Motion Trajectory Prediction with Multi-Model Ensembling

Jingyu Qian
Horizon Robotics Inc.
qianjingyu0322@gmail.com

Yifan Zhuang
Horizon Robotics Inc.
zhuangyf2012@gmail.com

Zhening Yang
Horizon Robotics Inc.
zheningy@gmail.com

Jiajie Chen
Horizon Robotics Inc.
chenjiajie_bm@163.com

Qiang Liu
Horizon Robotics Inc.
proliu@gmail.com

Abstract

This research proposes an ensembling method for motion trajectory prediction, which builds upon multiple base models and employs an expectation-maximization (EM) process as a post-processing step to enhance the trajectory precision. In our implementation, we have employed both raster-based model and vector-based model as the base models. The ensemble model with the EM process results in a significantly higher mAP (mean Average Precision) than each of the base models.

1. Introduction

Given a road map and the historical trajectories of the observed agents, a motion prediction model forecasts the future trajectories of the agents of interest. Based on the encoding format of the map and history trajectories, motion prediction methods may be categorized as raster-based, vector-based and graph-based. The raster-based methods usually use CNN to encode the spatial information, while the other two may use transformers or other vector operations to encode the structured road and agent features. The raster-based models may have more limitations on the range scope (due to memory constraints) and implicit encoding of agent attributes (e.g., velocity), but the CNN structure has been well established and may be preferred by some deep learning accelerators. The vector-/graph-based models, on the other hand, are usually featured with a lighter memory footprint, larger receptive fields, and explicit encoding of agent's attributes, forming a more promising methodology.

From an engineering point of view, if the range scope is small, e.g., for pedestrians and cyclists whose velocity is much smaller than vehicles, the raster-based methods may be much more affordable and can benefit from well-established CNN networks and hardware. However, for vehicles, it makes much more sense to utilize vector or graph-based methods. Considering both, we design a general motion prediction framework that may be an ensemble of one or both of these models, aiming to take advantage of both to achieve better prediction accuracy.

2. Methodology

In this section, we briefly review some recent motion prediction methods, which contribute to our ensemble model. Fig. 1 shows the overview of our method. For each target agent in a scene, we forward its data (past trajectories and surrounding environments) through a group of base models, which can be either raster-based or vector/graph-based. The outputs from each of these models are fed to a post-processing module for ensembling, which consolidates all the outputs into a desired number of trajectories (e.g. 6 final trajectories). For different classes, i.e. vehicle, pedestrian or cyclist, we use different sets of hyper-parameters for the ensembling, which are obtained by a grid-search beforehand. In our case, the base models are chosen to be a raster-based model modified from HOME [3], and a vector-based model with similar structure as Multipath++ [4].

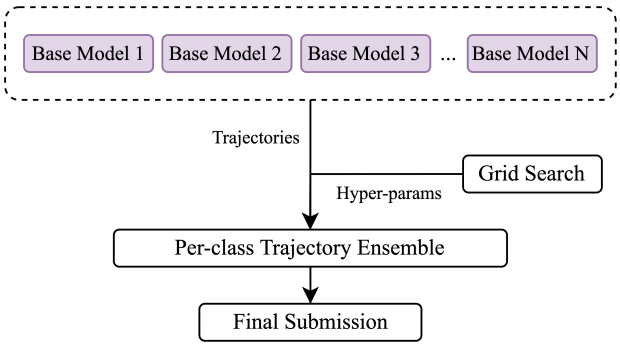
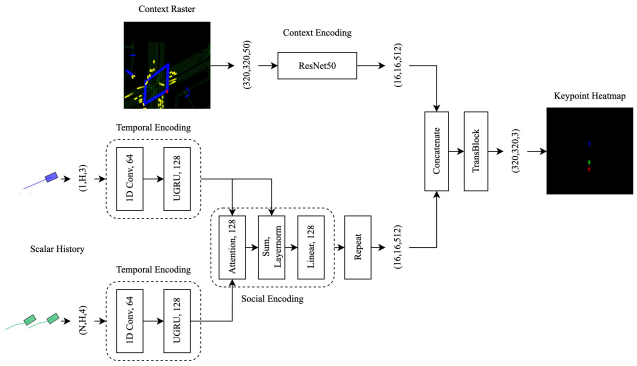


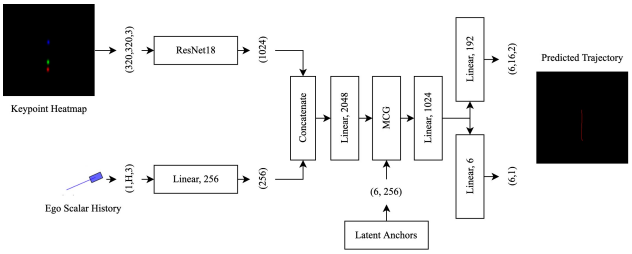
Figure 1. **Method Overview**: Each base model proposes a number of trajectories for an agent, and the ensemble post-process refines these results based on grid-searched hyper-parameters.

