Kargar, Eshagh; Kyrki, Ville

Vision Transformer for Learning Driving Policies in Complex and Dynamic Environments

Vision Transformer for Learning Driving Policies in Complex and Dynamic Environments

Eshagh Kargar, Ville Kyrki

Abstract— Driving in a complex and dynamic urban environment is a difficult task that requires a complex decision policy. In order to make informed decisions, one needs to gain an understanding of the long-range context and the importance of other vehicles. In this work, we propose to use Vision Transformer (ViT) to learn a driving policy in urban settings with birds-eye-view (BEV) input images. The ViT network learns the global context of the scene more effectively than with earlier proposed Convolutional Neural Networks (ConvNets). Furthermore, ViT's attention mechanism helps to learn an attention map for the scene which allows the ego car to determine which surrounding cars are important to its next decision. We demonstrate that a DQN agent with a ViT backbone outperforms baseline algorithms with ConvNet backbones pre-trained in various ways. In particular, the proposed method helps reinforcement learning algorithms to learn faster, with increased performance and less data than baselines.

I. INTRODUCTION

Driving in unstructured and dynamic urban environments is an arduous task. The decisions of drivers are influenced by many moving objects and the driver must be able to identify the importance and effect of each one on its behavior in order to act accordingly. Additionally, understanding map data and the global context of the environment around a car is essential for making decisions. As a result, autonomous driving in such complex environments remains a challenge for researchers and industry.

Reinforcement Learning (RL) is one of the emerging solutions to solve the decision-making problem in autonomous driving [1]–[4]. The use of RL in image-based environments has been widely explored [2], [5]–[7]; however, the environments in which RL is used are usually single-agent environments, or if there are multiple agents, they are considered as part of the environment. In comparison, policy and interaction learning is more challenging in environments with multiple agents interacting with each other.

Convolutional Neural Networks (ConvNets) are used widely as they are good at capturing local context. However, they are not able to capture long-range context in the scene [8] and lack an attention mechanism to handle settings with multiple interacting agents.

We propose to use Vision Transformers (ViTs) to learn a driving policy that can capture long-range context information and has an attention mechanism that allows the car to know where and how much to pay attention to. The attention is learned through interaction with other vehicles, without the

[1]Eshagh Kargar and Ville Kyrki are with School of Electrical Engineering, Aalto University, Finland. firstname.lastname@aalto.fi
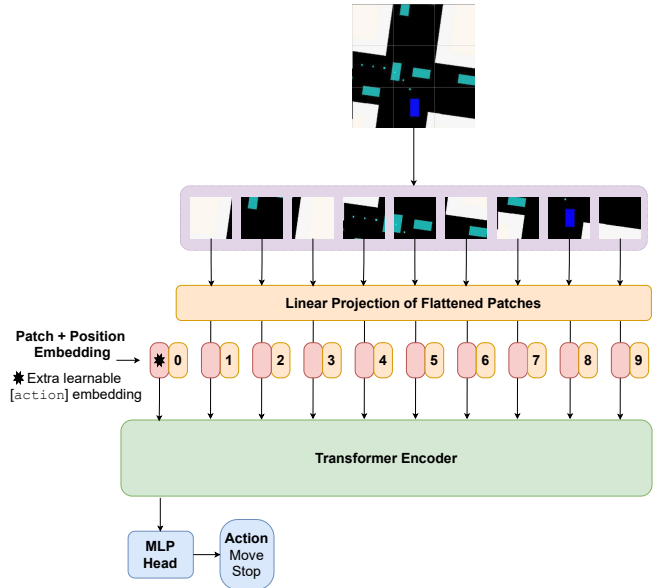
Fig. 1: The proposed architecture with a Vision Transformer as backbone. We split the input top-down view image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to output an action, we add an extra learnable "action" token to the sequence. Also, we use the Deep Q-Network algorithm to train the model. The architecture for the transformer encoder is shown in Fig. 2.

need for heuristics or supervision. The driving policy uses birds-eye-view (BEV) images as input and outputs a high-level discrete action.

