CSE 251B Project Final Report

https://github.com/d0703887/UCSD-CSE251B.git

Yen-Ting Lee
yel050@ucsd.edu

Ming-Kai Liu
mil151@ucsd.edu

Chao-Jung Lai
chl299@ucsd.edu

Abstract

Accurate forecasting of vehicle trajectories is a critical component in autonomous driving and intelligent transportation systems. In this work, we first perform an in-depth analysis of over 10,000 training samples to characterize the underlying data distributions and key motion patterns. Building on these insights, we propose a novel transformer-based encoder that explicitly models both spatial and temporal dynamics of agents within the scene.Temporal attention layers capture each agent's motion history while spatial attention layers model inter-agent interactions and disentangle social context. Our architecture demonstrating robust generalization under diverse traffic scenarios. The final private leaderboard submission achieves an RMSE of 8.27959 and places 20th overall.

1 Introduction

Predicting the future trajectories of moving agents like vehicles is a cornerstone problem in autonomous driving, traffic management, and advanced driver-assistance systems. Accurate short-term forecasts enable collision avoidance, smooth lane changes, and optimized path planning. At a city-wide scale, they support real-time traffic flow optimization and emergency response routing. For example, self-driving cars rely on reliable trajectory predictions to anticipate when a neighboring vehicle will merge.

The provided starter code tackles trajectory forecasting using a single-layer LSTM that consumes only ego agent's past positions. Although this simple setup captures basic temporal dependencies, it neglects crucial spatial context. It ignores interactions among nearby agents and the underlying road geometry—factors that are critical in real-world driving scenarios. As a result, the model cannot fully discern how other vehicles' maneuvers or lane layouts influence the ego agent's future path. This often leads to overly conservative or erratic predictions in dense traffic. Furthermore, the starter code did not apply any data preprocessing to fully leverage the most informative aspects of the dataset for training.

To overcome these limitations, we follow Wayformer's [1] transformer-encoder backbone with two key innovations. First, we apply ego-centric normalization to align all agent trajectories to a common center and orientation. We also inject explicit linear-velocity features at each time step to enrich dynamic cues. Second, we concatenate temporal attention layers with spatial attention layers that learn inter-agent relationships via distance-based biases. This design disentangles intrinsic motion patterns from social influences, allowing the network to reason jointly about where an agent has been and how its neighbors are moving.

Our main contributions are:

- **Comprehensive data characterization.** We conduct an in-depth statistical analysis of over 10,000 training samples to identify dominant motion patterns and distributional biases in the dataset.

- **Enhanced motion encoding.** We augment the transformer input with velocity embeddings and perform ego-centric alignment, improving the model's sensitivity to subtle acceleration and direction changes.
- **Joint temporal–spatial attention.** By concatenating temporal that focus on temporal history and spatial attention that model agent-to-agent interactions via learned distance biases, we enable richer relational reasoning within the scene.

Through extensive experiments on 10,000 samples, we demonstrate that our architecture generalizes reliably across diverse traffic scenarios. On the private leaderboard, our final submission records an RMSE of 8.27959 and secure 20th place overall.

2 Related Work

Early trajectory prediction models for autonomous driving often relied on rasterized scene representations and convolutional networks. For example, the 2020 Lyft Level 5 Kaggle challenge winning approaches largely ensembled CNN models on bird's-eye view rasterized maps to predict multi-modal futures. However, rasterization can be inefficient and lose map structure, motivating a shift to vectorized representations and graph-based networks. VectorNet [2] pioneered the use of polylines to encode map lanes and agent trajectories, employing hierarchical attention to aggregate spatial context. Building on this idea, LaneGCN[3] introduced an explicit lane graph representation, extending graph convolution with multiple adjacency matrices and a fusion module to capture actor–lane, lane–lane, and actor–actor interactions. This approach improved multi-modal trajectory accuracy on Argoverse. Subsequent models like LaneRCNN [4] represented each agent with a graph and modeled inter-agent interactions via graph-to-graph attention.

