ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

DiffuseDrive

An Exploration of Diffusion Models for Autonomous Driving

Robot Learning Course Project - Spring 2023

Jacopo Graldi
Marcus Leong
Minxuan Qin
Department of Information Technology and Electrical Engineering

Advisors: Yan Wu, Dr. Yifan Liu
Supervisor: Prof. Dr. Fisher Yu

July 27, 2023

Abstract

Autonomous Driving has long been a deeply studied field in both academia and industry for its terrific societal and economic potential. It is however a difficult task that still has not reached a satisfactory level of trust and capability for a widespread deployment. An end-to-end, interpretable architecture - in contrast to the current state-of-the-art models - is needed for its pervasive adoption. In this project, we implement a proof-of-concept end-to-end Autonomous Driving pipeline based on Diffusion Models. Based on previous attempts at diffusing trajectories for planning pipelines, we employ diffusion models to generate and predict future waypoints and trajectories, conditioning on the past history of the vehicle such as past visual information, high-level commands, and waypoints. Despite lacking understanding and reasonable decision-making in complex traffic situations (e.g. at intersections), our agent is able to learn some basic behavior such as simple turning and lane-following actions. This demonstrates the interesting potential and opportunities given by Diffusion Models for Autonomous Driving, and we pave the way for an interpretable architecture.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The world has witnessed in the past year an unprecedented and terrific advance in the field of Artificial Intelligence (AI), where OpenAI's DALL-E 2[1] - an image generation system - and ChatGPT[2] - a natural language processing (NLP) model for written interaction between user and machine - are only the most prominent examples. Despite these mindblowing AI improvements, some problems are still far from being solved. One of these, despite recent and rapid progress, is autonomous driving. Some of the reasons for its difficulty are the large number of dynamic objects, actors, and obstacles involved in the decision-making; the sudden emergence of safety concerns; difficult verification of the decision-making process. The latter in particular, is a crucial point for understanding and improving the current models, as well as for guaranteeing a level of safety that is absolutely necessary for widespread adoption acceptance of these systems by both users and legislators. An important step forward was introduced in 2017 by [6] with CARLA (CAR Learning to Act), an open-source urban driving simulator for training and assessing the autonomous driving capabilities of the deployed models. A plethora of implementations populated its leaderboard[3] in recent years, with the InterFuser [17] being currently the best-ranked implementation.

The InterFuser approach is based on multi-modal sensor fusion via Vision Transformers (ViT) [5], and it differs from other popular implementations - as the TransFuser [3] - for addressing the interpretability requirements and concerns by providing intermediate interpretable features. Despite its performance and interpretability, the InterFuser presents some major drawbacks: it plans at a highly abstract level that requires costly annotations, and it is far from being end-to-end, with multiple hardcoded human heuristics inserted into its planning pipeline.

In the latest months, a new and fast-developing paradigm of generative models has drawn a lot of attention in the computer vision community: diffusion models ([7], [15]). These have demonstrated an unprecedented image-generation quality and real-world fidelity, and have the potential to allow image generation of the traffic scenes predicted by the autonomous-driving pipeline as an interpretable middle layer. Diffusion models have been also recently employed not only for the generation of images but also in the planning of trajectories of robotic agents in complex environments ([10], [1]) and thus might be promising tools for autonomous driving. In this project, we indeed explore how to leverage their potential to create an end-to-end autonomous driving pipeline that is reliable and interpretable. The code of this project is available on the DiffuseDrive GitHub repository[4].

This project report is structured as follows. In Chap. 2 we introduce the relevant literature for our work. Next, in Chap. 3 we present the materials and the methods used throughout the project and in Chap. 4 we

[1] https://openai.com/dall-e-2
[2] https://openai.com/blog/chatgpt
[3] https://paperswithcode.com/sota/autonomous-driving-on-carla-leaderboard
[4] https://github.com/graldij/DiffuseDrive

show the results of our experiments. Finally in Chap. 5 we critically discuss our findings, and in Chap. 6 we conclude this report.

Chapter 2

Related Work

2.1 Autonomous Driving in urban scenarios

The research on end-to-end autonomous driving has become more and more popular. The CARLA simulator [6] (Car Learning to Act) offers an open-source platform to train and validate autonomous driving models in realistic urban traffic situations with various weather conditions. Based on the CARLA simulator, TransFuser [3] proposed a new experimental setting in CARLA for imitation learning policies and a novel multi-modal fusion transformer architecture to incorporate the global context of image and LiDAR modalities. Furthermore, the SOTA model InterFuser [17] on the CARLA leaderboard proposed a more efficient way to fuse different sensor modalities. It designed a safety-enhanced and interpretable driving policy by the exploitation of the interpretable outputs from the fusion transformer (future waypoints, object density map, and traffic rules). However, it needs numerous annotations, contains many human heuristics, and is not fully end-to-end.

2.2 Diffusion Probabilistic Models

Diffusion models have come up as new state-of-the-art deep generative models with a record-breaking performance in many applications such as image and video generation ([13], [15]). In this section, we will first introduce DDPM (Denoising Diffusion Probabilistic Model) [7], the first work to utilize diffusion models in high-fidelity image generation area, its improved version improved DDPM [12], DDIM (Denoising Diffusion Implicit Model) [19], latent diffusion [15], and guided diffusion [8], which diminish the training cost, accelerate the inference, and improve the image generation quality.

2.2.1 Denoising Diffusion Probabilistic Models

DDPM [7] uses the Markov chain to model the forward diffusion process which is inspired by non-equilibrium thermodynamics [18]. Formally, given a data distribution $x_0 \sim q(x_0)$, in the forward diffusion process, we gradually perturb the sample with a small amount of Gaussian noise, producing a sequence of noisy samples, $x_1, \ldots, x_T$. By the property of the Markov chain, the joint conditional distribution, denoted by $q(x_1, \ldots, x_T)$, is factorized into:

$$q(x_1, \ldots, x_T) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{2.1}$$

In DDPM, the step sizes are controlled by a schedule $\{\beta_t\}_{t=1}^{T}$ with $\beta_t \in (0, 1)$, and the transition kernel is parametrized by

$$
\begin{aligned}
q(x_t|x_{t-1}) &= \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \\
q(x_t|x_0) &= \mathcal{N}(x_t; \sqrt{\tilde{\alpha}_t}x_0, (1-\tilde{\alpha}_t) I)
\end{aligned}
\tag{2.2}
$$

To obtain the latter, let $\alpha_t := 1 - \beta_t$ and $\tilde{\alpha}_t := \prod_{s=0}^{t} \alpha_s$. By using the reparametrization trick and the property of Gaussian distribution, the conditional distribution of the noisy sample x_t given the original data x_0 is again a Gaussian distribution. Given x_0 and $\tilde{\alpha}_T \approx 0$, x_T is almost a standard Gaussian in distribution.

