

Learning Sampling Distribution and Safety Filter for Autonomous Driving with VQ-VAE and Differentiable Optimization

Simon Idoko, Basant Sharma, Arun Kumar Singh

Abstract—Sampling trajectories from a distribution followed by ranking them based on a specified cost function is a common approach in autonomous driving. Typically, the sampling distribution is hand-crafted (e.g a Gaussian, or a grid). Recently, there have been efforts towards learning the sampling distribution through generative models such as Conditional Variational Autoencoder (CVAE). However, these approaches fail to capture the multi-modality of the driving behaviour due to the Gaussian latent prior of the CVAE. Thus, in this paper, we re-imagine the distribution learning through vector quantized variational autoencoder (VQ-VAE), whose discrete latent-space is well equipped to capture multi-modal sampling distribution. The VQ-VAE is trained with demonstration data of optimal trajectories. We further propose a differentiable optimization based safety filter to minimally correct the VQ-VAE sampled trajectories to ensure collision avoidance. We use backpropagation through the optimization layers in a self-supervised learning set-up to learn good initialization and optimal parameters of the safety filter. We perform extensive comparisons with state-of-the-art CVAE-based baseline in dense and aggressive traffic scenarios and show a reduction of up to 12 times in collision-rate while being competitive in driving speeds.

I. INTRODUCTION

Trajectory sampling is a conceptually simple approach that has found widespread adoption in autonomous driving community [1], [2]. As the name suggests, the process involves sampling trajectories from a distribution and evaluating their utility based on a specified cost function. The least cost trajectory from the sample is chosen for execution. The sampling itself can be encoded in the form of interpretable parameters. For example, instead of trajectories, we can sample set-points for velocity and lateral offset for the vehicle and pass it through a simple quadratic program to obtain the resulting trajectory samples [3]. This encoding has the advantage of assigning some physical interpretation to each sampled trajectory.

Irrespective of the exact strategy, the underlying sampling distribution is often hand-crafted, e.g in the form of a Gaussian or a pre-fixed grid [4], [2]. Recently, there has been efforts towards learning the sampling distribution [3], [5]. The distribution itself is represented in the form of a Conditional Variational AutoEncoder (CVAE) [6] and can be conditioned on the environment observations. These cited approaches have shown superior performance compared to

All authors are with the University of Tartu. This research was in part supported by financed by European Social Fund via ICT program measure, grants PSG753 from Estonian Research Council and collaboration project LLTAT21278 with Bolt Technologies. Our code is available at <https://github.com/cisimon7/VQOptMain>. Emails: cisimon7@gmail.com, aks1812@gmail.com

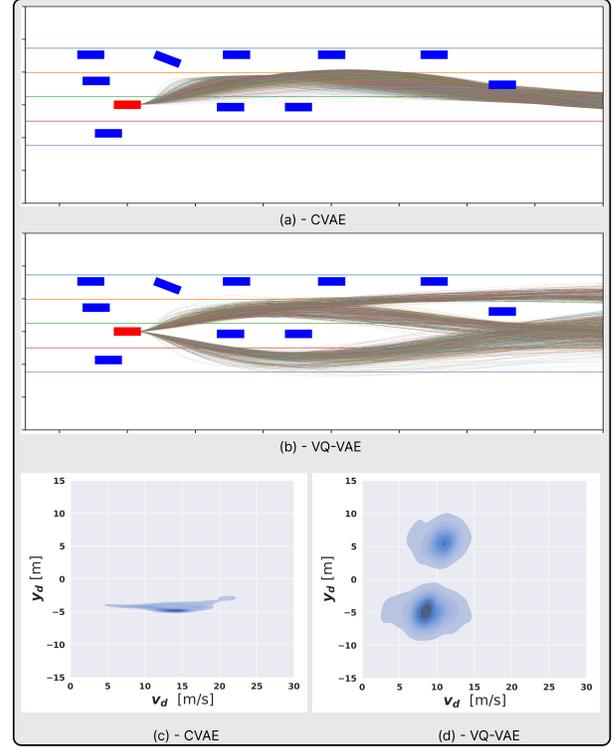


Fig. 1. Comparison between trajectory distribution sampled from CVAE (a) and VQ-VAE (b). As can be seen, VQ-VAE shows higher diversity and multi-modality in the sampled trajectories. This can be further validated by Fig. (c) and (d) which show the Kernel Density Estimation plots of the forward velocity (v_d) and lateral-offset (y_d) associated with each sampled trajectory.

hand-crafted approaches. In this paper, we take this line of research further to solve one of the fundamental bottlenecks of CVAE, namely the posterior collapse. Intuitively, it refers to the inadequacy of CVAE in capturing multi-modal distributions, similar to what is commonly encountered in autonomous driving. This in turn, can be attributed to the Gaussian latent prior of the CVAE.

In this paper, we present a novel improvement based on Vector-Quantized Variational Autoencoder (VQ-VAE) [7], whose discrete latent space is better suited to capturing multi-modal distribution. For example, the discrete latent space can capture the different homotopies of the optimal driving trajectories (see Fig.1). Our key algorithmic contributions along with their benefits are summarized below

Algorithmic Contribution: We train a VQ-VAE using multi-modal demonstration of optimal trajectories to learn the underlying discrete latent space. We also train a PixelCNN [8] to sample from the learned latent space while

conditioning it on the observation. We embed a differentiable QP within the VQ-VAE decoder to generate an intermediate interpretable representation of the each sampled trajectory from the VQ-VAE in terms of velocity and lateral-offset setpoints.

We show that while VQ-VAE based trajectory sampling is enough for collision-free navigation in low density traffic, more complicated scenarios require explicit consideration of collision avoidance and lane boundary constraints. With this motivation, we propose an optimization-based safety filter modeled in terms of barrier function [9]. The parameters of the filter along with good initialization for the underlying optimizer is learned in a self-supervised manner. We show how reformulations of barrier constraints can be exploited to simplify the differentiation through the safety filter optimization layer.

State-of-the-Art Performance: We compare our approach with recent CVAE-based baseline presented in [5] which has shown impressive performance in dense traffic scenarios. We show that our VQ-VAE pipeline achieves up to 12 times reduction in collision-rate over [5], while being competitive in achieved forward velocity. We further show that the discrete latent prior of VQ-VAE ensures superior diversity in sampled trajectories. As a result, a near-perfect performance is achieved in low traffic scenarios even without the computationally demanding safety layer. Finally, we show that our approach also maintains good performance even at reduced computation and sampling budget.

