Trajeglish: Learning the Language of Driving Scenarios

Jonah Philion[1,2,3], **Xue Bin Peng**[1,4], **Sanja Fidler**[1,2,3]
[1]NVIDIA, [2]Vector Institute, [3]University of Toronto, [4]Simon Fraser University

Abstract

A longstanding challenge for self-driving development is the ability to simulate dynamic driving scenarios seeded from recorded driving logs. In pursuit of this functionality, we apply tools from discrete sequence modeling to model how vehicles, pedestrians and cyclists interact in driving scenarios. Using a simple data-driven tokenization scheme, we discretize trajectories to centimeter-level resolution using a small vocabulary. We then model the multi-agent sequence of motion tokens with a GPT-like encoder-decoder that is autoregressive in time and takes into account intra-timestep interaction between agents. Scenarios sampled from our model exhibit impressive realism; on the WOMD "sim agents" benchmark for simulating up to 128 agents per scenario, our model sets a new state-of-the-art, topping the realism meta metric by 3.3% and the interaction metric by 9.9%.

1 Introduction

In this work, we propose an autoregressive model of the motion of roadusers that can be used to simulate how humans might react in the real-world if a self-driving system were to choose a given sequence of actions. At a high level, as visualized in Fig. 2, the model outputs a categorical distribution over a set of possible states an agent might move to at the next timestep, conditional on previous state information as well as the actions of any other agents in the scene that have already chosen an action for the current timestep. While prior work generally models the actions of each agent as independent within a single timestep, we assume access to an agent-by-agent simulator in which all agents not controlled by our model choose actions, and then the remaining agents sample actions one by one, each time conditional on all actions already chosen. We call our approach "trajeglish" due to the fact that we model multi-agent trajectories as a sequence of discrete tokens, similar to the representation used in language modeling, as well as to draw an analogy between the motion that road users use to communicate and verbal languages people use to communicate.

A selection of samples from our model are visualized in Fig. 1. In contrast to prior work [11, 7, 10] that generally requires at least a second of historical motion to begin sampling, our model is capable of generating realistic scenarios given only the initial heading and position of the agents. Within a row, the initialization is the same and outlined in black. Our model is able to model diverse possible outcomes for a given scenario, while maintaining the scene-consistency of the trajectories.

Our main contributions are:

- A simple data-driven method for tokenizing trajectory data we call "k-disks" that enables us to tokenize the Waymo Open Dataset (WOMD) [3] to, on average, within 1 cm discretization error using a small vocabulary size of 384.
- A transformer-based architecture for modeling sequences of motion tokens that conditions on map information and one or more initial states per agent. Our model outputs a distribution of actions for agents one at a time which we show is ideal for interactive applications.

NeurIPS 2023 Workshop on Robot Learning: Pretraining, Fine-Tuning, and Generalization with Large Scale Models, New Orleans, United States of America