The primary contributions of this work are: (a) using a Vision Transformer in Autonomous Driving with vehicles and map data rendered as a BEV image to learn a global context of the scene and where to pay attention to, (b) an "action" token which allows to use Vision Transformers for RL, and (c) evaluation in an urban driving scenario with multiple interacting vehicles, showing better performance, faster convergence, lower crash percentage, and meaningful attention map for multiple cars in the scene in situations such as four-way and T-intersections.

II. RELATED WORK

Learning a decision-making policy using reinforcement learning is a popular approach in autonomous driving [2]–[4], [7]. Although RL is typically data-hungry, Kendall et

al. [2] recently demonstrated the learning of a lane following task using real-world data gathered in a single day. Nevertheless, the complexity and multi-agent interaction of complex traffic environments require a method that takes into account both agent-agent and agent-environment interactions.

Autonomous driving systems also rely on HDMap data to make decisions. Map data can be rendered on a BEV image and encoded using ConvNets [7], [9]. While ConvNets are very good at capturing local patterns, the global context can not be captured very well. Graph neural networks were proposed by Jiyang et al. [8] to capture better global context and encode map and agent information.

In addition to the global context, which is crucial for decision-making, vehicles interacting with the ego car are also crucial. However, not all cars and not all parts of the scene are equally important. Therefore, an attention mechanism can be useful in paying more attention to important vehicles and critical parts of the map in the decision-making problem. Attention mechanism can be used with ConvNets to improve explainability and interpretability of end-to-end deep neural networks [10], [11]. A multi-task attention-aware network with a ConvNet backbone was proposed by Ishihara et al. [12] to learn a driving policy via conditional imitation learning. Chen et al. [13] designed a hierarchical Deep Reinforcement Learning algorithm and used an attention mechanism with ConvNets to learn faster, safer, and smoother lane change behaviors.

In addition to using of attention mechanisms with ConvNets, recently, Transformer architecture has become the de-facto standard for natural language processing tasks [14] and has been used in computer vision applications [15]. However, its application in RL remain limited and is not studied. Leurent et. al [16] used Transformers in an RL task for autonomous driving but with state-vector inputs. Transformers are also used in some works for the motion prediction task [17], [18].

In this work, we propose to use Vision Transformers as backbone for RL methods in dynamic environments with multiple interacting agents in which could be used to handle both (1) the dynamic and interactive nature of the environment via its attention mechanism, (2) and encoding the map and vehicles' information simultaneously, as well as capturing global context.

III. BACKGROUND

In this section, we will outline the required theoretical concepts behind the proposed method, including Reinforcement learning, Deep Q-Networks, and Transformers.

A. Reinforcement Learning

Reinforcement learning aims to find an optimal control policy $\pi : S \to A$ from states to actions that maximizes total expected future rewards

$$R(\pi) = E_\pi \left[\sum_t \gamma^t r(s_t, a_t) \right] \qquad (1)$$

where s_t, a_t, r, and γ are state at time t, action at time t, reward, and discount factor, respectively.

Value-function based RL solves the RL problem by determining an optimal value function $Q : S, A \to \mathbb{R}$ that describes the expected cumulative rewards when starting from a particular state and choosing a particular action

$$Q^*(s,a) = \max_\pi E \left[\sum_{t=0} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right]. \qquad (2)$$

Knowing the optimal value function, an optimal policy π^* can then be determined as $\pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$.

B. Deep Q-Network

It is usually impossible to determine the value function of continuous state spaces with unknown dynamics. Deep Q-Networks (DQNs) [5] are a successful RL algorithm that use a deep neural network $Q(s,a;\psi)$ to approximate the value function where ψ are the parameters of the neural network. DQN also stores past experiences in a replay buffer $D = \{(s,a,r,s')\}$ to accelerate and stabilize learning. To further stabilize the learning, DQN defines a target Q-network with parameters ψ' which are updated only every τ steps to the current ψ. To optimize ψ, the Q-learning loss

$$L_{DQN}(\psi) = E_{(s,a,s',r) \sim D} \left[\left(r + \gamma \max_{a'} Q(s',a';\psi') - Q(s,a,\psi) \right)^2 \right] \qquad (3)$$

is minimized for a uniform sample of transitions sampled from D.

