Golfer: Trajectory Prediction with Masked Goal Conditioning MnM Network

Xiaocheng Tang, Soheil Sadeghi Eshkevari, Haoyu Chen, Weidan Wu, Wei Qian, Xiaoming Wang
DiDi Research, Autonomous Driving

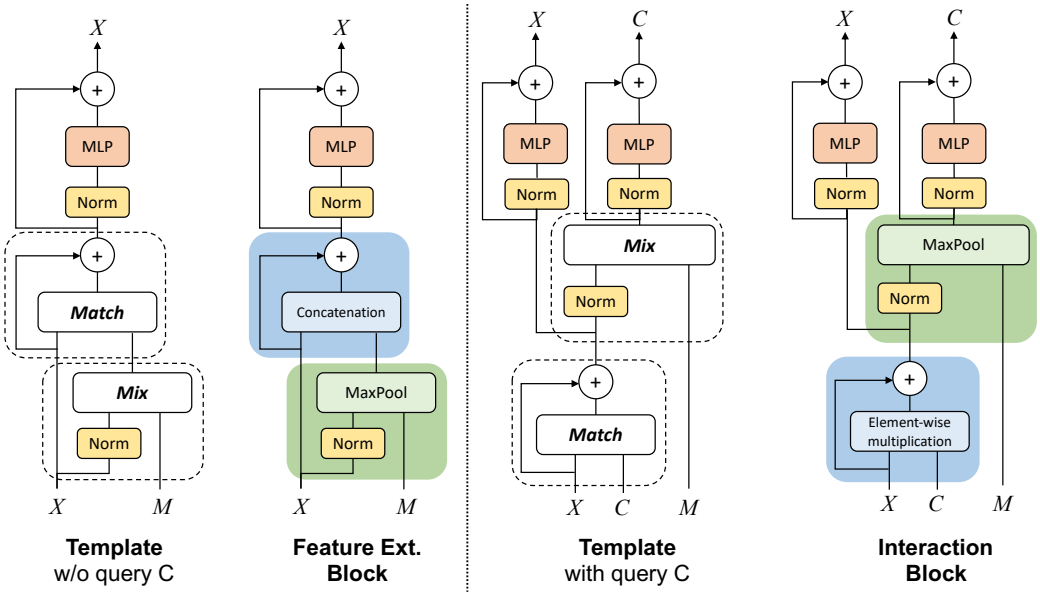

Figure 1: The general structure of **MnM Network** and the specific instantiation in our **golfer** model that was ranked **1st** according to minADE and **2nd** according to Soft mAP in the 2022 Waymo Open Dataset Motion Prediction challenge.

1 Introduction

Transformers have achieved remarkable success and gained much interest in multiple fields including NLP and computer vision. Its performance in trajectory prediction, however, remains unsatisfying [2, 5]. Particularly, trajectory prediction in an autonomous vehicle setting poses unique challenges to Transformers partly due to its inefficiency in obtaining a conditioned interactive encoding for sets of elements, such as modeling the relationships between the ego agent and the road objects. Additionally, recent work have started to raise questions on core design choices of Transformers especially the attention component and whether it is essential to the remarkable effectiveness of Transformers [6, 4, 1]. One of the very recent work [4], for example, replaces the attention module with MLPs and still attains competitive performance on image classification benchmarks. In fact, even earlier work such as PointNet [3] already shows that using a simple max pooling as the token mixers can achieve strong performance for 3D classification and segmentation.

Here we explore the effectiveness of Transformer-like architectures in AV trajectory prediction. Specifically, we propose a general architectural module, **Mix and Match** (MnM) block, that is especially effective at modeling conditioned interactive relationships and that encompasses Transformer as a special instance. The resulted **MnM network**, consisted of only the stacked MnM blocks, is

shown to achieve superior performance by directly predicting the trajectories from the raw point-like sets of agent and road inputs. Particularly, equipped with the novel masked goal conditioning and the MnM network our trajectory prediction model named **golfer** ranked the second place on the Waymo Open Motion Dataset leaderboard as of May 23, 2022.

