Efficient Autonomous Driving with Foundation Models

by

Shadi Sameh Hamdan

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Engineering

KOÇ ÜNİVERSİTESİ

July 17, 2025

Efficient Autonomous Driving with Foundation Models

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Shadi Sameh Hamdan

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assist. Prof. Fatma Guney (Advisor)

_____

Professor Deniz Yuret

_____

Assist. Prof. Hongyang Li

Date: _____

To a Free Palestine.

ABSTRACT

Efficient Autonomous Driving with Foundation Models
Shadi Sameh Hamdan
Master of Science in Computer Engineering
July 17, 2025

Driving safely in urban environments is a task that requires a deep understanding of complex, dynamic environments and the ability to react swiftly and effectively to unexpected events. Recent advances in large foundation models have shown impressive capabilities to reason and generalize. In this thesis, we explore the potential and feasibility of foundation models for driving. First, we start by building on top of previous work showing promising performance in robotics tasks by formulating reinforcement learning as a language modeling problem using auto-regressive transformers. These approaches, however, assume the state is relatively simple, often representable as a single vector. In driving, the state is more complex, involving multiple agents that interact with each other, raising the need to find a compact, effective state representation. In Carformer, we propose a language model using learned, self-supervised, object-centric representation based on slot attention. We analyze different possible state representations in a privileged setting and find that our proposed representation outperforms both scene-level and hand-crafted object-centric representations, achieving the state-of-the-art on the Longest6 benchmark.

After showing strong results with a language modeling approach in a privileged setting, we then focus on a more realistic, end-to-end setting. While foundation models further improve with scale, this results in additional inference latency. Latency is one of the main hurdles facing the deployment of large foundation models for real-time applications like driving. To address this, we propose a dual approach pipeline, ETA, that utilizes a large foundation model in tandem with a smaller, lightweight model. To minimize latency, we shift the computation of the large foundation model to an earlier time-step following it with a forecasting module to adapt the features to the present. Unlike previous work on dual approaches, we batch the large model inference to enable the decisions to benefit from the large model features at every time-step. Outperforming all previous approaches at a fraction of the latency, we highlight the feasibility of large foundation models in self-driving.

ÖZETÇE

Yüksek Lisans Tez Başlığı
Shadi Sameh Hamdan
Bilgisayar Mühendisliği, Yüksek Lisans
17 Temmuz 2025

Kentsel ortamlarda güvenli sürüş, karmaşık ve dinamik çevrelerin derinlemesine anlaşılmasını ve beklenmedik durumlara hızlı ve etkin biçimde yanıt verebilme yetkinliğini gerektirir. Son dönemdeki büyük temel yapay zekâ modelleri (foundation models) üzerine gelişmeler, akıl yürütme ve genelleme konusunda dikkate değer yetenekler ortaya koymaktadır. Bu tezde, temel modellerin sürüş bağlamındaki potansiyelini ve uygulanabilirliğini inceliyoruz. İlk olarak, pekiştirmeli öğrenme problemini otoregresif transformer modelleri aracılığıyla bir dil modelleme problemi olarak formüle ederek, robotik görevlerde umut verici performans sergileyen önceki çalışmalara dayandırıyoruz. Bununla birlikte, bu yaklaşımlar genellikle durumun görece basit olduğunu varsayar ve tek bir vektör ile temsil edilebileceğini öngörür.

Sürüşte durum daha karmaşıktır; birbirleriyle etkileşim hâlinde çok sayıda ajan içerir. Bu nedenle, daha kompakt ve etkili bir durum temsili geliştirmemiz gerekir. Bu doğrultuda, Carformer modeli kapsamında slot attention mekanizmasına dayalı, öz-denetimli olarak öğrenilmiş, nesne-merkezli bir temsil ile çalışan bir dil modeli öneriyoruz. Ayrıcalıklı (privileged) bir senaryoda farklı durum temsillerini karşılaştırmalı olarak analiz ediyor, önerdiğimiz temsilin hem sahne düzeyinde hem de elle tasarlanmış nesne-merkezli temsillerden üstün olduğunu ve Longest6 kıyaslama testinde en iyi (state-of-the-art) performansı elde ettiğini gösteriyoruz.

Ayrıcalıklı senaryoda dil modelleme yaklaşımıyla güçlü sonuçlar elde ettikten sonra, daha gerçekçi uçtan uca bir senaryoya odaklanıyoruz. Temel modeller ölçek büyüdükçe daha gelişmiş sonuçlar üretse de, bu durum ek çıkarım gecikmesine yol açar. Gecikme, büyük ölçekli temel modellerin sürüş gibi gerçek zamanlı uygulamalarda kullanılmasının önündeki temel engellerden biridir. Bu sorunu aşmak için, büyük bir temel modeli daha küçük ve hafif bir modelle birlikte kullandığımız ikili yaklaşıma dayalı bir uygulama, ETA, öneriyoruz. Gecikmeyi en aza indirmek amacıyla, büyük temel modelin hesaplamalarını daha erken bir zaman adımında

gerçekleştiriyor ve elde edilen özellikleri, bir tahmin modülü aracılığıyla güncel zamana uyarlıyoruz. Mevcut ikili yaklaşımlardan farklı olarak, büyük model çıkarımlarını toplu biçimde işleyerek her bir zaman adımında kararların büyük model özelliklerinden yararlanmasını sağlıyoruz. Böylelikle, çok daha düşük gecikme ile tüm önceki yaklaşımları geride bırakıyor ve otonom sürüşte büyük temel modellerin uygulanabilirliğini vurguluyoruz.

ACKNOWLEDGMENTS

This work is the culmination of almost two decades of my education journey. I am immensely grateful and privileged to be where I am today, and I could not have done this without the help, guidance, and unwavering support of many friends and family along the way. First of all, I must start with my utmost gratitude to my father, Sameh, and my mother, Eiman. My father, Sameh, with his relentless focus on giving me the best possible education and consistently pushing me to further improve myself has gotten me to places I would have never reached otherwise. My mother, Eiman, was always there for me, making sure I have whatever I need to keep going. I am forever grateful for all the sacrifices and sleepless nights from my parents. I am also very grateful to my sisters, Shahad, Sara, Noor, and Zaina for all the love and support, and I am extremely lucky to have them. I would also like to shout out cockatiels Riko, Riki, Abood, Ghadeer, Miko, and Miki for being amazing little puffs of joy.

I have been very fortunate to have many people that I have the pleasure of calling my friends. Starting with my high school days, I would like to start by thanking:

Seif: For being a great friend, for being the first person I go to for tech questions, and for carrying me through all the games we play :).

Talal: For all the amazing time biking around in the streets of Beijing and Shanghai, even through the worst typhoon Shanghai has seen in almost 75 years.
My time at Koç experience was a blast, and I would like to thank:

Besher: For all the C debugging help in COMP132, career advice, and being a great inspiration for me.

Ahmed Masry: For all the long nights we spent studying and all the late night pizzas, as well as for all the mentorship.

Mohammed Al Saqqa: For all the Sadd-Radd throughout the years and for being

a great roommate and friend.

Moayad: For all the fun we had in programming contests being on team Shawerma, and for also being a great roommate.

Nazir: For being an amazing friend since undergrad, a vital member of team Shawerma, for teaching me all sorts of things and mentoring me about competitive programming, for all the ideas we bounced off each other, and for sharing our mutual love of Shawerma.

And of course, special thanks to all the Al Ghurair Foundation for Education scholars at Koç for all the fun times.

I would also like to thank all the members of the Autonomous Vision Group for creating a great working environment and being always ready to help each other. Special thanks to Volkan for being part of the crow army, Taher for always being down to go eat food wherever it is, Gorkay for all the research discussions, Sadra for being always there for any math help, Onat for answering and helping with all my CARLA questions, Rabia for all the shared suffering we went through with VQ-VAEs, and Eray, Misra, Kaan, Utku and Yaman for being great friends and amazing colleagues to work alongside. Additionally, I would like to thank the "honorary AVG members" Mert, Alessio, and Mustafa.

I would also like to express my gratitude to Prof. Deniz Yuret for his guidance and mentorship throughout the years, and to Asst. Prof. Hongyang Li for hosting me and advising me as an intern at the OpenDriveLab. Of course, I would also like to thank my advisor, Asst. Prof. Fatma Güney for all her mentorship and guidance.

My life trajectory would have been entirely different if not for the Abdulla Al Ghurair Foundation for Education supporting me generously throughout my undergraduate studies, and I am forever in debt to them.

Finally, I have been very blessed to have gone through my Master's journey with the love of my life and my best friend, Ekin. You have always been on my side through thick and thin, thank you very much for being there for me. My life is so much better with you in it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

RGB	Red Green Blue
BEV	Bird's Eye View
2D	Two Dimensional
RL	Reinforcement Learning
VQ-VAE	Vector Quantized-Variational Autoencoder
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
MLP	Multi-Layer Perceptron
MLLM	Multimodal Large Language Model
VLM	Vision-Language Model
Async	Asynchronous
BA	Block Attention
CE	Cross-Entropy
mIoU	mean Intersection over Union
ARI	Adjusted Rand Index
DS	Driving Score
RC	Route Completion
IS	Infraction Score
LR	Learning Rate
SR	Success Rate

Chapter 1

INTRODUCTION

The task of safe driving necessitates a deep understanding of the dynamic interactions between various objects within a scene and an ability to promptly handle rare, unexpected events. In self-driving, a scene is observed by the ego-vehicle via its sensors, typically multiple cameras. Then, for each observation step, an action is predicted and performed by the vehicle. Recent work in robotics formulates Reinforcement Learning (RL) as a sequence modeling problem [Chen et al., 2021a, Janner et al., 2021a, Furuta et al., 2022, Zheng et al., 2022], achieving strong performance on various benchmarks. However, while these methods show promise, they are designed for and typically evaluated on tasks where the state space is low-dimensional or can be straightforwardly encoded into a single vector. Driving, however, is a task in which the state is much more high dimensional (i.e. RGB sensor pixels). Additionally, advances in large-scale language and vision-language models [OpenAI, 2022, Chen et al., 2024b, Dosovitskiy et al., 2021] have demonstrated impressive generalization and reasoning capabilities across a range of perceptual tasks [Mao et al., 2023, Xu et al., 2024, Chen et al., 2024a, Hu et al., 2023]. Despite their perceptual strength and impressive performance, such models suffer from high inference latency due to their large size, which is a critical bottleneck in real-time applications like driving. In this work, inspired by these developments, we explore the use of language models and subsequently large foundation models for autonomous driving, where the ability to reason about complex and dynamic environments and to make decisions in real time is paramount.

To bridge the gap between recent progress in sequence modeling RL approaches and the task of self-driving, a suitable, compact representation for the state is es-

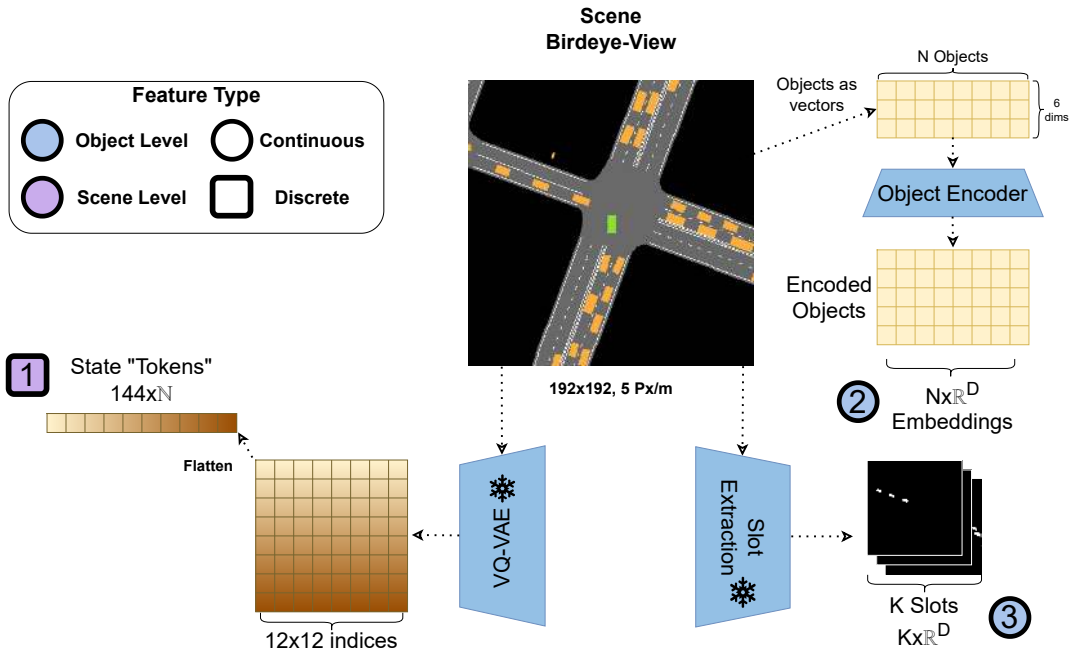

Figure 1.1: **Possible representations of the state.** In order to get discrete tokens, one possible way is to use a VQ-VAE to encode the BEV. Additionally, we experiment with object-level representations following [Renz et al., 2022], where each object is represented with a continuous vector of attributes. Finally, we propose using slot-based representations based on SAVi, where each slot captures one or more objects in the scene [Kipf et al., 2022].

sential. To tackle the prohibitively high-dimensional state space of self-driving, approaches often start by either defining or learning a lower-dimensional representation space. There are various representation spaces used in self-driving, ranging from pixels and coordinates to stixels and 3D points [Janai et al., 2020]. In recent years, efforts converged to the bird's eye view (BEV) representation [Chen et al., 2024a] which provides a top-down summary of the scene with scene elements that are relevant to driving, such as lanes and vehicles. It enables the agent to concentrate on relevant semantics, free of distractions.