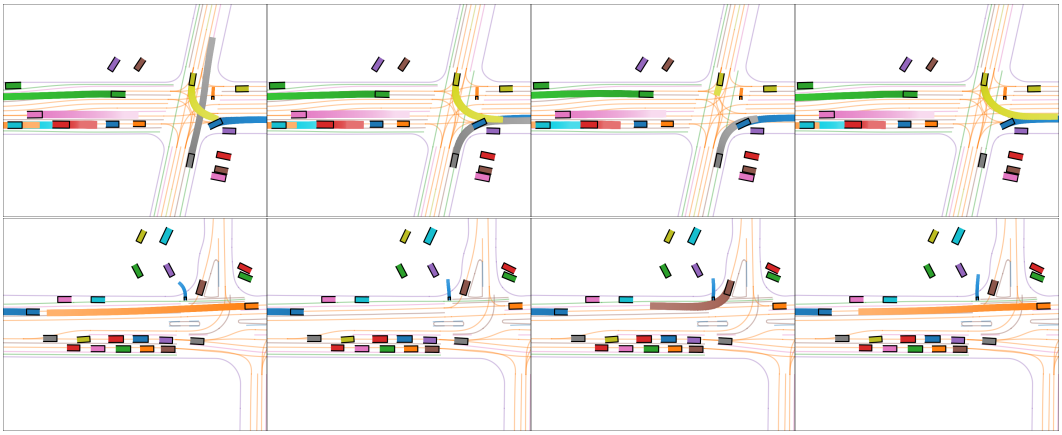

Figure 1: **Trajeglish** Visualizations of samples from our model. Rollouts within each row are given the same initialization, outlined in black. The timestep is visualized by lightening the agent's future trajectory. While some tracks overlap in the figure, they do not overlap when time is taken into account; there are no collisions in these rollouts.

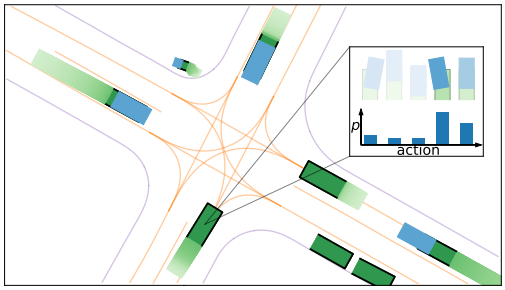

Figure 2: **Inputs and outputs** At a given timestep, our model predicts a distribution over a fixed set of V states defined relative to an agent's current location and heading, and conditions on map information, actions from all previous timesteps (green), and any actions that have already been chosen by other agents within the current timestep (blue). We model all motion of all agents relevant to driving scenarios, including vehicles, pedestrians, and cyclists.

- State-of-the-art quantitative and qualitative results when sampling rollouts given real world initializations both when the traffic model controls all agents in the scene as well as when the model must interact with agents outside its control.

2 Tokenization

The continuous distributions that we seek to tokenize for the case of traffic modeling are distributions over changes in position and heading. Given the low dimensionality of the input data, we propose a simple approach for tokenizing trajectories based on a fixed set of template state-to-state transitions.

Method Let $\boldsymbol{s}_0$ be the state of an agent with length l and width w at the current timestep. Let $\boldsymbol{s}$ be the state at the next timestep that we seek to tokenize. We define $V = \{\boldsymbol{s}_i\}$ to be the set of *template actions*, each of which represents a change in position and heading in the coordinate frame of the most recent state. We use the notation $a_i \in \mathbb{N}$ to indicate the index representation of token template $\boldsymbol{s}_i$ and $\hat{\boldsymbol{s}}$ to represent the raw representation of the tokenized state $\boldsymbol{s}$. Our tokenizer f and renderer r are defined by

$$f(\boldsymbol{s}_0, \boldsymbol{s}) = a_{i^*} = \arg\min_i d_{l,w}(\boldsymbol{s}_i, \text{local}(\boldsymbol{s}_0, \boldsymbol{s})) \tag{1}$$

$$r(\boldsymbol{s}_0, a_i) = \hat{\boldsymbol{s}} = \text{global}(\boldsymbol{s}_0, \boldsymbol{s}_i) \tag{2}$$

where $d_{l,w}(\boldsymbol{s}_0, \boldsymbol{s}_1)$ is the average of the L2 distances between the ordered corners of the bounding boxes defined by $\boldsymbol{s}_0$ and $\boldsymbol{s}_1$, "local" converts $\boldsymbol{s}$ to the local frame of $\boldsymbol{s}_0$, and "global" converts $\boldsymbol{s}_{i^*}$ to the global frame out of the local frame of $\boldsymbol{s}_0$. We use $d_{l,w}(\cdot, \cdot)$ throughout the rest of the paper to refer to this mean corner distance metric. Importantly, in order to tokenize a full trajectory, this process of converting states $\boldsymbol{s}$ to their tokenized counterpart $\hat{\boldsymbol{s}}$ is done iteratively along the trajectory instead of in parallel, using tokenized states as the base state $\boldsymbol{s}_0$ in the next tokenization step. The final tokenized

representation of a trajectory of length T is given by the initial state followed by $T-1$ indices. Tokens generated with our approach have three convenient properties for the purposes of traffic modeling: they are affine invariant, equivariant under temporal shift, and they supply efficient access to a measure of similarity between tokens, namely the distance between the raw representations.