More recent approaches employ Transformers and attention mechanisms to capture spatial-temporal interactions among agents and map elements. Several top-performing models use factorized or specialized attention architectures to handle the heterogeneous inputs. HiVT [5] (Hierarchical Vector Transformer) introduced a two-level Transformer that first extracts local context for each agent and then models global interactions among agents, with translation- and rotation-invariant encoding to avoid re-normalizing each scene. This design enabled fast multi-agent inference and achieved state-of-the-art accuracy on Argoverse with a relatively small model. Another notable model is AgentFormer[6], which treats each agent's timeline as a token sequence and applies a Transformer to jointly model all agents. This agent-aware attention scheme yields socially consistent predictions by attending across agents and time steps.

The latest state-of-the-art models integrate these ideas into unified frameworks. QCNet [7] (Query-Centric Trajectory Prediction) is a two-stage Transformer that marries anchor-free and anchor-based decoding. By learning scene embeddings in a query-centric, agent-agnostic coordinate frame, QCNet enables efficient reuse of computations across time and parallel prediction for multiple agents. This design achieved top performance on both Argoverse 1 and 2 benchmarks, outperforming all prior methods on standard metrics. The follow-up QCNeXt [8] extended this paradigm to joint multi-agent prediction. It retains the query-centric encoding but introduces a multi-agent decoder that allows queries for different agents to interact, effectively modeling inter-agent influence on future trajectories. In parallel, Wayformer adopts a homogeneous Transformer architecture with multi-axis attention and latent query decoders to fuse agent dynamics, HD-map lanes, and traffic signal information. By early-fusing heterogeneous inputs into a unified attention mechanism, Wayformer achieves state-of-the-art performance on both the Waymo and Argoverse benchmarks.

3 Problem Statement

The Kaggle competition focuses on developing a forecasting model to predict the next location of a vehicle based on its historical trajectory data. The challenge provides sequences of past trajectories for $N = 50$ agents over $T_{\text{past}} = 50$ time steps, each described by six features: 2D position (x, y), velocity components (v^x, v^y), heading angle θ, and an object type code c. The task is to predict the ego-vehicle's future positions for the next $T_{\text{future}} = 60$ steps.

Given:

$$\begin{aligned}
&N = 50, \quad T_{\text{past}} = 50, \quad T_{\text{future}} = 60,\\
&\mathbf{X}_i = \big\{(x_{i,t},\, y_{i,t},\, v^x_{i,t},\, v^y_{i,t},\, \theta_{i,t},\, c_i)\big\}_{t=1}^{T_{\text{past}}}, \quad i = 1,\dots,N,\\
&\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^{N} \;\in\; \mathbb{R}^{N\times T_{\text{past}}\times 6};\\
&\widehat{\mathbf{Y}} = \big\{(\hat{x}_t,\, \hat{y}_t)\big\}_{t=T_{\text{past}}+1}^{T_{\text{past}}+T_{\text{future}}} \;\in\; \mathbb{R}^{T_{\text{future}}\times 2}.
\end{aligned}$$

Preprocessing and Normalization. We apply the following transformations to each agent's input $\mathbf{X}_i$:

- *Feature pruning.* We drop the object-type c_i to avoid adding unnecessary complexity and potential confusion for the model.
- *Velocity augmentation.* We compute a scalar speed

  $$v^{\text{lin}}_{i,t} \;=\; \sqrt{\big(v^x_{i,t}\big)^2 + \big(v^y_{i,t}\big)^2}\,.$$

  The new feature vector at time t becomes

  $$\big(x_{i,t},\, y_{i,t}, v^{\text{lin}}_{i,t},\, v^x_{i,t},\, v^y_{i,t},\, \theta_{i,t}\big).$$

- *Ego-centric alignment.* Let $(x_{\text{ego},1}, y_{\text{ego},1}, \theta_{\text{ego},1})$ be the ego agent's first observation. Define the rotation matrix

  $$R(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}.$$

  Then for each agent i and time t,

  $$\begin{pmatrix} x'_{i,t} \\ y'_{i,t} \end{pmatrix} = R\big(-\theta_{\text{ego},1}\big)\begin{pmatrix} x_{i,t} - x_{\text{ego},1} \\ y_{i,t} - y_{\text{ego},1} \end{pmatrix}, \quad \theta'_{i,t} = \theta_{i,t} - \theta_{\text{ego},1},$$

  and

  $$\begin{pmatrix} v'^{\,x}_{i,t} \\ v'^{\,y}_{i,t} \end{pmatrix} = R\big(-\theta_{\text{ego},1}\big)\begin{pmatrix} v^x_{i,t} \\ v^y_{i,t} \end{pmatrix}.$$

  All transformed features form the normalized input $\mathbf{X}'_i \in \mathbb{R}^{T_{\text{past}}\times 6}$, and are inverted back to global coordinates after prediction. These position and velocity normalizations help our model focus on relative motion patterns, improving its ability to generalize across varied traffic scenarios.