For generating new data samples, DDPM starts with sampling from the standard Gaussian distribution, then removes the noise step by step by estimating the noise with a deep neural network $p_\theta(x_{t-1}|x_t)$. Similar to the equation (2.1), we describe the reverse diffusion process by

$$
p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.3}
$$

The key to the success of the reverse process is to approximate the forward process $q(x_{0:T}) = q(x_0) \cdot q(x_1, \ldots, x_T)$ (see equation (2.1)) by the parametrized reverse process $p_\theta(x_{0:T})$(see equation (2.3)). The concept is very similar to the variational autoencoder. So we can use the variational lower bound to maximize the log-likelihood:

$$
\begin{aligned}
KL(q(x_{0:T})||p_\theta(x_{0:T})) &= \mathbb{E}_{q(x_{0:T})}[-\log(p(x_T) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t)}] + const \\
&= -L_{VLB} + const \\
&\geq \mathbb{E}[-\log p_\theta(x_0)] + const
\end{aligned}
\tag{2.4}
$$

2.2.2 Improvements of DDPM

Ho et al. (2020) [7] improved the training loss by reweighting various terms in the L_{VLB} of the above equation (2.4). In addition, it set the forward variances linearly increasing from $\beta_1 = 10^{-4}$ to $\beta_t = 0.02$. For the estimation of the diagonal reverse variances $\Sigma_\theta(x_t, t) = \sigma_t^2 I$, they experimentally found that setting it to β_t or $\tilde{\beta}_t = \frac{1-\tilde{\alpha}_{t-1}}{1-\tilde{\alpha}_t} \cdot \beta_t$ leads better performance. Contrastively, Nichol & Dhariwal (2021) [12] proposed several improvement techniques over DDPM: they used a cosine-based variance schedule for β_t and interpreted the reverse variance as an interpolation between β_t and $\tilde{\beta}_t$:

$$
\begin{aligned}
\beta_t &= clip(1 - \frac{\tilde{\alpha}_t}{\tilde{\alpha}_{t-1}}, 0.999) \\
\tilde{\alpha}_t &= \frac{f(t)}{f(0)}, \quad where \quad f(t) = \cos(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}) \\
\Sigma_\theta(x_t, t) &= \exp(v \log \beta_t + (1-v) \log \tilde{\beta}_t)
\end{aligned}
\tag{2.5}
$$

Furthermore, they run a strided sampling schedule for the sampling process by taking the sampling update every $\lceil T/S \rceil$ steps, denoted by $\{\tau_1, \ldots, \tau_S\}$, where $\tau_1 < \tau_2 < \cdots < \tau_S \in [1, T]$. Based on the strided sampling, denote the variance $\sigma_t^2 := \eta \cdot \tilde{\beta}_t$, DDIM (Denoising Diffusion Implicit Model) [19] defined $\eta = 0$ to make the sampling process deterministic, which also brings the "consistency" property to the diffusion model.

2.2.3 Latent Diffusion and Guided Diffusion

LDM (Latent Diffusion Model) [15] runs the diffusion process in latent space instead of pixel space, decreases the training cost, and makes the inference phase faster. It loosely decomposes the perceptual compression and the semantic compression with generative modeling. More concretely, it first uses an autoencoder with regularization to extract image features, then estimates the perturbed noise from the diffusion process with a time-conditioned U-Net, which contains a cross-attention mechanism for different modalities, e.g., generating images from textual inputs.

In order to further improve the quality of generated images, Nichol & Dhariwal (2021) [12] trained a classifier $f_\phi(y|x_t, t)$ to guide the denoising direction using the conditioning information y. We can write the score function of the joint distribution $q(x_t, y)$ into following:

$$\begin{aligned}\nabla_{x_t} \log q(x_t, y) &= \nabla_{x_t} \log q(x_t) + \nabla_{x_t} \log q(y|x_t) \\ &\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t) + \nabla_{x_t} \log f_\phi(y|x_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}(\epsilon_\theta(x_t, t) + \sqrt{1-\bar{\alpha}_t}\nabla_{x_t} \log f_\phi(y|x_t))\end{aligned} \tag{2.6}$$

Therefore, a new classifier-guided predictor $\bar{\epsilon}_\theta$ can be defined as following:

$$\bar{\epsilon}_\theta(x_t, t) = \epsilon_\theta(x_t, t) + \sqrt{1-\bar{\alpha}_t} w \nabla_{x_t} \log f_\phi(y|x_t) \tag{2.7}$$

where w is a parameter for controlling the guidance strength. In practice, we can use a dropout mechanism controlled by w to implement the sampling among conditional and unconditional diffusion. Ho & Salimans (2021) [8] proposed a novel approach to implicitly build classifier guidance. Let the conditional model be parametrized by $\epsilon_\theta(x_t, t, y)$ and unconditional model by $\epsilon_\theta(x_t, t) = \epsilon_\theta(x_t, t, y = \emptyset)$. These two models can be learned simultaneously by a single neural network. Similar to the equation (2.6), the classifier-free predictor $\bar{\epsilon}_\theta$ can be represented by following:

$$\begin{aligned}\nabla_{x_t} \log p(y|x_t) &= \nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t)) \\ \bar{\epsilon}_\theta(x_t, t, y) &= \epsilon_\theta(x_t, t, y) - \sqrt{1-\bar{\alpha}_t} w \nabla_{x_t} \log p(y|x_t) \\ &= \epsilon_\theta(x_t, t, y) + w(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t)) \\ &= (w+1)\epsilon_\theta(x_t, t, y) - w\epsilon_\theta(x_t, t)\end{aligned} \tag{2.8}$$