C. Transformers and the Vision Transformer

Transformer architectures with attention mechanism are commonly used in sequence modeling tasks such as natural language processing (NLP). With the attention mechanism, dependencies can be modeled without regard for their distance in the input or output sequence. Moreover, Transformers' parallelization makes them more efficient than ConvNets and Recurrent Neural Networks, and they can train significantly faster. Additionally, they are very adept at identifying global dependencies between inputs and outputs [14].

Transformers have a structure of encoders and decoders. In this work we only use the encoder part, illustrated in Fig. 2. The transformer encoder consists of L stacked identical layers, each with a multi-head attention layer and a fully connected layer. Both sub-layers have residual connection and layer-normalization.

In computer vision, a variant called Vision Transformers [15] (ViT) has been used extensively. In order to feed a $2D$ image into the Transformer model, ViT [15] converts the input image into non-overlapping patches. The $2D$ patches are then flattened and mapped into a vector using a trainable linear projection. The patch embeddings are then concatenated with learnable 1D position embeddings, which preserve positional information, and fed into the Transformer Encoder.
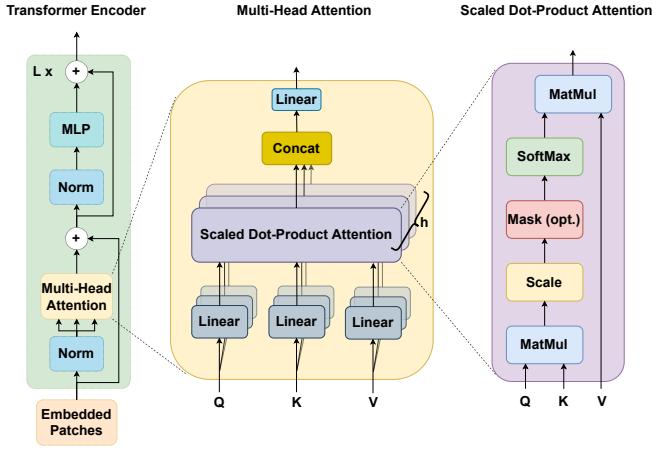
Fig. 2: (left) Transformer Encoder architecture with L stacked layers. (middle) Multi-Head Attention with several attention layers running in parallel. (right) Scaled Dot-Product Attention. The architecture is inspired by [14].

IV. METHOD

In this section, we first explain the problem setting and review the proposed idea to solve it. We then proceed with describing a detailed description of the proposed method including the Vision Transformer equations and the RL algorithm on top of that. Finally, we present the input representation.

A. Problem Setting and Solution Overview

The architecture of the autonomy system of self-driving cars is typically organized into the perception system and the decision-making system. In general, each of these systems is subdivided into numerous subsystems [19]. In this work, we assume that there is a perception system that detects all objects in the scene. Furthermore, we presume we have access to a high-definition map of the city.Then the map information and other vehicles' information are rendered on a BEV RGB image $x \in R^{H \times W \times C}$ which serves as the input to our method. The input is then processed by a Vision Transformer which generates a high level discrete action as output. The Vision Transformer is trained using the DQN algorithm. The structure is illustrated in Fig. 1.

B. Vision Transformer

In order to feed the input image $x \in R^{H \times W \times C}$ into the Vision Transformer, we reshape the BEV image into flattened patches $x_p \in R^{N \times (P^2 C)}$, where (H, W) is the input image resolution, C is the number of channels, (P, P) is the size of each patch, and $N = HW/P^2$ is the number of patches and the input sequence length.

In order to use Vision Transformer in RL, we propose to prepend a learnable "action" token ($z_0^0 = x_{action}$) to the input sequence of embedded patches. The idea is inspired by the original Vision Transformer work which used a "class" token [15] for classification tasks.

