REDMOTION: MOTION PREDICTION VIA REDUNDANCY REDUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Predicting the future motion of traffic agents is vital for self-driving vehicles to ensure their safe operation. We introduce RedMotion, a transformer model for motion prediction that incorporates two types of redundancy reduction. The first type of redundancy reduction is induced by an internal transformer decoder and reduces a variable-sized set of road environment tokens, such as road graphs with agent data, to a fixed-sized embedding. The second type of redundancy reduction is a self-supervised learning objective and applies the redundancy reduction principle to embeddings generated from augmented views of road environments. Our experiments reveal that our representation learning approach can outperform PreTraM, Traj-MAE, and GraphDINO in a semi-supervised setting. Our RedMotion model achieves results that are competitive with those of Scene Transformer or MTR++. We provide an anonymized open source implementation that is accessible via Colab: `https://colab.research.google.com/drive/16pwsmOTYdPpbNWf2nm1olXcx1ZmsXHB8`

1 INTRODUCTION

It is essential for self-driving vehicles to understand the relation between the motion of traffic agents and the surrounding road environment. Motion prediction aims to predict the future trajectory of traffic agents based on past trajectories and the given traffic scenario. Recent state-of-the-art methods (e.g., Shi et al. (2022); Wang et al. (2023); Nayakanti et al. (2023)) are deep learning methods trained using supervised learning. As the performance of deep learning methods scales well with the amount of training data (Sun et al., 2017; Kaplan et al., 2020; Zhai et al., 2022), there is a great research interest in self-supervised learning methods, which generate supervisory signals from unlabeled data. While self-supervised methods are well established in the field of computer vision (e.g., Chen et al. (2020); Radford et al. (2021); He et al. (2020)), their application to motion prediction in self-driving has only recently started to emerge (e.g., Xu et al. (2022); Azevedo et al. (2022)). One of the main reasons contributing to this is the limited availability and relatively small size of datasets for motion prediction in self-driving until recently (e.g., highD (Krajewski et al., 2018) 147 hours recorded vs. Waymo Open Motion dataset (Ettinger et al., 2021) 570 hours recorded).

In this work, we focus on HD map assisted motion prediction. We introduce RedMotion, a transformer model for motion prediction that incorporates two types of redundancy reduction for road environments. Specifically, our model learns augmentation-invariant features of road environments as self-supervised pre-training. We hypothesize that by using these features, relations in the road environment can be learned, providing important context for motion prediction.

We specifically target transformer models for three reasons: **(a)** Transformers are successfully applied to a wide range of applications in natural language processing (e.g., Vaswani et al. (2017); Brown et al. (2020); OpenAI (2023)), computer vision (e.g., Dosovitskiy et al. (2020); Carion et al. (2020); Meinhardt et al. (2022)), and time-series prediction (e.g., Zhou et al. (2021; 2022)). Therefore, it is likely that enhancements in training mechanisms in a particular application will also apply to other applications. **(b)** Transformers have no inductive biases for generating features based on spatial correlations (Raghu et al., 2021). Therefore, appropriate mechanisms must be learned from data. **(c)** The performance of transformers on various downstream tasks scales very well with datasets (Kaplan et al., 2020; Zhai et al., 2022).

Our main contributions are the two types of redundancy reduction incorporated in our model:

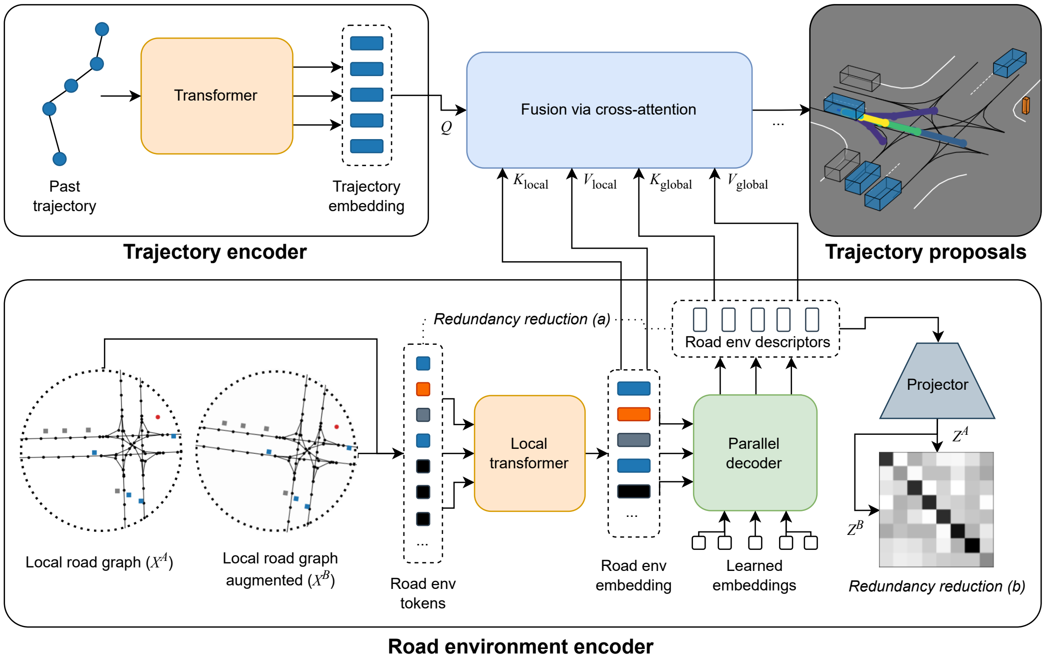


Figure 1: **RedMotion model.** Our model consists of two encoders. The trajectory encoder generates an embedding for the past trajectory of the current agent. The road environment encoder generates sets of local and global road environment embeddings as context. All embeddings are fused via cross-attention to yield trajectory proposals per agent.

1. Redundancy reduction induced by an internal transformer decoder that reduces a variable-sized set of road environment tokens, such as road graphs with agent data, to a fixed-sized embedding.
2. Self-supervised redundancy reduction between embeddings generated from augmented views of road environments.