Optimizing template sets A good set of template actions is one that results in tokenized trajectories that have a small average corner distance to the raw input trajectories. We propose to sample candidate templates by collecting a large number of state transitions observed in data, sampling one of them, filtering any transitions that are within ϵ meters under the corner distance metric, and repeating $|V|$ times. We call this method for sampling candidate templates "k-disks" given it's similarity to KMeans++, the standard algorithm for seeding the anchors for KMeans [1], as well as the Poisson disk sampling algorithm from computer graphics [2]. We show in Fig. 3 that we find template sets that discretize vehicle, pedestrian, and cyclist trajectories to within 1-2 centimeters on average.

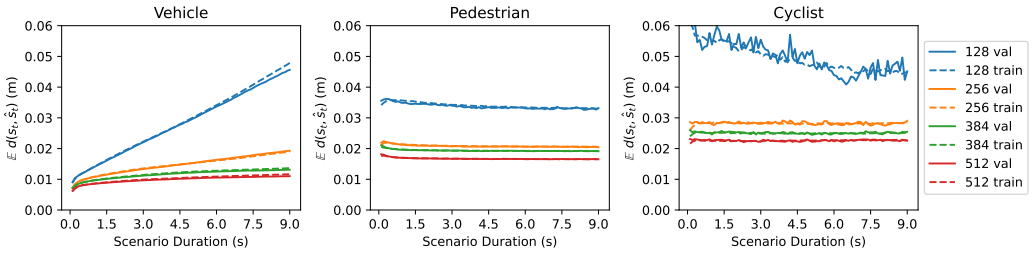

Figure 3: **K-disk discretization error** Average corner distance for each of the k-disk vocabularies of sizes 128, 256, 384, and 512. Gains to resolution drop off quickly after 384.

3 Modeling

The second component of our method is an architecture for learning a distribution over the sequences of tokens output by the first. Our model follows a transformer-based encoder-decoder structure very similar to those used for LLMs [13, 8, 9]. A diagram of the model is shown in Fig. 4. During training, a random ordering of agents is chosen and actions are listed in that order for the first timestep, followed by the next timestep, etc. ensuring the ability to rollout autoregressively in time at test time.

4 Experiments

We use the Waymo Sim Agents Benchmark to compare the performance of Trajeglish to prior work based on VAEs and mixture-of-x modeling. Details on how we use our model to sample scenarios

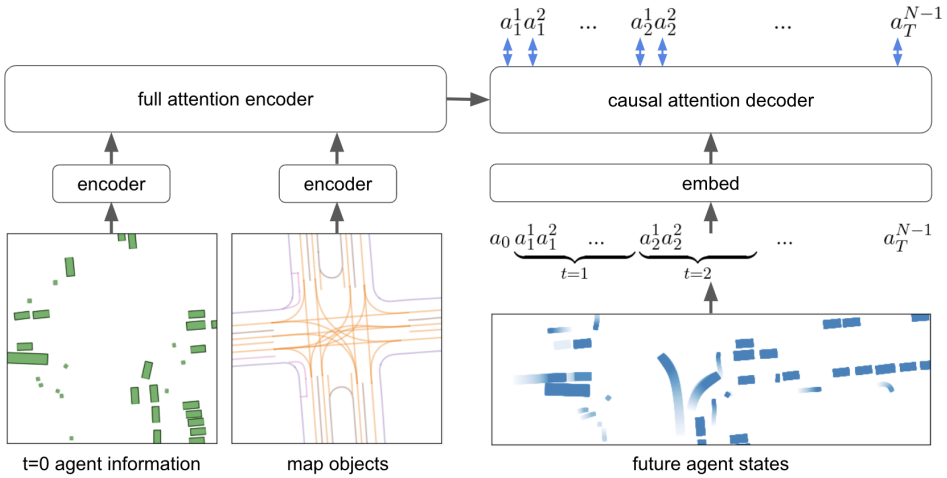

Figure 4: **Trajeglish modeling** We train an encoder-decoder transformer that predicts the action token of an agent conditional on previous action tokens, map information, and agent information available at $t=0$. The diagram represents the forward pass of the network during training in which $t=0$ agent information, map objects, and motion tokens are passed into the network using a causal mask and the model is trained to predict the next motion token, shown in the top right.