2.1. HOME-based Model

The overall structure of the HOME-based model is shown in Fig. 2, which encloses a two-stage algorithm, including a keypoint heatmap prediction stage and a trajectory generation stage. The first stage will predict heatmaps



(a) Heatmap Prediction of Keypoints



(b) Trajectory Generation

Figure 2. **Home-based Motion Prediction Model**: A two-stage motion prediction model – (a) predicts the keypoint heatmaps at specified time-steps (e.g. at 3s, 5s, 8s); (b) takes the heatmaps and the target agent history and outputs the full trajectories and the corresponding scores.

2.1.1 Heatmap Prediction

The first stage follows the original work from HOME [3]. The primary difference of this research lies in the output. HOME only predicts the heatmap representing the last point of the future trajectory. However, such a design may have weak supervision by using a single trajectory point. In contrast, we generate three heatmaps, corresponding to the time-steps at the 3, 5 and 8 seconds respectively. In fact, one may choose to employ all the points along the trajectory for supervision. But we have found it much redundant, as the trajectories are usually quite smooth.

2.1.2 Trajectory Generation

As shown in Fig. 2b, the second stage will generate the whole trajectory using the heatmap from Sec. 2.1.1. HOME applies MR sampling or FDE sampling algorithms to estimate potential final points from the heatmap, then uses an MLP to generate the full future trajectory with the sampled keypoints. But we have found that these algorithms suffer from low-quality heatmaps, e.g. blurry heatmaps. As a result, we adopt a lightweight CNN, e.g. ResNet18, and a global pooling to generate a context vector, which is then concatenated with the feature vector projected from the ego agent history trajectory. We then utilize an attention operator to combine this context vector with a set of latent anchors, and finally forward the combined vector to a linear layer to output the regressed trajectories and their scores. Each output trajectory has 16 points. In our implementation, we have used a MCG as the attention module, which was inspired by [4].

2.2. MultiPath++ Based Model

2.2.1 Inputs

We have used two different model inputs, 256 closest map objects (a line segment of maximum 20 points) of all types (e.g. lane center, road edge, road line) and 128 nearest objects of lane center type only, respectively. 128 nearest lane objects are selected using a Breadth-First-Search algorithm based on road connection information. In this algorithm, we start from the lane center object closest to the target agent, and iteratively collect the lane object that are neighbors to the already collected, until reaching the specified number of lane objects. Note that the neighbors include the lanes both in front of and behind the collected lane nodes.

For each point in a map object, we use a flag vector to encode its attributes, which indicates if it is located in a specific traffic region (e.g. Crosswalk, Speed Bump) and/or close to a stop sign and/or controlled by a traffic light signals. Note that this is not a one-hot vector, as multiple attributes may co-exist.

2.2.2 Encoder

Agent history encoding. Different from [4], we have used GRU (instead of LSTM) for history features encoding of the target agent, without losing accuracy.
Agent interaction encoding. For surrounding agents, we also used GRU to encode their history in the target agent's coordinate frame. For the three types of surrounding agents (i.e. vehicle, pedestrian, cyclist), we have used three GRU encoders respectively, the outputs of which are fused with stacked MCG blocks.
Road network encoding. Instead of using custom lane node encoding defined by start point, end point in [4]. We still used GRU to encode all points for both the 256 all-type road elements and the 128 lane-center-only elements, which achieved similar performance in experiment.

2.2.3 Decoder

The encoded target agent, surrounding agents and road elements are concatenated to form a context vector. Similar to [4], we use latent anchors to fuse with the aggregated context by using a stacked MCG block. The fused embedding is then forwarded to an MLP layer to predict the K trajectories and their probabilities. Each trajectory has 80 points.

2.3. EM-based Ensembling

We adopt the model-agnostic ensembling method proposed in [4], so the trajectories can come either from multiple prediction heads of a single model or multiple models with single prediction heads.

Let L denote the number of base models or prediction heads, where each model or head predicts E trajectories for the target. The ensembling then takes $L \cdot E$ trajectories and consolidate to 6 trajectories required by the task.