2 Related Work

Recent works on motion prediction utilize a variety of deep learning model architectures, including transformers, graph-neural networks (GNNs), or convolutional neural networks (CNNs).

Transformer models for motion prediction. Ngiam et al. (2022) use PointNet (Qi et al., 2017) to encode polylines as a road graph. They fuse information from agent interactions across time steps and the road graph using attention mechanisms. Nayakanti et al. (2023) combine early and late fusion with two basic primitives: a self-attention encoder and a cross-attention decoder. They investigate how to use a pure transformer without domain-specific modifications resulting in a motion prediction architecture that is similar to transformer architectures from other domains.

GNN models for motion prediction. Gao et al. (2020) generate vectorized representations of HD maps and agent trajectories to use a fully-connected homogeneous graph for inspecting traffic scenes from the viewpoint of every agent. For their homogeneous graph, they learn a node embedding for every object in the scene. Later works utilizing GNNs use heterogeneous graphs (Monninger et al., 2023; Grimm et al., 2023; Cui et al., 2023). Monninger et al. (2023) and Cui et al. (2023) highlight that choosing a fixed reference coordinate frame is vulnerable to domain shift and aim for viewpoint-invariant representations. These works store spatial information of lanes and agents in edges. Compared to transformers, GNNs require additional modules to generate graph representations for multi-modal inputs. Furthermore, the scaling properties of GNNs can be undesirable.

CNN models for motion prediction. CNN-based approaches have emerged as straightforward yet effective baselines in motion prediction. These approaches often employ a fixed CNN-head, with only the head being adapted to the specific requirements of the application. Recent works remain competitive in motion prediction benchmarks (Chai et al., 2020; Konev et al., 2022; Varadarajan et al., 2022). However, CNN architectures typically tend to require larger models compared to GNNs or tranformers due to the low information content per pixel versus in vector representations.

Self-supervised learning for motion prediction. Labeled data requirements of preceding approaches motivate the application of self-supervised learning on motion prediction. Balestriero et al. (2023) categorize self-supervised representation learning methods into major families. **The deep metric learning family:** PreTraM (Xu et al., 2022) exploits for contrastive learning that a traffic agent's trajectory is correlated to the map. Inspired by CLIP (Radford et al., 2021), the similarity of embeddings generated from rasterized HD map images and past agent trajectories is maximized. Therefore, past trajectories are required, which limits the application of this method to annotated datasets. Ma et al. (2021) improve modeling interactions between traffic agents via contrastive learning with SimCLR (Chen et al., 2020). They rasterize images of intersecting agent trajectories and train the corresponding module by maximizing the similarity of different views of the same trajectory intersection. Accordingly, only a small part of the motion prediction pipeline is trained in a self-supervised manner and annotations are required to determine the trajectory intersections. **The masked sequence modeling family:** Chen et al. (2023) and Yang et al. (2023) propose masked autoencoding as pre-training for motion prediction. Inspired by masked autoencoders (He et al., 2022), they mask out parts of the road environment and/or past trajectory points and train to reconstruct them. When applied to past trajectory points, these approaches require annotated trajectory data. **The self-distillation family:** GraphDINO (Weis et al., 2023) is a self-supervision objective designed to learn rich representations of graph structures and thus can be applied to road graphs used for motion prediction. Following DINO (Caron et al., 2021), the learning objective is a self-distillation process between a teacher and a student model without using labels. Compared to the previously mentioned methods, self-distillation methods tend to require more hyperparameter tuning (e.g., for temperatures or teacher weight updates). Besides these families, Azevedo et al. (2022) use graph representations of HD maps to generate possible traffic agent trajectories. Trajectories are generated based on synthetic speeds and the connectivity of the graph nodes. The pre-training objective is the same as for the subsequent fine-tuning: motion prediction. While this method is well adapted to motion prediction, it requires non-trivial modeling of agent positions and synthetic velocities when applied to non-annotated data.

3 Method

In this work, we propose a transformer model for motion prediction that generates road environment descriptors via redundancy reduction as self-supervision objective.

3.1 Redundancy reduction for learning rich representations of road environments

We use the redundancy reduction principle (Barlow, 2001; Zbontar et al., 2021) to learn rich representations of road environments. Following Ulbrich et al. (2015), we define a road environment as lane network and traffic agent data. In the context of deep learning, Zbontar et al. (2021) define redundancy reduction as reducing redundant information between vector elements of embeddings generated by deep learning models. We implement two types of redundancy reduction:

(a) Redundancy reduction between token sets. Reduction from a variable length set of road environment tokens to a fixed set of road environment descriptors (RED). RED are a fixed sized set of tokens that represent agent and lane features (see input road environment encoder in Figure 1). To capture global context, i.e., environment and lane features, we use a global cross-attention mechanism between RED tokens and road environment tokens (see parallel decoder in Figure 1). Accordingly, every RED token can attend to every road environment token. The Wayformer (Nayakanti et al., 2023) model employs a similar decoding mechanism. It uses learned embeddings to decode a fixed set of trajectory proposals from a context embedding of variable length. Compared to Wayformer, we encode past agent trajectories and environment context with separated encoders, offering more flexibility for specialized self-supervised pre-training.

(b) Redundancy reduction between embeddings. These embeddings are generated from RED tokens (see projector in Figure 1). This self-supervision objective, Road Barlow Twins (RBT), is based on Barlow Twins (Zbontar et al., 2021), which aims to learn augmentation-invariant features via redundancy reduction. For each training sample (X^A in Figure 1) we generate an augmented view (X^B). We use uniform distributions to sample random rotation (max. +/- 10°) and shift augmentations (max. +/- 1m). Afterwards, the local transformer and parallel decoder within the road environment encoder generate a set of RED tokens per input view. Finally, an MLP-based projector generates two embeddings (Z^A and Z^B) from the RED tokens. For the two embedding vectors, a cross-correlation matrix is created. The training objective is to approximate this cross-correlation matrix to the corresponding identity matrix, while reducing the redundancy between individual vector elements. By approximating the identity matrix, similar RED token sets are learned for similar road environments. The redundancy reduction mechanism prevents that multiple RED tokens learn similar features of an environment, increasing diversity. Accordingly, our proposed self-supervision objective belongs to canonical correlation analysis family (Balestriero et al., 2023). The mentioned modules of the road environment encoder are shown with more details in Figure 2.