II. MATHEMATICAL PRELIMINARIES

Symbols and Notation: Scalars will be denoted by normal font lowercase letters, vectors by bold font lowercase letters, and matrices by uppercase bold font letters. The superscript T will indicate the transpose operation applied to either a matrix or a vector.

A. Frenet Frame and Trajectory Parametrization

We assume access to a lane center-line which allows us to perform motion planning in the so-called Frenet frame. In this set-up, the X and Y axes of the Frenet-frame are aligned with the longitudinal and lateral motion of the ego-vehicle. We parametrize the positional space $(x[k], y[k])$ of the ego-vehicle in the Frenet frame at any time instant k in terms of polynomials:

$$[x[0], x[1], \dots, x[k]] = \mathbf{W}\mathbf{c}_x, [y[0], y[1], \dots, y[k]] = \mathbf{W}\mathbf{c}_y, \quad (1)$$

where, \mathbf{W} is a matrix formed with time-dependent polynomial basis functions and $(\mathbf{c}_x, \mathbf{c}_y)$ are the coefficients of the polynomial. We can also express the derivatives in terms of $\dot{\mathbf{W}}, \ddot{\mathbf{W}}$.

B. Trajectory Sampling Via Setpoints

Instead of trajectories, we follow the intuition of [10], [5] and sample set-points for forward velocity and lateral-offset from the center-line. These are converted to a trajectory distribution by solving the following optimization problem.

$$\min \sum_k c_s + c_l + c_v \quad (2a)$$

$$(x^{(o)}[0], y^{(o)}[0]) = \mathbf{b}_0, (x^{(o)}[n], y^{(o)}[n]) = \mathbf{b}_f \quad (2b)$$

$$c_s(\ddot{x}[k], \ddot{y}[k]) = \ddot{x}[k]^2 + \ddot{y}[k]^2 \quad (3a)$$

$$c_l(\dot{y}[k], \dot{y}[k]) = (\dot{y}[k] - \kappa_p(y[k] - y_d) - \kappa_v \dot{y}[k])^2 \quad (3b)$$

$$c_v(\dot{x}[k], \dot{x}[k]) = (\dot{x}[k] - \kappa_p(\dot{x}[k] - v_d))^2 \quad (3c)$$

The first term $c_s(\cdot)$ in the cost function (2a) ensures smoothness in the planned trajectory by penalizing high accelerations at discrete time instants. The last two terms ($c_l(\cdot), c_v(\cdot)$) model the tracking of lateral offset (y_d) and forward velocity (v_d) set-points respectively and is inspired from works like [10]. For the former, we define a Proportional Derivative (PD) like tracking with gain (κ_p, κ_v) . It induces lateral accelerations that will make the ego-vehicle converge to the y_d . The derivative terms in c_l minimize oscillations while converging to the desired lateral offset. For velocity tracking, we only use a proportional term. Equality constraints (2b) ensures boundary conditions on the o^{th} derivative of the planned trajectory. We use $o = \{0, 1, 2\}$ in our formulation.

Optimization (2a)-(2b) can be converted into the following QP using the trajectory parameterization of (1).

$$\xi^* = \arg \min_{\xi} \frac{1}{2} \xi^T \mathbf{Q} \xi + \mathbf{q}^T(\mathbf{p}) \xi, \quad (4a)$$

$$\mathbf{A} \xi = \mathbf{b} \quad (4b)$$

where $\xi = (\mathbf{c}_x, \mathbf{c}_y)$ and $\mathbf{p} = (v_d, y_d)$.

III. MAIN RESULTS

The complete pipeline of our model is illustrated in fig 2. Our goal is to generate a distribution of safe trajectories corresponding to a given observation \mathcal{O} . This is accomplished through the different blocks as show in fig 2. Firstly, we sample from a learned VQ-VAE to obtain a distribution of velocity and lateral-offset set points (\mathbf{p}) , and the associated trajectories. The sampled trajectories are then processed by a learned safety filter block, which ensures that they conform to collision and lane boundary constraints. Following this safety filtration, the trajectories are evaluated by a cost function to identify the optimal (least-cost) trajectory. The VQ-VAE and the safety filter are trained separately in a heirarchical manner. We describe the main building blocks next.

A. Learning with VQ-VAE and Differentiable QP block

Our VQ-VAE pipeline is shown in Fig.3 and adapts [7] for trajectory inputs. The expert trajectories are mapped by an encoder to a continuous latent space $\mathbf{Z}_e \in \mathbb{R}^{L \times D}$, where L is the number of latent vectors and D is their individual dimensionality. For ease of notation, we represent the i^{th} latent vector (row) of \mathbf{Z}_e as $\mathbf{z}_{e,i}$. The core idea of VQ-VAE lies in the discretization of the latent space. i.e mapping continuous \mathbf{Z}_e to discrete \mathbf{Z}_q . To this end, we introduce a latent embedding space $\mathbf{E} \in \mathbb{R}^{K \times D}$ consisting of K discrete vectors \mathbf{e}_j with dimension D . Subsequently, we define the

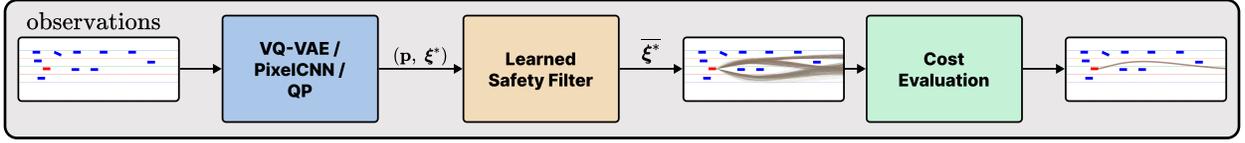


Fig. 2. Our overall pipeline that consists of sampling from a learned VQ-VAE and passing the samples to a learned safety filter. This is followed by cost evaluation on the trajectory samples and the selection of the best trajectory.

