End to End Planning for Autonomous Driving

Zheyang Lyu & Mathis Finckh

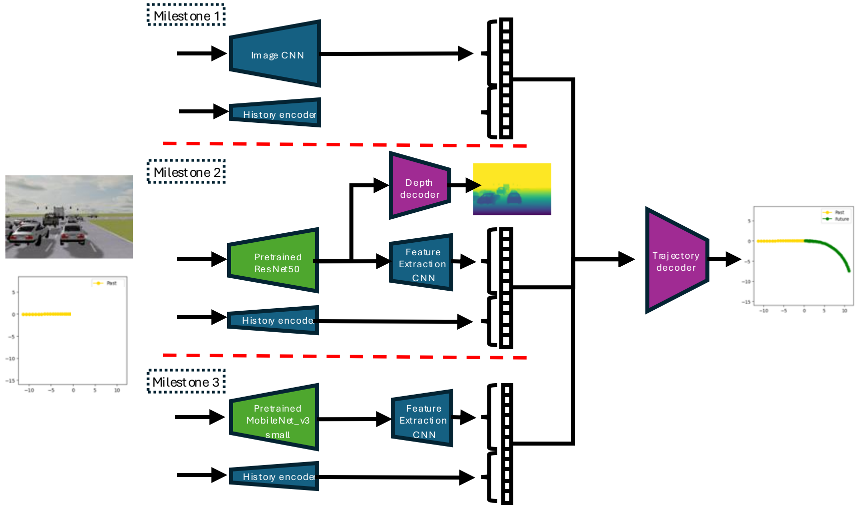
Abstract

Autonomous vehicles must accurately predict future trajectories in highly dynamic and complex environments. This challenge becomes even more critical in real-world deployment, where perception noise and sensor limitations are prevalent. According to Chitta et al. (2023), “the end-to-end approach will have enormous potential over modular stacks in terms of performance and effectiveness” [1]. That’s why our goal is to design an end-to-end trajectory planning model that is:

- **Accurate** under synthetic and real data settings
- **Perception-aware**, using auxiliary tasks like semantic segmentation and depth
- **Generalizable**, capable of adapting from simulation to reality without dense supervision

We tackle this problem in three progressive phases, improving representation power, supervision richness, and domain robustness step by step.

Models

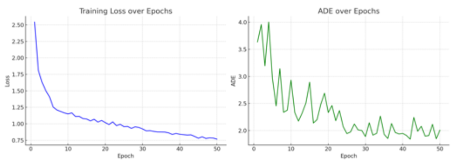


Dataset and Augmentation

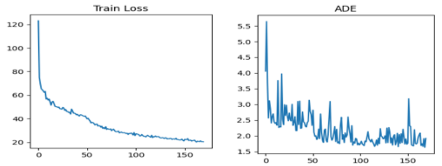
We used the nuPlan dataset using street images and past trajectories as input and future trajectories as output, with depth estimation and semantic segmentation as auxiliary tasks. We augmented our data with:

- Random affine transforms
- Color jitter (brightness, contrast, saturation, hue).
- Horizontal flips for images and trajectory labels.
- Gaussian noise on trajectories.
- Computation of velocity & acceleration, added to history trajectory

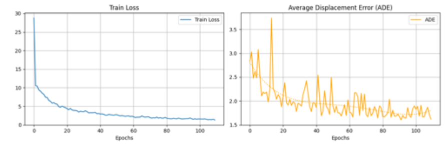
Training Metrics



Milestone 1:
ADE stagnating at 2 since epoch 50, training loss still linearly dropping



Milestone 2:
ADE stagnating at 2 since epoch 60, training loss still linearly dropping.
More complex model and losses doesn’t necessarily translate to performance



Milestone 3:
ADE dropping and stagnating with the training loss, which indicated mora appropriate model and learning

References

[1] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. *End-to-end Autonomous Driving: Challenges and Frontiers*. arXiv:2306.16927, 2024. URL: https://arxiv.org/abs/2306.16927

[2] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. *TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving*. arXiv:2205.15997, 2022. URL: https://arxiv.org/abs/2205.15997

Analysis and Results

Milestone 1: Baseline Establishment
This model served as a functional baseline plateaued above an ADE of 2.0 because of its simplicity and having to learn image feature extraction from scratch.
We implemented a heading loss, one-hot encoding for the driving commands, encoders for all inputs and a few extra layers than the given starter notebook.

Milestone 2: Escalation of Complexity
•**Data augmentation**
•**Backbone Replacement:** Pretrained & finetuned ResNet50 replaced the lightweight CNN.
•**Auxiliary Supervision:** Depth prediction.
•**Trajectory Regularization:** Smoothness and jerk loss.
•**Transformer Integration:** Early experiments with transformer layers to fuse inputs (after feature extraction) in a meaningful way. However, the performance was poor, so we came back to something simpler last minute. (Inspired by [2], to fuse image and trajectory like they fused image and lidar)
With this simpler architecture, we still noticed that the model was not generalizing well, as the training curve was still going down, but all the validation metrics were stagnating. We might have been experiencing overfitting, or just an ill-suited architecture. This model achieved a Kaggle ADE of 1.72.

Milestone 3: Simplification and Convergence
•**Backbone:** Pretrained & finetuned MobileNetV3-Small
•**Loss Function:** Single MSE on position vectors
We first tried to fine-tune the Milestone 2 model and got close to 1.8, but also saw that the given starter notebook was achieving 1.9 ADE, so with the previous conclusions, we thought we'd follow this “simple-better” approach. This architecture was **computationally efficient** and more robust. The pretrained CNN features aligned well with real-world images, while the minimal decoder generalized better on sparse data. This model achieved a **Kaggle ADE** of 1.43.

Learnt Principles:
•Simpler is better: architectural novelty is tempting, but doesn’t always pay off
•Auxiliary losses are high-risk: Additional signals (e.g., heading or depth) introduced instability
•Pretrained backbones matter
•Data augmentation is critical