of each actor with respect to the lanes. Following [26], we rasterize the map into M channels, each representing a different element, e.g., road, intersections, lanes, lane boundaries, traffic lights. The full input representation for a given frame is therefore a tensor of size $H_L \times W_L \times (ZN + M)$, where Z and H_L, W_L are the height and x-y dimensions respectively. The embedding for each frame $\mathbf{E}^{(t)}$, computed by f^{θ}_{e}, has size $H \times W \times D_e$, where D_e is the embedding dimension and the spatial dimensions H, W are obtained by downsampling H_L and W_L by a factor of r.



Scene Tag	Model	GT
Vehicles in the scene:	21.0	21.8
Pedestrians on a crosswalk:	1	1
Vehicle turning left in the intersection:	Yes	Yes
Maximum vehicle speed in front of the SDV:	10.9 m/s	10.9 m/s

Scene Tag	Model	GT
Maximum vehicle speed behind the SDV:	14.2 m/s	13.5 m/s
Vehicle density around the SDV:	13.0	13.2
Vehicle blocked by vehicle in front of the SDV:	Yes	Yes
4-way intersection in front of the SDV:	Yes	Yes

Figure 3: **Qualitative Scene Tags:** (Left) Visualization of the scene (**not** given to the model). (Center) Sensor observations for a central timestep (Right) Scene tags generated by our model.

The architecture of our network is inspired by recent works in object detection [40]. First, we process the voxelized LiDAR and the rasterized map with independent backbones. Then, their features at multiple resolutions are upsampled and concatenated together, which is given to a convolutional header to obtain our universal embedding. Please refer to the supplementary material for details.

Attribute Embedding: Tag attributes might be related to one another. For example, the action "vehicle braking" and interaction "vehicle is braking due to another vehicle" are clearly related. Other attributes, like turns and intersection types, may also be related in more complicated ways. Inspired by the success of word embeddings in NLP [41], we learn an embedding representation for each tag attribute. Specifically, we introduce a learnable embedding matrix $\mathbf{A}$ with dimension $|\mathcal{T}| \times D_e$, where $|\mathcal{T}|$ is the number of attributes. Given a tag attribute, τ, its embedding is obtained by indexing the attribute's corresponding row, $A[\tau]$.

Tagging Module: Given the log embedding representation $\mathbf{E}$ and the attribute representation $\mathbf{a} = A[\tau]$ as input, our tagging module returns a spatio-temporal tensor of tagged values with dimension $T \times H \times W$ using a pointwise dot product along the embedding dimension,

$$\mathbf{V} = \mathbf{E} \odot \mathbf{a} \ . \tag{3}$$

This approach is parameter free, forcing our learned embeddings to capture all relevant information.

Scene Tags: Our system can also compute a tag's value over an arbitrary region, R, via pooling,

$$v = f_p(\mathbf{V}^{(t)}, R) \ . \tag{4}$$

These regions can be defined with respect to the SDV, for example encoding the region in front of, or behind the SDV, or according to regions from an HD map, for example, encoding regions with a crosswalk. For example, if $\mathbf{V}^{(t)}$ is the pedestrian density at each spatial location, we can obtain an estimate of the region's density with $f_p = \texttt{sum}$, summing the values in $\mathbf{V}^{(t)}$ over R. We also use $f_p = \texttt{max}$ to pool logits, outputting the probability that an attribute is present *somewhere* in R.

Compositional Tags: Many interesting scenarios in self-driving scenes can be best expressed in terms of simple scene attributes. For example, tagging scenes with vehicle's stopped at a 4-way intersection could be solved by recognizing stopped vehicles and 4-way intersections independently, and then reasoning about the existence of the two outputs to obtain a final tag. Motivated by this compositionality, we extend our model to support compositional tags, which can be expressed as functions of the outputs of simpler tags. More formally, we define a compositional tag τ_c as,

$$f(\mathbf{L}, \tau_c) = g\left(\{\mathbf{V}_\tau : \tau \in \mathcal{T}_c\}\right) \ . \tag{5}$$

Compositional attributes are fully defined by the compositional function g and the subset of tag attributes $\mathcal{T}_c \subseteq \mathcal{T}$. In this work, we use simple compositional functions of the form,

$$g_{\text{AND}}\left(\cdot\right) = \prod_{\tau \in \mathcal{T}_c} h_\tau(\mathbf{V}_\tau) \ , \tag{6}$$