i^{th} latent vector (row) of \mathbf{Z}_q as the j^{th} \mathbf{e}_j vector closest to $\mathbf{z}_{e,i}$. This assignment is performed by the nearest neighbour optimization problem defined in (5)

$$\mathbf{z}_{e,i} = \mathbf{e}_r, \text{ where } r = \arg \min_j \|\mathbf{z}_{e,i} - \mathbf{e}_j\|_2^2$$

$$\text{for } i \in \{1, 2, \dots, L\} \quad (5)$$

The decoder of VQ-VAE takes in \mathbf{Z}_q and produces set-points \mathbf{p} , which is then passed through a QP layer defined by (4a)-(4b) to obtain the reconstruction of the expert trajectory ξ^* . The VQ-VAE is trained using the loss function defined in (6) which has three components. The first part is the reconstruction loss which makes the entire pipeline learn to reconstruct the input expert trajectory. The typical backpropagation cannot trace the gradient through the non-differentiable nearest neighbour assignment (5), which in turn poses problem in jointly training the encoder, decoder and the embedding space vectors. To counter this, straight-through gradient estimation method proposed in [7] is used, which in turn necessitates adding the last two terms in the loss function (6). The second term in the loss function employs the l_2 error to guide the movement of embedding vectors \mathbf{e}_i towards the encoder outputs \mathbf{z}_e , thereby updating the embedding space vectors parameters. And the last term seeks to make the encoder commit to the embedding space vectors. In other words, it prevents the encoder output to keep getting assigned to a different nearest neighbour in each forward pass.

$$\mathcal{L}_{VQVAE} = \|\overline{\mathbf{W}}\xi^* - \tau_e\|_2^2 + \|\text{sg}[\mathbf{Z}_e] - \mathbf{E}\|_2^2$$

$$+ \beta \|\mathbf{Z}_e(x) - \text{sg}[\mathbf{E}]\|_2^2 \quad (6)$$

B. Conditional sampling of the discrete latent space using PixelCNN

Let \mathbf{h}_q denote a vector wherein each element corresponds to the codebook index of the vector $\mathbf{z}_{q,i}$ in \mathbf{Z}_q , with $i \in 1, 2, \dots, L$. We want to generate different samples of \mathbf{h}_q such that they can be mapped to the learned latent space of VQ-VAE, \mathbf{Z}_q . These in turn, can be mapped to velocity and lateral offset setpoints (\mathbf{p}) and the associated trajectories through the learned decoder. Moreover, the \mathbf{h}_q generation should be conditioned on the observations \mathcal{O} . To this end, we adapt the PixelCNN architecture proposed in [8].

Our pipeline is shown in fig 4. We pass observation vector \mathcal{O} through CNN layers to get feature vector $\bar{\mathbf{o}}$. During training, this representation, together with the ground truth \mathbf{h}_q obtain from the trained VQ-VAE model is passed to the PixelCNN. The model then predicts the conditional

distribution of the codebook indices in an autoregressive manner using a set of masked CNN layers as described in [8]. We train the generated distribution $p(\mathbf{h}_q|\mathcal{O})$ using cross-entropy loss with the ground truth \mathbf{h}_q .

During the inferencing phase, we initialize \mathbf{h}_q to zeros and recursively generate the element of \mathbf{h}_q conditioned on the observation using the trained PixelCNN. This generates a multinomial distribution for each generated \mathbf{h}_q element. We sample from this distribution at each step to construct the corresponding \mathbf{h}_q vector which then passes through the embedding layer to get \mathbf{Z}_q .

C. Learnable Safety Filter

Our safety filter is defined through the following optimization problem.

$$\bar{\xi}_j^* = \arg \min_{\bar{\xi}_j^*} \frac{1}{2} \|\bar{\xi}_j^* - \xi_j^*\|_2^2 \quad (7)$$

$$\mathbf{A}\bar{\xi}_j^* = \mathbf{b}, \quad \mathbf{g}(\bar{\xi}_j^*, \gamma) \leq \mathbf{0} \quad (8)$$

As can be seen, the safety filter takes in the j^{th} trajectory generated by the VQ-VAE and projects them onto the constraint set. The equality constraints ensure that the trajectories satisfy the initial and terminal states. The inequality constraints model collision avoidance, lane-boundary constraints along with velocity and acceleration bounds. We defined the algebraic form of these entities in Appendix. But we highlight that the collision avoidance and lane-boundary constraints are designed using the barrier function approach of [9]. In this context, γ represents the parameter of the barrier function.

The safety filter has one explicit learnable parameter namely γ . Additionally, we also aim to learn good initialization for optimization problem underlying the safety filter.

Training Pipeline: The overall architecture of safety filter is shown in Fig.5. A multi-layer perceptron (MLP) block takes in the observations \mathcal{O} along with the trajectory generated by the VQ-VAE and outputs γ along with initialization λ (defined in Appendix VII), ${}^0\bar{\xi}^*$ for the optimization solver. These predictions are then passed onto the safety filter. The MLP is trained in a self-supervised manner using the loss function defined in (9). The first term in the loss function minimizes the projection residual and is designed to enforce minimum possible correction to the input trajectory. The second term $c(\bar{\xi}^*)$ minimizes the constraint residuals of the safety filter. The constant s is used to trade-off between the two loss terms.

$$\mathcal{L}_{mlp} = \|\xi^* - \bar{\xi}^*\|_2^2 + s \sum_k c(\bar{\xi}^*) \quad (9)$$

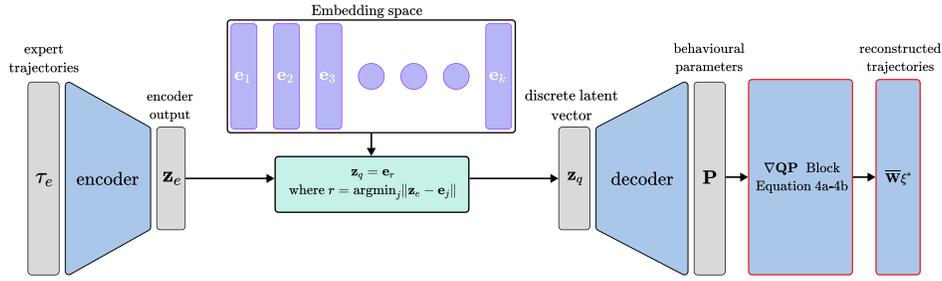


Fig. 3. VQVAE with a differentiable QP block for learning discrete latent prior over the forward velocity v_d , lateral offset y_d and the associated trajectory.