2.3 Applications of Diffusion Models in Reinforcement Learning

Recently, several works showed the possibility of applying generative modeling in the field of reinforcement learning (RL) and decision-making, outperforming some classical RL benchmarks. Diffuser [10] utilized DDPM ([18], [7]) to trajectory optimization. It predicts all timesteps of a plan concurrently (that means non-autoregressive) and can recover trajectories with high returns or satisfying a set of constraints. Furthermore, it allows task and temporal composition based on the diffusion model described in Sec. 2.2. The detailed problem setting is the following: Given a system with the discrete-time dynamics $s_{t+1} = f(s_t, a_t)$ at state s_t with an action a_t. The goal is finding a sequence of actions $a^*_{0:T}$ that maximizing the objective:

$$a^*_{0:T} = \arg\max_{a_{0:T}} \mathcal{J}(s_0, a_{0:T}) = \arg\max_{a_{0:T}} \sum_{t=0}^{T} r(s_t, a_t) \tag{2.9}$$

Let $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$ be the whole trajectory. Diffuser proposed a tighter coupling between modeling and planning by perturbing the sampling distribution (the sampling distribution see equation (2.3)):

$$\tilde{p}_\theta(\tau) \propto p_\theta(\tau) h(\tau) \tag{2.10}$$

The function $h(\tau)$ can contain information about prior evidence, desired outcomes, or general functions to optimize. In the case of optimizing cumulative reward, Diffuser needs to separately train a model J_ϕ to predict it, and similar to the classifier guided diffusion (see equation (2.6)), let $y := \mathcal{O}_{1:T}$ be the binary variables representing the optimality of the distribution ($p(\mathcal{O}_t = 1) = \exp(r(s_t, a_t))$, the diffusion process is described as following:

$$\begin{aligned} p_\theta(\tau^{i-1}|\tau_i, y) &= p_\theta(\tau^{i-1}|\tau_i, \mathcal{O}_{1:T}) \approx \mathcal{N}(\tau_{i-1}; \mu + \Sigma g, \Sigma) \\ g &= \nabla_\tau \log p(\mathcal{O}_{1:T}|\tau)|_{\tau=\mu} \\ &= \sum_{t=0}^{T} \nabla_{s_t, a_t} r(s_t, a_t)|_{(s_t, a_t)=\mu_t} = \nabla \mathcal{J}(\mu) \end{aligned} \tag{2.11}$$

Similar to the Diffuser, Decision Diffuser [1] utilizes the classifier-free diffusion model to diffuse model states, and then uses inverse dynamics to predict the corresponding actions:

$$\begin{aligned} x_k(\tau) &:= (s_t . s_{t+1}, \ldots, s_{t+H-1})_k \\ a_t &:= f_\phi(s_t, s_{t=1}) \end{aligned} \tag{2.12}$$

Here k denotes the timestep in the forward diffusion process, t denotes the time at which the state was visited in trajectory τ, and H denotes the prediction horizon. The main conditional training process is similar to the equation (2.8) except for the training loss with inverse dynamics and the low-temperature sampling:

$$\begin{aligned} \mathcal{L}(\theta, \phi) &:= \mathbb{E}_{k, \tau \in \mathcal{D}, \beta \sim Bern(p)}[||\epsilon - \epsilon_\theta(x_k(\tau), (1-\beta) y(\tau) + \beta \emptyset, k)||^2] \\ &\quad + \mathbb{E}_{(s,a,s') \in \mathcal{D}}[||a - f_\phi(s, s')||^2] \\ x_{k-1} &\sim \mathcal{N}(\mu_{k-1}, \alpha \Sigma_{k-1}) \end{aligned} \tag{2.13}$$

Chapter 3

Materials and Methods

3.1 Data collection

Similar to the data collection process in [17], a rule-based agent with privileged information is used as the agent for data collection in the Carla simulator [6]. The Carla simulator provides 10 different maps with various weather conditions. The data collection process took around 8 to 12 hours for each town and recollection of data is required if the simulation crashes due to compatibility between the GPU driver and the CARLA simulator. Therefore, due to time constraints, we have decided to focus on one specific weather and collected 50000 frames of data from 8 different town maps, namely towns 1, 2, 3, 4, 5, 6, 7, and 10, under the same weather condition. Each frame consists of the front RGB view, the current GPS location of the agent, high-level command, information about the surrounding agents, and the bird's-eye view of the scene centered around the agent. The front RGB images were preprocessed and resized in order to speed up the training process.

3.2 Interpretable Architecture

The first variant of our architecture has the goal to be interpretable. To do so, we want to leverage the diffusion models' capabilities to predict and generate high-fidelity future frames of the RGB front view. In Fig. 3.1 we show the detailed pipeline.

First, the last sensed image of size $H \times W \times C$ is encoded with the encoder of a variational autoencoder (VAE) to a latent space of size $H' \times W' \times C'$. Together with the last waypoint, of size $X \times Y \times \theta$, it goes to a state buffer storing the last S timesteps. Next, both S last waypoints and frames tensors are expanded with T future timesteps with noisy data and then concatenated together. This tensor, of size $\text{Concat}((H' \times W' \times C' \times (S+T)), (X \times Y \times \theta \times (S+T)))$ is fed into the diffusion model which, using the last S timesteps as a condition, predicts and generates the future T waypoints and frames. At this stage of the pipeline, the predicted future frames can be decoded with the decoder of the VAE to reconstruct the predicted images in the human-understandable image space. The output of the diffusion model enters an ad-hoc event prediction network. This, first extracts the visual features of the T future frames (e.g. with a convolutional neural network (CNN) or a ViT), then, together with the T predicted future waypoints a multi-layer perceptron (MLP) is responsible to predict and classify various types of events occurring with that particular combination of future waypoints and future frames, such as collisions, and infractions, just to name a few.

We leverage the stochasticity of the diffusion model by sampling multiple possible trajectories and frames: these all are classified into events as described above and then enter a scoring network that scores the trajectories to identify the best possible trajectory. The best trajectory is selected among all sampled

trajectories and is fed to a proportional–integral–derivative (PID) controller that converts the trajectories (i.e. a sequence of the future T waypoints) into actions that the car has to take (e.g. accelerate, brake, steer the wheel).

3.2.1 Variational Autoencoder

For the perceptual compression of the image we started with the variational autoencoder from stable diffusion v1-4 [16], which includes 4 downsampling blocks to compress the original image 16 times in each dimension. The reconstructed images and their latent code are shown in Figure 3.2.

