ME292B Individual Project: Prediction

Jack Leckert, ID: 3039658377

4/5/2024

Abstract

This project builts upon Convolutional Neural Networks (CNN) and Multilayer Perceptrons (MLP) to develop a rasterized prediction model for autonomous driving. The process begins by rasterizing the map into a 3-channel representation and rasterizing the agents across different frames, capturing their history over 10 frames. This results in a comprehensive input with a total of 13 channels. Convolutional Neural Networks (CNNs) are then employed to effectively encode the combined map and agent information. In particular, MobileNetV2 performs remarkably well at predicting unimodal trajectories when integrating vehicle dynamics, reaching a Miss Rate score of 0.4762.

1 Virtual environment

The INTERPRET challenge is only compatible with Ubuntu systems and requires heavy computational resources. The first challenge of this project was therefore to find a virtual environment to run the project. A first try was done using the Virtual Box on an Intel MacBook Pro but one epoch would take approximately 12 hours to train, which is unacceptable given the deadline of this project. The project was therefore run on the Microsoft Planetary computer which gives freely access to a Linux environment including a Linux compatible T4 GPU. With this environment, each epoch would take approximately 1:20 hour to train, which is much more convenient to fine-tune the prediction model.

Figure 1: Microsoft Planetary Computer logo

2 Fully trained prediction model

This first part summarizes the modification done on the provided INTERACTION base code for prediction, training the model from scratch. Due to lack of resources, results are compared on a training basis of 5 epochs. Given minADE, minFDE and MR values are computed by the INTERACTION platform after running the test mode and submitting the csv files. At the end of this part, a table summarizes the different tried modifications. Learning curves are plotted only in the second part as I only figured out later a method to run TensorBoard on the virtual environment.

2.1 Model structure

The rasterized trajectory prediction model is built upon Convolutional Neural Networks (CNN) and Multilayer Perceptrons (MLP), its structure is depicted on Figure 2. After rasterizing the map and its agent to capture their history, the data is encoded through a CNN-based map encoder. This outputs a distribution of features (mean and variance) which is then given to the map decoder.The MLP decode the features and outputs one or multiple predicted trajectories using an MLP.

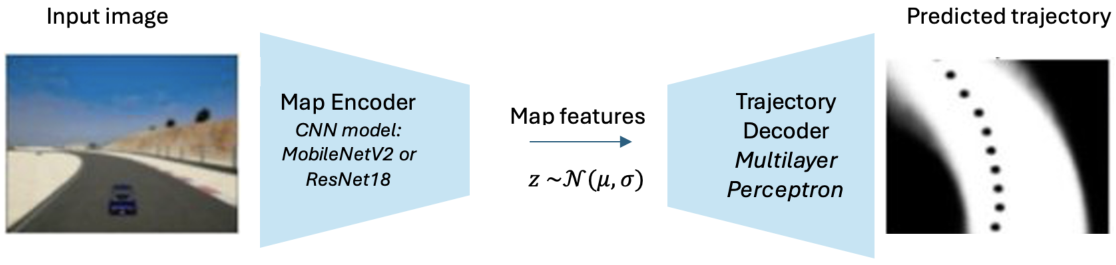


Figure 2: Rasterized Prediction Model structure

MobileNetV2
The base model relies on the MobileNetV2 encoder whose architecture is depicted in the following picture. It does not incorporate any vehicle dynamics constraints, so the number of possible trajectory predictions is bigger. The base model decoder is unimodal, meaning it will predict only one trajectory (the most probable one) given the encoded map features. Finally, the loss function is based on the mean squared error. Batch size is fixed to 64 and learning rate to 1e-3.
Results:

$$minADE = 1.6771$$
$$minFDE = 4.8232$$
$$MR = 0.8187$$

MobileNetV2 & Dynamics
Including vehicle dynamics constraints in the trajectory decoder limits the number of possible trajectories and can improve accuracy of prediction. Dynamics are kept as given, with following features:

- dynamics type: "Unicycle"
- max steering angle: $0.5rad$
- max yaw velocity: 2π
- acceleration bounds: $[-10, 8]m/s^2$
- ddh bounds: $[-2\pi, 2\pi]$
- max speed: $40m/s$

Results:

$$minADE = 1.7333$$
$$minFDE = 4.7076$$
$$MR = 0.7587$$

ResNet18 & Dynamics
The second change made on the prediction model was to change the encoder architecture to a residual neural network. This type of neural network normally does not overfit the data by including residual connections to later hidden layers but it becomes much heavier to train. Its architecture includes 3x more trainable parameters compared to the MobileNetV2. Using the same parameters as before, the training of 5 epochs would give:
Results:

$$minADE = 0.9989$$
$$minFDE = 2.7501$$
$$MR = 0.6781$$

2.2 Loss function

The total loss function can include several loss functions such as the trajectory loss (MSE), the goal loss, the collision loss and the yaw loss when dynamics are included. Trajectory loss is the main loss function and weight is kept equal to 1. The yaw loss weight is fixed to 0.001. Regarding the goal loss, it is set to 0.5, which improves very slightly the test MR score. The collision loss has not been implemented since it required missing utils functions.

2.3 Multi-modal trajectory decoder

As of now, the trajectory decoder is unimodal, meaning it would output only one possible trajectory given the encoded map features. However, real-world scenarios are complex and include uncertainties on the motion of the different agent in the environment. Multimodal predictions can capture diverse behaviors and outcomes, especially in scenarios involving multiple agents or decision-makers, leading to a more accurate representation of potential future states. The GitHub repository [XCIP23] gives some approaches to implement such a model.

First, a multimodal trajectory decoder is implemented (in *base models* line 283) by adding one more dimension to the prediction tensor, the modes. Then the MLP output activation is set to a Softmax to predict the *num modes* probabilities of each trajectories. The prediction loss function is switched to the available Multimodal trajectory loss which takes as inputs the targeted and predicted trajectories, associated with their respective probabilities. Finally, only the most probable trajectory is kept to compute the minADE and minFDE metrics and evaluate the performance on the test set.
Results:

$$
\begin{gathered}
minADE = 2.1696 \\
minFDE = 5.6987 \\
MR = 0.803
\end{gathered}
$$

The multimodal trajectory loss using the Softmax MLP performs significantly worse than the unimodal case. Probabilities seemed to converge rapidly to one for a given mode, leaving the other mode probabilities to 0. By changing the layer dimensions to (64), (128,64) or even (128,128,64), the final results would not change. The problem can arise from 2 sources: 1) the implementation is false or 2) the model is not trained enough and underfitting the data.

2.4 Fine tuning parameters and final results

Coming back to the ResNet18 encoder including vehicle dynamics, some more parameters fine-tuning have been done to optimize the training.
First of all, some randomness was included in the training dataset by shuffling the training examples (setting shuffle to True).
The batch size was increased to 128 and the learning rate was assigned to a schedule: $1e-2$ for 3 epochs, then $1e-3$ for the remaining epochs.
The goal loss weight is reduced to 0.3. 5 epochs of training gives the following results.
Results:

$$
\begin{gathered}
minADE = 0.8180 \\
minFDE = 2.1245 \\
MR = 0.6513
\end{gathered}
$$

The results of this first part are summarized in the following table. They were all obtained with 5 epochs of training the prediction model from scratch.