2.3.1 Centroid Selection

Let Eq. (1) denote a set of trajectories and corresponding probabilities. We begin by selecting K trajectories out of Ψ that are most representative of the entire set. We adopt the greedy criterion that maximizes the probability sum of the trajectories within τ (the distance threshold) from the selected centroid in Eq. (2). Once a trajectory is selected as a centroid, all the inliers are excluded in the next round of iteration.

$$\Psi = \{(\xi_{11}, p_{11}), ..., (\xi_{1E}, p_{1E}), ..., (\xi_{LE}, p_{LE})\} \tag{1}$$

$$\underset{k\in[1,K]}{\bar{\xi}_k} = \underset{\bar{\xi}_k \in \Psi}{\operatorname{argmax}} \sum_{i=1}^{L}\sum_{j=1}^{E} p_{ij}\mathbb{I}(\|\xi_{ij} - \bar{\xi}_k\| \leq \tau) \tag{2}$$

As an alternative, we've also implemented a non-maximum-suppression(NMS) method to select centroids. Different from the aforementioned greedy selection, NMS is based solely on the probabilities of trajectories, while the criterion of the inliers is the same as the greedy method.

2.3.2 Expectation Maximization

Same as [4], we view all the predictions from all the base models at a single time point as Gaussian mixture models (GMM), described as Eq. (3):

$$\mathbf{p}(x, y; \Psi) = \sum_{j=1}^{L\cdot E} q_j \mathcal{N}(x, y; \mu_j, \Sigma_j) \tag{3}$$

The aggregated results $\bar{\Psi}$ should minimize the KL-divergence $D_{KL}(\Psi||\bar{\Psi})$, which involves updating q, μ and Σ iteratively:

$$\bar{q}_h \leftarrow \sum_{i=1}^{L\cdot E} q_i p(h|\mu_i; \bar{\Psi}) \tag{4}$$

$$\bar{\mu}_h \leftarrow \frac{1}{\bar{q}_h}\sum_{i=1}^{L\cdot E} q_i p(h|\mu_i; \bar{\Psi})\mu_i \tag{5}$$

$$\bar{\Sigma}_h \leftarrow \frac{1}{\bar{q}_h}\sum_{i=1}^{L\cdot E} q_i p(h|\mu_i; \bar{\Psi})[\Sigma_i + (\mu_i - \bar{\mu}_h)(\mu_i - \bar{\mu}_h)^T] \tag{6}$$

In our implementation, we specified a maximum number of iterations as the stopping criterion.

3. Experiments

3.1. Evaluation of MultiPath++

Table 1. **Vehicle result of using Multipath++ as the base model**: We compare the two different inputs to the model. Both experiments were conducted on the vehicle data only, with 6 trajectories as output.

Soft mAP	3s	5s	8s	Average
256 all-type-elem	0.426	0.339	0.238	0.335
128 lane-center	0.405	0.329	0.251	0.328

Tab. 1 summarizes the evaluation of different model input types. One thing we have noticed : for the shorter time horizon, e.g. at 3s and 5s, encoding the road elements of all types showed significantly higher accuracy, but not at the long horizon, i.e at 8s time-step. This is probably due to the cases in which the target agents possess high speed and the collected all-type road elements didn't cover enough range, while the lane-center only elements may cover enough range, but lacks of the richness of the road information. So for better accuracy, one may need to use a larger number road elements with all the types.

3.2. Evaluation of HOME-based Model

We evaluate the HOME-based models on cyclists and pedestrians, as shown in Tab. 2 and Tab. 3. Here, "Basic" represents the baseline. "Data Augment" refers to the temporal augmentation instead of spatial augmentation. Compared to vehicles, which have over 10^6 training samples, pedestrians and cyclists face the challenge of insufficient training data. To mitigate this, we conducted two augmentation methods. First, we collected all the other agents besides the target agents that are already given, and added their future trajectories as additional training samples. Then, we augmented the trajectories horizons, by randomly forwarding the trajectory to 1 to 5 seconds into the future. After data augmentation, the size of the training set for pedestrians and cyclists becomes around 250,000 and 700,000.

Table 2. Pedestrian Motion Prediction Accuracy (Data Augmentation)

Algorithms	Soft mAP	mAP	Miss Rate
Basic	0.353	0.338	0.087
Data Augment	0.360	0.334	0.078

Table 3. Cyclist Prediction Accuracy (Data Augmentation)

Algorithms	Soft mAP	mAP	Miss Rate
Basic	0.268	0.249	0.241
Data Augment	0.297	0.279	0.226

As shown in Tab. 2 and Tab. 3, the temporal data augmentation can increase the accuracy significantly. The improvement is more obvious when one category has fewer training samples. For cyclists, the increasing scale is over 10% compared to the baseline. Besides evaluating the impact of the data augmentation, we also conducted ablation studies testing different trajectory regression algorithms. Tab. 4 depicts the results of the two methods. The first row shows the results of using CNN to regress the heatmap into trajectories. The second row shows the results of using the sampling method proposed by [2, 3] followed by an MLP. As shown by Tab. 4, the CNN-based method is much more accurate than using the MR sampling. Another ablation test evaluates the usage of latent anchors, as shown in Tab. 5. The first row shows the results of using latent anchors proposed by [4], and the second row shows the results of using fixed anchors proposed by [1]. As can be seen, the latent anchors showed superior performance, as they are more adaptive to different scenarios.