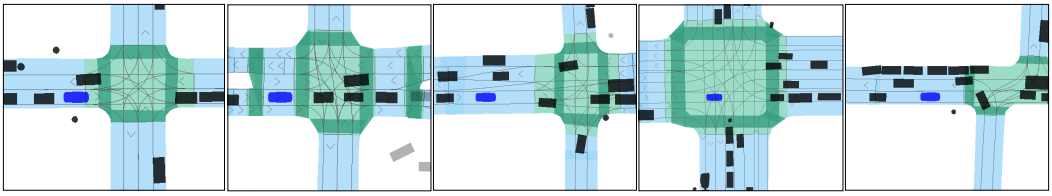


Figure 4: **Log Retrieval:** Visualizations of scenes retrieved by our model using the compositional tag for vehicles blocked by vehicle at a 4-way intersection in front of the SDV. Importantly, our model is given only raw sensor data and HD maps, not the actor labels used for visualization.

where $h_\tau(\cdot)$ is either the identity or an indicator function $\mathbb{1}\{\mathbf{V}_\tau \geq \kappa\}$ applied elementwise. As an example, imagine we have two attributes "left turn" and "4-way intersection", which have taggers which output the probability of each event occurring at locations across the scene. Then, the output of the multiplicative compositional function g_{AND} corresponds to the probability that there is a vehicle turning left at a 4-way intersection under the assumption that the events are independent. g_{OR} can be derived similarly, using a sum instead of product and subtracting by g_{AND} as the events may not be mutually exclusive. We leave the use of more advanced, potentially learned, compositional functions as future work.

3.2 Learning

We use supervised learning to jointly learn both the embedding network and the attribute embeddings end-to-end. Let $\Theta = \{\theta, \mathbf{A}\}$ be the collection of model parameters. Given a database of training logs $\mathcal{L} = \{\mathbf{L}\}$ with ground truth tag values for each attribute $\tilde{\mathbf{V}}_\tau$, we train our model to minimize a multi-task tagging loss which minimizes the loss across all attributes, timesteps, and data logs in the training set,

$$\min_{\Theta} \sum_{\mathbf{L}} \sum_{t}^{T} \left(\sum_{\tau \in \mathcal{D}_\tau} \ell_d(f_\Theta(\mathbf{L}^{(t)}, \tau), \tilde{\mathbf{V}}_\tau^{(t)}) + \sum_{\tau \in \mathcal{C}_\tau} \ell_c(f_\Theta(\mathbf{L}^{(t)}, \tau), \tilde{\mathbf{V}}_\tau^{(t)}) \right) , \tag{7}$$

where $\mathcal{D}_\tau$ and $\mathcal{C}_\tau$ are the discrete and continuous tag attributes, respectively. The loss for discrete attributes, ℓ_d is the standard cross entropy loss, applied per-pixel. For continuous-valued attributes, the loss ℓ_c is a standard regression loss (e.g., smooth L1) applied both independently per-pixel and, for density tags, after pooling via summation over the entire scene. Applying the loss after pooling ensures that errors will not be spatially correlated. Naively optimizing Equation 7 with these losses will be suboptimal given both large imbalances in the tagged value distributions and the multi-task nature of the objective. Therefore, we leverage two techniques to ensure stable and efficient learning.

Hard Negative Mining: In the context of self-driving, the most interesting scene attributes often occur infrequently. For example, interesting maneuvers, like lane changes and actor-to-actor interactions, will predominantly take on negative values, resulting in highly imbalanced supervision. We solve this using hard negative mining and only apply the loss to negative pixels in $\tilde{\mathbf{V}}^{(t)}$ with the highest predicted confidences in $\mathbf{V}^{(t)}$. In practice, we sort the predictions of each pixel where the ground truth is negative by their predicted confidences and only apply the cross entropy loss to the hardest examples, ensuring there are at most $K = 3$ negative pixels for each positive pixel in a frame.

Task Balancing: Hard negative mining ensures that for each attribute, $\tau \in \mathcal{T}$, the distribution of the tagged values is balanced. At the same time, it introduces imbalance between the amount of supervision each task receives. As an example, consider two attributes which we use in our system: parked and left lane changes. Almost all frames will have at least one parked vehicle producing useful signal for the network, while the many frames without left lane changes will produce no signal. To counter this imbalance, we preprocess our dataset and for each attribute, $\tau \in \mathcal{T}$, we compute the subset of frames that have at least one location with a positive label, $\mathcal{L}(\tau)$. Then, to construct minibatches for training, we first sample a tag attribute uniformly and then sample a log frame $\mathbf{L}^{(t)}$ that is "interesting" for the given task uniformly from $\mathcal{L}(\tau)$.