Table 1: Comparison of the different trials of 5 epochs

Encoder	Dynamics	Loss functions	Modality	Learning rate	Batch size	Shuffle	minADE	minFDE	MR
MobileNetV2	No	trajectory loss	Unimodal	1.00×10^{-3}	64	False	1.6771	4.8232	0.8187
MobileNetV2	Yes	trajectory loss (1.0), yaw loss (0.001)	Unimodal	1.00×10^{-3}	64	False	1.7333	4.7070	0.7587
ResNet18	Yes	trajectory loss (1.0), yaw loss (0.001)	Unimodal	1.00×10^{-3}	64	False	0.9989	2.7501	0.6781
ResNet18	Yes	trajectory loss (1.0), yaw loss (0.001)	Multimodal	1.00×10^{-3}	64	False	2.1696	5.6987	0.8030
ResNet18	Yes	trajectory loss (1.0), yaw loss (0.001), goal loss (0.3)	Unimodal	1.00×10^{-2}	128	True	0.8180	2.1245	0.6513

3 Pretrained model

A pretrained MobileNetV2 was later provided by the faculty to help achieving acceptable results even with lacking computing ressources. The remaining work was achieved using this model.

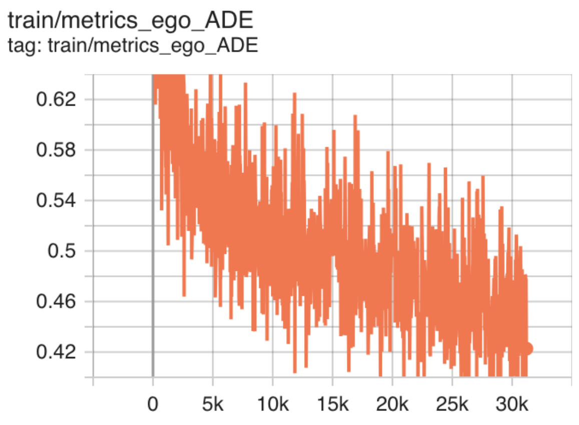
3.1 Model structure

The model structure uses the pretrained MobileNetV2. Goal loss weight is fixed to 0.3 and yaw loss weight to 0.001. The decoder is unimodal. Batch size is fixed to 128 and finally the train dataset is shuffled.
The model was trained for 10 epochs, with a learning rate starting at $1e-2$ for 5 epochs, then $1e-3$ for the remaining epochs.
The use of Tensorboard was finally figured out using the Planetary Computer and following training curves were obtained:

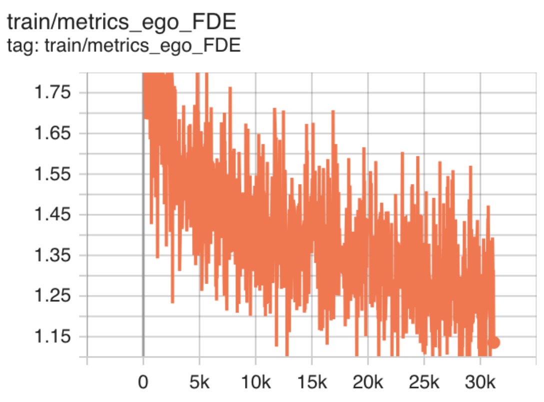
As we can observe on the following plots, the training and validation metrics are on a general trend decreasing and not reaching a converging value. 10 epochs even after pretraining the MobileNetV2 is not enough to optimally train the model.
On another note, we can observe that the training loss contains a strong noise. This is probably due to the strong sparsity available in the dataset. The validation metrics and losses remains however higher than the training metrics, which usually indicates overfitting.
Results:

$$minADE = 0.7227$$
$$minFDE = 2.005$$
$$MR = 0.4762$$

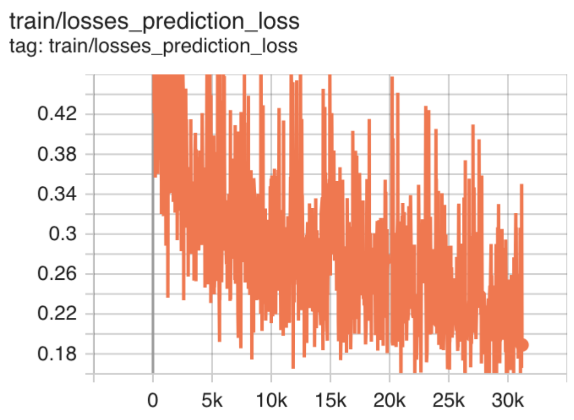
Finally, a multimodal trajectory decoder has been tested with the pretrained MobileNetV2 as well, which gave similar worse performance such as Part 1.