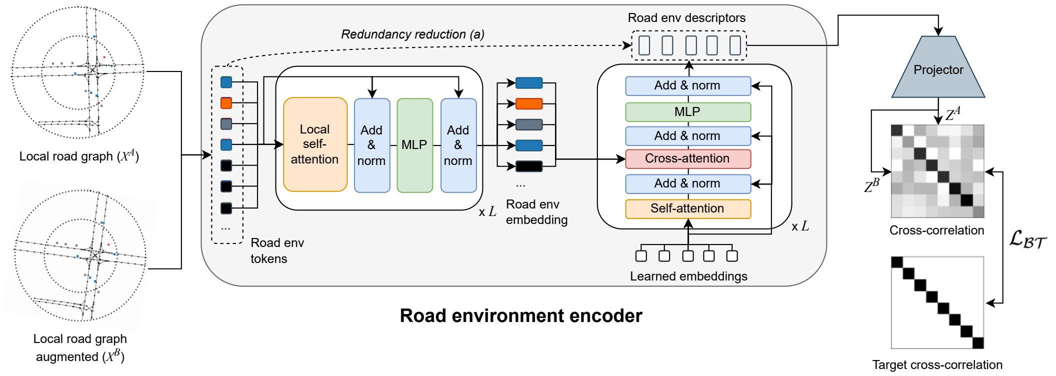


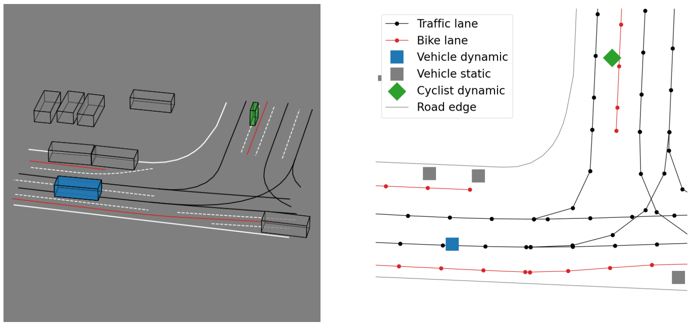
Figure 2: **Road environment encoder.** The circles in the local road graphs denote the maximum distance for considered lane network (outer) and agent nodes (inner). $\mathcal{L}_{\mathcal{BT}}$ is the Barlow Twins loss, L is the number of modules.

3.2 Road environment description and motion prediction model

Our proposed motion prediction model (RedMotion) is a transformer model that builds upon the aforementioned two types of redundancy reduction. As input, we generate road environment tokens for agents and lanes based on a local road graph (see Figure 1). In this local road graph, the current agent is in the center and we consider lane nodes within a radius of 50 meters and agent nodes within 25 meters. As input for the road environment encoder, we use a list of these tokens sorted by token type, polyline, and distance to the current agent. The road environment encoder learns a semantic embedding for each token type, which is concatenated with the position relative to the current agent. We use the current speed information to learn separate embeddings for static and dynamic agents, i.e., we define agents with a current speed greater than $0.0\,\mathrm{m/s}$ as dynamic. In Figure 3b static vehicles are marked as grey squares and dynamic vehicles as blue squares.

Figure 3c shows the vocabulary size (number of individual embeddings), the dimension of individual embeddings, and whether they are learned during training. In the road environment encoder, we additionally use a learned linear projection to project the concatenated semantic and positional embedding to the model dimension.

Since local relations are especially important for processing lanes, we use local attention (Beltagy et al., 2020) layers. Compared to classical attention layers, these have a local attention mechanism within a limited window instead of a global attention mechanism. In addition to the focus on local relations, this reduces memory requirements and allows us to process long input sequences (max. 1200 tokens). Correspondingly, the receptive field of each token grows with an increasing number of local attention layers. Figure 4 shows how a polyline-like representation is built up for a traffic lane token. For a better illustration, we show an example of an attention window of 2 tokens



(a) Local road environment

(b) Lane graph with agents

Type	Size	Dimension	Learned?
Local transformer in road env. encoder			
Semantic	11	4	✓
Positional	-	2	×
Parallel decoder in road env. encoder			
Semantic	100	118	✓
Positional	100	10	✓
Trajectory encoder			
Semantic	3	4	✓
Positional	-	2	×
Temporal	10	1	×
Fusion via cross-attention			
Semantic	2	128	✓

(c) Embedding tables

Figure 3: **Road environment description.** Local road environments are first represented as lane graphs with agents, afterwards, we generate token sets as inputs by using embedding tables for semantic types and temporal context. Positions are encoded relative to the current agent, except in the parallel decoder for RED tokens, where positional embeddings are learned.

per layer. In our model, we use an attention window size of 16 tokens. The polyline-like structures built in the local transformer are set in relation to all surrounding tokens in the parallel decoder of RED tokens. This is implemented by a global cross-attention mechanism from RED tokens to road env tokens (see Figure 1). Thus, global representation can be learned by RED tokens.