Region	Density (L1) ↓		Actions (F1) ↑										Map (F1) ↑		Speed (L1) ↓
	Veh	Ped	P	S	B	KL	RT	LT	RC	LC	BB	BF	3-way	4-way	Speed
Full	0.93	1.40	93	95	83	98	76	76	44	53	66	67	99	99	0.62
Around	0.46	0.90	94	94	75	96	76	77	45	55	68	67	100	99	0.48
Front	0.31	0.61	93	92	67	92	74	78	39	52	63	59	99	99	0.49
Behind	0.29	0.45	94	89	71	93	76	72	46	56	69	65	100	99	0.46
Intersection	0.31	0.27	-	83	58	91	78	77	37	44	59	48	99	99	0.59
Crosswalk	-	0.21	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 1: **Scene Tagging:** Metrics computed over different regions of interest. Values are omitted if tag is not applicable (e.g., parked at an intersection)

	Joint Embedding		Separate Embedding	
D_e	Disc. (F1) ↑	Cont. (L1) ↓	Disc. (F1) ↑	Cont. (L1) ↓
4	50	1.62	10	494*
16	71	0.98	50	1.20
64	**72**	**0.93**	**71**	**0.92**

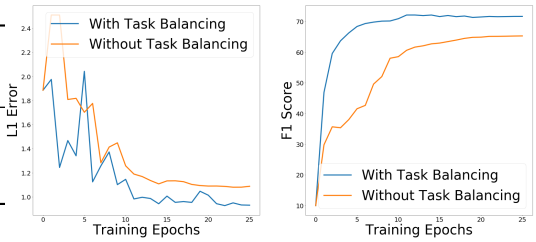


Figure 5: (Left) **Embedding Ablations:** Geometric mean of metrics for different embedding dimensions and configurations. *Model unable to learn given memory limitations. (Right) **Task Balancing:** Effect of our task balancing sampling scheme on learning.

4 Dataset and Experiments

We evaluate our approach on our new large scale dataset, **SDVScenes** (see Section 4.1), showcasing our model's ability to simultaneously tag a diverse and complex set of scene attributes. We also evaluate our model in the continuous learning setting, where new tag attributes are added at later stages of the training process. In particular, we show stronger performance when new attributes are learned from a pretrained universal embedding, demonstrating that information captured by our embedding generalizes to new attributes. Finally, we perform a set of ablation studies to understand how different inputs, embedding configurations & sizes, and training schemes affect performance.

4.1 SDVScenes Dataset

We introduce a novel large-scale dataset which contains sensor observations, HD maps, and annotations of 15 scene attributes relating to actor density, actions, interactions, map topology and vehicle speed, which will be released in a future benchmark.

For the data logs, we collected roughly 40 hours of driving over multiple cities across North America. We split the data into 4857 data logs for training, 477 for validation and 960 for testing. Each log is roughly 25 seconds, and our dataset provides observations and supervision at 10 Hz, resulting in roughly 1M training frames, 100K validation frames, and 200K testing frames.

Continuous Scene Attributes: Our dataset contains annotations for 3 continuous-valued scene attributes, divided into two categories: actor speed and density. We provide annotations for the speed of each moving vehicle, as it is an important aspect of the scene. Our dataset also provides bounding box annotations for both vehicles and pedestrians, allowing us to compute the density for any region in the scene. We represent this actor density as a continuous attribute to handle cases where vehicles on the boundary are only partially visible (e.g., 0.5 vehicle density).

Discrete Scene Attributes: Our dataset contains annotations for 12 discrete scene attributes. For each vehicle, we annotate 8 common actions: Parked, Stopped, Braking, Keeping Lane, Right (Left) Turn, Right (Left) Change. Additionally, our dataset contains annotations for two vehicle-to-vehicle interactions: whether a vehicle is blocked-by, or braking-for another vehicle. Precise definitions of these can be found in the supplementary material. All actions and interactions take binary values and are labeled independently. We also have annotations for the map's topology: intersections are either 3-way intersections, 4-way intersections or neither (intersections with more than 4 arms).