Waymo motion prediction competition has been a legacy platform for benchmarking best efforts for motion prediction in self-driving industry and research. In this competition, the objective is to predict agent trajectories for 8.0 seconds in the future and for at most eight heterogeneous agents in any given scene. The competition is an annual event that scores submissions based on a set of evaluation metrics including mAP, minADE, minFDE, miss rate, etc. In this report, we present an overview of the model that was ranked **1st** according to minADE and **2nd** according to Soft mAP in the 2022 Waymo motion prediction competition.

In summary, we propose a general form of set transformation that is found strongly effective in learning cross-correlations between items of a set as well as a set of items and a shared context. This transformation block is used in various settings throughout the model architecture. In terms of data encoding, we propose a hierarchical vector-based scene representation that enables a multi-level feature extraction scheme. To improve the efficacy of the predictive model, we use ensemble of models for the final prediction. More details on the main contributions of the designed model is given in the following sections.

2 Method

2.1 MnM Block

We present the core component of the golfer architecture, which we denote as the Mix and Match (MnM) Block. An illustration is shown in Figure 1. It is the generalization of the transformers such that the attention mechanism can be formulated as a special instance of the Mix and Match operation we will introduce below. We will see that with the abstraction of the MnM block we are able to replace the expensive attention computation with a much more efficient operation, as simple as a pooling layer, while retaining most expressive power, if not more, of the transformers.

Let the input be embedded as X which consists of a sequence of n tokens each with embedding dimension d. Let M denote the binary mask of X that for each token X_i we have one bit M_i indicating whether the corresponding token is valid or not. Additionally we define a query vector C with the same embedding dimension as one token in X. The query vector C is optional and depending on whether it is available the MnM block has two variants with minor differences, e.g., $X \leftarrow \text{MnM}(X, M)$ and $X, C \leftarrow \text{MnM}(X, C, M)$. The basic MnM block can be expressed as

$$
\begin{aligned}
C &\leftarrow \text{Mix}(\text{Norm}(X), M), \\
S &\leftarrow \text{Match}(C, X) + X, \\
X &\leftarrow \sigma(\text{Norm}(S)W_1)W_2 + S.
\end{aligned}
$$

where $\text{Norm}(\cdot)$ denotes the normalization layer such as Batch or Layer Normalization and $\sigma(\cdot)$ is a non-linear activation function such as ReLU or GELU. Here we introduce two new operators, $\text{Mix}(\cdot, \cdot)$ and $\text{Match}(\cdot, \cdot)$. `Mix` is a module that mixes token information and `Match` works by matching back the mixed information to each token. As an example, consider the self-attention mechanism, where `Mix` is multiplying X by itself followed by a softmax and `Match` takes the $n \times n$ attention matrix C and multiplies it with X. Alternatively, and in a much simpler fashion, `Mix` can be simply a pooling operator, in which case C will be a single vector, and `Match` can be either a concatenation or a element-wise product followed by a linear layer to map back to dimension d if necessary. We argue that this is a more efficient alternative compared to the attention with no less expressive power. The MnM block with a pre-given query vector C is largely the same as the basic one but with an additional MLP layer on the query as follow

$$
\begin{aligned}
S &\leftarrow \text{Match}(C, X) + X, \\
C &\leftarrow \text{Mix}(\text{Norm}(S), M), \\
X &\leftarrow \sigma(\text{Norm}(S)W_1)W_2 + S, \\
C &\leftarrow \sigma(\text{Norm}(C)W_3)W_4 + C.
\end{aligned}
$$

Finally we note that a multi-headed version of MnM block can be obtained in a similar manner as in the transformers. The main advantage is parameter efficiency such that for the same size of the network we observe the best performance with Multi-headed MnM blocks.

2.2 Auxiliary Masked Goal Conditioning

Inspired by recent advances in both language modeling and goal-conditioned RL, we apply a masking strategy to the targeted trajectory and use that as an additional input to the model. Note that the prediction task corresponds to using a fully masked input (100%), and that a zero masked input becomes trivial to predict, e.g., copying from input to the output. Anything in between transforms the problem to estimating the conditional probability. In practice we use a 85% mask. We further restrict each sample to contain at most one unmasked state in the targeted trajectory and remove that state from the loss computation. To encode the masked targets we treat it either as one of the agent or the road objects, and add it to the set of agents or road objects with a coin toss. We find that this helps the network to capture the nonlinear mappings between the input and the output, e.g., a small change in the current scene sometimes demands a dramatic behavior correction.