3.2.2 Image Diffusion

To tackle the image diffusion task, we started by implementing an unconditioned diffusion model using the HuggingFace library. This toy example allowed us to get acquainted with the implementation of diffusion models. We first implemented it in the easiest possible version, namely acting the diffusion on the image space with a simple 2D U-Net with 6 downsampling and 6 upsampling channels of size (128, 128, 256, 256, 512, 512) and a DDIM scheduler. We train this model from scratch on the CARLA images collected. In Fig. 3.3 an example of four sampled images is provided.

As a second step, with increasing difficulty, we have implemented diffusion on the latent space. We followed the key steps from the latent diffusion model described in Sec. 2.2.3, however, as shown in Fig. 3.4, the quality of generated images was not satisfying. They are always with noticeable noise and lose detailed information.

The next step is to implement conditioning on the images. However, as temporal consistency across all S past frames and T future frames is important for a realistic trajectory and interpretable prediction, a 3D video diffusion is needed. We have however soon realized that the computational burden of diffusing a video - even in a latent space - would be unbearable for our very limited computational resources and time constraints. To reduce the computational burden, we are therefore forced to postpone the interpretable goals of our architecture and not diffuse on the computationally-heavy images. This alternative architecture is presented in the next section (Sec. 3.3).

3.3 Non-Interpretable Architecture

We want to use a diffusion model to diffuse and generate meaningful trajectories (i.e. T future waypoints) taking the S past waypoints and S front image frames as a condition.

Two viable options exist and were considered. The first is based on the Diffuser [10] and considers the trajectory planning as an RL problem where the best trajectory maximizes a reward-like score. However, this approach too takes the image dimensionality (either in pixel space or in a latent space) for the diffusion space, and thus has a high computational load. We, therefore, opt for the second option, which is a classifier-free diffusion based on the Decision Diffuser [1]. This latter approach only denoises the trajectory space, and thus does not suffer from the load of the images.

3.3.1 Classifier Free Trajectory Diffusion

Our input data consists of a tensor of shape $(S+T)\times 3$ for the waypoints, where S corresponds to the past $S-1$ timesteps plus the current timestep, and T are the future timesteps. The waypoints are a vector of length 3 in the ego-vehicle frame, i.e. (x, y, θ) (position of the current timestep is $(0.0, 0.0, 0.0)$). Another tensor of size $S\times 128\times 128\times 3$ is responsible for the conditioning on the $128\times 128\times 3$ RGB images of the $S-1$ past and current timesteps.

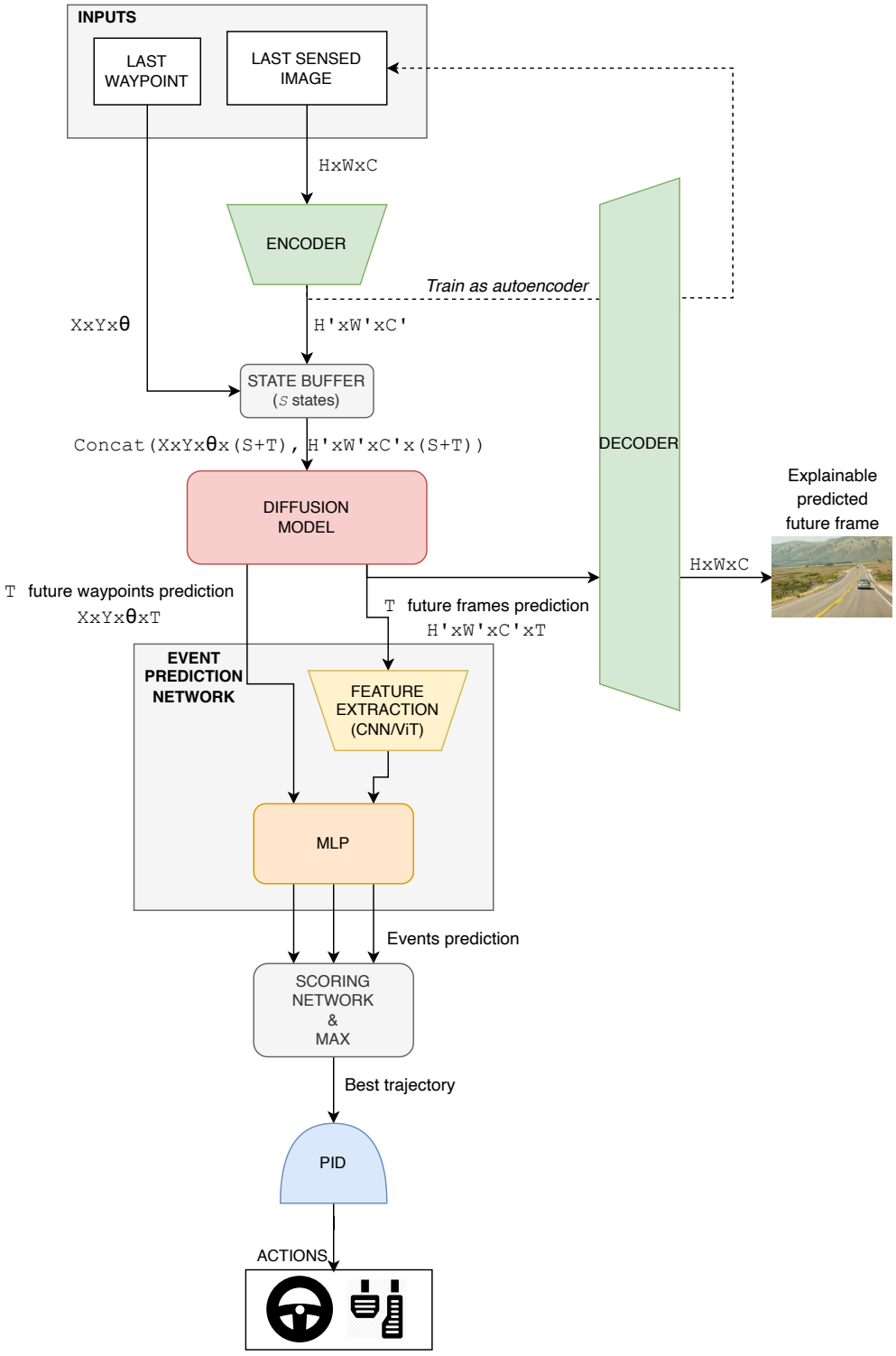


Figure 3.1: DiffuseDrive interpretable pipeline.