In order to evaluate different possible input representations on an even playing field, we propose **CarFormer**, an auto-regressive transformer inspired by Decision

Transformer [Chen et al., 2021b] and Trajectory Transformer [Janner et al., 2021b]. Assuming that we have perfect perception and access to the ground truth BEV, we experiment with different possible state representations. In language modeling, transformers typically operate on discretized inputs, or tokens, which are vastly different from high-dimensional continuous inputs such as camera images. In our experiments, we first explore discrete tokens by discretizing the BEV with a VQ-VAE to provide it as input. Although BEV provides a summary compared to six high-resolution camera images, it is still very high-dimensional. For example, most pixels belong to the road region and vehicles cover a relatively small portion of the BEV map despite being the primary cause of infractions. Our initial experiment with discretized BEV resulted in high infractions, motivating us to shift our focus to object-centric representations.

In object-centric approaches [Wu et al., 2023, Renz et al., 2022], the scene is represented in terms of objects for better modeling of scene dynamics. Previous work has proposed representing objects with vectors containing exact object attributes, including position, size, heading, and speed [Renz et al., 2022]. Ultimately, we propose to instead learn to extract relevant information about objects from their spatial and temporal context in BEV sequences. This allows our model to decide how to represent objects without explicitly specifying a set of attributes that may be incomplete, or subject to variations depending on the scene. Using this learned, self-supervised representation, we achieve state-of-the-art performance in the privileged setting of Longest6 benchmark, outperforming exact object-level attributes.

The reliance on perfect intermediate representations, however, limits practical applicability and scalability. For example, extracting accurate BEV maps from images has proven to be a challenging problem, especially for small object classes such as motorbikes and pedestrians [Philion and Fidler, 2020, Li et al., 2022, Harley et al., 2023]. This is further highlighted by the release of the CARLA Leaderboard-v2 benchmark, introducing more complex scenarios which rendered existing approaches ineffective. Large models, including multimodal foundation models [OpenAI, 2022, Chen et al., 2024b] and vision foundation models with huge amount of parameters [Lee et al., 2019, Dosovitskiy et al., 2021], have been increasingly used in

self-driving because of their impressive perception and reasoning capabilities [Mao et al., 2023, Xu et al., 2024, **?**, Hu et al., 2023]. Due to the real-time demand of self-driving, inference-time speed becomes a critical concern when using large models. For instance, the best-performing models [Jia et al., 2025, Jia et al., 2023a] on the Bench2Drive closed-loop benchmark [Jia et al., 2024] currently have latencies in the order of hundreds of milliseconds due to their large transformer structures, which do not satisfy the real-time requirement.

The efficiency bottleneck in large models has driven researchers to develop dual approaches for embodied systems [Bu et al., 2024, Li et al., 2024b, Zhang et al., 2024a, Yu et al., 2024, Shi et al., 2025, Figure, 2025], including self-driving [Tian et al., 2024, Zhang et al., 2024c, Mei et al., 2024, Sima et al., 2025]. Inspired by the System-1 and System-2 framework from cognitive science [Kahneman, 2011], these dual frameworks utilize a small model for fast, reactive decisions and a large model for slower, more thoughtful decisions. While the dual approach facilitates the use of large models in online systems, the inference results from these large models in existing paradigms [Tian et al., 2024, Zhang et al., 2024c] are not available at every decision step. As a result, most decisions are derived from only immediate small model inference coupled with occasional delayed large model inference (See Fig. 4.1).

Our key insight is to enable large model inference at every frame by proposing the **E**fficiency through **T**hinking **A**head (ETA) pipeline, shifting the time-consuming computations of large models on the current frame to a previous time step, and performing a batch inference for multiple frames on one time to trade space complexity for time complexity, as shown in Fig. 4.1 (right). Consequently, each online frame is processed using large models from prior time steps, yielding outcomes that integrate both timely large model inference and small model inference. However, shifting the computation of the current time step to a prior time step is non-trivial, as the information of the current time step is not available in previous frames. To address this issue, we update the information from the prior time step to the current time step through future prediction, while still processing the current time step with a small model to capture changes that are difficult to predict.

We evaluate the proposed ETA framework on the challenging Bench2Drive bench-

mark in the Leaderboard-v2 setting. Our extensive experiments demonstrate that using the large model asynchronously with the smaller model effectively balances performance and latency, demonstrating significant improvements over baselines while maintaining a low latency of 50ms.

The structure of the thesis is organised as follows:

Chapter 2 discusses the related work on autonomous driving and dual-system approaches.

Chapter 3 introduces CarFormer as well as our proposed self-supervised slot-based object-level representations.

Chapter 4 introduces ETA, enabling effective utilization of large foundation models for time-sensitive tasks like driving.

Chapter 5 concludes the thesis and outlines directions for possible future work.

Chapter 2

RELATED WORK AND BACKGROUND

2.1 Representation Spaces in Self-Driving

To tackle the prohibitively high-dimensional state space of self-driving, approaches typically start by either defining or learning a lower-dimensional representation space. Earlier methods [Chen et al., 2015, Sauer et al., 2018] start with hand-crafted affordances, which are indicators that restrict the valid action space of the ego vehicle in the current scene. DeepDriving [Chen et al., 2015] propose using the angle of the vehicle compared to the current lane, the distance to the markings of the lanes to either direction, and the distance to the closest vehicle in front of the ego vehicle as affordances to enable highway driving. Urban settings present a much wider range of critical variables, including traffic signals, pedestrian activity, varied speed regulations, and a higher likelihood of encountering unexpected obstacles, which are not adequately captured by DeepDriving's initial set of affordances. To tackle this limitation, the Conditional Affordance Learning (CAL) paper [Sauer et al., 2018] builds on this approach by defining additional affordances specifically tailored for urban contexts. These include crucial indicators such as an indicator that is on if the ego vehicle is affected by a red traffic light, the current speed limit, and a binary indicator for the presence of a general hazard.

Another common representation is semantic segmentation of the scene. Some methods resort to learning to predict a 2D semantic segmentation map of the camera sensor input [Sax et al., 2019, Müller et al., 2018, Zhou et al., 2019, Mousavian et al., 2019]. These 2D semantic segmentation maps range from simple binary maps that mark each pixel as road or not-road as in [Müller et al., 2018] to more finegrained segmentations into multiple classes as in [Zhou et al., 2019], which classify pixels into road, sidewalk, terrain, or other objects like stop signs and trees.

Another direction utilizes segmentation of a predicted bird-eye-view (BEV) of the scene instead of directly segmenting the pixels of the camera sensor [Chen et al., 2019, Zhang et al., 2021, Hanselmann et al., 2022, Chen et al., 2022, Hu et al., 2023]. Learning from All Vehicles (LAV) [Chen et al., 2019] utilizes 5 classes, dividing the pixels in BEV to either background, vehicles, roads, lane markings, and pedestrians. ROACH [Zhang et al., 2021] further extends this by including traffic lights, signs, and further segmentation of roads into general drivable areas, lane boundaries, and the desired route for the vehicle.

2.2 Self-Supervised Object-Centric Representations

Self-supervised representations are a promising direction in autonomous driving due to the lack of a requirement for labeling, enabling better scaling and utilization of unlabeled data. Some self-supervised object-centric approaches learn to group pixels in an image based on low-level features like color [Greff et al., 2019, Burgess et al., 2019, Crawford and Pineau, 2019, Engelcke et al., 2020, Eslami et al., 2016, Locatello et al., 2020] or in more recent work based on semantic features extracted using frozen pre-trained deep networks [Seitzer et al., 2023]. Another direction extends object-centric approaches to videos by utilizing temporal information [Kabra et al., 2021, Kipf et al., 2022, Jiang et al., 2019, Veerapaneni et al., 2019, Zoran et al., 2021, Kosiorek et al., 2018].

In addition to pixel-level or semantic features, another line of work proposes to inject inductive biases about individual objects in the scene directly into the learned architecture, as done in Slot Attention [Locatello et al., 2020]. Methods like Slot Attention identify and learn useful features about individual objects in the scene by following an auto-encoding paradigm consisting of a set of bottlenecks. These bottlenecks in the latent space, denoted as *slots*, are expected to bind to visually similar objects and be able to reconstruct them.

The datasets used to train these models started with synthetic images [Johnson et al., 2017, Karazija et al., 2021], followed by a shift to more real-world and diverse images [Everingham et al., 2010, Lin et al., 2014] and videos [Ochs et al., 2013,

Perazzi et al., 2016, Xu et al., 2018, Yang et al., 2019, Li et al., 2013]. Due to the difficulty of reconstructing objects, especially in real world datasets, methods have resorted to instead learn to reconstruct flow [Kipf et al., 2022], depth [Elsayed et al., 2022] or motion segmentation masks [Bao et al., 2022, Bao et al., 2023]. Despite promising results shown in recent work [Seitzer et al., 2023, Aydemir et al., 2023] which relies on reconstruction in the latent space of self-supervised models [Caron et al., 2021], extracting slots from complex driving sequences remains a significant challenge.

2.3 Auto-regressive Transformers for Sequential Modeling in Autonomous Driving

Transformers are a dominant architecture in sequential modeling tasks in vision such as image or video generation [Yan et al., 2021, Ren and Wang, 2022, Micheli et al., 2023, Nash et al., 2022]. Due to the high dimensionality of images, a common strategy is to first quantize the input by mapping it to discrete tokens [Esser et al., 2021] and then training the transformer with the discrete tokens as input.

Previous work in robotics formulates Reinforcement Learning as a sequence modeling problem [Chen et al., 2021a, Janner et al., 2021a, Furuta et al., 2022, Zheng et al., 2022]. [Chen et al., 2021a] propose the Decision Transformer, which when conditioned on the desired return, past states, and past actions, results in a policy. Janner et al. propose the Trajectory Transformer as both a policy and a dynamics model to predict the next states [Janner et al., 2021a]. Furuta et al. propose a generic formulation covering both variants [Furuta et al., 2022] and Zheng et al. extend the formulation in [Chen et al., 2021a] with online fine-tuning [Zheng et al., 2022]. While these methods show promise, they are tested on problems where the state space is low-dimensional or where the state can simply be encoded into a single vector. For instance, the design of Decision Transformer [Chen et al., 2021a] assumes that the state is representable with a single token, encoding input images with a CNN [Chen et al., 2021a]. Trajectory Transformer [Janner et al., 2021a] proposes working with discrete inputs. To handle continuous inputs, each dimension of the

input vector is discretized separately, which is prohibitive for high-dimensional inputs. Autonomous Driving involved significantly a high-dimensional state compared to the state in tasks tackled by Decision Transformer and Trajectory Transformer. However, their insight remains useful for the case of the action, which typically involves two scalar values, steering and acceleration.

2.4 Dual Approach to Embodied Systems

Autonomous Driving: Due to the large computational cost of LLMs and VLMs, a dual approach similar to our proposed approach has recently been explored for driving. DriveVLM-Dual [Tian et al., 2024] uses a larger VLM running asynchronously with a smaller traditional modular pipeline for 3D perception and motion planning by refining coarse waypoints predicted by the VLM. Despite being heavily optimized, 2B-parameter VLM takes 300ms to run, rendering closed-loop evaluation infeasible. Given observations and high-level textual instructions, AD-H [Zhang et al., 2024c] uses an LLM to output a higher level plan in the form of mid-level driving commands, such as turn left, etc., and conditioned on that, a small LLM predicts low-level waypoints.

LeapAD [Mei et al., 2024] proposes a dual approach with a heuristic system for fast decisions based on samples retrieved from a memory and a slow analytical system that uses GPT4 for reasoning in the case of failures of the fast part and then stores the corrected action in the memory.

In a dual paradigm, the frequency of querying large model vs. small model is an important design decision, resulting in a trade-off between accuracy and speed. Our work distinguishes from prior works in that we enable utilization of relevant large model features at every time step. We propose an efficient dual framework with batched predictive large model inference and timely small model adjustment.

Robotics: LLMs and VLMs are significantly welcomed in the robotics domain to provide open-world reasoning [Driess et al., 2023, Zitkovich et al., 2023]. Very recent works resort to a hierarchical structure to combine the large foundation models and low-level policy networks as a dual framework to realize both instruction

decomposition and real-time control [Zhang et al., 2024a, Shentu et al., 2024]. Li *et al.* [Li et al., 2024b] propose to encode observations to latents with a VLM in low frequency and employ an action decoder conditioned on the latents with high prediction frequency. Instead, RoboDual [Bu et al., 2024] and Hi Robot [Shi et al., 2025] integrate different models for the dual system, adopting vision-language-action models for System-2 and System-1, respectively. This methodology has also been successfully applied in industrial companies such as Figure AI [Figure, 2025], where generalized performance and efficient deployment are achieved for whole upper-body control of humanoid robots.

2.5 Recent Progress on CARLA

Closed-Loop Large Models on Leaderboard-v1: Earlier autonomous driving frameworks utilizing large models evaluate on Town05 [Wang et al., 2023, Mei et al., 2024] or LAV benchmark [Zhang et al., 2024b] in Leaderboard-v1 setting for closed-loop evaluation. These approaches first process the input using a large encoder, such as Qformer [Wang et al., 2023] or CLIP ViT [Zhang et al., 2024b], to tokenize the scene or a VLM to create a textual description of the scene [Mei et al., 2024] and then feed its output to an LLM/MLLM to predict action. LangAuto, another benchmark used by LLM/VLM-based agents [Shao et al., 2024, Zhang et al., 2024c], provides a semi-closed-loop evaluation but with predefined routes with explicit instructions as input instead of route waypoints.

Leaderboard-v2: Due to the challenging long scenarios in CARLA Leaderboard-v2 , there are only a few submissions, each scoring extremely low driving scores. The modular [Zhang et al., 2024c] or hybrid [Kaljavesi et al., 2024] approaches fall behind end-to-end approaches. End-to-end imitation learning approaches require high-quality driving demonstrations from an expert. There are two experts, i.e., with access to privileged data, commonly used for data collection in the Leaderboard-v2 setting: Think2Drive [Li et al., 2024a], which is a model-based RL agent, and PDMLite [Beißwenger, 2024, Sima et al., 2024], which is a rule-based planner. TransFuser++ [Jaeger et al., 2023], the best imitation learning agent in Leaderboard-v1

, performs the second best in Leaderboard-v2 using data collected by the PDMLite expert.