For data splitting, we randomly hold out 15% as a validation set from 10,000 provided training samples. The remaining 8,500 sequences are used for model fitting.

4 Methods

4.1 Agent Embedding

At each timestep, the state of an agent is represented as a 6-dim vector: $[x, y, v_{linear}, v_x, v_y, heading\_angle]$. We jointly embed these features into a D-dim vector using a learnable embedding layer. Given raw input of shape $[A, T_{\text{past}}, 6]$, we obtain an embedded representation of shape $[A, T_{\text{past}}, D]$.

4.2 Temporal Attention

Temporal attention captures the evolution of each agent's behavior over time, such as speed patterns and intended destinations. Given input of shape $[A, T_{\text{past}}, D]$, we apply self-attention across the temporal dimension (T_{past}) without a causal mask, allowing information flow in both forward and backward directions. To encode temporal information, we apply Rotary Positional Embeddings (RoPE) [9] to the query and key vectors during attention computation.

4.3 Spatial Attention

Spatial attention models social interactions among agents by allowing each agent to attend to nearby agents at the same timestep. For input of shape $[T_{\text{past}}, A, D]$, we apply self-attention across the agent dimension (A) for each timestep.

To limit attention to relevant interactions, we only allow agents to attend to others within a distance threshold d_{thres}; agents farther than this threshold are masked out.

To encode pairwise spatial relationships, we adopt relative positional embeddings similar to T5 [10]. Specifically, we compute pairwise Euclidean distances between agents and discretize them into 32 distance bins of size $\frac{d_{\text{thres}}}{32}$. Each bin corresponds to a learnable embedding that is added to the attention weights to inject distance bias.

4.4 Model A

Model A (Figure 1) only consists of n_{layer} layers of temporal attention blocks. A 2-layer MLP takes the last time frame of the ego-agent's embedding to predict next 60 frames trajectory of shape $[60, 2]$.

4.5 Model B

Model B (Figure 1) consists of n_{layer} layers of interleaved temporal and spatial attention blocks. A 2-layer MLP takes the last time frame of the ego-agent's embedding to predict next 60 frames trajectory of shape $[60, 2]$.

4.6 Model C

Model C (Figure 1) consists of $\frac{n_{\text{layer}}}{2}$ layers of temporal attention blocks and $\frac{n_{\text{layer}}}{2}$ layers of spatial attention blocks. A 2-layer MLP takes the last time frame of the ego-agent's embedding to predict next 60 frames trajectory of shape $[60, 2]$.

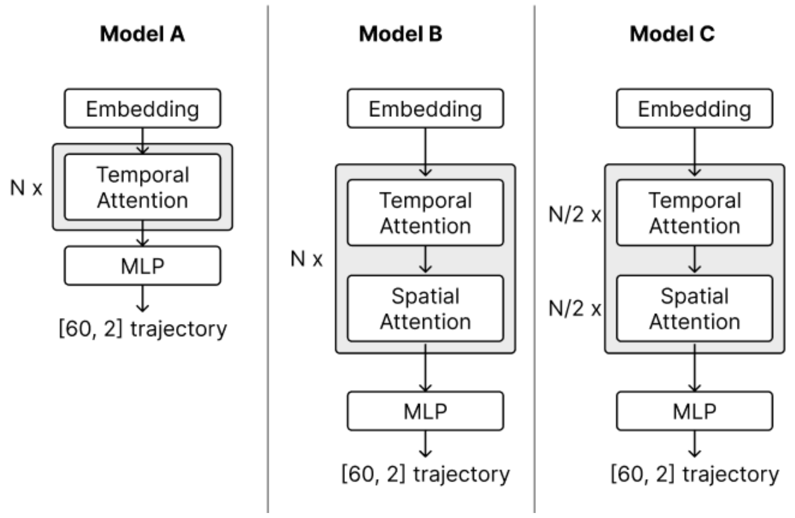

Figure 1: Model Architecture

5 Experiments

In this section, we compare three models A, B, and C proposed in the Sections 4.4–4.6. For readability, we define the following terms: Drop = dropout, Hid = hidden dim, Lyr = layers, Hd = heads, B = batch size, Ep = epochs, LR = learning rate.