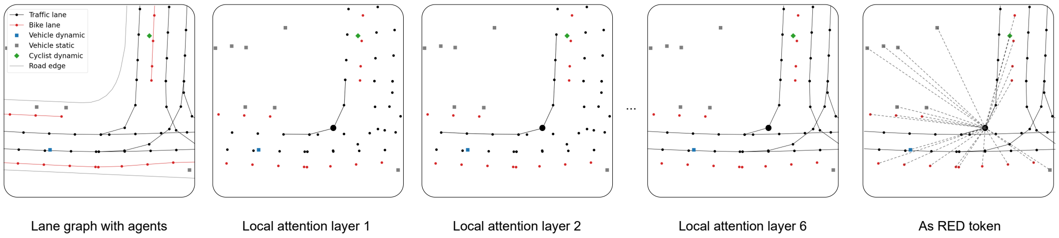


Figure 4: **Receptive field of a traffic lane token.** It expands with the increasing number of layers, thereby enabling the token to cover related tokens within a larger surrounding area. Consequently, the road environment tokens initially form a disconnected graph, but as the number of layers increases, they gradually transform into a fully connected graph. Best viewed, zoomed in.

In addition to the road environment, we encode the past trajectory of the current agent with a standard transformer encoder (trajectory encoder in Figure 1). The trajectory encoder learns a semantic embedding per agent type, which is concatenated with a temporal encoding and the position relative to the current position of the agent. As temporal encoding, we use the number of time steps between the encoded time step and the time step at which the prediction starts. The concatenated embedding is then projected to the model dimension using a learned linear projection.

Afterwards, we fuse the embedding of the past trajectory with local and global (RED) road environment embeddings in two steps. First, we use regular cross-attention to fuse the past trajectory tokens with local road environment tokens. Second, we use a memory efficient implementation of cross-attention to add information from our global RED tokens. As shown in Figure 5, we concatenate both input sequences with learned fusion tokens ([Fusion]). We use the local fusion token as queries vector and concatenate it with our RED tokens to generate keys and values matrices for a standard attention module. Hence, the attention module computes: $\text{softmax}(\frac{QK^T}{\sqrt{d_K}}) \cdot V$, where Q, K, and V are query, key, and value matrices, and d_K is the dimension of key vectors. Then, we add the local fusion token to the attention output to generate a local-global fusion token. Figure 5 focuses on the fusion mechanism of the local fusion token with our RED tokens (local-global fusion), we proceed analogously when fusing the global fusion token with trajectory and local environment tokens (global-local fusion). In this case, we replace the local fusion token with the global fusion token and RED tokens with trajectory and local environment tokens during fusion. For both ways, we compute this cross-attention mechanism in a token-to-sequence manner. This reduces the computational complexity compared to regular sequence-to-sequence attention from $O(n^2)$ to $O(2n)$, where n is the sequence length. Finally, the output sequence is generated by concatenating the fused tokens with trajectory and local environment tokens.

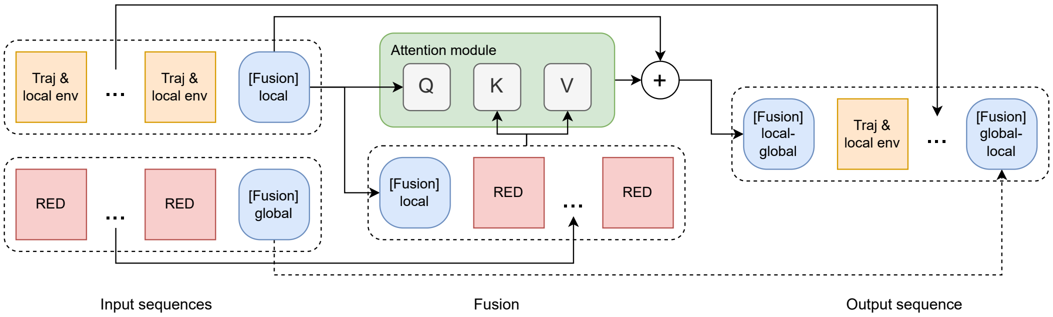


Figure 5: **Efficient cross-attention for feature fusion.** The output sequence contains features from the past trajectory, the local road environment, and global RED tokens.

After the fusion step, we use global average pooling over the feature dimension to reduce the input dimension for an MLP-based motion head. This MLP head regresses a configurable number of trajectory proposals and the corresponding confidences per agent. During inference, we use an agent-centric representation of the environment and predict trajectories marginally. In detail, each agent in the scene becomes the agent in the center when we predict his future trajectory.

4 Results

4.1 Comparing pre-training methods for motion prediction

In this set of experiments, we compare our representation learning approach with approaches from all other major families of self-supervised learning mechanisms (Balestriero et al., 2023). Specifically, we evaluate against contrastive learning with PreTraM (Xu et al., 2022), self-distillation using GraphDINO (Weis et al., 2023), and masked sequence modeling with Traj-MAE (Chen et al., 2023). Since self-supervised learning methods for motion prediction are only recently being developed, there are no common baseline models to compare such methods. Therefore, we use a modified version of our proposed model as baseline. Learning global environment features represented as RED tokens is our contribution, hence we remove the parallel decoder in the road environment encoder for the baseline version (see Figure 1). To allow a fair comparison, we increase the token dimension for the baseline from 128 to 192 to give both models a similar capacity (RedMotion baseline 9.9M params vs. RedMotion 9.2M params).

Dataset. We use the official training and validation splits of the Waymo Open Motion dataset (Ettinger et al., 2021) version 1.0 as training and validation data. Correspondingly, 100% of the training split is 2.2M agent-centric training samples. Since pre-training is particularly useful when little annotated data is available, we use 100% of the training data for pre-training and fine-tune on only 12.5%. Validation and evaluation are performed on 200K samples. In addition to the road environment, the trajectories of all traffic agents from the last second are used as input during fine-tuning for motion prediction. The dataset is sampled with 10 Hz, accordingly 10 past time points are used as input.