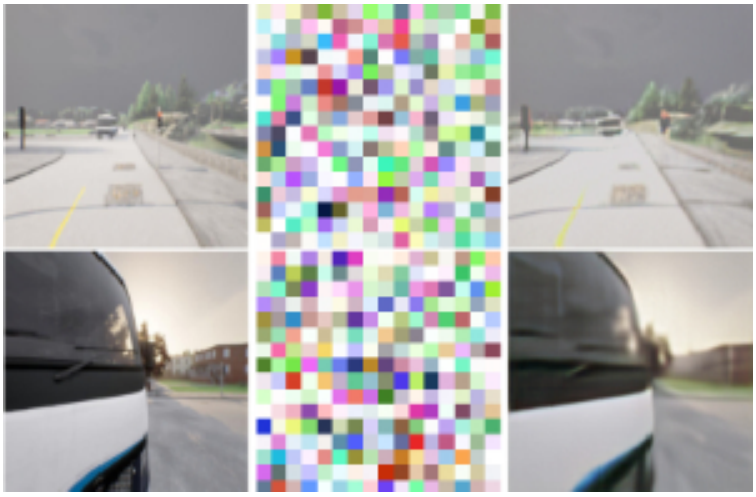


Figure 3.2: Results from the variational autoencoder. The original and reconstructed images are shown in the left and right columns, and visualizations of latent code (4 channels) are in the middle. The reconstruction looks successful.

Figure 3.3: Example of four frames sampled from the unconditional diffusion model acting on the image space.

Figure 3.4: Example of four frames sampled from the latent diffusion model acting on the image space.

We base our implementation on the Decision Diffuser [1] and modify it to our data type and objective. In particular, the diffusion model is trained to denoise the part of the input tensor consisting of the future timesteps, while the past and present part act as conditioning, namely the denoised and predicted future timesteps need to satisfy the temporal consistency with the S slice of the tensor (the past). To do so, we implement a Temporal U-Net model to fit our data and image conditioning. The architecture is from the Decision Diffuser, and it is based on residual blocks of downsampling and upsampling. The temporal information of the diffusion process is encoded with a sinusoidal positional embedding and a simple MLP with `Mish` activation function [11]. As the diffusion procedure and architecture - with its forward and backward process - follows the implementation of the Decision Diffuser, we do not go into further details here and refer to the original implementation.

Image Conditioning

To generate meaningful trajectories, we need to couple the trajectory prediction with the past and current views of the car in the traffic situation and ensure consistency with the two. To this end, we implement a feature extraction branch acting on the past and current images that is integrated into the diffusion pipeline.

As we do not have enough data to train both the diffusion model and the feature extraction backbone, we use a pre-trained image backbone. Since not only the presence of objects and obstacles is important, but also - and foremost - their position in the environment, we use pre-trained semantic segmentation backbones.

First, we have tested a non-specific segmentation backbone, the lightweight "Lite R-ASPP Network" with a MobileNetV3-Large backbone [9], with pre-trained weights from the PyTorch API[1]. To integrate it into the diffusion architecture, we take the low-resolution output of the pre-trained backbone of the segmentation model (of shape `(batch_size, 40, 16, 16)`), and we reduce its depth with a learnable 1D convolutional layer with 5 output channels. The S past and current images are sequentially fed into the segmentation-based feature extractor and their output is flattened to a vector of length $5 \cdot 16 \cdot 16 = 1280$ and then concatenated to a $1280 \cdot S$-long vector. This is then fed into a learnable linear layer that projects the stacked features to a 1280-long vector. This vector - which includes the conditioning on the past images - as it is common practice with classifier-free diffusion, gets either concatenated with the temporal information of the diffusion process or dropped out with a probability that we set to 0.1.

With only 3M parameters, this model allowed for a first image conditioning even with our resource-constrained setting, but its 20 classes are not specific for driving, and thus many important details (e.g. the lanes) are simply segmented and classified as *background*. We, therefore, tried also to substitute this generic backbone with a driving-specific backbone. The model we used is the DeeplabV3Plus [2] with MobileNetV2 as a backbone. The model is pre-trained on semantic segmentation of the urban street scene from the Cityscapes dataset [4]. Similar to the "Lite R-ASPP Network", we only take the feature output after passing through the backbone model (of shape `(batch_size, 320, 8, 8)`) and reduce the channel to 20 with learnable 1D convolutional layer. A similar feature extraction, flattening, and concatenation process is applied to the past images as in the "Lite R-ASPP Network" backbone implementation.

Other Conditioning

Image conditioning is indeed crucial information that the car has to incorporate into its decision-making, but not enough for a satisfying driving capability. In fact, one needs to condition also on the high-level commands coming from the CARLA simulator. Such commands include changing the lane left or right, following the lane, turning left or right, and keeping straight at the intersection. These six commands - as a one-hot-encoded vector - go through an MLP projecting them to a $6 \cdot 4$ vector and then back to length 6,

[1] `https://pytorch.org/vision/main/models/generated/torchvision.models.segmentation.lraspp_mobilenet_v3_large.html`

with `Mish` non-linearity at each layer. Same as the images, this encoded information is then concatenated along time information and image condition.

3.3.2 Planning

The diffused trajectory, as a sequence of waypoints, is directly transformed into actions by a PID Controller. In this stage of development, we, therefore, do not use any event prediction network to choose the best trajectory among the sampled ones, as we do not diffuse the images.

Our planning routine does not involve any hard-coded human heuristics, unlike the Interfuser's approach. In fact, it only consists of a PID Controller for each possible action (i.e. for turning and speeding). To calculate the target angle and speed, solely the current and next 2 waypoints are used. After every 0.5 seconds, we diffuse a new trajectory and use it as the new target trajectory. In Tab. 3.1 we summarize the configuration of the PID controllers.

K_p	turn-controller	1.25
	speed-controller	5.0
K_i	turn-controller	0.75
	speed-controller	0.5
K_d	turn-controller	0.3
	speed-controller	1.0

Table 3.1: PID Controllers configuration.

3.3.3 Training

We train our diffusion model in a supervised fashion to denoise future timesteps. We train with a learning rate of `1e-4` decayed every of a factor `0.8` every `10000` steps for `5M` steps with a gradient accumulation of 4 steps.