Table 4. Pedestrian Prediction Accuracy (Sampling)

Algorithms	Soft mAP	mAP	Miss Rate
Basic	0.353	0.338	0.087
MR Sampling	0.271	0.268	0.113

Table 5. Pedestrian Prediction Accuracy (Anchors)

Algorithms	Soft mAP	mAP	Miss Rate
Latent Anchors	0.353	0.338	0.087
Fixed Anchors	0.338	0.313	0.091

3.3. Evaluation of Ensembling

In our experiments, we trained a multipath++ based model on vehicle data, and two HOME-based models on pedestrian and cyclist data respectively. Each base model has 6 prediction heads and each head predicts 6 potential trajectories. As a result, each model predicts 36 trajectories as inputs for ensembling.

In order to get an optimal ensemble results, we employed a grid search algorithm to find the hyper-parameters. The initial searches allowed us to fix some hyper-parameters, reducing the search space size. Specifically:

- We limit the EM update to 3 iterations;
- Vehicle class uses the outputs from multipath++ based model only, and pedestrian and cyclist classes use the outputs from the HOME-based models only.

Our final submission for each category is shown in Tab. 6. "Cent." refers to centroid selection methods. τ represents the distance threshold used for the centroid selection. Note that for vehicle we have used $l1$ distance and $l2$ for pedestrian and cyclist. "EM" indicates whether the iterative EM algorithm is used. "std" is a heuristic standard deviation value used for the covariance matrix of the GMM's, in which we assumed an i.i.d distribution across all the trajectories and the time-steps. "Base" means the type of models during the ensemble.

Table 6. Hyper-params for Each Category

Category	Cent.	τ	EM	std.	Base
Vehicle	greedy	0.74	NO	N/A	MTP++
Pedestrian	nms	0.94	YES	1.4	HOME
Cyclist	nms	2.0	YES	3.4	HOME

Table 7. Ensemble Prediction Accuracy on Validation Set

Category	Soft mAP	mAP	Miss Rate
Vehicle	0.335	0.293	0.168
Pedestrian	0.360	0.334	0.078
Cyclist	0.297	0.279	0.226
Vehicle Ens.	**0.405**	**0.387**	**0.159**
Pedestrian Ens.	**0.381**	**0.374**	0.083
Cyclist Ens.	**0.367**	**0.363**	**0.222**
Average Ens.	**0.384**	**0.375**	**0.154**

Table 8. Final results on Test Set

Algorithms	Soft mAP	mAP	Miss Rate
MTRA	0.459	0.450	0.116
golfer	0.426	0.412	0.135
HBEns	**0.380**	**0.370**	**0.159**
DM	0.377	0.371	0.165

We show our ensemble results on the validation set in Tab. 7 and the test set in Tab. 8, both submitted against the online server. As can be concluded, the implemented model ensemble method significantly boosts the overall prediction accuracy. From grid-search, our best results were obtained

by using the greedy centroid selection on vehicles and NMS centroid selection on pedestrians and cyclists. The reason for this result may be that the trajectories of vehicles are not as diverged as the pedestrians or cyclists, due to the fact that vehicles do not make changes as easily as the latter two. So for vehicles, NMS-based sampling tends to select the trajectories close to each other, while the greedy centroid sampling would prevent such cases. On the other hand, cyclists' and pedestrians' trajectories tend to be more omnidirectional compared to vehicles, so it makes more sense to use the confidence-based sampling, such as NMS.

4. Conclusion

In this work, we have shown a multi-model ensemble method for motion trajectory prediction by combining raster-based and vector-based models to achieve superior prediction accuracy. An EM-based ensembling module is utilized as a post-processing step, which significantly enhances the mAP of the predicted trajectories.

References

[1] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 4

[2] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. *arXiv preprint arXiv:2109.01827*, 2021. 4

[3] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 500–507. IEEE, 2021. 1, 2, 4

[4] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. *arXiv preprint arXiv:2111.14973*, 2021. 1, 2, 3, 4