LLM4AD/CarLLaVA [Renz et al., 2024], the winner of the CARLA Autonomous Driving Challenge 2.0, encodes the input with LLaVA-Next and then feeds its output to an LLM to predict action. Our base model is similar to LLM4AD/CarLLaVA but is still far behind real-time constraints at 102 ms, which is improved to 50 ms with our dual approach. As of December 2024, the CARLA leaderboard test server is temporarily closed and does not accept submissions.

Bench2Drive Benchmark: Due to the difficulty of comparing methods without significant differences in driving scores on challenging, long test routes of Leaderboard-v2, Bench2Drive [Jia et al., 2024] creates a benchmark with short routes, each focused on evaluating a distinct skill set. Several previous end-to-end driving methods are benchmarked by training on official training data collected by the Think2Drive expert [Li et al., 2024a]. DriveTransformer [Jia et al., 2025] proposes a transformer framework with different types of attention, achieving SOTA performance on Bench2Drive; however, it still has a high latency. Our proposed dual framework, outlined in Chapter 4, achieves the best results with significantly lower latency. Recent work [Zimmerlin et al., 2024, Arasteh et al., 2024] reports results by collecting data from other experts, creating an unfair comparison due to changes to training data in terms of resolution, diversity, etc. To ensure a fair comparison with a fixed dataset, we restrict our comparison to models that are explicitly trained only on the publicly available Bench2Drive base training set.

Chapter 3

CARFORMER: SELF-DRIVING WITH LEARNED OBJECT-CENTRIC REPRESENTATIONS

The choice of representation plays a key role in self-driving. Bird's eye view (BEV) representations have shown remarkable performance in end-to-end learning of driving. In this work, we propose to learn object-centric representations in BEV to distill a complex scene into more actionable information for self-driving. We first learn to place objects into slots with a slot attention model on BEV sequences. Based on these object-centric representations, we then train a transformer to learn to drive as well as reason about the future of other vehicles. We found that object-centric slot representations significantly outperform a baseline based on directly encoding the BEV representation of the scene with a VQ-VAE and lead to similar performance compared to object-level approaches that use the exact attributes of objects. This shows that slot representations naturally incorporate information about objects from their spatial and temporal context such as position, heading, and speed despite this information not being explicitly provided during training. We analyze the factors that affect slot extraction and behavior learning with slots and find that increasing the number of slots and enlarging small objects improve performance significantly.

3.1 Background on Slot Extraction

Given the BEV representation of the scene in the last T time steps, we first use a frozen object-centric model to extract the objects into slots. For extracting slots, we build on slot attention for videos, SAVi [Kipf et al., 2022]. In this section, we provide a brief overview of SAVi for completeness and to introduce the notation with slots. Further details about SAVi training can be found in [Kipf et al., 2022].

As a reconstruction-based method, SAVi follows an auto-encoding paradigm.

Given a sequence of RGB frames $\mathbf{x}_{t-T:t}$ with T time steps for context, a CNN-based encoder is used to process each frame $\mathbf{x}_i$ with positional encoding. The output of the encoder is flattened into a set of vectors $\mathbf{h}_{t-T:t}$.

SAVi first initializes K slot vectors $\tilde{\mathcal{Z}}_{t-T} = \{\mathbf{z}^i_{t-T}\}^K_{i=1}$. The initial set of slot vectors $\tilde{\mathcal{Z}}_i$ is then updated with Slot Attention (SA) [Locatello et al., 2020] based on the visual features $\mathbf{h}_i$ from the encoder for each time step i within the interval, resulting in the updated set of slot vectors $\mathcal{Z}_i$:

$$\mathcal{Z}_i = f_{SA}\left(\tilde{\mathcal{Z}}_i, \mathbf{h}_i\right) \tag{3.1}$$

To ensure temporally consistent slot representations, the subsequent slots are initialized based on the slot representation of the previous time step: $\mathcal{Z}_{i+1} = f_{pred}\left(\mathcal{Z}_i\right)$. In SAVi, slots are decoded into RGB predictions, and an alpha mask per slot for computing the reconstruction loss. We use the latent slot features $\mathcal{Z}_t$ to describe time step t in our model, i.e., without decoding.

3.2 Methodology

We introduce CarFormer for learning to drive in the urban environment of CARLA [Dosovitskiy et al., 2017]. Urban driving presents complexity due to the interactions between the ego-vehicle and other vehicles. Our goal is to learn driving behavior by capturing scene dynamics through slot representations. We formulate the behavior learning as a sequence modeling problem, as illustrated in Fig. 3.1. This sequence comprises tokens representing the goal, state, and action. With this formulation, we can compare different state representations while keeping the representation of the goal and the action fixed. We first define representations for each aspect before detailing the model architecture.

3.2.1 Goal and State Representation

We encode the route to follow and the state of the world in terms of tokens, which can be continuous or discrete, and feed them into our model. Specifically, we provide the model with the next target point in the route (g^x_t, g^y_t), a flag signifying whether

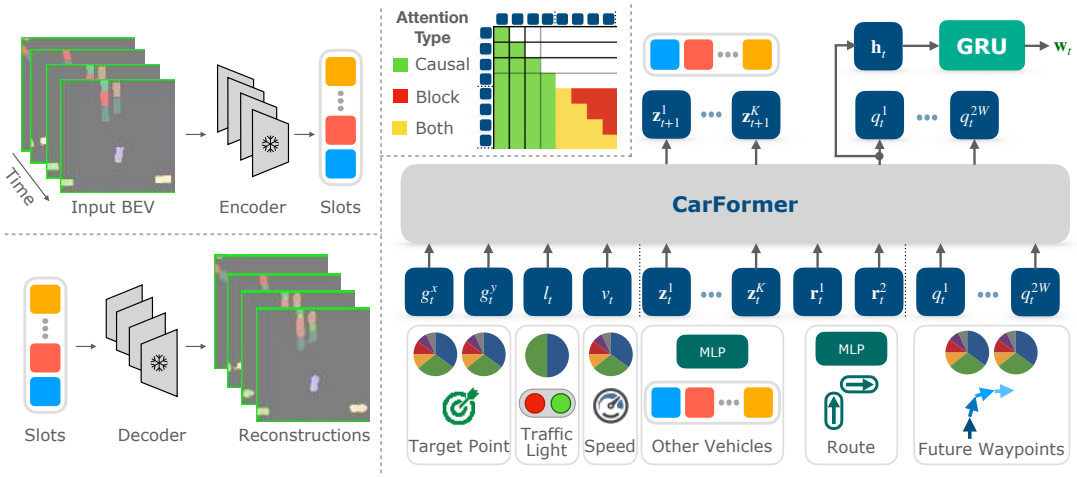

Figure 3.1: **Overview of CarFormer.** Given a trajectory τ_t consisting of discrete and continuous inputs, we first embed these tokens to the same hidden dimension H. For scalar inputs such as target point (g_t^x, g_t^y), traffic light flag l_t, speed v_t, and waypoints q_t^i, we discretize them, using k-means if not discrete, before we look them up from an embedding matrix. For continuous inputs like the K slot features $\{\mathbf{z}_t^i\}_{i=1}^K$ and the desired route $\mathbf{r}_t^1, \mathbf{r}_t^2$, we project them using an MLP. Conditioned on this context, CarFormer learns to jointly predict future slot features $\{\mathbf{z}_{t+1}^i\}_{i=1}^K$ and the waypoints autoregressively using the backbone $(\mathbf{q}_t)$ as well as the GRU head $(\mathbf{w}_t)$. The K slot features $\{\mathbf{z}_t^i\}_{i=1}^K$ are extracted from a pre-trained, frozen, SAVi model, shown on the left. The K slot features, along with the desired route, are considered a block, and block attention is applied in the attention layers as shown on the top.

the ego vehicle is affected by a red traffic light $l_t \in \{0, 1\}$, and the current speed of the ego vehicle $v_t \in \mathbb{R}_0^+$. For each of the continuous attributes, we apply k-means clustering and quantize them into k_{attr} bins. For scene representation, we initially consider a scene-level representation by directly encoding the BEV map, and then we explore object-level representations.

Scene-Level Representation

The input consists of a BEV representation of the scene at time t, denoted as $\mathbf{B}_t \in 0, 1^{192\times192\times8}$, centered around the ego-vehicle at time t (i.e. we use an ego-centric

coordinate frame at each time-step). Each of the 8 channels represents a binary map corresponding to a semantic class, such as road and vehicles [Zhang et al., 2021]. Following common practice [Yan et al., 2021], we use a VQ-VAE [van den Oord et al., 2017] to encode $\mathbf{B}$ into a grid of discrete integers: $\mathbf{b}_t \in \{1, \dots, C\}^{12\times12}$ where C denotes the codebook size. We then flatten this grid to obtain a representation of the scene as a set of discrete tokens. Despite successful reconstructions, our experiments show that learning successful driving behavior on top of these discretized tokens is challenging (see Table 3.1).

Object-Level Representations

An alternative approach to representing the entire scene as a rasterized BEV is to represent individual objects within the scene. In PlanT [Renz et al., 2022], both vehicles and the desired route are represented as 6-dimensional vectors representing essential information such as size, position, orientation, and speed. Following PlanT, we initially train our model using the exact attributes of objects to represent the scene. This allows us to attribute improvements directly to the proposed object-centric representation with slots rather than our modifications to the transformer, such as block attention.

In this work, we propose an alternative object-level representation based on slots. Building on advancements in self-supervised object-centric representations, we explore slots as a more natural way of representing objects compared to exact object attributes. With slots, objects can be implicitly represented with relevant information from the spatio-temporal context of the object. Given the BEV representation of the scene in previous T time steps, denoted as $\mathbf{B}_{t-T:t}$, we employ SAVi [Kipf et al., 2022] to extract objects into K slots, $\{\mathbf{z}_t^i\}_{i=1}^K$ where each $\mathbf{z}_t^i \in \mathbb{R}^{1\times d}$ is a d-dimensional slot vector corresponding to an object in the scene. As object-level representations do not include any information on the desired route, we also provide the model with the desired route represented by two vectors $\mathbf{r}_t^1, \mathbf{r}_t^2 \in \mathbb{R}^6$ following PlanT [Renz et al., 2022].

3.2.2 Action Representation

Following common practice in self-driving [Renz et al., 2022, Hanselmann et al., 2022, Chitta et al., 2023], we predict waypoints that are then used to calculate the corresponding control consisting of throttle, brake, and steering angle. We predict waypoints in two ways:

GRU: One way is to use a small GRU [Cho et al., 2014] as in PlanT [Renz et al., 2022] The GRU is fed with the last latent vector from the backbone, concatenated with a flag representing the traffic light status. With the GRU head, we sequentially predict W future waypoints $\mathbf{w}_t = \{(w_{x,i}, w_{y,i})\}_{i=t+1}^{t+W}$ one by one in an autoregressive manner.

Quantization: Another way is to use our autoregressive transformer backbone and treat waypoint prediction as a next-word problem, a common strategy in language modeling approaches for RL [Chen et al., 2021a, Janner et al., 2021a, Shafiullah et al., 2022]. To implement this, we quantize the 2D waypoints by using k-means clustering on each dimension separately. As a result, W waypoints are represented by a vector $\mathbf{q}_t \in \{1, \ldots, k_q\}^{2W \times 1}$, and then each dimension is predicted sequentially, conditioned on the previous predictions.

3.2.3 CarFormer

Our goal is to learn self-driving in urban environments while jointly reasoning about scene dynamics as a sequence prediction task. At each time step, we represent the state of the world with a set of tokens as previously defined. We define a trajectory τ_t at time step t as follows:

$$\tau_t = \{g_t^x, g_t^y, l_t, v_t, \mathbf{z}_t^1, \ldots, \mathbf{z}_t^K, \mathbf{r}_t^1, \mathbf{r}_t^2, q_t^1, \ldots, q_t^{2W}\} \tag{3.2}$$

where g_t^x, g_t^y denote the target point, l_t the traffic light, v_t the speed, $\{\mathbf{z}_t^i\}_{i=1}^K$ object-level slot representations, and $\mathbf{r}_t^1, \mathbf{r}_t^2$ two attribute vectors representing the desired route. In PlanT style representation, slot vectors are replaced by the object attributes. In the case of scene-level representation, object-level slot representations

and the desired route are replaced by the discrete tokens from the VQ-VAE, represented as $\{b_i^i\}_{i=1}^C$.

Encoding

For each input in the trajectory, we handle discrete inputs, such as quantized target points and traffic light status, by performing a lookup from an embedding matrix. In the case of continuous attributes, like slot vectors, we project the vectors into the desired dimensionality using an MLP, as illustrated in Fig. 3.1. Specifically, for each discrete attribute, we initialize a $k_{attr} \times H$ embedding matrix, where k_{attr} represents the number of possible values of the attribute after discretization, and H denotes the hidden dimension of the backbone. Conversely, for continuous attributes, we utilize an MLP to project the vectors into $\mathbb{R}^H$.

Architecture

The backbone of the CarFormer is an autoregressive transformer decoder adapted from the architecture of GPT-2 [Radford et al., 2019]. We modify the embedding layer to accommodate both continuous and discrete inputs simultaneously, enabling the incorporation of continuous representations such as slots while keeping other inputs like speed and traffic light discrete. Furthermore, we adjust the attention mechanism, which is causal in transformer decoders, to allow for cross-attention between certain blocks of the input. These blocks can be considered as a single unit, despite being composed of multiple tokens, such as the slot representations. To achieve this, we replace the triangular causal attention mask with a block triangular mask, enabling cross-attention within the block as shown on top in Fig. 3.1. We use this attention mechanism throughout our experiments and evaluate its impact in Table 3.2.