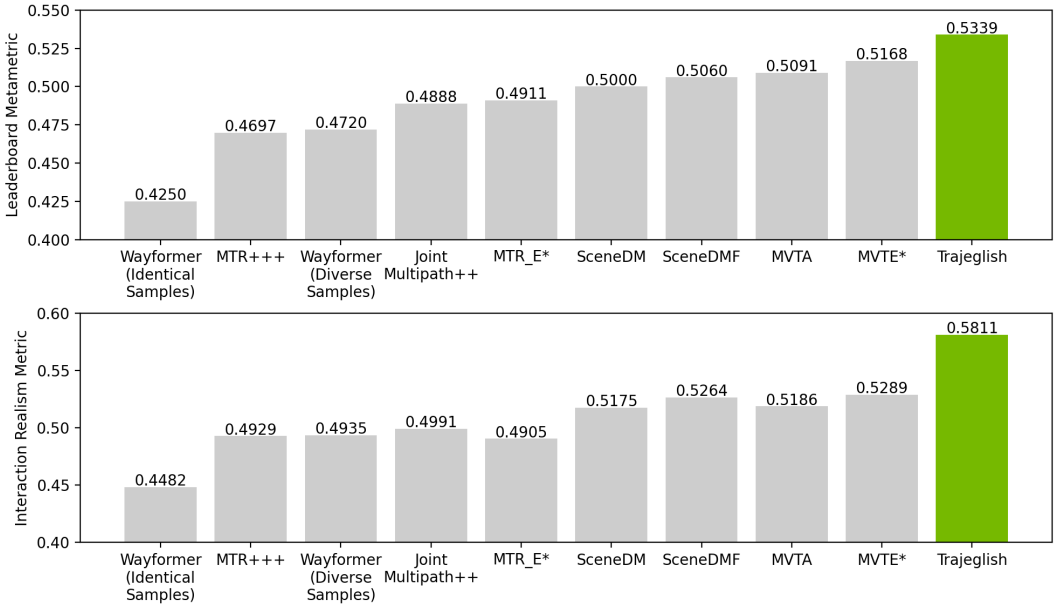

Figure 5: **Waymo Sim Agents Benchmark** Trajeglish sets a new state-of-the-art on the Waymo Sim Agents benchmark, outperforming the previous leader by 3.3% along the leaderboard metric as well as by 9.9% along the interaction metric. An asterisk indicates that the submission leverages ensembling.

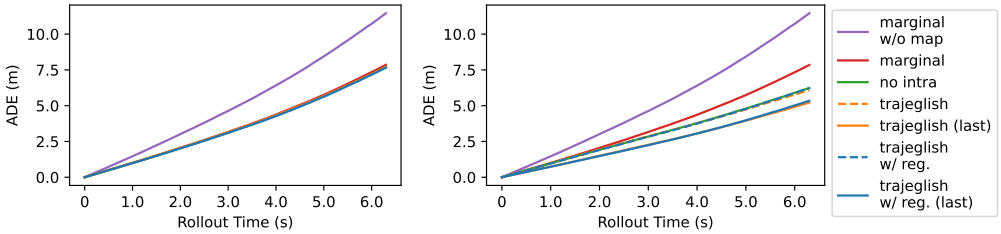

Figure 6: **Partial control ADE** Left shows the ADE for the vehicles selected for evaluation under partial control, but for rollouts where the agents are fully autonomous. Right shows the ADE for the same vehicles but with all other agents on replay. When agents controlled by trajeglish go first in the permutation order, they behave similarly to the no intra model. When they go last, utilize the intra-timestep information to produce interaction more similar to recorded logs, e.g. a lower ADE.

with an arbitrary number of agents for an arbitrary number of timesteps, as is required of submissions to the benchmark, can be found in Sec. A.2. Trajeglish is the top-performing submission, as shown in Fig. 5. Details on evaluation metrics for the benchmark can be found in [6].