Visual inspection of the predicted waypoints has shown that without any filtering on the waypoints, the car is not able to train a meaningful trajectory, collapsing to a never-moving state. This is caused by a severe unbalancedness in the dataset, as more than 40% of the data shows a situation where the car is not moving in the whole time horizon of length $S + T$ considered by the scene. We have therefore filtered these scenes where the car is not moving for the entire observed horizon. Another similar observation shows that there is also an unbalancedness for scenes that do not show a turn in the whole temporal horizon, namely where the car is always going straight for more the $S+T$ steps of the scene. We apply stochastic filtering for dropping out these scenes, where the never-turning scene is discarded with a probability of 0.9. Without this filtering, the imbalacedness yields a bias where the car is encouraged to always go straight even in turning situations.

Another crucial observation and step for effective training is to normalize (and standardize) the waypoints values. The naive way to normalize and standardize them with the mean and standard deviation of the temporal horizon over the whole dataset is however not the optimal way as the temporal dimension of the scene introduces the distance of movement, which is not accounted for. This would therefore mean that, in a normal sampling such as that happening in a diffusion model, far away points - even at later timesteps - are very unlikely. To overcome this, we capture the mean and standard deviation of each timestep for all scenes of length $S + T$ in the collected dataset, and normalize accordingly, so that - on average - every coordinate and timestep in the scene is normalized. This, therefore, allows for a much higher sample quality, in particular for further timesteps.

Chapter 4

Experiments and Results

4.1 Waypoint generation

We first evaluate the quality of the generated waypoints by sampling 10 waypoints for each future timestep and compute the average Euclidean distance between the generated waypoints and their corresponding ground truth waypoints. For the experiments, we train our proposed model with all the data collected from towns 1, 2, 3, 5, 9, 10, and 11. The model is then evaluated on data collected from town 4, where we randomly select 100 data points from it for our evaluation.

4.1.1 Conditioning

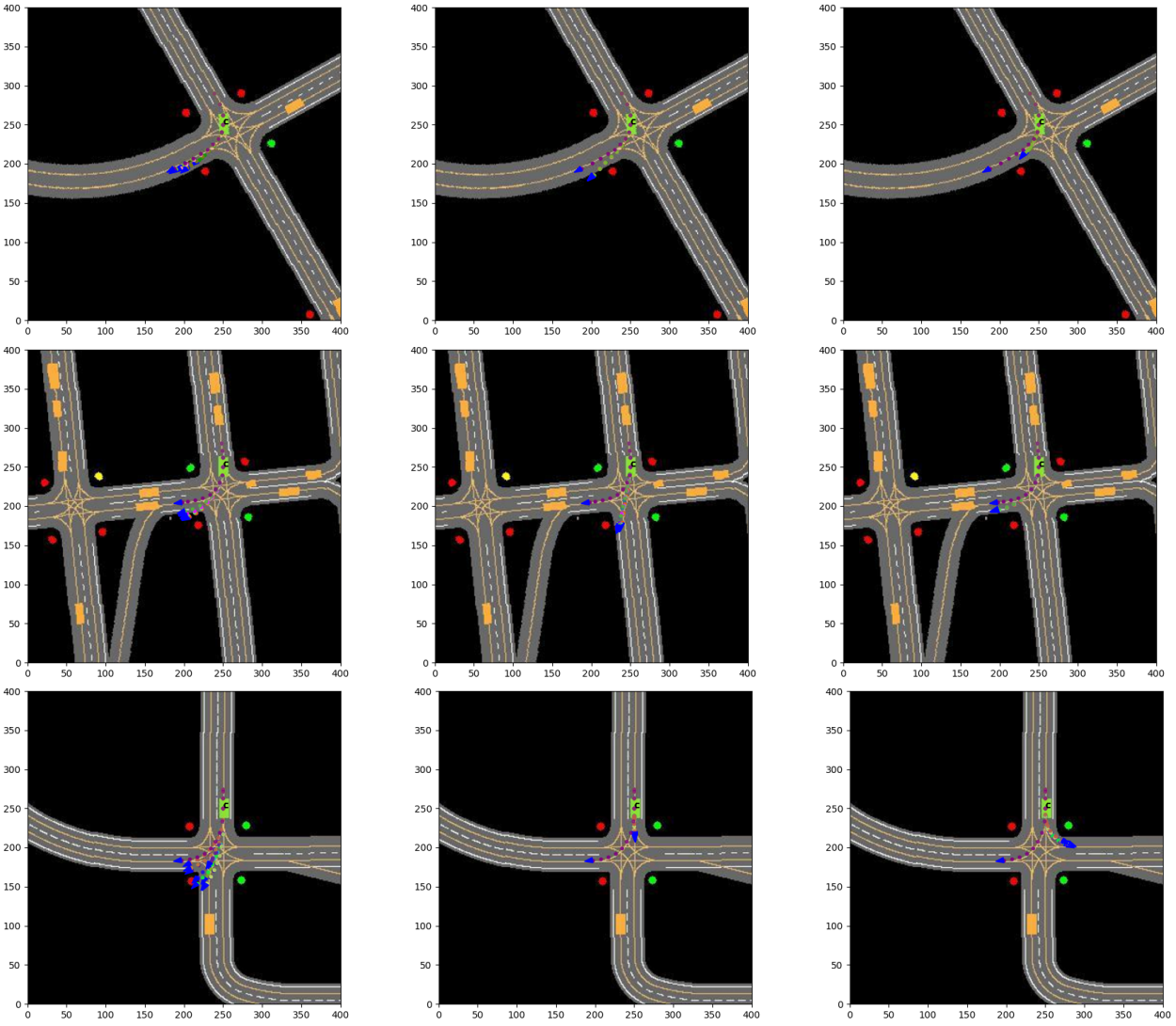
As mentioned, we tested the diffusion model with different conditioning, namely image and high-level commands. We also tested the performance of the diffusion model with two different image feature extraction models namely the Lite R-ASPP and the DeeplabV3+.

As shown in Table 4.1, the error in the predicted waypoints increases as the time step increases. This is to be expected as further into the future, the current information becomes more irrelevant thus increasing the uncertainty regarding the future waypoints. Conditioning only on high-level commands also shows the best performance than conditioning on the image only or on both image and high-level commands. The diffusion model with DeeplabV3+ as the image feature extractor outperforms the diffusion model with the Lite R-ASPP image feature extractor.