Training

While our formulation can be extended to RL with rewards and multi-step predictions, currently, it corresponds to imitation learning with a single-step policy to

predict action based on context. We train the model using imitation learning to learn to predict both the continuous waypoints $\mathbf{w}_t$ and their quantized form, $\mathbf{q}_t$ given the context as shown in (3.2). We supervise both the GRU head, responsible for predicting continuous waypoints, and the language modeling head, responsible for predicting the quantized waypoints. To achieve this, we use the following loss functions:

$$\begin{aligned} \mathcal{L}_{wp} &= \mathcal{L}_{\text{GRU}} + \mathcal{L}_{\text{LM}} \\ \mathcal{L}_{\text{GRU}} &= \sum_{i=1}^{W} |\mathbf{w}_t^i - \hat{\mathbf{w}}_t^i| \\ \mathcal{L}_{\text{LM}} &= \sum_{i=1}^{W} \sum_{j=1}^{2} \text{CE}(\mathbf{q}_t^{i,j}, \hat{\mathbf{q}}_t^{i,j}) \end{aligned} \tag{3.3}$$

Auxiliary Forecasting: Similar to PlanT [Renz et al., 2022], we additionally train the model to predict future scene representations jointly with action. In the case of the discretized scene representation using the VQ-VAE, we calculate the cross-entropy between the predicted logits and the ground truth future representation:

$$\mathcal{L}_{\text{scene}} = \sum_{i=1}^{C} \text{CE}(\mathbf{b}_{t+f}^i, \hat{\mathbf{b}}_{t+f}^i) \tag{3.4}$$

where f denotes the time horizon into the future for which we make predictions.

In the case of continuous scene representations, such as in object-level vectors or slot representations, we instead use the mean squared error between the representations:

$$\mathcal{L}_{\text{object}} = \sum_{i=1}^{K} \|\mathbf{z}_{t+f}^i, \hat{\mathbf{z}}_{t+f}^i\| \tag{3.5}$$

The final loss is a weighted sum of the losses on action (3.3) and forecasting:

$$\mathcal{L}_{wp} + \alpha \, \mathcal{L}_{\text{forecast}} \tag{3.6}$$

where $\mathcal{L}_{\text{forecast}}$ corresponds to $\mathcal{L}_{\text{scene}}$ (3.4) in case of scene-level and $\mathcal{L}_{\text{object}}$ (3.5) in case of object-level. We experimentally set α, the weight of forecasting, to 40.

3.3 Experiments

In this section, we first describe the experimental setup for training and evaluation in Section 3.3.1. Then, we examine the results of our experiments, quantitatively in Section 3.3.2 followed by qualitatively in Section 3.3.3.

3.3.1 Experimental Setup

CARLA Setting: We collect training data using the setup introduced in TransFuser [Chitta et al., 2023] on CARLA version 0.9.10.1, as in PlanT [Renz et al., 2022]. As PlanT [Renz et al., 2022] shows that scaling the dataset size by collecting the data multiple times with different seeds leads to performance improvements, we adopt the $3\times$ setting, i.e. collecting the data three times, as our default configuration. Our evaluation is conducted on the Longest6 benchmark, as proposed in TransFuser [Chitta et al., 2023], comprising the six longest routes across six different towns. Further details on the expert driver and data collection routes can be found in TransFuser [Chitta et al., 2023].

Data Filtering: During data collection, we filter out expert runs that exhibit problematic behavior, typically occurring with specific seeds. We perform filtering by repeating runs where the expert achieves a driving score of less than 50% with a different seed, ensuring consistency in dataset size. The filtered instances commonly involve vehicles spawned in orientations hindering movement, leading to expert timeouts, or scenarios failing to trigger, causing the expert to remain immobilized until timing out.

Evaluation Setting: During online evaluations, unless stated otherwise, we employ the model with slots as input, $t+4$ as the target for future prediction, and the GRU head for waypoint prediction. Additionally, we implement creeping to prevent the agent from becoming stuck.

Metrics: When assessing driving performance, we report the metrics used in the CARLA leaderboard [Dosovitskiy et al., 2017], namely *Route Completion (RC)*, *Infraction Score (IS)*, and *Driving Score (DS)* as the combination of the two. For

evaluating the predicted dynamics, we report the foreground version of the *Adjusted Rand Index (ARI)* and mean *Intersection over Union (mIoU)*. These metrics are computed by comparing predicted slot masks to target slot masks at a future time step.

Baselines: Since we operate in the privileged setting, we benchmark against other privileged baselines. Specifically, we compare with AIM-BEV introduced in [Hanselmann et al., 2022] and ROACH [Zhang et al., 2021] which use ground truth BEV as input. Additionally, we compare with PlanT [Renz et al., 2022], which incorporates ground truth object-level vehicles and routes as input. We obtain the results for AIM-BEV and ROACH directly from [Renz et al., 2022] and evaluate the publicly available PlanT-Medium checkpoint, trained on the 3× dataset on Longest6.

Implementation Details: Our dataset is divided into a training set (94% of the data), and validation and test sets (3% each). We train for 100 epochs on the training set and select the checkpoint with the best validation loss for online evaluation. We set the hidden dimension of the transformer as $H = 768$ and the number of layers to 6. In k-means, we use $k_g = 32$ for the target point, $k_v = 14$ for the speed, and $k_q = 48$ for the quantized waypoints. For the target point and waypoints, we allocate half of the dimension for x and the other half for y. In SAVi, we set the number of context frames T to 2 and the slot dimensionality d to 128 and train it from scratch on a subset of our training split. Additionally, we process the input BEV representations by assigning a different color to each vehicle, which we found to help in slot extraction. To color the vehicles, we sample a random color per vehicle from a set of 14 colors and fix this color by vehicle across time.

We train our model for 100 epochs on the full dataset and pick the checkpoint with the best validation loss to evaluate. We use a setup of either 4× A6000s or 4× A100s to train our models. We keep the effective batch size as 512 across runs.

3.3.2 Quantitative Results

Comparison: We present the results of online evaluations on the Longest6 benchmark in Table 3.1. The table is divided into three sections based on the type

Model	**Representation**	**DS↑**	**IS↑**	**RC↑**
CarFormer	BEV	17.07±3.78	0.30±0.04	59.54±4.34
AIM-BEV*		45.06±1.68	0.55±0.01	78.31±1.12
ROACH*		55.27±1.43	0.62±0.02	88.16±1.52
CarFormer	Attributes	71.53±3.52	0.78±0.06	90.01±1.60
PlanT (Rep.)		73.36±2.97	**0.84±0.01**	87.03±3.91
CarFormer	Slots	**74.89±1.44**	0.79±0.02	**92.90±1.28**

Table 3.1: **Comparison on Longest6.** In the top part, we report the scene-level comparison where CarFormer significantly falls behind the other two approaches, AIM-BEV [Hanselmann et al., 2022] and ROACH [Zhang et al., 2021]. In the middle, we report results using exact, object-level attributes, where CarFormer achieves a driving score within statistical significance of PlanT. With object-level slot representations shown in the bottom, CarFormer outperforms all scene-level approaches and even surpasses object-level approaches based on explicit object attributes. We report mean±std over 3 different runs. We report the results of PlanT [Renz et al., 2022] reproduced (Rep.) using their official code. *Results reported in PlanT.

of representation: scene-level representation at the top, followed by exact object-level attributes, and object-level slot representations at the bottom. In scene-level representations, CarFormer lags behind another imitation learning approach AIM-BEV [Hanselmann et al., 2022], and an RL approach, ROACH [Zhang et al., 2021]. Despite accurately reconstructing input BEV with a VQ-VAE, the model cannot focus on objects, as evidenced by the significantly lower infraction score (IS).

Compared to scene-level representation with VQ-VAE, we observe significant performance improvements with object-level representations. CarFormer with slots outperforms PlanT in RC despite a lower IS due to covering a longer distance, resulting in a higher mean DS with only half the variance compared to PlanT (Table 3.1). This achievement is particularly noteworthy for two reasons: First, the slots model (bottom row) achieves this solely from BEV. While PlanT and CarFormer with at-

Action	BA	Forecasting	Creeping	DS↑	IS↑	RC↑
GRU	✓	✓	✓	**74.89±1.44**	0.79±0.02	**92.90±1.28**
Q	✓	✓	✓	66.87±0.96	0.74±0.01	87.12±0.89
GRU	×	✓	✓	70.42±2.68	0.78±0.02	88.19±2.00
GRU	✓	×	✓	57.25±2.89	0.63±0.03	89.54±1.48
GRU	✓	✓	×	69.16±2.20	**0.83±0.01**	80.52±1.88

Table 3.2: **Ablation Study.** We first compare predicting continuous actions with the GRU to predicting quantized actions with the transformer (Q). We also ablate the effect of removing Block Attention (BA) and slot forecasting and creeping. We present the results as the mean ± std over 3 different runs on Longest6.

tributes have access to exact agent locations, the slots model learns to accurately place agents in slots. Second, the significantly lower variance with slots demonstrates increased stability across runs, affirming slots as a more reliable alternative to attribute vectors. Note that the improved performance of our model cannot be attributed to architectural changes, as CarFormer with attributes performs worse than PlanT with higher variance.

Ablation Study: In Table 3.2, we perform an ablation study to evaluate our design choices when training and evaluating our model with slots as input. Specifically, we evaluate the effect of removing block attention, forecasting slots, and creeping when the vehicle is stuck during online evaluation. In the complete version of our model, we compare the results when using action predictions from either the GRU or the transformer itself with quantization (Q). We observed a significant improvement in results with the GRU and consequently adopted it for the remaining experiments.

The negative effect of removing creeping can be seen in a notably lower route completion score. Without creeping, the agent suffers from more frequent instances of getting stuck, leading to a decreased completion rate of assigned routes. This cautious behavior leads to a slightly higher IS but a lower overall DS. With BA, the agent can better model relationships between all objects in the scene due to the

bi-directional attention within a block. This is shown in the third row, where removing BA results in increased infractions and a lower RC score. Adding forecasting significantly improves driving performance. The effect of forecasting is most visible in infraction scores, where removing it leads to a 0.17 decrease in IS ($0.78 \rightarrow 0.63$). This shows the importance of formulating the driving task jointly with forecasting, allowing the agent to anticipate the intentions of other vehicles and act accordingly.

Slot Representations: When examining the performance of SAVi on BEV sequences, we identified two primary factors contributing to failure in slot extraction. Firstly, a low number of slots leads to missing vehicles as all slots become filled, particularly in crowded urban driving scenes. As demonstrated in Table 3.3, the performance improvement from 7 to 30 slots clearly highlights the advantage of setting the number of slots high enough to capture the majority of vehicles in the driving environment. However, adding more slots increases the computational load of training SAVi. To address this, we developed a lighter decoder to maintain efficiency while accommodating more slots.

Increasing the number of slots leads to significant performance improvements, particularly in terms of infractions. Upon investigating the infractions with the lower number of slots, we discovered that most of them stemmed from small vehicles such as motorbikes or bicycles. Small vehicles, covering a relatively small area on the final BEV map, were often missed while decoding slots due to the small cost paid in terms of reconstruction loss. To test our hypothesis, we enlarged these small vehicles to increase their likelihood of being assigned to a slot. As shown at the bottom part of Table 3.3, enlarging small vehicles notably enhances the performance, particularly with a small number of slots. The best performance is obtained with 30 slots and enlarging small vehicles.

Ablation of Light SAVi: We modify the architecture of SAVi to be able to increase the number of slots while keeping slot extraction feasible. Specifically, we propose a lighter decoder by decreasing the number of parameters. To measure the effect of these changes quantitatively, we train our model with the light decoder while keeping the number of slots constant at 7. Note that, training the slot extraction

model is prohibitively slow with the original decoder when we increase the number of slots to 30.

We evaluate the forecasting performance of the model in Table 3.4. Although the performance degrades slightly with the light decoder in the case of 7 slots, increasing the number of slots to 30 improves performance noticeably. Increasing the number of slots is not only crucial to improve the performance in terms of slot extraction, it also proves to be crucial in terms of driving performance as we show in (Table 3.3).

To further evaluate its effect on driving performance, we compare CarFormer using the original SAVi checkpoint with 7 slots (full decoder) as well as the light decoder version with 7, 14, and 30 slots in Table 3.5. The results show that increasing the number of slots improves the driving performance and the best driving performance is achieved with 30 slots which was only made possible with the light decoder.

Forecasting Future Slots: So far, we evaluated the performance of our model as a policy function to predict action. Additionally, our model can also predict future states of objects by learning to match the slot representations of SAVi in future time steps via the loss function defined in (3.5). Therefore, CarFormer can be evaluated as a world model for predicting future states in addition to action. Similar to prior work on visual dynamics models [Wu et al., 2023], we evaluate our model's performance in directly predicting future slot representations 1 or 4 timesteps ahead in Table 3.6. To be able to use object discovery metrics ARI and mIoU, we decode the predicted slot representations using the frozen decoder of SAVi.

We report the performance of directly copying input as a sanity check (Input-Copy) and the performance of SAVi as an upper bound since SAVi has access to the current frame. Input-Copy works well for predicting the near future but degrades while predicting timestep $t+4$ due to objects moving away from their initial position. While there is a drop in our model's performance in predicting timestep $t+4$ as well, its predictions are more accurate in all metrics. Our model can learn the dynamics of objects in the scene and change their future position and orientation with respect to input.

3.3.3 Qualitative Results

In Fig. 3.2, we visualize predictions of our model (third column) in comparison to ground truth (first column) and SAVi reconstructions (second column) which we use to supervise our model for predicting future slot representations. In the first three columns, we display the initial locations of the vehicles in dark grey and the final locations that we aim to predict in light grey. In the last column, we overlay the final positions in all three columns, each represented in a different channel: ground truth in red, SAVi in green, and CarFormer in blue. This allows us to clearly identify various types of errors for our model in a specific color such as false negatives in yellow and false positives in blue.

Proper Dynamics: As illustrated in examples (a), (c), (d), and (e), our model effectively learns visual dynamics between objects. In (a), our model recognizes that the vehicle in the bottom left is moving backward (downward), and accurately predicts its absence in the future. In the same example, our model correctly infers that vehicles on the right are heading forward at a slower speed than the ego vehicle, and thus accurately predicts their future locations. In (d), our model accurately identifies that all vehicles are heading forward, with those on the top-left moving slightly faster than the ego vehicle, thus they end up slightly further up. These examples demonstrate that our model can deduce vehicle speeds from slot representations, without explicit information on heading direction or speed. Its capability extends beyond simply associating left/right lanes with heading direction, as can be seen from (a) and (d). In case of a lane change ((e)) or a turn ((c)), our model accurately predicts the change in its viewpoint.