The Transformer encoder module is shown in Fig. 2 and its mathematical formulation can be written

$$\mathbf{z}_0 = [\mathbf{x}_{action}; \mathbf{x}_p^1 \mathbf{E}; \ldots; \mathbf{x}_p^N \mathbf{E}] + \mathbf{E}_{pos},$$
$$\mathbf{E} \in R^{(P^2.C) \times D}, \quad \mathbf{E}_{pos} \in R^{N+1} \times D \quad (4)$$

$$\mathbf{z}_l' = MSA(LN(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \quad (5)$$

$$\mathbf{z}_l = MLP(LN(\mathbf{z}_l')) + \mathbf{z}_l' \quad (6)$$

where *LN* is Layernorm layer [20], *MSA* is a Multi-head Self Attention block, and *MLP* block contains two linear layers with a GELU non-linearity.

The *MSA* block is

$$[\mathbf{Q}, \mathbf{K}, \mathbf{V}] = \mathbf{z} \mathbf{U}_{QKV}, \quad \mathbf{U}_{QKV} \in R^{D \times 3D_h} \quad (7)$$

$$A = softmax(\mathbf{Q}\mathbf{K}^T / \sqrt{D_h}), \quad A \in R^{N \times N} \quad (8)$$

$$SA(\mathbf{z}) = A\mathbf{v} \quad (9)$$

$$MSA(\mathbf{z}) = [SA_1(\mathbf{z}); SA_2(\mathbf{z}); \ldots; SA_k(\mathbf{z})]\mathbf{U}_{msa}, \quad \mathbf{U}_{msa} \in R^{k.D_h \times D} \quad (10)$$

The first output vector of the Transformer Encoder ($\mathbf{z}_L^0$) corresponding to the learnable "action" token is used to output the action:

$$\mathbf{a} = MLP(\mathbf{z}_L^0). \quad (11)$$

The whole network can be trained using any RL algorithm with discrete action space. Here we consider only longitudinal control to show the effectiveness of the approach and use DQN to train the network weights. The whole proposed method is depicted in Fig. 1.

C. Input Bird-Eye View Image

One common way to consider map information in the decision making process of autonomous vehicles is to render them on a BEV image. It is also possible to render all the agents in the scene such as vehicles, pedestrians, and bicycles on the same image [9]. The map information and other agents' data come from other modules in a self-driving car.

The rendered data on the BEV input image are as follows:

1) **HDMap**: The HD-Map information includes drivable areas, road boundaries, and curbs. It is also possible to render more information such as lane lines and center of lane lines. Due to our intention to do longitudinal control only, this information is not used here. However, it is extremely useful for lateral control and steering angle generation.

2) **Route**: The planned route to the destination is assumed to come from an external planning system such as a standard vehicle navigator.

3) **Pose of other vehicles**: The current position, orientation, and size of other cars are rendered on the BEV image. This information can come from the perception module with object detection algorithms.

4) **Pose of the ego vehicle**: The current position, orientation, and size of the ego vehicle are also rendered on the BEV image. This information can come from the Localization module.

The generated BEV image will be used as input to the ViT network as shown in Fig. 1.

V. Experiments

In this section, we first introduce the simulator we used to do our experiments. Then we study the following questions:

1) Do ViT-based DQNs improve the performance compared to ConvNet-based DQNs?
2) How the proposed ViT-based DQN method performs compared to state-of-the-art RL algorithms?
3) How ViT backbone helps to pay attention to different parts of the scene and other cars in the environment and how the ego car reacts to them?
4) What is the effect of patch size in the Vision Transformer?

A. Simulation

In order to test our proposed method in a dynamic and complex setting with multiple interactive agents, we use Fluids [21] which is a self-driving car simulator. Fig. 3 shows the simulation environment.
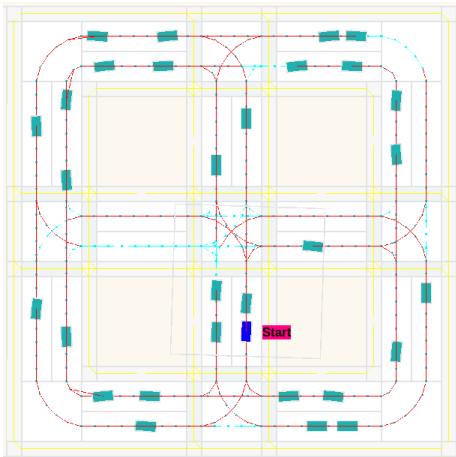

Fig. 3: Fluids simulation environment.