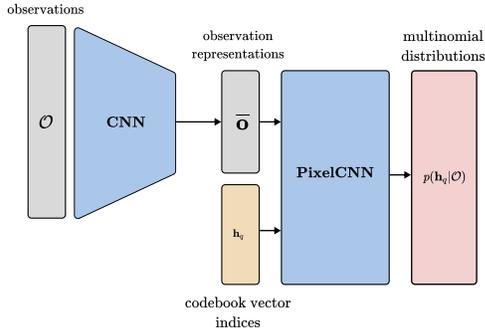


Fig. 4. The auto-regressive conditional PixelCNN architecture that is used to learn how to sample from the discrete latent space of a trained VQ-VAE.

Differentiation Through the Optimization Layer: Training the safety filter requires differentiating through the optimization problem underlying the safety filter. There are two possible approaches for it namely implicit differentiation and algorithm unrolling [11]. The former cannot be used in our case, as they are not suitable for learning initialization. More precisely, the initialization do not appear explicitly in the optimality conditions and thus, these cannot be learned using implicit differentiation. On the other hand, algorithm unrolling does not have this limitation but requires that every step of the optimization solver is differentiable. To this end, we propose a novel solution process that ensures differentiability across each numerical step. Detailed analysis can be found in Appendix VII, (26a)-(26f). Moreover, our solver is easily parallelizable over GPUs, does not require any matrix factorization and even has closed-form solution for many of the intermediate steps.

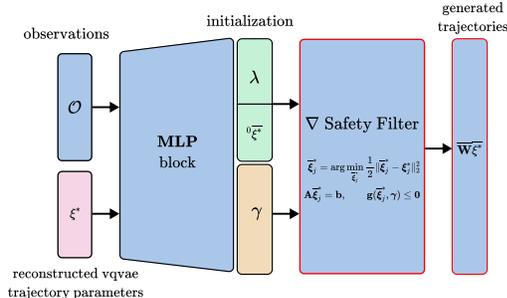


Fig. 5. Learned Safety filter that consists of an MLP augmented with an optimization layer. We train the safety filter in self-supervised setting to learn the parameters of inequality constraints along with good initialization for the optimization problem.

IV. CONNECTIONS TO EXISTING WORKS

Learning Trajectory Sampling Distribution: A large number of recent works have attempted to learn good sampling distribution. The application domain covers both general motion planning as well as those specific for autonomous driving. For example, [12] uses CVAE to approximate a good sampling distribution for motion planners. Along similar lines, [13] learned sampling distribution to accelerate MPPI [14] algorithm using normalizing flows [15]. Our prior work used CVAE along with differentiable optimization layers [3], [5]. The fundamental difference between these cited works stems from our usage of discrete latent priors, which we believe better captures the multi-modality for autonomous driving trajectories.

Differentiable Optimization Layers: The ability to differentiate optimization solvers have proved extremely useful for end-to-end learning [16] for control. However, the greatest success of these class of approaches have come while embedding convex optimization problems within neural network pipeline. Algorithms for differentiating non-convex solvers is limited, for example, unconstrained non-linear least squares problem [11]. Thus, current work develops its own custom differentiable optimization layers following our prior efforts [5]. In particular, we reformulate the underlying constraints to achieve an optimization routine where each step is differentiable. Subsequently, we use algorithm unrolling to learn both good initialization as well as the explicit learnable parameters of our safety filter.

Safety Filter: Safety filter based on barrier function are extensive used in the existing works [17]. Potentially, the parameters of the CBF can also be learned, for example through imitation learning [18]. However, most existing safety filter, especially the ones that are learned are based on one-step planning formulated as a QP [18]. In contrast, our proposed safety filter is based on trajectory optimization over a long horizon.

V. VALIDATION AND BENCHMARKING

This section presents extensive simulation results to answer the following research questions.

- **Q1** How does VQ-VAE compare with CVAE in capturing multi-modal trajectory distribution and how these respective approaches perform in complex driving scenarios.

- **Q2** How does the performance of VQ-VAE and CVAE scale with restriction in computational and sampling budget.

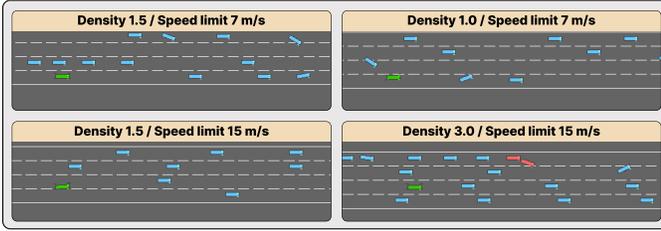


Fig. 6. Various traffic scenarios made from different densities and speed limit of neighboring vehicles used for benchmarking our approach with the CVAE-based baseline [5]

A. Implementation Details

We developed our entire inferencing pipeline in Python with JAX [19] as the GPU-accelerated linear algebra backend. The matrix \mathbf{W} in (1) is derived from a 10^{th} order polynomial. The VQ-VAE, PixelCNN and safety filter were trained in PyTorch. Our simulation framework is based on the Highway Environment (highway-env) simulator [20], where neighboring vehicles adhere to simple rule-based strategies for lateral and longitudinal control.

1) *Training of VQ-VAE and PixelCNN*: The training details of the VQ-VAE and PixelCNN models have been elaborated in earlier sections. Each expert trajectory consists of 100 $x - y$ coordinate points. The observation vector \mathcal{O} , sized 55, consisted of (i) distance from the left and right lane boundaries, (ii) lateral and longitudinal velocities, (iii) ego-vehicle heading, and (iv) state information (x - y position, latitudinal and longitudinal velocities, and heading) of the ten closest obstacles. All position measurements are relative to the ego-vehicle position. During inference, the PixelCNN uses \mathcal{O} from the simulator to yield vector samples of code-book indices h_q in the embedding space. These indices are then fed through the embedding layer to retrieve the discrete latent representation \mathbf{Z}_q . Subsequently, the VQ-VAE decoder first generates the corresponding velocity and lateral-offset vector \mathbf{p} and then the associated trajectories. To gather multi-modal optimal trajectory demonstrations for training both the VQ-VAE and PixelCNN, we use the approach presented in [21]

2) *Baseline*: We compare our VQ-VAE based approach with the CVAE-based baseline presented in [5], which has earlier demonstrated good performance in dense traffic scenarios. Both the CVAE and VQ-VAE models utilize identical observation vectors denoted as \mathcal{O} as input. Both the approaches also employ a safety filter. But the ones used in [5] does not have any explicit learnable parameter, while that proposed in current work learns the barrier function parameters through self-supervised learning.