Problems in Forecasting: We encounter three types of issues while forecasting slots. The first one occurs due to the perception errors by SAVi during slot extraction. Since we rely on slot representations extracted by SAVi both as input and also for supervision while forecasting slots, our model's performance is constrained by SAVi's performance. This can be observed from errors caused by blurry predictions of SAVi in turning scenarios in (e) and (c) or from failing to locate cars in crowded scenes in (b). The second type of problem involves False Positives, resulting from

hallucinated vehicles, as indicated by the blue color in the last column of (e), (c), and (f). Lastly, our model struggles to predict complex dynamics when multiple vehicles are moving at different speeds as shown in (c).

#Slots	Enlarged	DS↑	IS↑	RC↑
7		48.17±8.61	0.56±0.08	86.70±5.21
14	✗	49.34±5.66	0.54±0.07	92.33±2.43
30		71.48±5.25	0.75±0.06	93.00±0.23
7		62.93±6.78	0.73±0.07	80.20±1.87
14	✓	69.75±8.03	0.78±0.04	86.17±2.79
30		**74.89±1.44**	**0.79±0.02**	**92.90±1.28**

Table 3.3: **Effect of Slot Extraction on Driving.** We perform experiments to show the effect of enlarging small vehicles in order to overcome perception issues as well as varying the number of slots. We report mean±std over 3 different runs on Longest6.

Method	#Slots	LD	#Steps	ARI↑	mIoU↑
SAVi	7	✕	-	0.905	0.856
	7	✓	-	0.911	0.860
	30	✓	-	0.924	0.874
CarFormer	7	✕	1	0.742	0.644
	7	✓	1	0.704	0.602
	30	✓	1	0.795	0.702
	7	✕	4	0.513	0.436
	7	✓	4	0.462	0.385
	30	✓	4	0.540	0.454
Input-Copy	-	-	1	0.641	0.561
	-	-	4	0.412	0.375

Table 3.4: **Forecasting Results with Light Decoder (LD).** We show the results for SAVi reconstruction, which serve as the upper bound as the model has access to ground truth. We compare six versions of our model, with each of the three SAVi versions (7-base, 7-light, 30-light), and trained to predict 1 or 4 timesteps into the future. Finally, we show the results of predicting the current BEV to forecast the future as a baseline.

LD	#Slots	DS↑	IS↑	RC↑
✕	7	60.82±0.87	0.73±0.02	79.44±1.62
✓	7	62.93±6.78	0.73±0.07	80.20±1.87
✓	14	69.75±8.03	0.78±0.04	86.17±2.79
✓	30	74.89±1.44	0.79±0.02	92.90±1.28

Table 3.5: **Driving Performance with Light Decoder (LD).** We test three different SAVi versions. Light refers to the lighter version of SAVi we propose, and the base version otherwise. #Slots refers to the number of slots used in the model. We report mean±std of 3 different runs on the Longest6.

	$T = t + 1$		$T = t + 4$	
Method	**ARI↑**	**mIoU↑**	**ARI↑**	**mIoU↑**
Input-Copy	0.641	0.561	0.412	0.375
CarFormer	0.795	0.702	0.540	0.454
SAVi	0.924	0.874	0.924	0.874

Table 3.6: **Forecasting Results.** We evaluate CarFormer's ability to predict future slots at time $T = T+1$ and $T = T+4$ that are decoded with the frozen SAVi decoder.

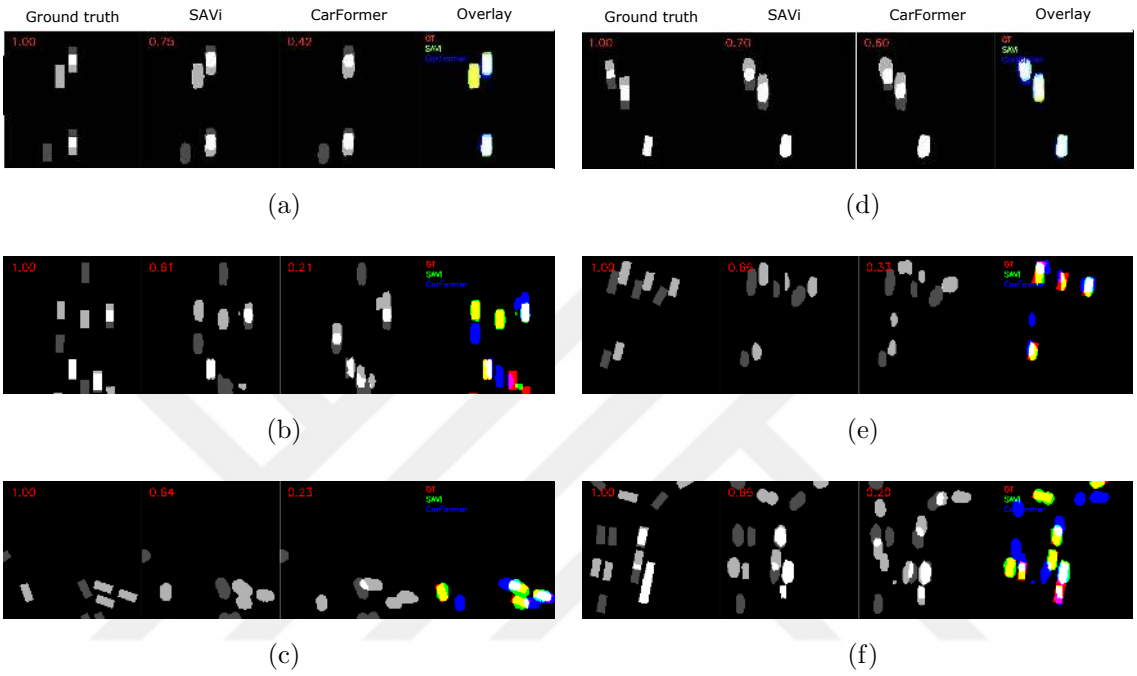

Figure 3.2: **Visualization of Slot Forecasting Results.** Each sub-figure shows an example of input (dark grey)-output (light grey) objects in the first column, SAVi reconstructions in the second column, and our model's predictions in the third column. The top left corner of each column shows the mIoU compared to the ground truth. For comparison, we overlay the three in the last column where the red channel (R) is the ground-truth location, the green channel (G) is SAVi reconstruction, and the blue channel (B) is our prediction. In the case of perfect alignment between the three, we see the vehicles in white, and different errors for our model can be seen in a unique color such as yellow (R+G) indicating misses and blue indicating false positives (B).

Chapter 4

ETA: EFFICIENCY THROUGH THINKING AHEAD, A DUAL APPROACH TO SELF-DRIVING WITH LARGE MODELS

How can we benefit from large models without sacrificing inference speed, a common dilemma in self-driving systems? A prevalent solution is a dual-system architecture, employing a small model for rapid, reactive decisions and a larger model for slower but more informative analyses. Existing dual-system designs often implement parallel architectures where inference is either directly conducted using the large model at each current frame or retrieved from previously stored inference results. However, these works still struggle to enable large models for a timely response to every online frame. Our key insight is to shift intensive computations of the current frame to previous time steps and perform a batch inference of multiple time steps to make large models respond promptly to each time step. To achieve the shifting, we introduce Efficiency through Thinking Ahead (ETA), an asynchronous system designed to: (1) propagate informative features from the past to the current frame using future predictions from the large model, (2) extract current frame features using a small model for real-time responsiveness, and (3) integrate these dual features via an action mask mechanism that emphasizes action-critical image regions. Evaluated on the Bench2Drive CARLA Leaderboard-v2 benchmark, ETA advances state-of-the-art performance by 8% with a driving score of 69.53 while maintaining a near-real-time inference speed at 50 ms.

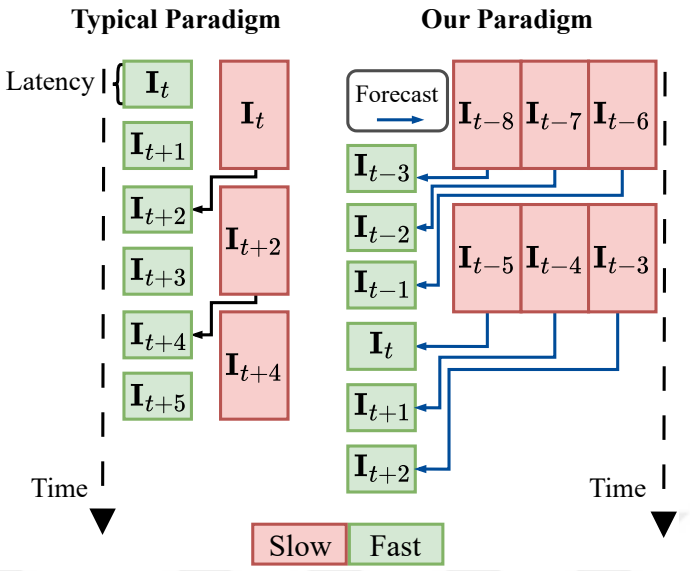

Figure 4.1: **Comparison of Dual Approaches.** Typically, dual-system approaches accommodate the larger latency of the slow, low-frequency model (red) by using the slow model predictions a fraction of the time. The fast, high-frequency model (green) outputs predictions at every time step, incorporating the slow model's outputs when available. In our approach, we batch and forecast in order to benefit from the larger model at every time step.

4.1 Methodology

4.1.1 Overview

In this work, we propose a dual system that leverages the concept of shifting intensive computations of large models from the current frame to previous frames. Additionally, we introduce the asynchronous batch inference technique, which performs feature extraction using large models across multiple frames simultaneously, thereby trading space complexity for time complexity. The integration of these two approaches enables timely inference from large models for every frame, effectively unleashing their potential in online systems.

Predictive Large Model: The predictive large model is specifically designed to transfer the computation of the current frame to a previous time step. Given two

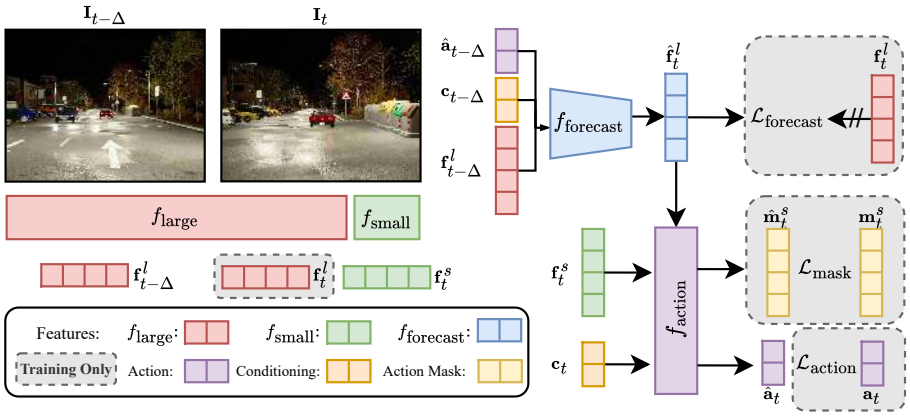

Figure 4.2: **Overview of the Asynchronous (Async) Model.** Our model processes two frames, Δ apart in time, using the large model f_{large} for the previous frame $\mathbf{I}_{t-\Delta}$ and the small model f_{small} for the current frame $\mathbf{I}_t$. Based on the previous frame's features from the large model, $\mathbf{f}^l_{t-\Delta}$, along with conditioning inputs $\mathbf{c}_t$ and $\hat{\mathbf{a}}_t$, we predict the current time-step features, $\hat{\mathbf{f}}^l_t$. The action $\hat{\mathbf{a}}_t$ is then predicted using the action model f_{action}, which takes as input the forecasted features $\hat{\mathbf{f}}^l_t$, the current frame's features from the small model $\mathbf{f}^s_t$, and the conditioning input $\mathbf{c}_t$. The forecasted features are supervised using the large model's features at the current time step, $\mathbf{f}^l_t$. Since this supervision occurs during training alone, inference-time speed remains unaffected. Training-time computations are highlighted with gray boxes. In addition to forecasting and action losses, we apply a mask loss to better align observations with actions.

RGB images, $\mathbf{I}_t$ and $\mathbf{I}_{t-\Delta}$, captured Δ seconds apart, along with the current speed $\mathbf{v}_t$ and the target waypoints $\mathbf{w}_t$, our objective is to predict the corresponding driving action $\mathbf{a}_t$. This action comprises a path made up of equidistant points and varying distance waypoints, as in [Renz et al., 2024].

To accomplish this, we first introduce a base model that employs a large encoder (Section 4.1.2), which, while accurate, is inefficient in terms of processing time. Subsequently, we develop our dual framework (Section 4.1.3), which processes the two time steps asynchronously. This method not only reduces latency but also retains the advantages of utilizing a large model.

Asynchronous Batch Inference: The concept of asynchronous batch inference is illustrated in fig. 4.1. As demonstrated, to facilitate the use of large models for every frame, we parallelize the inference of previous frames to enable large models for per frame. By simultaneously processing multiple frames, we can significantly reduce the latency burden typically associated with large models, as multiple frames are processed at the same time, enabling more responsive and efficient online systems. This approach not only optimizes resource utilization but also enhances the overall performance of the predictive large model, ensuring timely and accurate driving actions.

4.1.2 Base Large Model

By using a large model in all time steps, we can train a base model to perform well but with high latency. This allows us to explore the upper bounds of driving performance with various large models if latency were not a concern.

For example, following the approach proposed by the winner [Renz et al., 2024] of the CARLA Autonomous Driving Challenge 2.0, we can use a large vision encoder f_{large}, e.g. a VLM or MLLM, to extract features from sensory input $\mathbf{I}_t$ at time t, resulting in $\mathbf{f}_t^l$. We then feed it to an action model f_{action} to predict action $\hat{\mathbf{a}}_t$ conditioned on $\mathbf{c}_t$, which is the concatenation of speed v_t and target waypoints $\mathbf{w}_t$ at time t:

$$\mathbf{f}_t^l = f_{\text{large}}(\mathbf{I}_t), \tag{4.1}$$

$$\hat{\mathbf{a}}_t = f_{\text{action}}(\mathbf{I}_t, \mathbf{c}_t). \tag{4.2}$$

Following LLaVA's anyres [Liu et al., 2024, Liu et al., 2023] approach, we divide the image into two patches and then process them with the vision encoder. To reduce the number of output tokens, we apply 2×2 spatial pooling, decreasing the token count by a factor of 4. Despite this reduction, the best-performing base models with a large encoder still fail to achieve a latency close to real-time constraints.