Fluids simulator supports intersection, left and right turn scenarios with and without traffic light. It also has some capabilities to add pedestrians and enable traffic lights for cars and pedestrians. In this work, in order to study the interaction between cars in a complex scenario, we spawned 30 cars randomly in the city. We also spawned the ego car, the blue one, before an unsignalized intersection. The destination for the ego car is a random point in the city. Also, we disabled all traffic lights and did not spawn any pedestrian in the city. The state space is a 80×80 top-down view image which is shown in Fig. 1 and the actions are just two commands, move with a constant velocity or stop, to control the longitudinal movement. For lateral control we used a PID controller to generate steering angle.

The reward function is a sum of two terms: collision (r_c) and travelled distance (r_d),

$$r = r_c + r_d \tag{12}$$

where r_c is -100 if a collision happens and 0 otherwise, and r_d is the travelled distance from starting point in meters. The first term encourages the car to avoid collision, and the second term pushes it to go forward and prevents it from stopping.

B. Implementation Details

Our proposed method has two parts: a ViT backbone and an MLP head for action outputs. For the ViT backbone, we used a ViT-small version used in [22] with 12 Transformer Encoder blocks and 6 heads in its multi-head attention layers. The dimension of projected layers and the output of Transformer Encoder layers is 384. The MLP head consists of two linear layers of size 64 and 2 for two actions with GELU non-linearity.

C. Ablation Study

In order to evaluate the performance of the proposed ideas and answer questions 1 and 4, we used DQN algorithm with various backbones to encode the input. As the input is an image and needs to be encoded, we considered several ConvNets, pre-trained in various ways, as the backbone for the DQN agent and fine-tuned them on our task. We selected ResNet-50 because of its similarity with ViT-small along several axes: number of parameters, throughput, and supervised performance on ImageNet [22]. We also considered ResNet-18 which is a smaller version of ResNet-50 as a baseline to see the effect of model size.

We show two versions of our proposed method in the comparison:

- **ViT-scratch-p2,4,5,8,10**: ViT-small with $pathc\_size = 2, 4, 5, 8, 10$ and without pre-training phase. This version is presented to evaluate the effect of the pre-training stage and patch size in the proposed method's performance.
- **ViT-DINO**: ViT-small with $patch\_size = 8$ pre-trained on ImageNet dataset using a self-supervised algorithm called DINO [22]. However $patch\_size = 4$ outperforms other versions, there was no available pre-trained model for that and we used $patch\_size = 8$.

These are compared against the following ablations with ConvNet backbones and DQN algorithm are as follows:

- **ResNet18-ImageNet**: ResNet-18 pre-trained on ImageNet in a supervised learning classification task.
- **ResNet50-ImageNet**: ResNet-50 pre-trained on ImageNet in a supervised learning classification task.
- **ResNet18-VAE**: ResNet-18 pre-trained on ImageNet in a supervised learning classification task and then fine-tuned using a VAE (used as the encoder) on a collected dataset of size 50k from Fluids environment to reconstruct the input scene [7].
- **ResNet50-VAE**: ResNet-50 pre-trained on ImageNet and then fine-tuned using a VAE (used as the encoder) on a collected dataset of size 50k from Fluids environment to reconstruct the input scene [7].
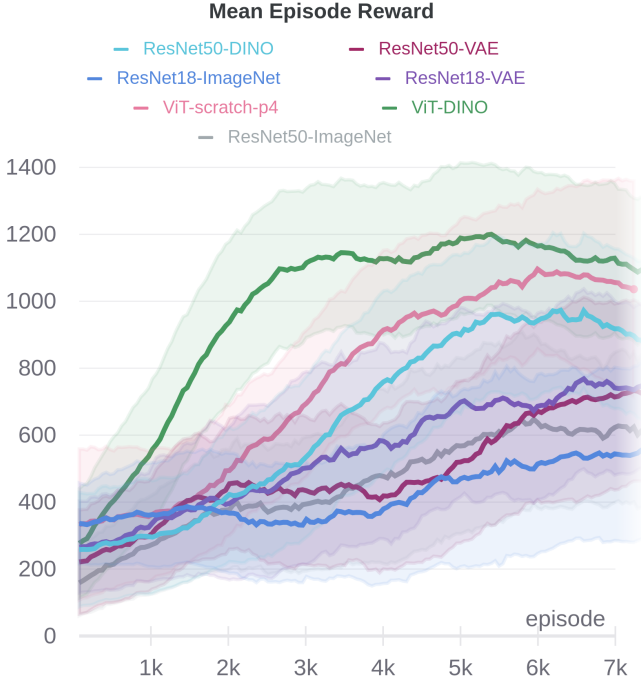
Fig. 4: Comparison of our methods against DQN baselines in an unsignalized 4-way intersection.