3) *Environments, Tasks, and Metrics*: The highway driving scenarios are depicted in Figure 6. Utilizing the highway-env simulator, we are able to configure parameters such as

vehicle density and average speed limit within each traffic scenario. Benchmarking was conducted across multiple scenarios, each evaluated using two distinct seeds, with the resulting metric averages being derived from these seeds.

The primary objective is collision-free navigation of the ego vehicle amidst traffic, while attempting to move as fast as possible. Consequently, the evaluation metric comprises two key components: (i) collision rate and (ii) average velocity attained within each episode, with collision rate taking precedence as the more critical measure.

B. Qualitative Comparison Between VQ-VAE and CVAE

Fig.1 (a) shows a scenario with static obstacles. The ego-vehicle is positioned in a way that multiple feasible trajectories are possible. However, CVAE-based approach of [5] is only able to generate trajectory samples in one homotopy class. In sharp contrast, our VQ-VAE (Fig.1(b)) can leverage its learned discrete latent space to generate more diverse trajectories. The core difference between the two models can be further understood by Fig.1(c)-(d), which shows the Kernel Density Estimation (KDE) over the velocity and lateral-offset setpoints associated with the sampled trajectories. While VQ-VAE generated samples exhibit sharp multi-modality, a distinct posterior collapse can be viewed for the CVAE.

C. Benchmarking Driving Performance with VQ-VAE and CVAE

All tasks referred to in this section were performed across 50 different episodes and two different seeds. The collision rate measures the percentage number of crashed episodes in the 50 episode runs. For forward velocity, we present both mean and standard-deviation to capture the variation within episodes.

1) *Performance Across Different Densities*: Fig.7 compares the performance of CVAE based trajectory sampling of [5] with our VQ-VAE pipeline. The former works well for low-traffic scenarios and where the speed limit of the neighboring vehicle is high creating large vacant spaces. As the density increases or the neighboring vehicles start moving slowly, [5] demonstrates higher collision-rate. In contrast, the superior exploration properties of our VQ-VAE sampling allows for upto 12 times reduction (density 3 scenarios) in collision-rate over the CVAE baseline. Both the approaches result in similar velocity profiles.

2) *Performance Scaling with Computational Budget*: In this study, we examine the performance of the VQ-VAE model under reduced computational resources. Specifically, we focus on variations in the number of iterations of the safety layer optimization and the number of trajectories sampled from VQ-VAE. Fig. 8 summarizes the results. We observe that the collision-rate is mere 3% even when the number of iterations is reduced to 50. Similarly, while reducing the sample-size from 1000 to 750, the collision-rate has only increased to 4%. It is worth pointing out that these experiments were conducted under conditions where the traffic density was set at 3.0 and the speed limit of neighboring vehicles was 15 m/s.

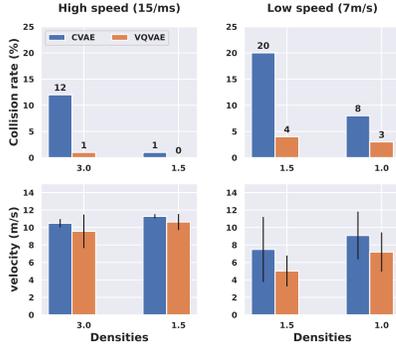


Fig. 7. Comparison of CVAE and VQ-VAE based models across different densities and speed limit of the neighboring vehicles. In both scenarios, the neighboring vehicles randomly chooses their forward velocity between zero and the speed limit. VQ-VAE massively outperforms the CVAE baseline in terms of collision-rate.

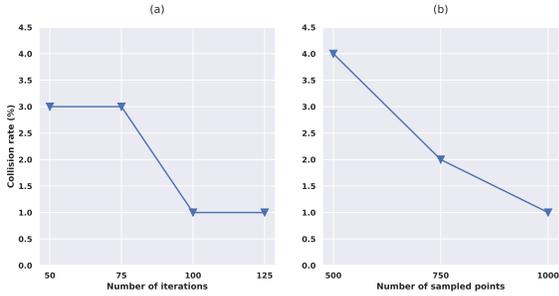


Fig. 8. (a) shows the comparison across different number of iterations of the safety filter. (b) shows the comparison across different number of sampled set points/trajectories.

D. Effect of Learned Safety Filter

Fig. 9 illustrates the significance of a safety filter. We perform experiments for CVAE baseline [5] and our VQ-VAE approach as both of them have similar safety with minor subtle differences that our proposed safety filter is based on barrier function whose parameters are also learned. The results are summarized in Fig.9. It can be seen that up to traffic density 1.5, the multi-modal sampling ensured by VQ-VAE itself is enough to maintain a low collision-rate. In contrast, the CVAE-baseline without safety filter demonstrates an eight times higher collision-rate. At higher densities both CVAE and VQ-VAE based approaches require the explicit constraint handling provided by the safety filter to ensure low collision-rate. It can also be seen that the incorporation of safety filter typically makes the vehicle more conservative by reducing its forward speed.

VI. CONCLUSION AND FUTURE WORKS

We demonstrated how we can train a VQ-VAE model to learn a discrete prior over the space of optimal trajectories. Moreover, we can train a PixelCNN to sample from the learned discrete latent space. We showed that VQ-VAE is more capable of capturing the inherent multi-modality in driving behaviours and thus demonstrates superior performance than CVAE based approaches. In particular, we compare against the SOTA approach of [5] and showed

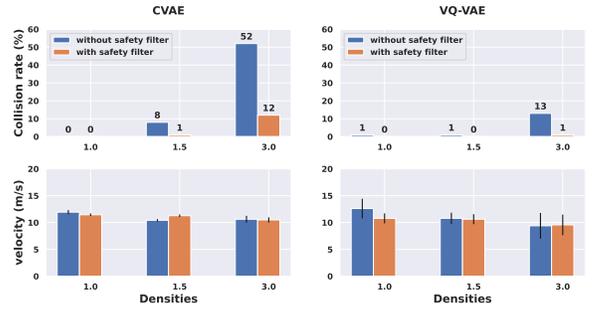


Fig. 9. Comparison of CVAE baseline [5] and our VQ-VAE approach with and without the safety filter. The superior samples generated from VQ-VAE allows for safe navigation even without the safety layer in some less/moderately dense scenarios

a reduction of up to 12 times in collision-rate. We also proposed a learnable safety filter that uses the concept of barrier function to enhance safety in dense traffic scenarios. The safety layer is trained in a self-supervised manner. Our future efforts are geared towards extending our approach to navigation in unstructured dynamic environments like human crowds and manipulation for high-dimensional systems.