Action Mask: When we visualized the attention maps, we observed that the model struggles to focus on image patches that are relevant to action. To better

align the predicted actions with the observations, we compute attention between patch features and the encoded queries corresponding to the action. This process generates a mask, $\hat{\mathbf{m}}_t$, showing the regions of the input image toward which the ego vehicle is likely to move (Fig. 4.3). We supervise this action mask with the ground truth mask, which is obtained by projecting expert actions onto the image (Section 4.1.4).

4.1.3 Asynchronous (Async) Model

As illustrated in Fig. 4.2, we propose a dual framework with a slow, large model f_{large} and a smaller, faster model f_{small}. Our small model is designed with real-time performance in mind. However, large models with strong performance typically exceed real-time constraints. To utilize the large model without increasing latency, we process the input at a previous time step $t - \Delta$, rather than the current time step t:

$$\mathbf{f}^l_{t-\Delta} = f_{\text{large}}(\mathbf{I}_{t-\Delta}). \tag{4.3}$$

$\mathbf{I}_{t-\Delta}$ denotes the input image at time $t - \Delta$ with the corresponding features $\mathbf{f}^l_{t-\Delta}$ obtained from the large model.

Forecasting: Given features $\mathbf{f}^l_{t-\Delta}$ from the large model at a previous time step, we train a model f_{forecast} to predict features at the current time step. Similar to a world model [Li et al., 2025], we condition this model on additional inputs, including the action predicted, $\hat{\mathbf{a}}_{t-\Delta}$, and $\mathbf{c}_{t-\Delta}$, which is the concatenation of speed $v_{t-\Delta}$ and target waypoints $\mathbf{w}_{t-\Delta}$:

$$\hat{\mathbf{f}}^l_t = f_{\text{forecast}}(\mathbf{f}^l_{t-\Delta}, \hat{\mathbf{a}}_{t-\Delta}, \mathbf{c}_{t-\Delta}). \tag{4.4}$$

The result of forecasting is the predicted features $\hat{\mathbf{f}}^l_t$ for the current time step t.

Small Model: In addition to forecasting the features for the current time step, we use the small model, f_{small}, to process the current input $\mathbf{I}_t$, resulting in features $\mathbf{f}^s_t$:

$$\mathbf{f}^s_t = f_{\text{small}}(\mathbf{I}_t). \tag{4.5}$$

Figure 4.3: **Action Mask.** In the top row, we show two examples of the RGB image with the path (purple) and waypoints (blue) projected onto it. Patches containing a path or waypoint are marked as 1 (yellow), and all other patches are marked as 0 (purple), creating the binary mask, overlayed on the image below for visualization.

The small model is designed to capture changes between the previous and current time steps, particularly those that are difficult to predict, such as a traffic light's state or the sudden appearance of a pedestrian.

Action Prediction: We concatenate the predicted features $\hat{\mathbf{f}}_t$ from forecasting with the features $\mathbf{f}_t^s$ from the small model and provide them as input to the action model, f_{action}, along with the conditioning input $\mathbf{c}_t$, resulting in the predicted action $\hat{\mathbf{a}}_t$:

$$\hat{\mathbf{a}}_t = f_{\text{action}}(\hat{\mathbf{f}}_t^l, \mathbf{f}_t^s, \mathbf{c}_t). \tag{4.6}$$

It is observed that the key to performance lies in forecasting and the small model working together to efficiently update past information from the large model to the current state.

4.1.4 Training

Action Loss: For the prediction of action, we apply an L_1 loss between the predicted actions and the expert action, including the path and waypoints. Instead of directly predicting absolute positions, we learn to predict residuals. At test time, we sum these residuals to reconstruct the original waypoints. A similar approach is applied to the path tokens. We find that this formulation stabilizes training compared to directly regressing path and waypoint coordinates.

Action Mask Loss: To generate supervision for the action mask, we project the expert action onto the input RGB image using the camera parameters. Patches containing a waypoint or path point are assigned a value of 1, while all others are set to 0 (Fig. 4.3). We apply a binary cross-entropy loss between the ground truth mask $\mathbf{m}_t$ created this way and the predicted mask $\hat{\mathbf{m}}_t$.

The final loss of the base model is a weighted sum of the two loss terms where $\lambda_1 = 1/16$:

$$\mathcal{L}_{\text{base}} = \mathcal{L}_{\text{action}} + \lambda_1 \, \mathcal{L}_{\text{mask}}. \tag{4.7}$$

Async Model with Forecasting Loss: For the async model, we optimize the predicted action loss and mask loss as in the case of the base model (4.7), but the mask loss is applied to the features of the small model in the async case. Additionally, we introduce a forecasting loss, $\mathcal{L}_{\text{forecast}}$, to supervise the predicted features based on the large model's features from the previous time step. With the forecasting loss, we aim to bring the predicted features closer to the ground truth features extracted from the current time step using the same large model. Specifically, we compute the absolute difference between ground truth features $\mathbf{f}_t^l$ and the predicted features $\hat{\mathbf{f}}_t^l$ from (4.4):

$$\mathcal{L}_{\text{forecast}}(\mathbf{f}_t^l, \hat{\mathbf{f}}_t^l) = |\mathbf{f}_t^l - \hat{\mathbf{f}}_t^l|. \tag{4.8}$$

We stop gradients from flowing through the ground truth features to prevent collapse. The weight of the mask loss remains the same as in the base model, and we set the weight of the forecasting loss to 0.5.

Table 4.1: **Comparison to SOTA on Bench2Drive.** In addition to the existent baselines provided by Bench2Drive [Jia et al., 2024], we compare our method to the recent DriveTransformer [Jia et al., 2025], selecting the best-performing variant for each model. Both our base and async models outperform *all* prior arts, with the async model achieving significantly lower latency with the proposed dual framework. Latency is reported in milliseconds.

Method	**DS↑**	**SR↑(%)**	**Efficiency↑**	**Comfort↑**	**Latency↓**
AD-MLP	18.05	0.00	48.45	22.63	3
UniAD-Base	45.81	16.36	129.21	43.58	663.4
VAD	42.35	15.00	157.94	46.01	278.3
TCP-traj	59.90	30.00	76.54	18.08	83
ThinkTwice	62.44	31.23	69.33	16.22	762
DriveTransformer-L	63.46	35.01	100.64	20.78	211.7
DriveAdapter	64.22	33.08	70.22	16.01	931
Base Model (Ours)	74.33±0.99	48.33±3.56	186.04±6.47	25.77±1.13	102
Async Model (Ours)	69.53±1.62	38.64±0.98	184.51±1.54	28.43±1.50	50

4.2 Experiments

4.2.1 Training Setting

We train all models for 40 epochs on 8× A100 GPUs. We use CLIP-ViT-L-336px as the base large model encoder and the first 8 layers of CLIP-ViT-L-336px as the small model encoder, which we ablate in Table 4.4. For the action decoder, we use a 12-layer transformer decoder. We use the base Adam optimizer with an initial learning rate of $3 \cdot 10^{-5}$ and a cosine annealing schedule. The learning rate (LR) resets 4 times before reaching the minimum LR near the end of training. We set the global batch size to 160. For the Async Model, we set Δ to 0.5s.

4.2.2 Evaluation Details

Data: The Bench2Drive [Jia et al., 2024] Benchmark comes with two data splits collected using the Think2Drive RL agent [Li et al., 2024a]. The base split, which consists of 1000 routes, is used to train and evaluate multiple models in the original paper and following work. To maintain direct comparability to the evaluated models, we only train on the base split and do not use any additional data. We divide the data into a 94%-3%-3% training-validation-test split during our experiments. Similarly to CarLLaVA [Renz et al., 2024], we create different buckets containing different interesting scenarios, such as turning, near-collision, or intersections. During training, we use a weighted sampling approach to sample from the buckets.

Metrics: We report the Driving Score (DS) and Success Rate (SR), along with Comfortness and Efficiency scores. Additionally, we provide per-ability scores to assess the model's performance across different scenario categories. Finally, we include the latency of each model in milliseconds (ms). For our main models, we report results averaged over three runs with different seeds to account for variance in initialization and closed-loop evaluation. However, due to time and computational constraints, we report results from a single seed for ablation studies.

4.2.3 Comparison to State-of-the-Art

In Table 4.1, we compare our method to other approaches, including the originally reported AD-MLP [Zhai et al., 2023], UniAD [Hu et al., 2023], VAD [Jiang et al., 2023], TCP [Wu et al., 2022], ThinkTwice [Jia et al., 2023b], and DriveAdapter [Jia et al., 2023a]. Note that we are unable to quantitatively compare to other dual approaches [Mei et al., 2024, Tian et al., 2024, Zhang et al., 2024c] due to the lack of comparable closed-loop results.

Our base model, without asynchronous future prediction, achieves the highest driving score among all counterparts, with a latency of 102 ms. However, despite the strong performance of our base model, its 102ms latency imposes an upper bound of approximately 10Hz on the control frequency, which is relatively low for real-time control.

Table 4.2: **Ablation Study.** We conduct an ablation study by removing each component of the model (**A**–**C**). Additionally, we report the performance of the small model alone (**D**) and further analyze forecasting by using ground truth features during both training and testing (**E**) or only during testing (**F**). 'GT' indicates ground truth. See the text for a detailed analysis.

ID	Method	DS↑	SR↑(%)	Comfort↑	Latency↓
	Base Model (Ours)	74.33	48.33	25.77	102
	Async Model (Ours)	69.53	38.64	28.43	50
A	Without Forecasting	54.92	30.45	18.71	50
B	Without Small Model	42.49	17.27	14.10	31
C	Without Action Mask	42.67	17.27	27.99	50
D	Small Model Only	61.30	35.00	43.91	50
E	GT Forecasting	74.12	48.18	24.76	124
F	GT Forecasting (Test Time)	60.72	32.27	17.25	124

Table 4.3: **Ablation Study according to Distinct Abilities.** To identify which components contribute to improvements in different scenarios, we report model performance in the ablation study based on distinct capabilities. For instance, removing the small model (**C**) leads to significant performance drops in the emergency brake (**E. Brake**) and traffic sign (**T. Sign**). See the text for a detailed analysis.

ID	Method	Merging↑	Overtaking↑	E. Brake↑	Give Way↑	T. Sign↑	Mean↑	Latency↓
	Base Model (Ours)	35.24	54.70	66.67	80.00	59.43	59.21	102
	Async Model (Ours)	26.66	50.42	60.13	80.00	43.64	52.17	50
A	Without Forecasting	20.00	35.90	41.18	80.00	36.18	42.65	50
B	Without Small Model	14.86	17.50	16.36	40.00	30.56	23.85	31
C	Without Action Mask	20.00	17.95	9.80	80.00	30.92	31.73	50
D	Small Model Only	27.14	41.03	47.06	60.00	44.74	43.99	50
E	GT Forecasting	29.33	57.50	56.36	60.00	53.89	51.41	124
F	GT Forecasting (Test)	28.57	38.46	43.14	60.00	43.42	42.72	124

Using our asynchronous model, which performs the computation of the full vision encoder asynchronously until the time it is needed, we reduce latency to 50ms, enabling a control frequency of 20Hz. This marks a significant improvement over the base model, allowing for much more responsive control. Despite not having access to the large model at the current time step, our async model still performs very well in terms of driving score and success rate, ranking second after our base model.

4.2.4 Ablation Studies

We first perform an ablation study of our design choices in Table 4.2 to address the following research questions:

- Does supervising the large model for forecasting improve performance, or is the improvement solely due to scaling up the model parameters?
- Is the small model necessary, or can we rely solely on the forecasted features?
- Is additional supervision with the action mask essential?
- How does the small model perform without forecasting?
- Would better forecasting lead to better performance?
- In which scenarios do we observe improvements, and what are the underlying reasons?

A: Without supervising forecasting, the driving score drops to 54.92, highlighting the importance of training the large model to learn forecasting. The performance improvement cannot be solely attributed to the increased parameter count, as the large model's features do not enhance performance without being explicitly trained for future prediction. By supervising the model to forecast with ground truth features, the large vision encoder effectively improves the driving score by approximately 15 points. Interestingly, the overall performance remains lower than when

Figure 4.4: **Hard Brake without Small Model. Left:** The initial state of the scene where the yellow car ahead suddenly brakes, requiring the ego vehicle to perform a hard brake. **Middle:** The Async Model detects the hazard using the small model and stops in time. **Right:** The version without the small model (**B**) relies solely on forecasting and fails to detect the sudden stop in time, resulting in a crash. The predicted action for each model is overlaid on the image, with waypoints shown in blue and the path in red.

using only the small model (**D**), possibly due to confusion caused by suboptimal features from the large model.

B: Without small model, the performance drops to 42.49, indicating that forecasting with the large model alone is insufficient without up-to-date information from the small model. This may be due to scenarios where the future state is difficult to predict, such as changes in traffic light status. As shown in Table 4.3, removing the small model (**B**) results in a significant decline in emergency braking ability (60.13→16.36) and proper handling of traffic signs (43.64→30.56). These results highlight the critical role of the small model in providing the model with up-to-date information by allowing it to glance at the current state.

C: Without action mask, the performance drops to 42.67, highlighting the importance of aligning the predicted actions with the observations. It might be surprising that such a small adjustment has such a significant impact on the async model, but this was one of the first insights we gained while attempting to make the base model work by examining the visualization of attention maps. This additional supervision helps the model better relate the predicted action to the observations.