- **ResNet50-DINO**: ResNet-50 pre-trained on ImageNet dataset using a self-supervised algorithm called DINO [22].

The whole network, the backbone and the MLP head, is then fine-tuned using DQN algorithm with the same hyperparameters for all cases.

To evaluate the performance of the proposed method with ablations, we report two types of results:

a) Evaluation reward during the training phase: In each training epoch, the trained model is evaluated for two episodes and the average achieved reward is reported.

b) Crash Percentage: After the training is finished, the trained model for each method is used to be run in the simulation for 200 episodes and number of crashes are reported. In order to have a fair comparison, the same set of random seeds are used.

Fig. 4 shows the evaluation reward during the training phase. We just reported the mean episode reward for ViT-scratch-p4, because it performs better than other patch sizes and also to keep the plots clean. ViT-DINO has the best mean evaluation reward of around 1200. The next best performance is for ViT-scratch-p4 which has a maximum mean evaluation reward of 1100. ResNet-DINO follows these two ViT-based models. As can be seen in the plots, ViT-DINO reaches the performance of ViT-scratch-p4 and ResNet-DINO with less than half and one-third of their training data, respectively. This shows the effect of pre-training and then fine-tuning on a down-stream task with much less data. It also indicates the impact of the ViT backbone even without a pre-training phase. The performance can improve if we pre-train the model using DINO on a large driving dataset, which is
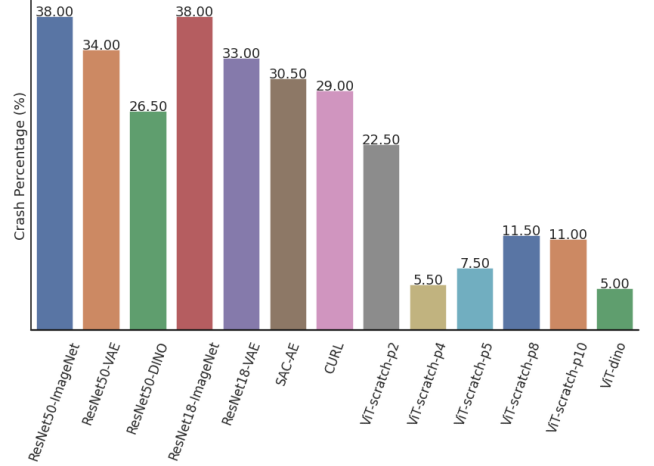

Fig. 5: Crash Percentage in 200 runs.

available to most of self-driving car companies.

Other methods with ConvNet backbones are after these ViT-based models. ResNet50-DINO outperforms ResNet50-ImageNet which shows the effect of the pre-training method and better learned representation. We can also see a small improvement of ResNet18-VAE and ResNet50-VAE over ResNet18-ImageNet and ResNet50-ImageNet that demonstrates training of the ResNet models as encoder in a VAE can lead to a better representation and improve the performance. It can also be seen that ViT-scratch-p4 can reach the best performance of ConvNet methods with almost half of their training data which shows the effect of ViT backbone even without pre-training phase.

In addition, as shown in Fig. 5, ViT-DINO has the best performance compared to other methods with the crash percentage of 5%. For ViT backbones trained from scratch, as we decrease the patch size, we see a small improvement in crash percentage, but for smaller patch sizes than 4, the crash percentage increases again. The next best performance after ViT-DINO is for ViT-scratch-p4 with the crash percentage of 5.5%. In general, the performance of all ViTs with different patch sizes are better than ConvNet methods. However, ResNet50-DINO has a lower crash percentage compared to other ConvNet-based methods which indicates the effect of the pre-training method.