Evaluation metrics. We use the L5Kit (Houston et al., 2021) to evaluate the multimodal trajectory predictions of our models. Following common practice (Houston et al., 2021; Ettinger et al., 2021), we use the final displacement error (FDE) and the average displacement error (ADE) to evaluate trajectory proposals. The ADE and FDE scores are evaluated in the oracle/minimum mode. Accordingly, the distance errors of the trajectory proposal with the lowest distance error are measured. minADE and minFDE metrics are computed at different prediction horizons of 3s and 5s and averaged. Additionally, we introduce the $\Delta\text{minADE}_{\text{rel}}$ and $\Delta\text{minFDE}_{\text{rel}}$ scores, which measure the relative change w.r.t. the baselines without pre-training. Accordingly, the $\Delta\text{minADE}_{\text{rel}}$ is computed with

$$\Delta\text{minADE}_{\text{rel}} = \frac{\text{minADE}_{\text{pre}} - \text{minADE}_{\text{base}}}{\text{minADE}_{\text{base}}} \cdot 100, \tag{1}$$

where $\text{minADE}_{\text{pre}}$ is the minADE score achieved with pre-training and $\text{minADE}_{\text{base}}$ without pre-training. We compute all metrics considering 6 trajectory proposals per agent.

Experimental setup. For PreTraM, GraphDINO, and our pre-training method, we use the same augmentations described in Section 3. For Traj-MAE pre-training, we mask 60% of the road environment tokens and train to reconstruct them. For all methods, we train the pre-training objectives using our local road environment tokens, so that the lane network and social context are included. For PreTraM, we evaluate two configurations, map contrastive learning (MCL) and trajectory-map contrastive learning (TMCL). For MCL, the similarity of augmented views of road environments is maximized. For TMCL, the similarity of embeddings from road environments and past agent trajectories of the same scene is maximized. For our method, we evaluate four configurations of redundancy reduction: with mean feature aggregation (mean-ag), with learned feature aggregation (learned-ag), with reconstruction (red-mae), and between environment and past trajectory embeddings (env-traj). Mean-ag refers to using the mean of the RED tokens as input to the projector in Figure 1. For learned-ag, we use an additional transformer encoder layer to reduce the dimension of RED tokens to 16 and concatenate them as input for the reduction projector. The red-mae configuration is inspired by masked sequence modeling and a form of redundancy reduction via reconstruction. In detail, we generate two views (X^A and X^B) of road environments, randomly mask 60% of their tokens, and reconstruct X^A from the masked version of X^B and vice versa. Since we reconstruct cross-wise, the similarity between embedding representations of the augmented views is maximized during pre-training. The env-traj configuration is inspired by TMCL and reduces the redundancy between embeddings of past agent trajectories and RED tokens. Therefore, this configuration is inherently cross-modal but requires annotations of past agent trajectories.

For pre-training and fine-tuning, we use AdamW (Loshchilov & Hutter, 2019) as the optimizer. The initial learning rate is set to 10^{-4} and reduced to 10^{-6} using a cosine annealing learning rate scheduler. Following Konev et al. (2022), we minimize the negative multivariate log-likelihood loss for fine-tuning on motion prediction.

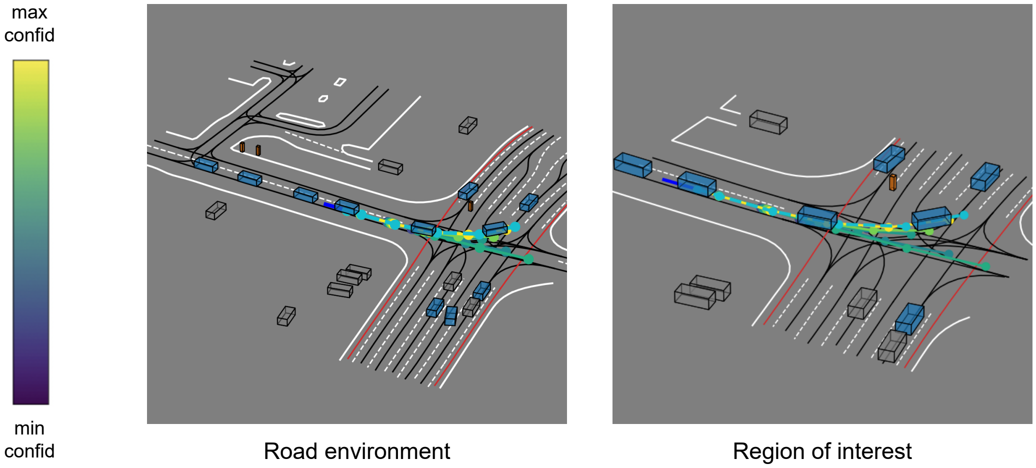
Results. Table 1 shows the results of this experiment. Overall, all pre-training methods improve the prediction accuracy in terms of minFDE and minADE. For our baseline model, our redundancy reduction reduction mechanism (b) (see Section 3) ranks second for the minFDE metric, marginally behind Traj-MAE and PreTraM in the TMCL configuration (only 0.3% worse). In terms of minADE, our mechanism ranks third behind Traj-MAE and PreTraM-TMCL. However, in this comparison our mechanism is much less complex and has less data requirements. Compared to Traj-MAE, no random masking and no complex reconstruction decoder (transformer model) are required. Compared to PreTraM-TMCL, no past agent trajectory data is required. When comparing to methods with similar requirements, our method outperforms PreTraM-MCL (-9.2% vs. - 15.8% in minFDE) and GraphDINO (-8.9% vs. -15.8% in minFDE). For PreTraM-MCL, the question arises, what is a good negative road environment. Road environments of agents close to each other (e.g., a group of pedestrians) are much more similar than, for example, images of different classes in ImageNet (e.g., cars and birds). During contrastive pre-training, all samples in a batch other than the current one are treated as negative examples. Therefore, the pre-training objective becomes to learn dissimilar embeddings for rather similar samples. For GraphDINO, we hypothesize that more hyperparameter tuning could further improve the performance (e.g., loss temperature or teacher weight update decay). When we combine our two redundancy reduction mechanisms a and b (lower group in Table 1), our RedMotion model outperforms all related methods by at least 4% in minFDE and achieves similar performance in the minADE metric. We hypothesize that the reason for the comparable worse performance in the minADE score is our trajectory decoding mechanism. Our MLP-based motion head regresses all points in a trajectory at once, thus individual points in a predicted trajectory are less dependent on each other than in recurrent decoding mechanisms. When fine-tuning, the error for the final trajectory point is likely to be higher than for earlier points and our model can learn to focus more on minimizing this loss term. Therefore, if pre-training improves the learning behavior of our model, this will effect the minFDE error more.