We additionally compare performance in Fig. 6 across 5 variants of our model in a partial controllability setting. Both "trajeglish" and "trajeglish w/ reg." refer to our model described in Sec. 3, the latter with additional data augmentation of the input tokens. The "no intra" model is an important baseline designed to mimic the behavior of behavior cloning baselines used in Symphony [4] and "Imitation Is Not Enough" [5]. For this baseline, we keep the same architecture but adjust the masking strategy in the decoder to not attend to actions already chosen for the current timestep. The "marginal" baseline is designed to mimic the behavior of models such as Wayformer [7] and MultiPath++ [12] that are trained to model the distribution over single agent trajectories instead of multi-agent scene-consistent trajectories. For this baseline, we keep the same architecture but apply a mask to the decoder that enforces that the model can only attend to previous actions chosen by the current agent. Our final baseline is the same as the marginal baseline but without a map encoder.

5 Conclusion

In this work, we introduce a discrete autoregressive model of the interaction between roadusers. By improving the realism of self-driving simulators, we hope to enhance the safety of self-driving systems as they are increasingly deployed into the real world.

References

[1] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[2] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, jan 1986.

[3] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurélien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021.

[4] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation, 2022.

[5] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Rebecca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, Dragomir Anguelov, and Sergey Levine. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios, 2023.

[6] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nick Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, Brandyn White, and Dragomir Anguelov. The waymo open sim agents challenge, 2023.

[7] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple and efficient attention networks, 2022.

[8] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[9] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

[10] Davis Rempe, Jonah Philion, Leonidas J. Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. *CoRR*, abs/2112.05077, 2021.

[11] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors, 2021.

[12] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S. Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi-Pang Lam, Dragomir Anguelov, and Benjamin Sapp. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. *CoRR*, abs/2111.14973, 2021.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

A Appendix

A.1 Tokenization Baselines

A comparison of k-disks to other tokenization approaches is shown in Fig. 7. For (x, y, h)-grid, we independently discretize change in longitudinal and latitudinal position and change in heading, and treat the template set as the product of these three sets. For (x, y)-grid, we independently discretize only x and y, and for each possible pair, we find the motion transition with change in x and y that was closest to that translation and assign the corresponding heading. The issue with both grid-based methods is that they distribute templates evenly instead of focusing them in regions of the support where the most state transitions occur. The coarseness of the grid results in oscillation. Finally, we compare against k-means by running k-means many times on a collection of state transitions from WOMD. The main difference between k-disks and k-means is that k-means optimizes the euclidean distance between locations and does not take into account heading. Additionally, the distribution of state to state transitions collected from human data is different from the distribution of transitions observed when tokenization is performed iteratively across a full trajectory, so optimizing with k-means does not necessarily result in a better template set. The simple k-disk approach finds a template set of size 384 that discretizes WOMD vehicle, pedestrian, and cyclist trajectories to about 1 cm error in expectation.

Algorithm 1 Samples a candidate vocabulary of size N. The distance $d(x_0, x)$ measures the average corner distance between a box of length 1 meter and width 1 meter with state x_0 vs. state x.

procedure SAMPLEKDISKS(X, N, ϵ)
  $S \leftarrow \{\}$
  while len(S) $< N$ **do**
    $x_0 \sim X$
    $X \leftarrow \{x \in X \mid d(x_0, x) > \epsilon\}$
    $S \leftarrow S \cup \{x_0\}$
  return S

A.2 Extended Rollouts for WOMD Sim Agents

Our model is trained on multi-agent scenarios with up to $A = 24$ agents that are no more than $D = 60.0$ meters away from the origin for up to $T = 32$ future timesteps. In order to sample scenarios for evaluation on the WOMD sim agents benchmark, we require the ability to sample scenarios with an arbitrary number of agents arbitrarily far from each other for an arbitrary number of future timesteps. While it may become possible to train a high-performing model on 128-agent scenarios on larger datasets, we found that training our model on 24-agent scenarios and then applying the model to overlapping 24-agent subsets of the scene achieved top performance.