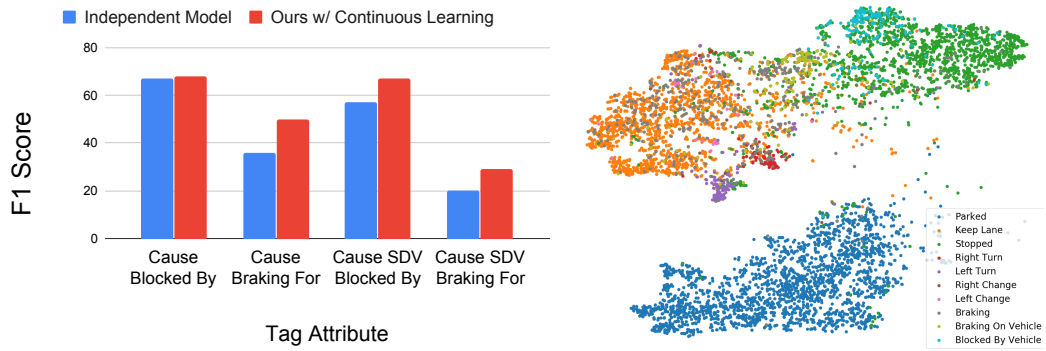


Figure 6: (Left) **Continuous Learning:** New attributes on the x-axis. (blue) independently trained baseline model, (red) continuously learned approach. (Right) **Embedding Visualization:** t-SNE [42] of the universal embedding space. Points correspond to embedding vectors at spatial locations in $\mathbf{E}^{(t)}$ with vehicles. Colors indicate the primary action or interaction of the vehicle.

Implementation Details: For all experiments, we use $H_L = 160$, $W_L = 280$ (at a 0.5 meter per pixel resolution), $Z = 3$ (at a 1 meter per pixel resolution), a spatial downsampling rate of $r = 2$ and $M = 15$ map channels. We use $N = 10$ frames, sampled at 5 Hz. Unless otherwise noted, the embedding dimension is $D_e = 64$. For each epoch, we sample 25000 examples per attribute without replacement and minimize the loss in Eq. (7). To train the parameters, we use the Adam optimizer [43] with an initial learning rate $\alpha = 0.0001$ that is decayed by 0.1 every 10 epochs. We use a batch size of 10 examples per GPU and employ data-parallelism to train with 32 GPUs. We notice that density tags require more examples than others in order to reach convergence. Hence, we train our final model in two stages. First, we train on density tags for 30 epochs. Then, we add all remaining tags and train for 25 more epochs.

4.2 Tagging Performance

Scene Tagging: In Table 1, we evaluate the performance of our model in the scene tagging setting for a variety of regions, R. For regions, we use the entire scene, the region in front of, behind and around the SDV (each defined precisely in the supplementary). Additionally, we evaluate on 2 map-based regions: intersections and crosswalks. Given that actions, interactions and speed tags are only trained on locations with vehicles, tagging one of these attributes τ, over a larger region requires reasoning about both the presence of a vehicle and the value of τ at each location. Therefore, we use a compositional tag that depends on both vehicle density and the tag attribute τ, where g is a compositional function which thresholds the density to obtain locations with vehicles which is multiplied by $f(\mathbf{L}, \tau)$ to obtain the final tensor of tagged values. We use L1 error and F1 score to measure the performance of continuous and discrete attributes, respectively. Generally, tagging vehicles in intersections appears most challenging, likely because vehicle behavior is more complicated in these regions compared to regular roads. Fig. 3 shows sample scene tags output by our model.

Retrieval Setting: We test our model qualitatively in the retrieval setting by searching for "gridlocked" scenarios at intersections, known to be challenging for SDVs. For all logs in the evaluation set, we tag whether there is a vehicle blocked by vehicle at a 4-way intersection in front of the SDV. In Fig. 4, we show visualizations of the scenes with actor labels shown (not given to the model, but useful for qualitative evaluation) that our model tagged as positives. The results show a diverse set of scenes where the SDV is at a crowded intersection, which could be used in practice to test whether the SDV would plan a safe maneuver without blocking the intersection.