CONDITIONING	T+1	T+2	T+3	T+4	T+5	T+6	T+7	T+8
HIGH-LEVEL COMMAND	0.05	0.31	0.77	1.31	2.06	3.06	4.09	5.16
IMAGE (LITE R-ASPP)	0.09	0.35	0.84	1.66	2.67	3.99	5.54	7.21
HIGH-LEVEL COMMAND + IMAGE (LITE R-ASPP)	0.07	0.42	1.02	1.99	3.05	4.30	5.60	6.90
HIGH-LEVEL COMMAND + IMAGE (DEEPLABV3+)	0.06	0.39	0.94	1.65	2.52	3.50	4.52	5.51

Table 4.1: Average Euclidean distance between ground truth and sampled waypoints across time horizon.

In Figure 4.1, the ground truth future waypoints of the vehicle are colored indigo and the sampled waypoints are denoted in other colors. We can observe that the model with only high-level command conditioning was able to learn to generate waypoints that satisfy the current high-level command which is turning right. However, for the model conditioning on both the high-level command and the image,



(a) Command only

(b) Command + Image (Lite R-ASPP)

(c) Command + Image (DeeplabV3+)

Figure 4.1: Qualitative comparison for turning command

the model is heavily influenced by the image conditioning which results in the waypoints being generated without much influence from the high-level command.

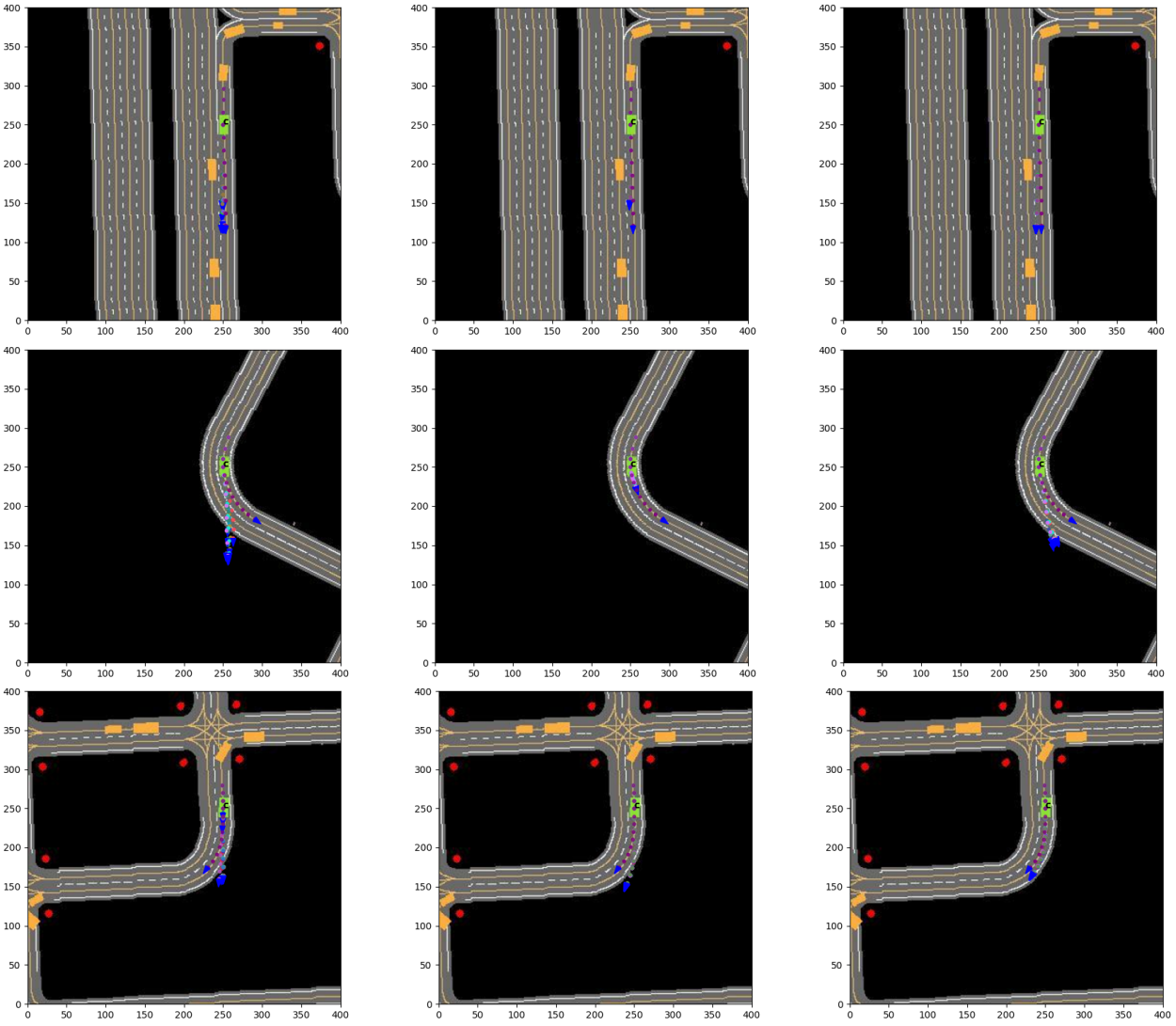
On the other hand, for cases where the vehicle is given the command to follow a lane and the lane is not straight, the diffusion model conditioning only on high-level commands is only able to maintain its motion based on its current pose and past waypoints. As shown in Figure 4.2, the diffusion model with only high-level conditioning sampled waypoints to move straight which is consistent with its past waypoints. However, for the diffusion model with high-level command and image conditioning, it is able to generate waypoints that follow the curvature of the lane.

4.2 Navigation Performance

We evaluate the navigation performance of our proposed model using the Town05 Benchmark [14]. In the benchmark, the agent is tasked to navigate the vehicle along a certain path and reach a predetermined goal. As mentioned, the agent is provided with the ego positions of the vehicle, front view images, and high-level commands for the past and current timesteps.

As observed in section 4.1.1, for the model conditioned on high-level command only, the model tends to stay fixed or move in a straight line. We hypothesize it is because of the lack of information about the environment, the model tends to sample waypoints that fit the trend of the past waypoints. We also observed that the agent exhibits random deceleration and stopping action during the simulation test.