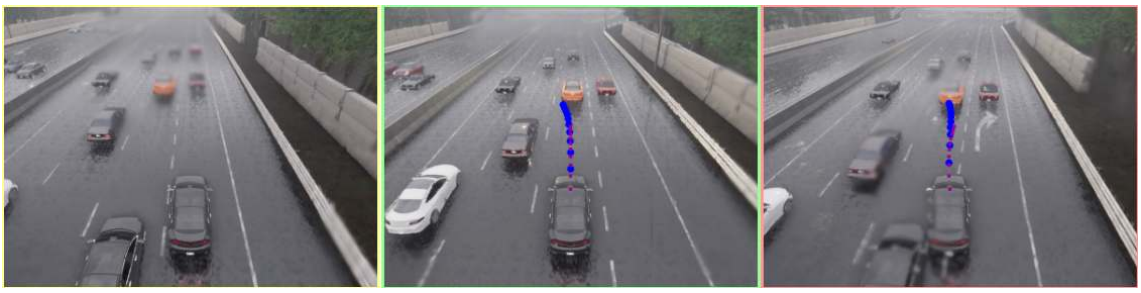

Figure 4.5: **Lane Change without Forecasting. Left:** The initial state of the scene, where the ego vehicle is tasked with switching to the left lane. **Middle:** The Async Model successfully executes the lane change while avoiding collisions by leveraging forecasting. **Right:** The version without forecasting (**A**) fails to anticipate the future position of the vehicle behind, resulting in a rear-end collision. The predicted action for each model is overlaid on the image, with waypoints shown in blue and the path in red.

D: Using only the small model, performance drops to 61.30, demonstrating that the predicted features from the large model enhance overall performance. With our dual formulation, we can leverage these features without impacting latency. Comparing the async model with forecasting to using only the small model without forecasting (**D**) in terms of abilities in Table 4.3, we observe a notable decline in overtaking (50.42→41.03) and emergency braking (60.13→47.06) behaviors. A possible explanation is that the large model provides strong features for predictable scenarios, enabling the small model to focus on detecting deviations, such as emergencies. Without the large model, the small model must handle all situations, leaving fewer resources for effectively managing emergencies.

E & F: Using ground truth features for forecasting, we further examine the impact of forecasting on performance. When ground truth features replace forecasted features during both training and testing (**E**), the driving score improves to 74.12, surpassing the score achieved with our async model's forecasted features (69.53). This gap suggests that forecasting can still be improved, and with perfect forecasting, the model can perform at the same level as our base model (74.33),

ruling out potential design limitations. However, when ground truth features are provided only at test time (**F**), the driving score drops significantly to 60.72. This suggests a distribution mismatch between the predicted and ground truth features, even though the model is trained with supervision from ground truth features.

Ablating the large encoder: Furthermore, we ablate different base encoders in Table 4.4

4.2.5 Qualitative Analysis

We visually investigate the contributions of the small model and the large model with forecasting through two example scenarios: a hard brake without the small model (Fig. 4.4) and a lane change without forecasting supervision (Fig. 4.5). These correspond to rows **B** and **A** in the ablation study (Table 4.2), respectively. In Fig. 4.4, the version in **B** fails to perform a hard brake, as it cannot detect the

Table 4.4: **Varying the Large Encoder in the Base Model.** We experiment with various sizes of large encoders in the base model to observe the differences when changing our default large encoder in **A** to smaller encoders (**B**–**E**) and larger encoders (**F, G**). While larger encoders generally perform better, smaller encoders achieve lower latencies, as expected. Notably, the model in **B** has a similar latency to our Aysnc model but performs significantly worse (61.30 vs. 69.53). **Lat.** stands for latency in milliseconds.

ID	Model	Size	DS↑	SR↑(%)	Lat.↓
A	CLIP-ViT-L	308M	74.33	48.94	102
B	CLIP-ViT-L8	114M	61.30	35.00	50
C	InternViT	304M	71.53	46.82	132
D	ViT-Base	86.9M	67.53	44.55	67
E	EVA02-Base	85.8M	65.20	37.73	80
F	CLIP ViT-H	631M	70.74	43.64	144
G	CLIP-ViT-g	1011M	73.19	45.91	192

sudden stop of the yellow vehicle ahead without the small model, highlighting its role in detecting emergencies that appear in the current time step. Similarly, in Fig. 4.5, the version in **A** results in a rear-end crash due to its failure to anticipate the future position of the vehicle behind without forecasting.

Chapter 5

CONCLUSION

5.1 Summary

In this thesis, we explore the feasibility and the potential of using large foundation models for the real-time, safety-critical task of self-driving. The motivation of this work is to enable self-driving approaches to benefit from the impressive performance and generalization capabilities of constantly improving, expensive, large foundation models. In Carformer we build on previous promising work in utilizing large, auto-regressive language models for RL in robotics tasks. These works, while promising, are designed for and evaluated on tasks where the state space is relatively small and possibly adequately representable with a single vector. To overcome this shortcoming and to effectively utilize these approaches for self-driving, a task where the state is significantly more complex and involves dynamic interactions between agents in a scene, we set out to find a compact yet effective representation. To create a fair experimental setting where we can fairly compare possible representations, we start by proposing Carformer, extending models in previous work to be able to more flexible, accommodating both discrete and continuous representations while maintaining auto-regressive properties. Then, while keeping everything else constant, we analyze the performance of different possible state representations. Ultimately, we propose using an object-level, slot-based, self-supervised representation. Using this proposed representation and our model, we achieve state-of-the-art performance on the Longest6 benchmark in CARLA Leaderboard-v1 , outperforming previous related work which uses hand-crafted object-level representations. With our work, we show that language modeling approaches to driving, when paired with compatible and strong representations, are a promising and scalable direction to explore.

In our earlier experiments, we assumed that the perception problem is solved,

and that we have access to perfect intermediate representations in the form of BEV. Perception approaches, however, are still far from perfect, and extracting accurate BEV maps from images is still a challenging problem [Philion and Fidler, 2020, Li et al., 2022, Harley et al., 2023]. One advantage of large foundation models is that their performance improves with scaling, showing impressive perception and reasonable capabilities. Simply scaling the foundation model used results in an increased decision latency, which is unacceptable in a task like autonomous driving where timely reactions to emergency situations is crucial. In our follow-up work, we tackle the problem of high decision latency with scaling, and explore the question of whether we are able to tap into the potential of large foundation models without sacrificing latency. To achieve this, we propose **E**fficiency through **T**hinking **A**head (ETA), where we accomodate the latency of the large foundation model by shifting the computation to an earlier timestep. The assumption at the core of our approach is that the computationally expensive representations learned by the larger models are much more informative and useful, and that a slightly outdated representation is still beneficial. As a result, instead of processing the current input with the large model for the decision at the current timestep, we shift the processing to an earlier point by instead processing a previous input. Then, we adapt these expensive features to the current timestep, we find a very lightweight forecasting module to be sufficient. In tandem with the larger, more expensive model, we also use a lightweight model to process the current timestep. This lighter model proves essential to handle cases where forecasting alone cannot accurately predict the future, such as in the case of a pedestrian suddenly crossing the street. By integrating the two models together, we achieve a very strong performance on the Bench2Drive benchmark [Jia et al., 2024], outperforming the next best models at a fraction of the latency. We ablate our design decisions, and show that each part of the pipeline is crucial to enabling strong performance. With our proposed paradigm, we show that it is feasible to use large foundation models in driving while meeting latency requirements.

5.2 Future Work

Our work attempts to highlight the potential of foundation models for self-driving, but leaves plenty of space for further improvement In this section, we describe some possible future directions to extend this work.

In CarFormer, we proposed our model inspired by language modeling approaches for RL. However, there are multiple limitations. First, we assume access to ground truth BEV, limiting real world applicability. Secondly, while our formulation can integrate rewards to do offline RL, we currently simply do imitation learning.

Given the promising results of our slot-based object-level representations in BEV, an interesting research question to explore is whether these slot-based representations are instead learnable directly from RGB inputs. Instead of a two-stage approach by first estimating BEV and then extracting slots from it, a more direct approach of extracting slots in BEV from RGB inputs might be preferable, both in terms of efficiency and avoiding cascading errors. However, self-supervised learning in BEV [Gosala et al., 2023] is in its infancy at the time of our work. Considering the complexity of self-driving and the challenge of slot extraction from videos, we presented a necessary first step before transitioning to an end-to-end approach based on object-centric representations. With the advancements in slot extraction from real-world videos, any object that can be placed into a slot can be part of the reasoning in our model.

While CarFormer achieves state-of-the-art performance on the Longest6 benchmark, it does not perform better than the rule-based expert used for training data collection. Previous work proposing language modeling approaches for reinforcement learning highlight the capability of these approaches to outperform the expert. As our model is directly compatible with these approaches, a direct line of future work is to integrate a meaningful reward function into the model to use RL instead of imitating the expert, potentially further improving performance.

In our follow-up work, we propose ETA, a dual-system paradigm for autonomous driving that enables the usage of large, expensive foundation models without sacrificing latency. In our experiments, we rely on using a single front-facing camera

as input. While this design works very well, outperforming previous approaches, a single front-facing camera is not adequate to handle all scenarios, such as performing a required stop for a trailing ambulance with active sirens. Integrating additional cameras and possibly other sensors like lidar could further improve performance and address these particular scenarios.

While introducing the forecasted features from the large foundation model in ETA significantly improves driving performance compared to only using the small model, it does not match the performance of using only the large model without forecasting. This shows that the current forecasting has a lot of room for improvement, which is further demonstrated by our experiments using ground truth forecasting instead of the forecasted features improves performance and bridges the gap.

Furthermore, in our ETA experiments we rely on the training data provided in Bench2Drive, which is collected using an RL agent as an expert. Due to the expert being imperfect, this puts a soft upper bound on the performance of our model trained with imitation learning. Recent work has proposed improved experts that outperform previous experts. Further work on designing better experts can further improve the performance of our model, as well as other imitation learning approaches.

BIBLIOGRAPHY

[Arasteh et al., 2024] Arasteh, F., Elmahgiubi, M., Khamidehi, B., Mirkhani, H., Zhang, W., Tongtong, C., and Rezaee, K. (2024). Validity learning on failures: Mitigating the distribution shift in autonomous vehicle planning. *arXiv preprint arXiv:2406.01544*.

[Aydemir et al., 2023] Aydemir, G., Xie, W., and Güney, F. (2023). Self-supervised Object-centric Learning for Videos. In *NeurIPS*.

[Bao et al., 2022] Bao, Z., Tokmakov, P., Jabri, A., Wang, Y.-X., Gaidon, A., and Hebert, M. (2022). Discovering objects that can move. In *CVPR*.

[Bao et al., 2023] Bao, Z., Tokmakov, P., Wang, Y.-X., Gaidon, A., and Hebert, M. (2023). Object discovery from motion-guided tokens. In *CVPR*.

[Beißwenger, 2024] Beißwenger, J. (2024). PDM-Lite: A rule-based planner for carla leaderboard 2.0.

[Bu et al., 2024] Bu, Q., Li, H., Chen, L., Cai, J., Zeng, J., Cui, H., Yao, M., and Qiao, Y. (2024). Towards synergistic, generalized, and efficient dual-system for robotic manipulation. *arXiv preprint arXiv:2410.08001*.

[Burgess et al., 2019] Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M. M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *ARXIV*, 1901.11390.

[Caron et al., 2021] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *ICCV*.

[Chen et al., 2019] Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. (2019). Learning by cheating. In *CORL*.

[Chen et al., 2021a] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021a). Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*.

[Chen et al., 2021b] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021b). Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*.

[Chen et al., 2024a] Chen, L., Wu, P., Chitta, K., Jaeger, B., Geiger, A., and Li, H. (2024a). End-to-end autonomous driving: Challenges and frontiers. *IEEE TPAMI*.

[Chen et al., 2022] Chen, X., Jiang, T., Song, J., Yang, J., Black, M., Geiger, A., and Hilliges, O. (2022). gdna: Towards generative detailed neural avatars. In *CVPR*.

[Chen et al., 2015] Chen, Z., Sun, X., Wang, L., Yu, Y., and Huang, C. (2015). A deep visual correspondence embedding model for stereo matching costs. In *ICCV*.

[Chen et al., 2024b] Chen, Z., Wu, J., Wang, W., Su, W., Chen, G., Xing, S., Zhong, M., Zhang, Q., Zhu, X., Lu, L., Li, B., Luo, P., Lu, T., Qiao, Y., and Dai, J. (2024b). InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*.

[Chitta et al., 2023] Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., and Geiger, A. (2023). Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE TPAMI*.

[Cho et al., 2014] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceed-*

ings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[Crawford and Pineau, 2019] Crawford, E. and Pineau, J. (2019). Spatially invariant unsupervised object detection with convolutional neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3412–3420.

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.

[Dosovitskiy et al., 2017] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *CORL*.

[Driess et al., 2023] Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., and Florence, P. (2023). PaLM-E: An embodied multimodal language model. In *ICML*.

[Elsayed et al., 2022] Elsayed, G. F., Mahendran, A., van Steenkiste, S., Greff, K., Mozer, M. C., and Kipf, T. (2022). SAVi++: Towards end-to-end object-centric learning from real-world videos. In *NeurIPS*.

[Engelcke et al., 2020] Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2020). GENESIS: generative scene inference and sampling with object-centric latent representations.

[Eslami et al., 2016] Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., kavukcuoglu, k., and Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[Esser et al., 2021] Esser, P., Rombach, R., and Ommer, B. (2021). Taming transformers for high-resolution image synthesis. In *CVPR*.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338.

[Figure, 2025] Figure (2025). Helix: A vision-language-action model for generalist humanoid control.

[Furuta et al., 2022] Furuta, H., Matsuo, Y., and Gu, S. S. (2022). Generalized decision transformer for offline hindsight information matching. In *ICLR*.

[Gosala et al., 2023] Gosala, N., Petek, K., Drews-Jr, P. L. J., Burgard, W., and Valada, A. (2023). SkyEye: self-supervised bird's-eye-view semantic mapping using monocular frontal view images. In *CVPR*.

[Greff et al., 2019] Greff, K., Kaufmann, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *ICML*.

[Hanselmann et al., 2022] Hanselmann, N., Renz, K., Chitta, K., Bhattacharyya, A., and Geiger, A. (2022). King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *ECCV*.

[Harley et al., 2023] Harley, A. W., Fang, Z., Li, J., Ambrus, R., and Fragkiadaki, K. (2023). Simple-BEV: What really matters for multi-sensor bev perception? In *ICRA*.