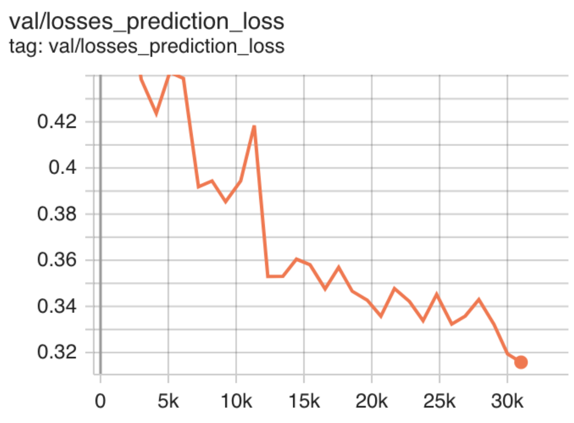
(a) Train ADE



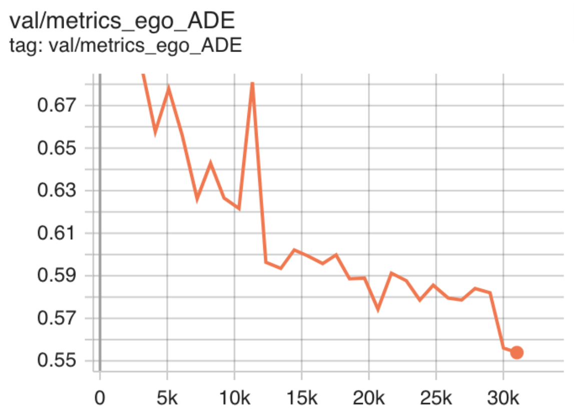
(b) Train FDE



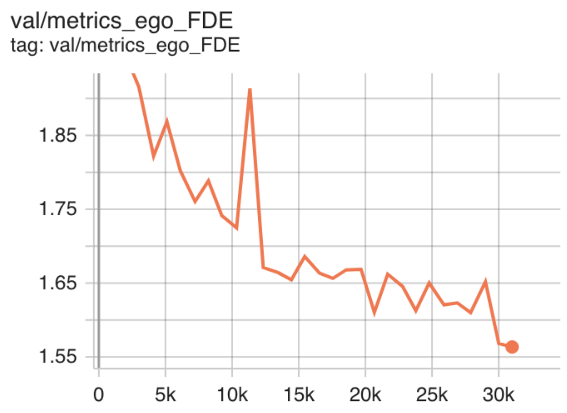
(c) Train trajectory loss



(d) Validation trajectory loss



(e) Validation ADE



(f) Validation FDE

Figure 3: ADE, FDE metrics and trajectory losses comparison between training and validation

4 Conclusion

To conclude, this project has advanced the development of a rasterized prediction model for autonomous driving, utilizing CNNs and MLPs to process map and agent data. Implementing MobileNetV2 with vehicle dynamics improved prediction accuracy, while transitioning to ResNet18 significantly enhanced model performance. Despite the promise shown by multimodal predictions, they did not yield the expected gains, indicating a potential need for further training. Fine-tuning parameters and leveraging a pre-trained MobileNetV2 allowed for continued improvement within computational constraints. The project underscores the potential of deep learning in autonomous systems, though optimal training and multimodal model refinement remain areas for future exploration.

References

[XCIP23] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Bits: Bi-level imitation for traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2929–2936. IEEE, 2023.