When comparing different configurations of our combined redundancy reduction objective, the env-traj configuration performs best and the mean-ag configuration performs worst. However, similar to PreTraM-TMCL our env-traj configurations learns to map corresponding environment embeddings

Model	Pre-training	Config	minFDE ↓	ΔminFDE$_{\text{rel}}$	minADE↓	ΔminADE$_{\text{rel}}$	#Params
RedMotion baseline	None		1.375		0.668		9.9M
	Traj-MAE (Chen et al., 2023)		1.154	**-16.1**	0.541	-19.0	9.9M
	PreTraM (Xu et al., 2022)	MCL	1.249	-9.2	0.579	-13.3	9.9M
	PreTraM (Xu et al., 2022)	TMCL*	1.154	**-16.1**	0.526	**-21.3**	9.9M
	GraphDINO (Weis et al., 2023)		1.252	-8.9	0.587	-12.1	9.9M
	RBT (ours)	mean-ag	1.158	-15.8	0.557	-16.6	9.9M
RedMotion	RBT (ours)	mean-ag	1.110	-19.3	0.568	-15.0	9.2M
	RBT (ours)	learned-ag	1.098	-20.1	0.555	-16.9	9.2M
	RBT (ours)	red-mae	1.093	-20.5	0.557	-16.6	9.2M
	RBT (ours)	env-traj*	1.055	**-23.3**	0.530	**-20.7**	9.2M

Table 1: **Comparing pre-training methods for motion prediction.** Best scores are bold, second best are underlined. *Denotes methods that require past trajectory annotations.

and past trajectory embeddings close to each other in a shared embedding space. Therefore, past trajectory data is required, which makes this objective less self-supervised and rather an improvement in data (utilization) efficiency. The learned-ag and red-mae configurations perform both better than the mean-ag configuration (1% improvement in minFDE and 2% improvement in minADE) and rather similar to each other. Since the learned-ag configuration has a lower computational complexity (no transformer-based reconstruction decoder but a simple MLP projector), we choose this pre-training configuration in the following. Figure 6 shows some qualitative results of our RedMotion model. Further qualitative results can be found in the Appendix.



Road environment

Region of interest

Figure 6: **Vehicle predictions.** Dynamic vehicles are marked as blue boxes, pedestrians as orange boxes, cyclists as green boxes, and static agents as grey boxes. Road markings are shown in white, traffic lane centerlines are black lines, and bike lane centerlines are red lines. The past trajectory of the ego agent is a dark blue line. The ground truth trajectory is cyan blue, the predicted trajectories are color-coded based on the associated confidence score using the viridis colormap on the left.

4.2 Comparing motion prediction models

We compare our RedMotion model with other recent models for motion prediction. We do not use ensembling for our model, therefore we compare our model with the best single model version of the other approaches. However, we include methods that employ ensembling for reference. As described in Section 4.1, our model with a basic MLP-based head (MLP-head) tends to focus more on later than on earlier trajectory points, which worsens the minADE score. Therefore, we additionally train a version of our model with a transformer decoder (tra-dec) as head, which is common amongst recent related methods (Girgis et al., 2022; Nayakanti et al., 2023; Zhang et al., 2023). In detail, we use a decoder with learned query tokens, which are transformed into trajectory proposals via attending to fused trajectory and road environment embeddings. For both variants, we use our redundancy reduction mechanisms to learn road environment embeddings. As aggregation method, we are using the learned-ag configuration from Section 4.1.

Dataset. We use 100% of the Waymo Open Motion training set for training to compare the performance of our model with that of other recent models. We perform evaluation on the validation and test splits.

Evaluation metrics. We use the same metrics for trajectories as in the previous set of experiments. However, this time, as in the Waymo Open Motion Challenge (Ettinger et al., 2021), the minFDE and minADE scores are computed for three prediction horizons of 3s, 5s, and 8s and then averaged. Additionally, we report the minFDE and minADE scores for the prediction horizon of 8s. For the test split, we also report this years challenge main metric, the Soft mAP score. Following the current challenge rules, we compute the metrics for 6 trajectory proposals per agent.

Results. Table 2 shows the performance of our model in comparison to other motion prediction models. On the validation split, our model with a basic MLP-based motion head achieves the lowest minFDE$_6$@8s score. Our model with a transformer decoder as motion head achieves the best scores for the minFDE$_6$, minADE$_6$, and minADE$_6$@8s metrics. This shows that a transformer decoder adds modeling capacity and prevents our model from focusing too much on the final trajectory points during training. On the test spilt, our model with a transformer decoder ranks second in terms of minADE$_6$ and minADE$_6$@8s behind HPTR. For the Soft mAP$_6$ score, our model ranks fourth behind MTR++, MTR, and HPTR. For reference, the best scores are achieved by methods that employ ensembling. However, in this work we focus on self-supervised representation learning, rather than on the advantages of ensembling.