Continuous Learning: In a real-world setting, the tag attributes supported by our system will grow as requirements change, and new supervised data becomes available. Ideally, models should support continuous learning, in which new tag attributes are trained on top of a learned model. Therefore, we explore whether our learned embedding can generalize to new attributes. In this experiment, we add an additional set of 4 attributes to create a new set $\mathcal{T}_{new}$. We introduce two attributes to tag whether a vehicle *causes* another vehicle to be blocked or braking and two attributes

Inputs		Density (L1) ↓		Actions (F1) ↑										Map (F1) ↑		Speed (L1) ↓
Map	LiDAR	Veh	Ped	P	S	B	KL	RT	LT	RC	LC	BB	BF	3-way	4-way	Speed
		7.98	4.91	52	45	16	45	3	4	1	2	4	4	42	83	4.0
✓		5.06	3.58	98	77	28	79	51	55	29	37	17	8	**99**	**100**	2.20
	✓	1.02	1.49	95	87	58	90	70	76	27	39	48	52	98	99	0.67
✓	✓	**0.93**	**1.40**	**99**	**92**	**60**	**91**	**74**	**77**	**43**	**53**	**51**	**56**	**99**	**100**	**0.62**

Table 2: **Input Ablation:** Model performance by input. First row computed from dataset statistics.

to tag whether a vehicle is specifically causing the SDV to be blocked or braking. We train our previously converged model with $\mathcal{T}_{\text{new}}$ for an additional 25 epochs and compare performance against an independently trained model for each new attribute. Fig. 6 shows that our model always outperforms the independently trained baseline and can provide up to **45%** increase in F1 score. Importantly, our continuously trained model maintained similar performance on all other attributes.

4.3 Ablation Studies and Visualization

For the following experiments, we evaluate the performance of each tag attribute at its relevant locations (e.g., everywhere for actor density, locations with vehicles for actions, intersections for map topology and locations with moving vehicles for speed.)

Joint vs. Separate Model: In Fig. 5, we compare the performance of our model trained with different embedding dimensions and configurations. For each dimension, we compare our universal embedding versus an approach with separated embeddings for each tag category, each with an independently trained model. Despite requiring *one fourth* the model capacity, our approach significantly outperforms the separated approach when embedding memory is limited ($D_e = 4, 16$) as it learns to share the embedding space more efficiently. Given a large enough embedding ($D_e = 64$), both approaches perform similarly. The separated approach, however, does not scale as it requires both large embedding memory and a new network to be trained for each new tag category.

Leveraging Inputs: In Table 2, we ablate model inputs to understand the relative gain from adding LiDAR and HD maps. The first row uses no input, outputting tag values based on dataset statistics. L1 error is minimized by predicting the tag's median and F1 score is maximized by always predicting positive. Unsurprisingly, the poor performance of this baseline demonstrates that our model cannot simply exploit dataset biases and must utilize sensor input. Overall, we find maximal performance is attained when leveraging both LiDAR and HD maps. However, we find that HD map information alone is sufficient to tag map topology attributes and parked vehicles. Similarly, vehicle speed and turns achieve strong performance with only LiDAR as these attributes are primarily motion-based.

Task Balancing: Fig. 5 plots validation metrics throughout training with and without our task balancing scheme. We notice that our approach to sampling examples converges in fewer epochs and achieves better performance across both continuous and discrete tag attributes.

Visualizing Embeddings: Fig. 6 visualizes the embeddings of evaluation logs using t-SNE [42], considering spatial locations in the embedding tensor $\mathbf{E}$ that are associated with vehicles. Three dominant clusters are apparent: parked vehicles (blue), stopped vehicles (green), and vehicles keeping lane (orange). Additional clusters are present for left and right turns. This demonstrates that the embedding captures high level semantics about each spatial location (e.g., actions at that location).

5 Conclusion

We introduced a novel dataset and approach for spatio-temporal tagging of self-driving scenes. Our model's output can be directly applied to generating interpretable scene tags and data log retrieval. Our approach is designed to be efficient through the use of a universal embedding, and achieves good performance across a diverse set of tags. Additionally, we demonstrate that our embedding is generalizable and can be used to improve the performance of new tags. We plan to release a benchmark, as we believe there are many exciting directions for future work including model improvements, more complex compositional tags and additional sensor inputs (e.g., RGB cameras, Radar).

References

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[2] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, pages 2446–2454, 2020.

[3] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, pages 8748–8757, 2019.

[4] 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. In *CVPR*, pages 11621–11631, 2020.