[Hu et al., 2023] Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., Lu, L., Jia, X., Liu, Q., Dai, J., Qiao, Y., and Li, H. (2023). Planning-oriented autonomous driving. In *CVPR*.

[Jaeger et al., 2023] Jaeger, B., Chitta, K., and Geiger, A. (2023). Hidden biases of end-to-end driving models. In *ICCV*.

[Janai et al., 2020] Janai, J., Güney, F., Behl, A., and Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308.

[Janner et al., 2021a] Janner, M., Li, Q., and Levine, S. (2021a). Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*.

[Janner et al., 2021b] Janner, M., Li, Q., and Levine, S. (2021b). Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*.

[Jia et al., 2023a] Jia, X., Gao, Y., Chen, L., Yan, J., Liu, P. L., and Li, H. (2023a). DriveAdapter: breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *ICCV*.

[Jia et al., 2023b] Jia, X., Wu, P., Chen, L., Xie, J., He, C., Yan, J., and Li, H. (2023b). Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *CVPR*.

[Jia et al., 2024] Jia, X., Yang, Z., Li, Q., Zhang, Z., and Yan, J. (2024). Bench2Drive: towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *NeurIPS 2024 Datasets and Benchmarks Track*.

[Jia et al., 2025] Jia, X., You, J., Zhang, Z., and Yan, J. (2025). DriveTransformer: unified transformer for scalable end-to-end autonomous driving. In *ICLR*.

[Jiang et al., 2023] Jiang, B., Chen, S., Xu, Q., Liao, B., Chen, J., Zhou, H., Zhang, Q., Liu, W., Huang, C., and Wang, X. (2023). VAD: vectorized scene representation for efficient autonomous driving. In *ICCV*.

[Jiang et al., 2019] Jiang, J., Janghorbani, S., De Melo, G., and Ahn, S. (2019). Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations*.

[Johnson et al., 2017] Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

[Kabra et al., 2021] Kabra, R., Zoran, D., Erdogan, G., Matthey, L., Creswell, A., Botvinick, M., Lerchner, A., and Burgess, C. (2021). Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20146–20159. Curran Associates, Inc.

[Kahneman, 2011] Kahneman, D. (2011). *Thinking, fast and slow*. Farrar, Straus and Giroux.

[Kaljavesi et al., 2024] Kaljavesi, G., Su, X., and Diermeyer, F. (2024). Integrating end-to-end and modular driving approaches for online corner case detection in autonomous driving. *arXiv preprint arXiv:2409.01178*.

[Karazija et al., 2021] Karazija, L., Laina, I., and Rupprecht, C. (2021). Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. *ARXIV*, 2111.10265.

[Kipf et al., 2022] Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. (2022). Conditional Object-Centric Learning from Video. In *ICLR*.

[Kosiorek et al., 2018] Kosiorek, A., Kim, H., Teh, Y. W., and Posner, I. (2018). Sequential attend, infer, repeat: Generative modelling of moving objects. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Lee et al., 2019] Lee, Y., Hwang, J.-w., Lee, S., Bae, Y., and Park, J. (2019). An

energy and gpu-computation efficient backbone network for real-time object detection. In *CVPR Workshop*.

[Li et al., 2024a] Li, Q., Jia, X., Wang, S., and Yan, J. (2024a). Think2Drive: efficient reinforcement learning by thinking in latent world model for quasi-realistic autonomous driving (in carla-v2). In *ECCV*.

[Li et al., 2024b] Li, Q., Liang, Y., Wang, Z., Luo, L., Chen, X., Liao, M., Wei, F., Deng, Y., Xu, S., Zhang, Y., Wang, X., Liu, B., Fu, J., Bao, J., Chen, D., Shi, Y., Yang, J., and Guo, B. (2024b). CogACT: a foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*.

[Li et al., 2013] Li, W., Cosker, D., Brown, M., and Tang, R. (2013). Optical flow estimation using laplacian mesh energy. In *CVPR*, pages 2435–2442.

[Li et al., 2025] Li, Y., Fan, L., He, J., Wang, Y., Chen, Y., Zhang, Z., and Tan, T. (2025). Enhancing end-to-end autonomous driving with latent world model. In *ICLR*.

[Li et al., 2022] Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., and Dai, J. (2022). BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*.

[Liu et al., 2024] Liu, H., Li, C., Li, Y., and Lee, Y. J. (2024). Improved baselines with visual instruction tuning. In *CVPR*.

[Liu et al., 2023] Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2023). Visual instruction tuning. In *NeurIPS*.

[Locatello et al., 2020] Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. In *NeurIPS*.

[Mao et al., 2023] Mao, J., Qian, Y., Ye, J., Zhao, H., and Wang, Y. (2023). Gpt-driver: Learning to drive with gpt. *arXiv preprint arXiv:2310.01415*.

[Mei et al., 2024] Mei, J., Ma, Y., Yang, X., Wen, L., Cai, X., Li, X., Fu, D., Zhang, B., Cai, P., Dou, M., Shi, B., He, L., Liu, Y., and Qiao, Y. (2024). Continuously learning, adapting, and improving: A dual-process approach to autonomous driving. In *NeurIPS*.

[Micheli et al., 2023] Micheli, V., Alonso, E., and Fleuret, F. (2023). Transformers are sample-efficient world models. In *ICLR*.

[Mousavian et al., 2019] Mousavian, A., Toshev, A., Fiser, M., Kosecká, J., Wahid, A., and Davidson, J. (2019). Visual representations for semantic target driven navigation. In *ICRA*.

[Müller et al., 2018] Müller, M., Dosovitskiy, A., Ghanem, B., and Koltun, V. (2018). Driving policy transfer via modularity and abstraction. In *CORL*.

[Nash et al., 2022] Nash, C., Carreira, J., Walker, J., Barr, I., Jaegle, A., Malinowski, M., and Battaglia, P. (2022). Transframer: Arbitrary frame prediction with generative models. *ARXIV*, 2203.09494.

[Ochs et al., 2013] Ochs, P., Malik, J., and Brox, T. (2013). Segmentation of moving objects by long term video analysis. *IEEE TPAMI*.

[OpenAI, 2022] OpenAI (2022). OpenAI: Introducing ChatGPT.

[Perazzi et al., 2016] Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*.

[Philion and Fidler, 2020] Philion, J. and Fidler, S. (2020). Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *ECCV*.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[Ren and Wang, 2022] Ren, X. and Wang, X. (2022). Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In *CVPR*.

[Renz et al., 2024] Renz, K., Chen, L., Marcu, A.-M., Hünermann, J., Hanotte, B., Karnsund, A., Shotton, J., Arani, E., and Sinavski, O. (2024). CarLLaVA: Vision language models for camera-only closed-loop driving. *arXiv preprint arXiv:2406.10165*.

[Renz et al., 2022] Renz, K., Chitta, K., Mercea, O.-B., Koepke, A. S., Akata, Z., and Geiger, A. (2022). Plant: Explainable planning transformers via object-level representations. In *CORL*.

[Sauer et al., 2018] Sauer, A., Savinov, N., and Geiger, A. (2018). Conditional affordance learning for driving in urban environments. In *CORL*.

[Sax et al., 2019] Sax, A., Emi, B., Zamir, A. R., Guibas, L. J., Savarese, S., and Malik, J. (2019). Learning to navigate using mid-level visual priors. In *CORL*.

[Seitzer et al., 2023] Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C.-J., He, T., Zhang, Z., Schölkopf, B., Brox, T., et al. (2023). Bridging the gap to real-world object-centric learning. In *ICLR*.

[Shafiullah et al., 2022] Shafiullah, N. M. M., Cui, Z. J., Altanzaya, A., and Pinto, L. (2022). Behavior transformers: Cloning k modes with one stone. In *NeurIPS*.

[Shao et al., 2024] Shao, H., Hu, Y., Wang, L., Waslander, S. L., Liu, Y., and Li, H. (2024). LMDrive: Closed-loop end-to-end driving with large language models. In *CVPR*.

[Shentu et al., 2024] Shentu, Y., Wu, P., Rajeswaran, A., and Abbeel, P. (2024). From LLMs to Actions: Latent codes as bridges in hierarchical robot control. *arXiv preprint arXiv:2405.04798*.

[Shi et al., 2025] Shi, L. X., Ichter, B., Equi, M., Ke, L., Pertsch, K., Vuong, Q., Tanner, J., Walling, A., Wang, H., Fusai, N., Li-Bell, A., Driess, D., Groom, L., Levine, S., and Finn, C. (2025). Hi Robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*.

[Sima et al., 2025] Sima, C., Chitta, K., Yu, Z., Lan, S., Luo, P., Geiger, A., Li, H., and Alvarez, J. M. (2025). Centaur: Robust end-to-end autonomous driving with test-time training. *arXiv preprint arXiv:2503.11650*.

[Sima et al., 2024] Sima, C., Renz, K., Chitta, K., Chen, L., Zhang, H., Xie, C., Luo, P., Geiger, A., and Li, H. (2024). DriveLM: Driving with graph visual question answering. In *ECCV*.

[Tian et al., 2024] Tian, X., Gu, J., Li, B., Liu, Y., Zhao, Z., Wang, Y., Zhan, K., Jia, P., Lang, X., and Zhao, H. (2024). DriveVLM: The convergence of autonomous driving and large vision-language models. In *CoRL*.

[van den Oord et al., 2017] van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. In *NeurIPS*.

[Veerapaneni et al., 2019] Veerapaneni, R., Co-Reyes, J. D., Chang, M., Janner, M., Finn, C., Wu, J., Tenenbaum, J. B., and Levine, S. (2019). Entity abstraction in visual model-based reinforcement learning. In Kaelbling, L. P., Kragic, D., and Sugiura, K., editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 1439–1456. PMLR.

[Wang et al., 2023] Wang, W., Xie, J., Hu, C., Zou, H., Fan, J., Tong, W., Wen, Y., Wu, S., Deng, H., Li, Z., Tian, H., Lu, L., Zhu, X., Wang, X., Qiao, Y., and Dai, J. (2023). DriveMLM: Aligning multi-modal large language models with behavioral planning states for autonomous driving. *arXiv preprint arXiv:2312.09245*.

[Wu et al., 2022] Wu, P., Jia, X., Chen, L., Yan, J., Li, H., and Qiao, Y. (2022). Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *NeurIPS*.

[Wu et al., 2023] Wu, Z., Dvornik, N., Greff, K., Kipf, T., and Garg, A. (2023). SlotFormer: unsupervised visual dynamics simulation with object-centric models. In *ICLR*.

[Xu et al., 2018] Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., and Huang, T. (2018). Youtube-vos: A large-scale video object segmentation benchmark. In *ECCV*.

[Xu et al., 2024] Xu, Z., Zhang, Y., Xie, E., Zhao, Z., Guo, Y., Wong, K.-Y. K., Li, Z., and Zhao, H. (2024). DriveGPT4: Interpretable end-to-end autonomous driving via large language model. *RA-L*.

[Yan et al., 2021] Yan, W., Zhang, Y., Abbeel, P., and Srinivas, A. (2021). VideoGPT: video generation using VQ-VAE and transformers. *ARXIV*, 2104.10157.

[Yang et al., 2019] Yang, L., Fan, Y., and Xu, N. (2019). Video instance segmentation. In *CVPR*.

[Yu et al., 2024] Yu, P., Xu, J., Weston, J., and Kulikov, I. (2024). Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.

[Zhai et al., 2023] Zhai, J.-T., Feng, Z., Du, J., Mao, Y., Liu, J.-J., Tan, Z., Zhang, Y., Ye, X., and Wang, J. (2023). Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. *arXiv preprint arXiv:2305.10430*.

[Zhang et al., 2024a] Zhang, J., Guo, Y., Chen, X., Wang, Y.-J., Hu, Y., Shi, C., and Chen, J. (2024a). HiRT: Enhancing robotic control with hierarchical robot transformers. In *CoRL*.

[Zhang et al., 2024b] Zhang, J., Huang, Z., Ray, A., and Ohn-Bar, E. (2024b). Feedback-guided autonomous driving. In *CVPR*.

[Zhang et al., 2021] Zhang, Z., Liniger, A., Dai, D., Yu, F., and Van Gool, L. (2021). End-to-end urban driving by imitating a reinforcement learning coach. In *ICCV*.

[Zhang et al., 2024c] Zhang, Z., Tang, S., Zhang, Y., Fu, T., Wang, Y., Liu, Y., Wang, D., Shao, J., Wang, L., and Lu, H. (2024c). AD-H: autonomous driving with hierarchical agents. *arXiv preprint arXiv:2406.03474*.

[Zheng et al., 2022] Zheng, Q., Zhang, A., and Grover, A. (2022). Online decision transformer. In *ICML*.

[Zhou et al., 2019] Zhou, B., Krähenbühl, P., and Koltun, V. (2019). Does computer vision matter for action? *Science Robotics*, 4(30).

[Zimmerlin et al., 2024] Zimmerlin, J., Beißwenger, J., Jaeger, B., Geiger, A., and Chitta, K. (2024). Hidden biases of end-to-end driving datasets. *arXiv preprint arXiv:2412.09602*.

[Zitkovich et al., 2023] Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., Vuong, Q., Vanhoucke, V., Tran, H., Soricut, R., Singh, A., Singh, J., Sermanet, P., Sanketi, P. R., Salazar, G., Ryoo, M. S., Reymann, K., Rao, K., Pertsch, K., Mordatch, I., Michalewski, H., Lu, Y., Levine, S., Lee, L., Lee, T.-W. E., Leal, I., Kuang, Y., Kalashnikov, D., Julian, R., Joshi, N. J., Irpan, A., Ichter, B., Hsu, J., Herzog, A., Hausman, K., Gopalakrishnan, K., Fu, C., Florence, P., Finn, C., Dubey, K. A., Driess, D., Ding, T., Choromanski, K. M., Chen, X., Chebotar, Y., Carbajal, J., Brown, N., Brohan, A., Arenas, M. G., and Han, K. (2023). RT-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*.

[Zoran et al., 2021] Zoran, D., Kabra, R., Lerchner, A., and Rezende, D. J. (2021). Parts: Unsupervised segmentation with slots, attention and independence maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10439–10447.