VII. APPENDIX

This section extends the formulation of [5]. Specifically, we incorporate collision and lane boundary constraints in terms of barrier functions, while retaining the differentiability and efficiency of the resulting optimization steps.

A. Reformulating Constraints:

Table I presents the list of all the constraints included in our projection optimizer. The collision avoidance, velocity and acceleration constraints presented there can be re-written in the following polar form by introducing additional variables.

$$\mathbf{f}_{o,i} = \left\{ \begin{array}{l} x[k] - x_{o,i}[k] - d_{o,i}[k] \cos \alpha_{o,i}[k] \\ y[k] - y_{o,i}[k] - d_{o,i}[k] \sin \alpha_{o,i}[k] \end{array} \right\} d_{o,i}[k] \geq 1 \quad (10)$$

$$\mathbf{f}_v = \left\{ \begin{array}{l} \dot{x}[k] - d_v[k] \cos \alpha_v[k] \\ \dot{y}[k] - d_v[k] \sin \alpha_v[k] \end{array} \right\}, v_{min} \leq d_v[k] \leq v_{max} \quad (11)$$

$$\mathbf{f}_a = \left\{ \begin{array}{l} \ddot{x}[k] - d_a[k] \cos \alpha_a[k] \\ \ddot{y}[k] - d_a[k] \sin \alpha_a[k] \end{array} \right\}, 0 \leq d_a[k] \leq a_{max} \quad (12)$$

The variables $\alpha_{o,i}[k]$, $\alpha_v[k]$, $\alpha_a[k]$, $d_{o,i}[k]$, $d_v[k]$, and $d_a[k]$ will be computed by the safety layer.

Incorporating obstacle barrier constraints based on [22]: In (10), $d_{o,i}[k] = 1$ represents the boundary of the feasible set of the collision avoidance constraints in Table I. In other words $d_{o,i}[k] = 1$ ensures $h_{1,i}[k] = 0$. Similarly, $d_{o,i}[k] > 1$ signify the interior of the set. With this insight, we can define the polar reformulation of barrier constraints in the following form:

$$d_{o,i}[k] \geq 1 + (1 - \gamma_{obs})(d_{o,i}[k-1] - 1), \forall k \quad (13)$$

The constraints (10) and (13) differ only in the feasible region definition of $d_{o,i}[k]$. When $\gamma_{obs} = 1$, the constraints are equivalent.

TABLE I
LIST OF INEQUALITY CONSTRAINTS USED IN THE SAFETY FILTER

Constraint Type	Expression	Parameters
Discrete-time Barrier for Collision Avoidance at step k $g_1 = (g_{1,1}, g_{1,2}, \dots, g_{1,m})$	$g_{1,i}[k] : h_{1,i}[k+1] - h_{1,i}[k] > -\gamma_{obs}h_{1,i}[k],$ $h_{1,i}[k] = -\frac{(x[k]-x_{o,i}[k])^2}{a^2} - \frac{(y[k]-y_{o,i}[k])^2}{b^2} + 1$	$\frac{a}{2}, \frac{b}{2}$: dimension of combined elliptical footprint of vehicle and obstacle . $x_{o,i}[k], y_{o,i}[k]$: trajectory way-point of obstacles at time step k m : number of obstacles
Velocity bounds at step k $g_2 = (g_{2,lb}, g_{2,ub})$	$g_{2,ub}[k] : \sqrt{\dot{x}[k]^2 + \dot{y}[k]^2} \leq v_{max}$ $g_{2,lb}[k] : \sqrt{\dot{x}[k]^2 + \dot{y}[k]^2} \geq v_{min}$	v_{min}, v_{max} : min/max velocity of the ego-vehicle
Acceleration bounds at step k g_3	$g_3[k] : \sqrt{\ddot{x}[k]^2 + \ddot{y}[k]^2} \leq a_{max}$	a_{max} : max acceleration of the ego-vehicle
Discrete-time barrier for Lane boundary at step k $g_4 = (g_{4,lb}, g_{4,ub})$	$g_{4,ub}[k] : h_{4,ub}[k+1] - h_{4,ub}[k] \geq -\gamma_{lane}h_{4,ub}[k],$ $g_{4,lb}[k] : h_{4,lb}[k+1] - h_{4,lb}[k] \geq -\gamma_{lane}h_{4,ub}[k]$ $h_{4,ub}[k] = -y[k] + y_{ub}$ $h_{4,lb}[k] = y[k] - y_{lb}$	y_{lb}, y_{ub} : Lane limits

We integrate (13) into our algorithm through a minor modification in the feasible region definition of $d_{o,i}[k]$. Let ${}^t d_{o,i}[k]$ be the value of $d_{o,i}[k]$ obtained at the t -th iteration of our optimization algorithm (presented later). We use it to approximate the lower bound on $d_{o,i}[k]$ for obstacle barrier constraints at the $(t+1)$ -th iteration as

$$d_{o,i}[k] \geq 1 + (1 - \gamma_{obs})({}^t d_{o,i}[k-1] - 1) \quad (14)$$

The right-hand side of (14) is constant, and thus the feasible region of $d_{o,i}[k]$ for barrier constraints is approximated through a simple lower bound.