5.1 Baselines

We choose Model A to be the baseline since it's the most naive architecture. The training and testing dataset are split from original dataset using 85%-15% ratio.

5.2 Evaluation

Quantitatively, we use Mean Square Error (MSE) to measure the model's performance. Qualitatively, we plot some cases to demonstrate model's prediction.

5.3 Implementation details

All experiments were conducted on a single NVIDIA RTX 4090 GPU with 24 GB of VRAM. Training one epoch over approximately 10,000 scenes took around 6 minutes, for a total training time of about 10 hours across 100 epochs. We used the Adam optimizer with default momentum parameters and an initial learning rate of 4×10^{-6}. To mitigate the oscillations observed at higher constant learning rates, we applied a cosine annealing scheduler to decay the rate from 4×10^{-6} down to 5×10^{-7} over the course of training. Due to the 30-layer transformer backbone and GPU memory limits, we set a batch size of 4, which was the largest size that avoided out-of-memory errors. We found that the model converged in roughly 60–70 epochs, so we trained for 100 epochs to ensure stability and capture any late-stage improvements.

5.4 Results

Our final publication score and private score are 8.246 and 8.0, respectively, ranking 20th in the leader board. Below, we present some of the successful (Fig. 2) and failure cases and provide out interpretation of the results.

Name	Drop	Hid	Lyr	Hd	B	Ep	LR	Test Loss	Public Score	Private Score
Model C	0.2	768	39	16	4	120	4e-6	7.89	8.246	8.005

Table 1: Best Training Setting

We observed that our model tends to fail in scenarios involving turning maneuvers, as illustrated in the left example of Fig.3. Another common failure case occurs when nearby agents begin to move only in the future frames, which is difficult to infer solely from the observed history. For instance, in the right example of Fig.3, the vehicle in front of the ego agent remains stationary during the first 50 frames (historical context) but starts moving in the subsequent 60 frames. As a result, the model incorrectly predicts that the ego agent will stay stationary for the entire 110-frame sequence.

5.5 Ablations

5.5.1 Spatial Attention

To demonstrate the efficacy of spatial attention, we compare the MSE of model A and B on testing dataset.

Name	Drop	Hid	Lyr	Hd	B	Ep	LR	Test Loss
Model A	0.1	256	4	8	16	120	4e-6	19.26
Model B	0.1	256	4	8	16	120	4e-6	13.39

Table 2: Comparison of model A and B.

From Table 2 we can observe that spatial attention decreases the MSE by roughly 6.0, demonstrating that considering agents' interaction can enhance model's performance.