The model conditioned on both high-level command and front view image is able to perform simple turning and lane following actions. However, the model does not exhibit any clear obstacle avoidance behavior and the traffic rules (such as traffic lights) are ignored. It is also unable to recover to its original lane when it deviates too much from its original lane. Random deceleration and stopping action were also observed during the simulation test.

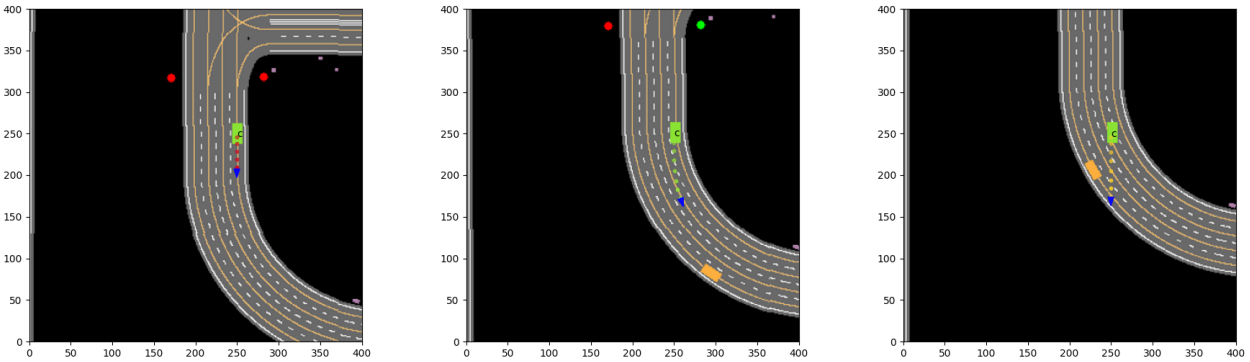


(a) Command only

(b) Command + Image (Lite R-ASPP)

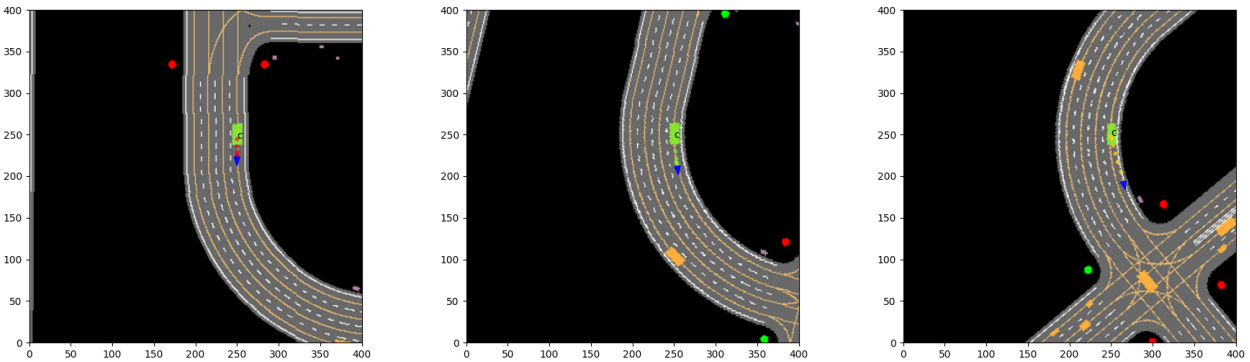
(c) Command + Image (DeeplabV3+)

Figure 4.2: Qualitative comparison for follow lane command



T = 4s T = 8s T = 12s

Figure 4.3: Sequence of trajectory generated for model conditioned on high-level command only



T = 4s T = 8s T = 12s

Figure 4.4: Sequence of trajectory generated for model conditioned on high-level command and image

Chapter 5

Discussion

Autonomous Driving is a difficult problem involving complex decision-making. This is especially challenging with an end-to-end architecture, as suggested by the CARLA leaderboard where the best implementation - the InterFuser - is a highly-specific and hard-coded implementation. Our proof-of-concept implementation, as described in Chap. 4, shows some promising trajectory predictions in simple traffic situations. It however still lacks to effectively comprehend the surrounding environment and agents, causing some detrimental predictions. In particular, it is not fully able to understand the concept of ”lane”, as it is visually difficult to decouple it from the full road and obey traffic rules such as stopping in front of red lights. Even though a driving-specific segmentation backbone slightly improves the result, combining both the high-level command and image conditioning is not a trivial task as both conditioning influence one another and a careful weighting of both conditions is needed to indeed improve the outcome. A reason for this might be that the conditioning used is too weak, and more coupling with the conditioning, e.g. with attention layers, is needed. Also, the responsiveness of the car to other agents to avoid collisions is usually too slow.

These observations suggest that further optimizations and extensions of the architecture, as well as of the training procedure, are needed to improve the planning results to an acceptable level. Our approach consists only of training the diffusion model naively, without a notion of ”goodness” of the trajectory. It would be interesting to integrate it as an RL problem with a reward-like score.

Furthermore, the current diffusion model is only trained to duplicate the actions in an ideal scenario and the dataset does not consist of scenarios where the vehicle is off track. Thus the diffusion model is unable to generate a proper recovery action when the vehicle deviates from its original track.

Finally, with enough computational resources, future work could concentrate on implementing the interpretable pipeline of Fig. 3.1 where also the future frames are diffused and predicted as interpretable middle-layer. This, besides allowing for human interpretability, could open the possibility of sampling multiple trajectories and then picking the best one among those.

Chapter 6

Conclusion

This report presents a proof-of-concept end-to-end Autonomous Driving pipeline based on Diffusion Models as a step towards developing an interpretable architecture capable of showcasing the potential benefits of using Diffusion Models in the field of Autonomous Driving. We evaluate the model with different conditioning and test it on the CARLA simulator, demonstrating simple behavior capabilities, but failing in complex scenarios. Our project provides a foundation upon which future research can build to further advance this perspective of autonomous driving.

Bibliography

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making?, 2022.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[3] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. 2022.

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[6] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: an open urban driving simulator. *CoRR*, abs/1711.03938, 2017.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020.

[8] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019.

[10] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[11] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *CoRR*, abs/1908.08681, 2019.

[12] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[13] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2022.

[14] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021.

[15] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021.

[16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[17] Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Conference on Robot Learning*, pages 726–737. PMLR, 2023.

[18] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[19] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.