The results represent the performance improvement caused by using ViT compared to ConvNet backbones, whether it is pre-trained or not.

Videos of example driving using the learned model, DQN agent with ViT-DINO backbone, can be found at *https://youtu.be/-9TBLZEeCDs*.

D. Comparison to State of the Art

In order to answer question 2, we compared the proposed approach to two other state-of-the-art RL algorithms capable of handling image inputs, SAC-AE [23] and CURL [24]. These two methods use ConvNets in their architecture and output a continuous action. The Fluids simulator's environment was modified to receive continuous velocity commands
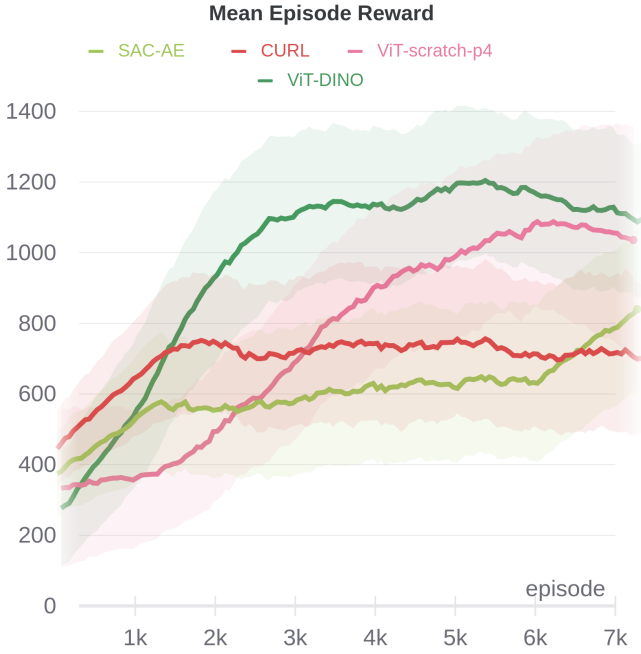
Fig. 6: Comparison of our methods against state-of-the-art baselines in an unsignalized 4-way intersection.
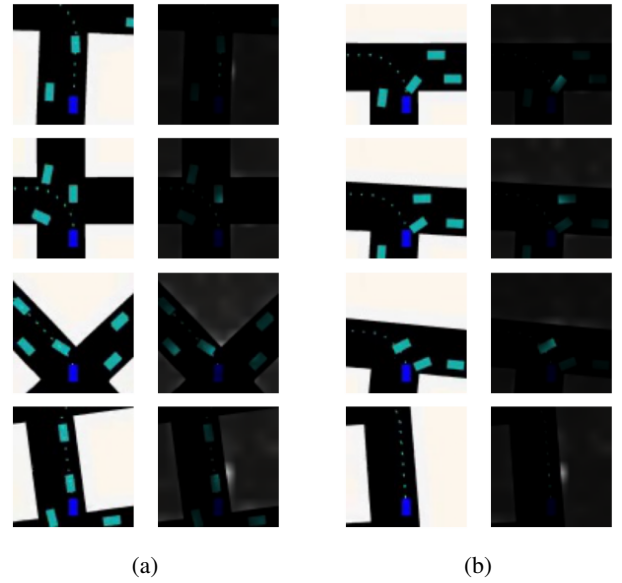

Fig. 7: Attention map visualization for two scenarios. (a) unprotected left-turn in a four-way intersection, (b) unprotected left-turn in a T-intersection. In each scenario, the left column shows the raw BEV image. For better visual understanding, the right column depicts the learned attention map as it is rendered on the BEV image. High attention is indicated by white areas in the images.

in order to support these methods. We used the hyperparameters from the main papers for these two baselines. Fig. 6 depicts the superior performance of the proposed method, trained from scratch or pre-trained with DINO, compared to SAC-AE and CURL.

Furthermore, the crash percentage reported in Fig. 5 indicates that the proposed method performs significantly safer. This shows the effect of the attention mechanism and the improved learning of the global context in ViTs. CURL, SAC-AE, and any other RL algorithm can easily use the ViT instead of ConvNet as their backbone.