In detail, at a given timestep during sampling, we determine the 24-agent subsets with the following approach. First, we compute the 24-agent subset associated with picking each of the agents in the scene to be the center agent. We choose the subset associated with the agent labeled as the self-driving car to be the first chosen subset. Among the agents not included in a subset yet, we find which agent has a 24-agent subset associated to it with the maximum number of agents already included in a subset, and select that agent's subset next. We continue until all agents are included in at least one of the subsets.

Importantly, when choosing the order for agents within the subset, we place padding at the front of the order, followed by all agents that have already selected an action at the current timestep, followed by the remaining agents sorted by distance to the center agent. In keeping this order, we enable the agents to condition on the maximum amount of information possible. Additionally, this ordering guarantees that the self-driving car is always the first to select an action at each timestep, in accordance with the guidelines for the WOMD sim agents challenge [6].

To sample an arbitrarily long scenario, we have the option to sample up to $t < T = 32$ steps before computing new 24-agent subsets. Computing new subsets every timestep makes sure that the agents within a subset are always close to each other, but has the computational downside of requiring the transformer decoder key-value cache to be flushed at each timestep. For our submission, we compute the subsets at timesteps 10, 34 and 58.

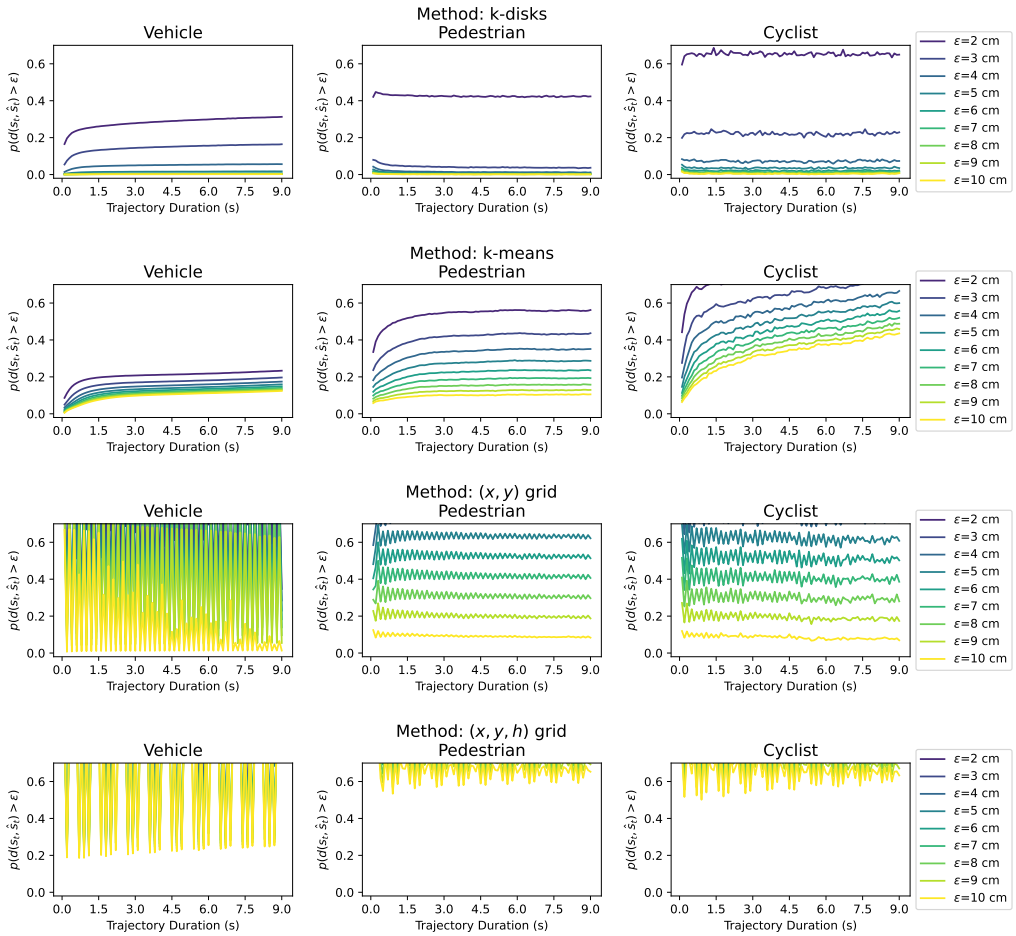