1) *Incorporating lane barrier constraints:* The standard form of lane constraints is as follows:

$$y_{lb} \leq y[k] \leq y_{ub} \forall k \quad (15)$$

where y_{lb}, y_{ub} are the lane limits. The discrete-time control barrier functions corresponding to these are given in I:

Consequently, the lane upper bound barrier constraints are,

$$y[k+1] + (\gamma_{lane} - 1)y[k] \leq \gamma_{lane}y_{ub} \forall k \quad (16)$$

In terms of trajectory parametrization (1), this becomes:

$$\mathbf{W}[k+1]\mathbf{c}_y + (\gamma_{lane} - 1)\mathbf{W}[k]\mathbf{c}_y \leq \gamma_{lane}y_{ub} \forall k \quad (17)$$

where $\mathbf{W}[k]$ represents the k^{th} row of the \mathbf{W} matrix and \mathbf{c}_y is the coefficient vector for \mathbf{y} . Since we want a combined inequality corresponding to all the time steps $k = 0, 1, 2, \dots, n-1$, we can rewrite the above as :

$$\mathbf{W}_{1,ub}\mathbf{c}_y + (\gamma_{lane} - 1)\mathbf{W}_{0,ub}\mathbf{c}_y \leq \mathbf{b}_{ub} \text{ or,} \quad (18)$$

$$(\mathbf{W}_{1,ub} + (\gamma_{lane} - 1)\mathbf{W}_{0,ub})\mathbf{c}_y \leq \mathbf{b}_{ub}$$

where $\mathbf{W}_{1,ub} := \mathbf{W}[1 : n-1]$, $\mathbf{W}_{0,ub} := \mathbf{W}[0 : n-2]$ and $\mathbf{b}_{ub} := \gamma_{lane}y_{ub}\mathbf{1}_{(n-1) \times 1}$.

Define $\mathbf{G}_{ub} := \mathbf{W}_{1,ub} + (\gamma_{lane} - 1)\mathbf{W}_{0,ub}$. This results in the following inequality constraint:

$$\mathbf{G}_{ub}\mathbf{c}_y \leq \mathbf{b}_{ub} \quad (19)$$

Similarly, we get the lane lower bound barrier constraints as:

$$\mathbf{G}_{lb}\mathbf{c}_y \leq \mathbf{b}_{lb} \quad (20)$$

where $\mathbf{G}_{lb} := \mathbf{W}_{1,lb} + (1 - \gamma_{lane})\mathbf{W}_{0,lb}$, $\mathbf{W}_{1,lb} := -\mathbf{W}[1 : n-1]$, $\mathbf{W}_{0,lb} := \mathbf{W}[0 : n-2]$ and $\mathbf{b}_{lb} := -\gamma_{lane}y_{lb}\mathbf{1}_{(n-1) \times 1}$.

2) *Reformulated Problem:* Using the developments in the previous section and the trajectory parametrization presented in (1), we can now replace the safety layer optimization (7)-(8) with the following. Note the subscript j that signifies the problem is defined for projecting the j^{th} sampled trajectory.

$$\bar{\xi}_j^* = \arg \min_{\xi_j^*} \frac{1}{2} \|\bar{\xi}_j - \xi_j^*\|^2 \quad (21a)$$

$$\mathbf{A}\bar{\xi}_j^* = \mathbf{b} \quad (21b)$$

$$\tilde{\mathbf{F}}\bar{\xi}_j^* = \tilde{\mathbf{e}}(\alpha_j, \mathbf{d}_j) \quad (21c)$$

$$\mathbf{d}_{min} \leq \mathbf{d}_j \leq \mathbf{d}_{max}(\gamma_{obs}) \quad (21d)$$

$$\mathbf{G}_{\gamma_{lane}}\bar{\xi}_j^* \leq \mathbf{y}_{lane} \quad (21e)$$

$$\tilde{\mathbf{F}} = \begin{bmatrix} \mathbf{F}_o & \mathbf{0} \\ \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} \mathbf{F}_o \\ \mathbf{W} \\ \mathbf{W} \end{bmatrix} \end{bmatrix}, \tilde{\mathbf{e}} = \begin{bmatrix} \mathbf{x}_o + \mathbf{a}d_{o,j} \cos \alpha_{o,j} \\ \mathbf{d}_{v,j} \cos \alpha_{v,j} \\ \mathbf{d}_{a,j} \cos \alpha_{a,j} \\ \mathbf{y}_o + \mathbf{a}d_{o,j} \sin \alpha_{o,j} \\ \mathbf{d}_{v,j} \sin \alpha_{v,j} \\ \mathbf{d}_{a,j} \sin \alpha_{a,j} \end{bmatrix}, \quad (22)$$

$$\mathbf{G}_{\gamma_{lane}} = [\mathbf{G}_{ub} \quad \mathbf{G}_{lb}]^T, \mathbf{y}_{lane} = [\mathbf{b}_{ub} \quad \mathbf{b}_{lb}]^T \quad (23)$$

$$\alpha_j = (\alpha_{o,j}, \alpha_{a,j}, \alpha_{v,j}), \quad \mathbf{d}_j = (\mathbf{d}_{o,j}, \mathbf{d}_{v,j}, \mathbf{d}_{a,j})$$

Constraints (21c)-(21e) acts as substitutes for $\mathbf{g}(\bar{\xi}_j^*) \leq 0$ in the optimization (7)-8.

We form matrix \mathbf{F}_o by stacking the matrix \mathbf{W} from (1) as many times as the number of neighbouring vehicles considered for collision avoidance at a given planning cycle. The vector $\mathbf{x}_o, \mathbf{y}_o$ is formed by appropriately stacking $x_{o,i}[k], y_{o,i}[k]$ at different time instants and for all the neighbours. Similar construction is followed to obtain $\alpha_{o,j}, \alpha_{v,j}, \alpha_{a,j}, \mathbf{d}_{o,j}, \mathbf{d}_{v,j}, \mathbf{d}_{a,j}$. The vector \mathbf{y}_{lane} is formed by stacking the upper and lower lane bounds after repeating them n times (planning horizon). Similarly, vectors $\mathbf{d}_{min}, \mathbf{d}_{max}$ are formed by stacking the lower and upper bounds for $\mathbf{d}_{o,j}, \mathbf{d}_{v,j}, \mathbf{d}_{a,j}$ (recall (10)-(12)). Note that the upper bound for $\mathbf{d}_{o,j}$ can be simply some large number,

while the lower bound depends on γ_{obs} (recall (10), (14)). Moreover, these bounds are the same across all batches.