As the training objective, we follow the common practice and use a double-headed loss with multi-mode Gaussian Mixture and a maximum likelihood classification head. The classification head predicts a softmax distribution over the mixture components. Each component is consisted of a time series of (x, y) gaussian. We compare each component with the targeted trajectory and only compute the loss based on the one closest to the groundtruth in Euclidean distance.

3 Golfer

Our high-level architecture design consists of two main modules: Encoder and Decoder. The encoder module is designed to extract salient features from multi-model input channels (including maps and agents' past trajectories), encode interactions, and finally, produce a latent feature vector to the decoder module for final predictions. In motion planning task, the input data representation is a design choice that would affect the performance and efficacy. In this study, we propose a hierarchical scene and past trajectory representation that is concise and compatible with the MnM design. In this approach, roadmap objects as well as agents' histories are encoded as a sequence of points with certain embeddings. This representation shows multiple advantages: (1) extremely high data efficiency compared to the rasterized presentation, and (2) inherent capability of capturing semantic information compared to vector-based representations that encoded polylines into separate line segments with no context.

The encoder architecture includes three main steps: (a) feature extraction from agents' history, (b) feature extraction from roadmap polylines, (c) interaction modeling between extracted features and ego features. The outputs from last step are concatenated and passed to the decoder module for trajectory regression and classification.

The feature extractor design, as shown in Figure 1, is inspired by graph neural networks (GNNs). Each input object (road polyline or agent history) is transposed into a sequence of multiple tokens (points in road polylines and agent coordinates in agent history) S as well as a context vector c that includes shared or global features for the group of tokens (for road polylines, it can be lane type and traffic signal encodings). In addition, binary mask M indicates the availability of the sequential embeddings in case of encountering partial or incomplete poylines. Given these, the following feature extractor (FE) block is proposed based on the MnM block with:

$$\text{Mix} \leftarrow \text{MaxPool} \quad \& \quad \text{Match} \leftarrow \text{Concat}.$$

This block extracts nonlinear features from raw sequential embeddings, passes massages between sequential nodes via *MaxPool(.)*, and incorporates global message via contextual vector c. For further improvement, a stacked and multi-headed version of the FE block is used in the final model.

In this design, the FE block extracts features from points of each polyline (local feature extraction). The polyline transformed features are then aggregated by a permutation-invariant transformation to produce a vector of latent information for the given poyline. In the next level, latent features from all polylines in the current scene interact particularly with the vector of latent features for the ego agent. The process of interaction is also modeled by a multi-headed MnM block. Here, the matching

operation is an element-wise multiplication between the ego feature vector (as context) and latent features tensor. The mixing operation is a *MaxPool(.)*.

The ego context f_E interacts with latent features tensors from roadmap polylines and agent histories separately to produce f_R and f_A. The final product of the encoder module is computed as: $f_{enc} = \text{MLP}(\text{Concat}[f_E, f_R, f_A])$. The decoder design is more straight-forward: the encoded feature f_{enc} is passed through independent branches of *MLP(.)*'s to predict a set of probable future trajectories along with corresponding occurrence probabilities.

Note that due to the multi-level process of feature extraction starting from point-level feature transformation to polyline-level feature aggregation and finally, scene-level feature transformation and aggregation, we term the encoder design as a "hierarchical scene encoding" that is compatible with heterogeneous data.

The final output is an ensemble of multiple models. In particular, we collect all outputs from the models and apply a weighted clustering using kmeans with a predefined number of clusters, e.g., 6. The centroid of each cluster and its corresponding aggregated and normalized weight is used as the predicted trajectory and the associated probability score.

References

[1] H. Liu, Z. Dai, D. So, and Q. V. Le. Pay attention to mlps. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9204–9215. Curran Associates, Inc., 2021.

[2] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. Weiss, B. Sapp, Z. Chen, and J. Shlens. Scene Transformer: A unified architecture for predicting multiple agent trajectories. *arXiv.org*, June 2021.

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[4] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. P. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy. MLP-mixer: An all-MLP architecture for vision. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[5] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, and B. Sapp. MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction. *arXiv.org*, Nov. 2021.

[6] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan. MetaFormer is Actually What You Need for Vision. *arXiv.org*, Nov. 2021.