Figure 7: **Discretization error distribution** We plot the probability that the discretized trajectory is greater than 2 cm $\leq \epsilon \leq$ 10 cm away from the true trajectory as a function of trajectory length. Lower is better. Each row visualizes the error distribution for a different method, each with a vocabulary size of 384. We keep the y-axis the same across all plots. We note that k-means discretizes more trajectories to within 2 cm than k-disks, but does not improve past 5 cm, whereas k-disks is able to tokenize nearly all trajectories in WOMD to within 6 centimeters.

Finally, while the hyperparameters above are primarily optimized for computational efficiency and only affect performance slightly, we found that the temperature that we use when sampling from the model did significantly affect performance along the benchmark metrics. We use a temperature of 1.5 to achieve the results in Fig. 5. Our intuition for why larger temperatures resulted in higher leaderboard numbers is that the log likelihood used by the leaderboard metric penalizes lack of coverage much more strongly than lack of calibration, and higher temperature greatly improves the coverage.

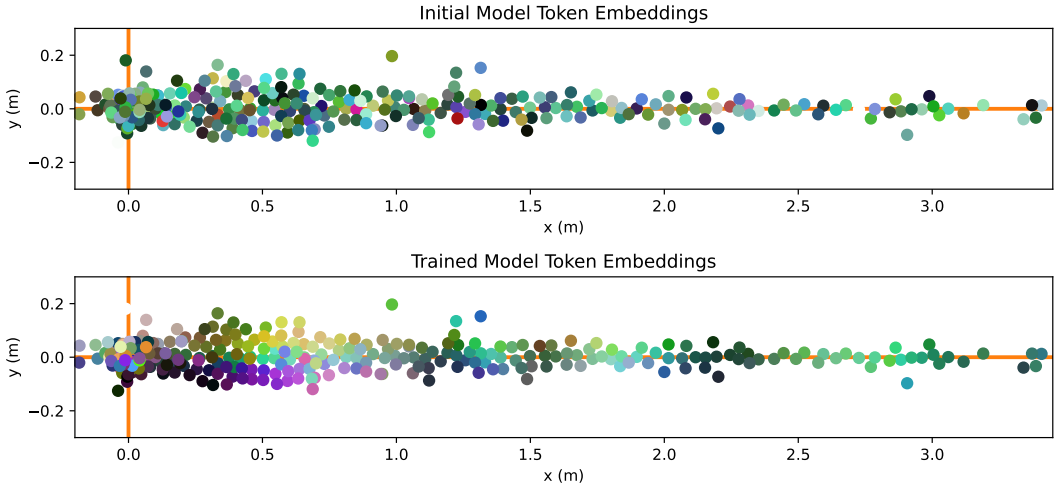

Figure 8: **Token Embedding Visualization** We run PCA on the model embeddings at initialization and at convergence, and plot the (x, y) location of each of the token templates using the top 3 principal component values of the corresponding embedding to determine the hue, lightness, and saturation of the point. The model learns that tokens that correspond to actions close together in euclidean space represent similar actions. Note that the heading of each action is not visualized, which also affects the embedding similarity. Additionally, the top 3 principal components include only 35% of the variance, explaining why some colors appear in multiple regions of the support.