Split	Method	Config	minFDE$_6$ ↓	minADE$_6$ ↓	minFDE$_6$@$8s$ ↓	minADE$_6$@$8s$ ↓	SoftmAP$_6$ ↑
Val	MotionCNN (Konev et al., 2022)	ResNet-18	1.640	0.815	-	-	-
	MotionCNN (Konev et al., 2022)	Xeption	1.496	0.738	-	-	-
	MultiPath++ (Varadarajan et al., 2022)		-	-	2.305	0.978	-
	Scene Transformer (Ngiam et al., 2022)	joint	1.804	0.837	3.113	1.353	-
	Scene Transformer (Ngiam et al., 2022)	marginal	1.220	0.613	2.070	0.970	-
	MTR (Shi et al., 2022)		1.225	0.605	-	-	-
	MTR++ (Shi et al., 2023)		1.199	0.591	-	-	-
	RedMotion (ours)	MLP-head	1.271	0.701	**1.952**	1.110	-
	RedMotion (ours)	tra-dec	**1.169**	**0.563**	2.044	**0.924**	-
Test	Scene Transformer (Ngiam et al., 2022)	joint	1.788	0.832	3.067	1.347	-
	Scene Transformer (Ngiam et al., 2022)	marginal	1.212	0.612	2.053	0.980	-
	MTR (Shi et al., 2022)		1.221	0.605	2.067	0.983	0.422
	MTR++ (Shi et al., 2023)		1.194	0.591	2.024	0.961	**0.433**
	HPTR (Zhang et al., 2023)		**1.139**	**0.557**	**1.954**	**0.910**	0.397
	RedMotion (ours)	tra-dec	1.199	0.577	2.084	0.948	0.391
	Wayformer* (Nayakanti et al., 2023)	multi-axis	1.128	0.545	1.942	0.892	0.434
	MTR* (Shi et al., 2022)	adv-ens	1.134	0.564	1.917	0.915	0.459

Table 2: **Comparing motion prediction models.** Best scores are bold, second best are underlined. *Denotes methods that employ ensembling.

5 Contribution and Future Work

In this work, we introduced a novel transformer model for motion prediction in the field of autonomous driving. Our proposed model incorporates two types of redundancy reduction, an architecture-induced reduction and a self-supervision objective for augmented views of road environments. Our evaluations indicate that this pre-training method can improve the accuracy of motion prediction and outperform contrastive learning, self-distillation, and autoencoding approaches. The corresponding RedMotion model attains results that are competitive with those of state-of-the-art methods for motion prediction. The method for creating RED tokens provides a universal approach to perform redundancy reduction, transforming a context of variable length into a fixed-size embedding. In future work, this approach can be applied to other context representations, extending to further multi-modal inputs beyond agent and environment data.

References

Caio Azevedo, Thomas Gilles, Stefano Sabatini, and Dzmitry Tsishkou. Exploiting map information for self-supervised learning in motion forecasting. *arXiv:2210.04672*, 2022.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A Cookbook of Self-Supervised Learning. *arXiv preprint arXiv:2304.12210*, 2023.

Horace Barlow. Redundancy reduction revisited. *Network: Computation in Neural Systems*, 12(3): 241, 2001.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv:2004.05150*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *European Conference on Computer Vision (ECCV)*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformer. In *International Conference on Computer Vision (ICCV)*, 2021.

Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction. In *Conference on Robot Learning (CoRL)*, 2020.

Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. Learning To Drive From a World on Rails. In *International Conference on Computer Vision (ICCV)*, 2021.

Hao Chen, Jiaze Wang, Kun Shao, Furui Liu, Jianye Hao, Chenyong Guan, Guangyong Chen, and Pheng-Ann Heng. Traj-MAE: Masked Autoencoders for Trajectory Prediction. *arXiv:2303.06697*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning (ICML)*, 2020.

Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. GoRela: Go Relative for Viewpoint-Invariant Motion Forecasting. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*, 2020.

Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset. In *International Conference on Computer Vision (ICCV)*, 2021.

Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. In *International Conference on Learning Representations (ICLR)*, 2022.

Daniel Grimm, Philip Schörner, Moritz Dreßler, J Zöllner, et al. Holistic Graph-based Motion Prediction. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Long Chen, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. In *Conference on Robot Learning (CoRL)*, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models. *arXiv:2001.08361*, 2020.

Stepan Konev, Kirill Brodt, and Artsiom Sanakoyeu. MotionCNN: A Strong Baseline for Motion Prediction in Autonomous Driving. *arXiv:2206.02163*, 2022.

Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Hengbo Ma, Yaofeng Sun, Jiachen Li, and Masayoshi Tomizuka. Multi-Agent Driving Behavior Prediction across Different Scenarios with Self-Supervised Domain Knowledge. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2021.

Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. TrackFormer: Multi-Object Tracking with Transformers. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Thomas Monninger, Julian Schmidt, Jan Rupprecht, David Raba, Julian Jordan, Daniel Frank, Steffen Staab, and Klaus Dietmayer. SCENE: Reasoning about Traffic Scenes using Heterogeneous Graph Neural Networks. *IEEE Robotics and Automation Letters*, 8(3):1531–1538, 2023.

Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion Forecasting via Simple & Efficient Attention Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David Weiss, Benjamin Sapp, Zhifeng Chen, and Jonathon Shlens. Scene Transformer: A unified architecture for predicting multiple agent trajectories. In *International Conference on Learning Representations (ICLR)*, 2022.

OpenAI. GPT-4 Technical Report. *arXiv:2303.08774*, 2023.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do Vision Transformers See Like Convolutional Neural Networks? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. MTR++: Multi-Agent Motion Prediction with Symmetric Scene Modeling and Guided Intention Querying. *arXiv preprint arXiv:2306.17770*, 2023.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *International Conference on Computer Vision (ICCV)*, 2017.

Simon Ulbrich, Till Menzel, Andreas Reschka, Fabian Schuldt, and Markus Maurer. Defining and substantiating the terms scene, situation, and scenario for automated driving. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2015.

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. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Xishun Wang, Tong Su, Fang Da, and Xiaodong Yang. ProphNet: Efficient Agent-Centric Motion Forecasting With Anchor-Informed Proposals. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Marissa A. Weis, Laura Pede, Timo Lüddecke, and Alexander S Ecker. Self-Supervised Graph Representation Learning for Neuronal Morphologies. *Transactions on Machine Learning Research*, 2023.

Chenfeng Xu, Tian Li, Chen Tang, Lingfeng Sun, Kurt Keutzer, Masayoshi Tomizuka, Alireza Fathi, and Wei Zhan. PreTraM: Self-Supervised Pre-training via Connecting Trajectory and Map Supplementary Material. In *European Conference on Computer Vision (ECCV)*, 2022.