[5] S. Malla, B. Dariush, and C. Choi. Titan: Future forecast using action priors. In *CVPR*, pages 11186–11196, 2020.

[6] Y. Xu, X. Yang, L. Gong, H.-C. Lin, T.-Y. Wu, Y. Li, and N. Vasconcelos. Explainable object-induced action decision for autonomous vehicles. In *CVPR*, pages 9523–9532, 2020.

[7] D. Lin, S. Fidler, C. Kong, and R. Urtasun. Visual semantic search: Retrieving videos via complex textual queries. In *CVPR*, pages 2657–2664, 2014.

[8] M. Guangyu Li, B. Jiang, Z. Che, X. Shi, M. Liu, Y. Meng, J. Ye, and Y. Liu. Dbus: Human driving behavior understanding system. In *ICCV Workshops*, pages 0–0, 2019.

[9] D. Y. Fu, W. Crichton, J. Hong, X. Yao, H. Zhang, A. Truong, A. Narayan, M. Agrawala, C. Ré, and K. Fatahalian. Rekall: Specifying video events using compositions of spatiotemporal labels. *arXiv preprint arXiv:1910.02993*, 2019.

[10] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, pages 568–576, 2014.

[11] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014.

[12] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 6299–6308, 2017.

[13] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *ECCV*, pages 803–818, 2018.

[14] H. Yang, L. Chaisorn, Y. Zhao, S.-Y. Neo, and T.-S. Chua. Videoqa: question answering on news video. In *eleventh ACM international conference on Multimedia*, pages 632–641. ACM, 2003.

[15] K.-H. Zeng, T.-H. Chen, C.-Y. Chuang, Y.-H. Liao, J. C. Niebles, and M. Sun. Leveraging video descriptions to learn video question answering. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[16] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler. Movieqa: Understanding stories in movies through question-answering. In *CVPR*, pages 4631–4640, 2016.

[17] L. Zhu, Z. Xu, Y. Yang, and A. G. Hauptmann. Uncovering the temporal context for video question answering. *International Journal of Computer Vision*, 124(3):409–421, 2017.

[18] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *2012 CVPR*, pages 1346–1353. IEEE, 2012.

[19] B. Gong, W.-L. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *NeurIPS*, pages 2069–2077, 2014.

[20] Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *CVPR*, pages 2714–2721, 2013.

[21] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang. Automatic video summarization by graph modeling. In *Proceedings Ninth ICCV*, pages 104–109. IEEE, 2003.

[22] K. Zhang, K. Grauman, and F. Sha. Retrospective encoders for video summarization. In *ECCV*, pages 383–399, 2018.

[23] Z. Ji, K. Xiong, Y. Pang, and X. Li. Video summarization with attention-based encoder-decoder networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[24] T. Streubel and K. H. Hoffmann. Prediction of driver intended path at intersections. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 134–139. IEEE, 2014.

[25] Y. Hu, W. Zhan, and M. Tomizuka. Probabilistic prediction of vehicle semantic intention and motion. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 307–313. IEEE, 2018.

[26] S. Casas, W. Luo, and R. Urtasun. IntentNet: Learning to Predict Intention from Raw Sensor Data. In *Conference on Robotic Learning (CoRL)*, 2018.

[27] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

[28] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, and S. Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *CVPR*, 2018.

[29] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, F.-F. Li, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. *2016 CVPR*, pages 961–971, 2016.

[30] L. L. Li, B. Yang, M. Liang, W. Zeng, M. Ren, S. Segal, and R. Urtasun. End-to-end contextual perception and prediction with interaction transformer. *IROS*, 2020.

[31] S. Casas, C. Gulino, R. Liao, and R. Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *arXiv preprint arXiv:1910.08233*, 2019.

[32] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

[33] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, pages 3569–3577, 2018.

[34] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, pages 7482–7491, 2018.

[35] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *CVPR*, pages 8660–8669, 2019.

[36] S. Chowdhuri, T. Pankaj, and K. Zipser. Multinet: Multi-modal multi-task learning for autonomous driving. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1496–1504. IEEE, 2019.

[37] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, pages 7345–7353, 2019.

[38] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*, 2017.

[39] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei. Dynamic task prioritization for multitask learning. In *ECCV*, pages 270–287, 2018.

[40] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, pages 7652–7660, 2018.

[41] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119, 2013.

[42] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[43] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.