E. Visualize the Learned Attention Map in ViT

In order to understand the attention map in ViT models and answer question 3, we present visualizations of it in two unprotected left-turns, which are among the most challenging in autonomous driving. Fig. 7 illustrates these two examples. ViT-DINO, the best model from the previous section, was used for this visualization.

In the 4-way intersection scenario, Fig. 7a, the ego car does not pay attention to any other vehicle in the first image. In the second image, as it approaches the intersection, it begins to pay attention to the car in front of it and a little bit to the car coming from the top. The third image shows that the ego car concentrates more on what is in front of it and its left side, which are more critical than other cars in that state. At the end, when the ego car passes the intersection, it just pays attention to its lead vehicle and follows it. Attention values in each image are relative and should be interpreted independently.

In the T-intersection scenario shown in Fig. 7b, the ego car pays attention to its lead vehicle first. Then, in the second and

third images, it focuses on the upcoming car, which wants to take a left turn and is more critical than other cars for the left-turn decision of the ego car. The ego car waits for the upcoming car to pass, then makes the left turn and continues on its way in the last image.

VI. CONCLUSIONS

Reinforcement learning presents an important avenue to handle the complexity of situations in autonomous driving with multiple interacting cars and map information that need to be encoded and take into account in decision-making, but its applicability is significantly hindered by the information encoding method. In this paper, we proposed a framework to use Vision Transformers in RL settings with multiple interacting agents. The map and vehicles' information can be rendered on a BEV images. Vision Transformers can encode this information while learning the global context and an attention map to identify which car in the environment is more important than others to the ego car. Experimental comparison against ablations and baselines showed that the use of Vision Transformer improves policy learning in several ways: the policy quality is better, the learning converges faster, and the learned policy is safer and has a lower crash percentage. The proposed approach was tested using DQN algorithm for longitudinal control and can be easily extended to lateral control too. It is also possible to replace DQN with other algorithms with discrete action space.

Using the proposed approach in real traffic environments benefits from a pre-training step on real world datasets. The

pre-training improves the quality of the policy and reduces the need for real world interaction in the reinforcement learning phase. Real world also has many constraints such as traffic rules that have to be taken into account. This may pose a challenge for methods based on learning, also because learning methods may lack interpretability, even if that can be alleviated partly by techniques such as the attention mechanism proposed in this paper. Thus, integration of logic and knowledge with learning based systems remains the greatest open issue in building robust and generalizable decision making for autonomous cars.

ACKNOWLEDGMENT

REFERENCES

[1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[2] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8248–8254.

[3] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, "Driving in dense traffic with model-free reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5385–5392.

[4] X. Ma, J. Li, M. J. Kochenderfer, D. Isele, and K. Fujimura, "Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships," *arXiv preprint arXiv:2011.04251*, 2020.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *arXiv preprint arXiv:2004.14990*, 2020.

[7] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 2019, pp. 2765–2771.

[8] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.

[9] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[10] L. Cultrera, L. Seidenari, F. Becattini, P. Pala, and A. Del Bimbo, "Explaining autonomous driving by learning end-to-end visual attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 340–341.

[11] J. Kim and J. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2942–2950.

[12] K. Ishihara, A. Kanervisto, J. Miura, and V. Hautamaki, "Multi-task learning with attention for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2902–2911.

[13] Y. Chen, C. Dong, P. Palanisamy, P. Mudalige, K. Muelling, and J. M. Dolan, "Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[16] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," *arXiv preprint arXiv:1911.12250*, 2019.

[17] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7577–7586.

[18] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 10 335–10 342.

[19] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

[20] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[21] H. Zhao, A. Cui, S. A. Cullen, B. Paden, M. Laskey, and K. Goldberg, "Fluids: A first-order lightweight urban intersection driving simulator," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2018, pp. 697–704.

[22] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," *arXiv preprint arXiv:2104.14294*, 2021.

[23] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," *arXiv preprint arXiv:1910.01741*, 2019.

[24] A. Srinivas, M. Laskin, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," *arXiv preprint arXiv:2004.04136*, 2020.