Remark 1: The vector $\gamma = (\gamma_{lane}, \gamma_{obs})$ form the learnable barrier function parameters introduced in (8), Section III-C

3) *Solution Process:* We relax the non-convex equality (21c) and affine inequality constraints as l_2 penalties and augment them into the projection cost (21a).

$$\begin{aligned} \mathcal{L} &= \frac{1}{2} \left\| \bar{\xi}_j^* - \xi_j^* \right\|_2^2 - \lambda_j^T \bar{\xi}_j^* + \frac{\rho}{2} \left\| \tilde{\mathbf{F}} \bar{\xi}_j^* - \tilde{\mathbf{e}} \right\|_2^2 \\ &\quad + \frac{\rho}{2} \left\| \mathbf{G}_{\gamma_{lane}} \bar{\xi}_j^* - \mathbf{y}_{lane} + \mathbf{s}_j \right\|_2^2 \quad (24) \\ &= \frac{1}{2} \left\| \bar{\xi}_j^* - \xi_j^* \right\|_2^2 - \lambda_j^T \bar{\xi}_j^* + \frac{\rho}{2} \left\| \mathbf{F} \bar{\xi}_j^* - \mathbf{e} \right\|_2^2 \end{aligned}$$

$$\mathbf{F} = \begin{bmatrix} \tilde{\mathbf{F}} \\ \mathbf{G}_{\gamma_{lane}} \end{bmatrix}, \mathbf{e} = \begin{bmatrix} \tilde{\mathbf{e}} \\ \mathbf{y}_{lane} - \mathbf{s}_j \end{bmatrix} \quad (25)$$

Note, the introduction of the Lagrange multiplier λ that drives the residual of the second and third quadratic penalties to zero [23]. We minimize (24) subject to (21b) through Alternating Minimization (AM), which reduces to the following steps [24], wherein the left superscript t represents the iteration index.

$${}^{t+1} \alpha_j = \arg \min_{\alpha_j} \mathcal{L}({}^t \bar{\xi}_j^*, {}^t \mathbf{d}_j, \alpha_j, {}^t \lambda_j, {}^t \mathbf{s}_j) \quad (26a)$$

$${}^{t+1} \mathbf{d}_j = \arg \min_{\mathbf{d}_j} \mathcal{L}({}^t \bar{\xi}_j^*, \mathbf{d}_j, {}^{k+1} \alpha_j, {}^t \lambda_j, {}^t \mathbf{s}_j) \quad (26b)$$

$${}^{t+1} \mathbf{s} = \max \left(0, -\mathbf{G}_{\gamma_{lane}} {}^t \bar{\xi}_j^* - \mathbf{y}_{lane} \right) \quad (26c)$$

$${}^{t+1} \lambda_j = {}^t \lambda_j + \rho \mathbf{F}^T (\mathbf{F} {}^t \bar{\xi}_j^* - {}^t \mathbf{e}_j) \quad (26d)$$

$${}^{t+1} \mathbf{e}_j = \begin{bmatrix} \tilde{\mathbf{e}}({}^{t+1} \alpha_j, {}^{t+1} \mathbf{d}_j) \\ \mathbf{y}_{lane} - {}^{t+1} \mathbf{s}_j \end{bmatrix} \quad (26e)$$

$${}^{t+1} \bar{\xi}_j^* = \arg \min_{\bar{\xi}_j^*} \mathcal{L}(\bar{\xi}_j^*, {}^{t+1} \lambda_j, {}^{t+1} \mathbf{e}_j) \quad (26f)$$

As can be seen, we optimize over only one group of variables at each AM step while others are held fixed at values obtained at the previous updates. Steps (26d)-(26e) have a closed-form solution, while (26f) is an equality constrained QP [5]. Thus, we can trace the gradients through the unrolled optimization step and train the safety filter proposed in Section III-C. Moreover, as shown in [5], the QP (26f) is easily batchable and can be made free of matrix factorization.

REFERENCES

- [1] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.
- [2] K. Moller, R. Trauth, G. Wuersching, and J. Betz, "Frenetix motion planner: High-performance and modular trajectory planning algorithm for complex autonomous driving scenarios," *arXiv preprint arXiv:2402.01443*, 2024.
- [3] A. K. Singh, J. Shrestha, and N. Albarella, "Bi-level optimization augmented with conditional variational autoencoder for autonomous driving in dense traffic," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023, pp. 1–8.
- [4] V. K. Adajania, A. Sharma, A. Gupta, H. Masnavi, K. M. Krishna, and A. K. Singh, "Multi-modal model predictive control through batch non-holonomic trajectory optimization: Application to highway driving," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4220–4227, 2022.

- [5] J. Shrestha, S. Idoko, B. Sharma, and A. K. Singh, "End-to-end learning of behavioural inputs for autonomous driving in dense traffic," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 020–10 027.
- [6] D. P. Kingma and M. Welling, "Auto-encoding variational {Bayes}," in *Int. Conf. on Learning Representations*.
- [7] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv preprint arXiv:1711.00937*, 2017.
- [8] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," *Advances in neural information processing systems*, vol. 29, 2016.
- [9] J. Li, L. Sun, J. Chen, M. Tomizuka, and W. Zhan, "A safe hierarchical planning framework for complex driving scenarios based on reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2660–2666.
- [10] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE transactions on intelligent vehicles*, vol. 5, no. 2, pp. 294–305, 2019.
- [11] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R. T. Chen, J. Ortiz, D. DeTone, A. S. Wang, S. Anderson *et al.*, "Theseus: A library for differentiable nonlinear optimization," in *Advances in Neural Information Processing Systems*.
- [12] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.
- [13] J. Sacks and B. Boots, "Learning sampling distributions for model predictive control," in *Conference on Robot Learning*. PMLR, 2023, pp. 1733–1742.
- [14] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.
- [15] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.
- [16] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.
- [17] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.
- [18] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "Barriernet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, 2023.
- [19] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: <http://github.com/google/jax>
- [20] E. Leurent, "An Environment for Autonomous Driving Decision-Making," 5 2018. [Online]. Available: <https://github.com/eleurent/highway-env>
- [21] F. Rastgar, H. Masnavi, K. Kruusamäe, A. Aabloo, and A. K. Singh, "Gpu accelerated batch trajectory optimization for autonomous navigation," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 718–725.
- [22] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "Amswarm: An alternating minimization approach for safe motion planning of quadrotor swarms in cluttered environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1421–1427.
- [23] T. Goldstein and S. Osher, "The split bregman method for l1-regularized problems," *SIAM journal on imaging sciences*, vol. 2, no. 2, pp. 323–343, 2009.
- [24] H. Masnavi, J. Shrestha, M. Mishra, P. Sujit, K. Kruusamäe, and A. K. Singh, "Visibility-aware navigation with batch projection augmented cross-entropy method over a learned occlusion cost," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9366–9373, 2022.