Yi Yang, Qingwen Zhang, Thomas Gilles, Nazre Batool, and John Folkesson. RMP: A Random Mask Pretrain Framework for Motion Prediction. *arXiv preprint arXiv:2309.08989*, 2023.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning (ICML)*, 2021.

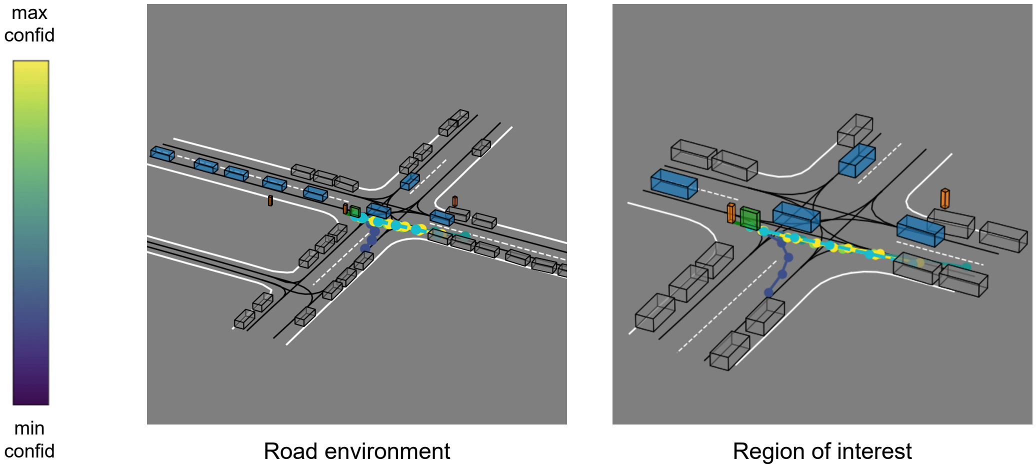
Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling Vision Transformers. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Zhejun Zhang, Alexander Liniger, Christos Sakaridis, Fisher Yu, and Luc Van Gool. Real-time motion prediction via heterogeneous polyline transformer with relative pose encoding. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *International Conference on Machine Learning (ICML)*, 2022.

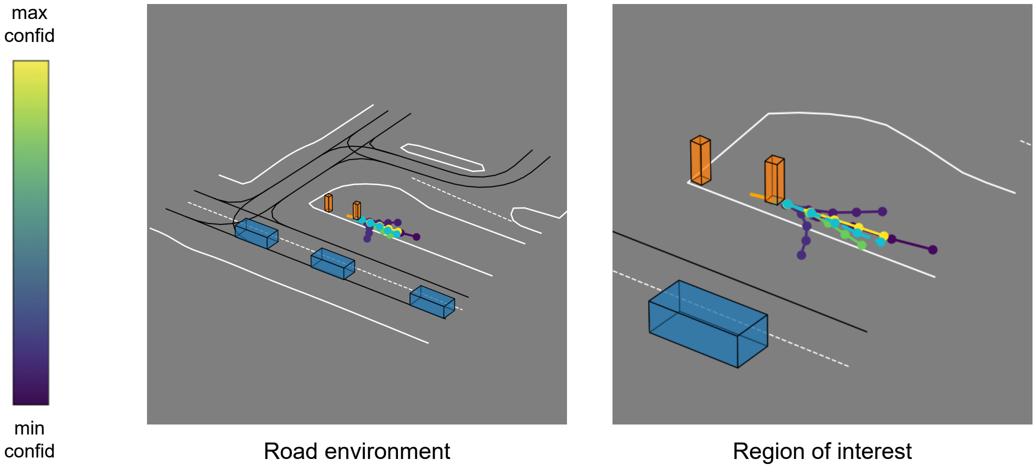
A Appendix



Road environment

Region of interest

Figure 7: **Cyclist predictions.** We use the same color-coding as in Figure 6. This plot shows an error case as the blueish trajectory pointing downwards enters the inbound lane.



Road environment

Region of interest

Figure 8: **Pedestrian predictions.** We use the same color-coding as in Figure 6.

More qualitative results can be generated using our anonymized Colab notebook.

B Limitations

We use a world on rails assumption (Chen et al., 2021) for our model and predict trajectories marginally. Specifically, we predict the trajectory of each agent in a scene individually only considering the current state but not the predicted trajectories of the surrounding agents. Therefore, we can not ensure consistency across all predictions in a scene as in joint prediction approaches (Ngiam et al., 2022; Cui et al., 2023; Zhang et al., 2023).

C Increasing the number of trajectory proposals

In this experiment, we increase the number of predicted trajectory proposals in our RedMotion model with a basic MLP-based head. Figure 9 shows the results on the validation split of the Waymo Open Motion dataset. Increasing the number of proposals from 6 to 16 decreases the minFDE score from 1.271 to 0.912 meters and the minFDE @8s score from 1.952 to 1.387 meters. This shows that our models performance scales well with the amount of trajectory proposals.

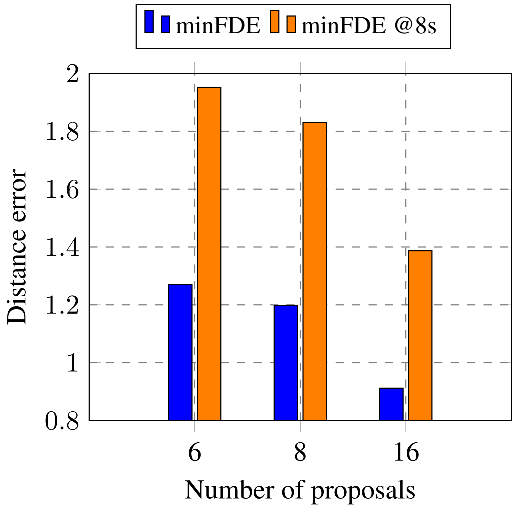


Figure 9: Increasing the number of predicted trajectory proposals