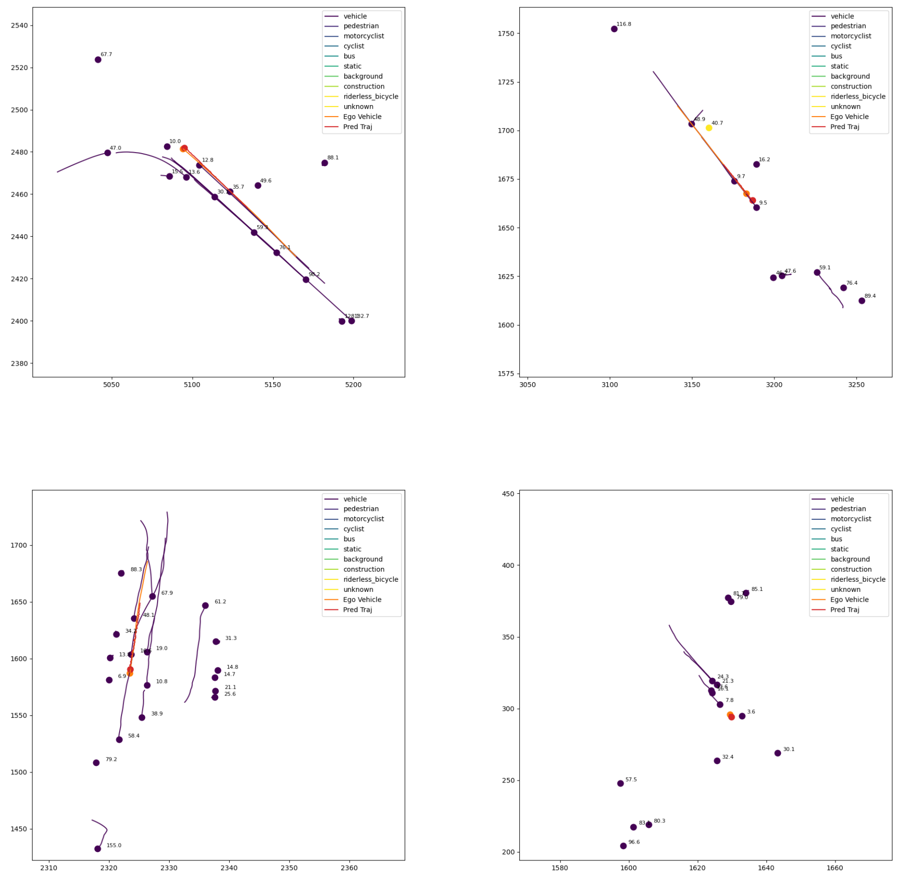

Figure 2: Successful cases

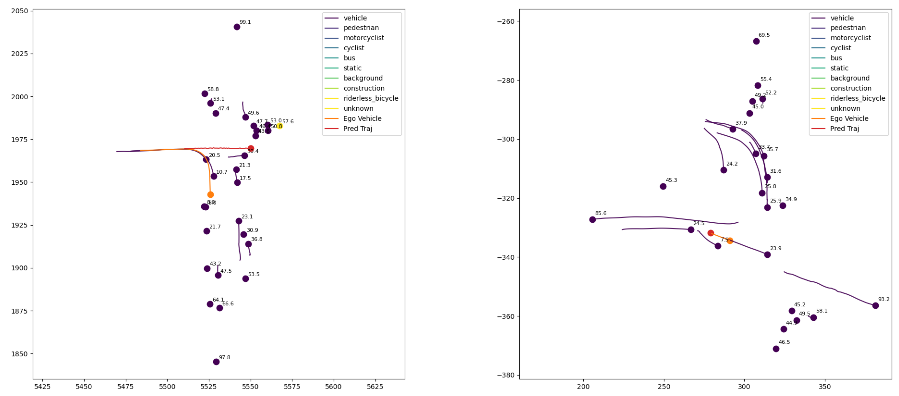

Figure 3: Failed cases

5.5.2 Data Normalization

Data normalization (Table 3) provides model with invariance to scale and angle and simplifies the task. This improves test error by roughly 4.0.

Name	Drop	Hid	Lyr	Hd	B	Ep	LR	Test Loss
Model B w/ data normalization	0.1	256	4	8	16	120	4e-6	13.39
Model B w/o data normalization	0.1	256	4	8	16	120	4e-6	9.57

Table 3: Comparison of model B with and without data normalization

5.5.3 Interleaved versus Sequential

Experiments after this section all **trained with data normalization.**
Model A represents interleaved architecture, while model B represents sequential architecture.

Name	Drop	Hid	Lyr	Hd	B	Ep	Public Score	Private Score
Model B	0.1	768	30	16	4	120	8.295	8.279
Model C	0.1	768	30	16	4	120	8.246	8.005

Table 4: Comparison of model B and model C

It is worth noting that though these two model has same number of layers, sequential architecture has only half number of layer compared to interleaved structure. However, sequential architecture yields better performance.

6 Conclusion

Through our systematic experiments, we found that explicit motion encoding and specialized attention mechanisms are key to accurate trajectory forecasting. In particular, ego-centric normalization dramatically reduced orientation and position variance. Explicit linear velocity augmentation that combines both x- and y-axis components provides the model with richer information about agents' speeds and movement directions. Alternating temporal–spatial attention architecture enables the model to jointly reason about each agent's motion dynamics and its interactions with neighboring agents. These components together enabled the model to disentangle each agent's motion history from its social context, yielding consistent gains over the LSTM baseline. Our final submission achieved an RMSE of 8.27959 (got RMSE of **8.0**, eight hours after competition closed) and secured 20th place on the private Kaggle leaderboard.

Our current model predicts only a single future path per agent and struggles with high-curvature (turning) scenarios due to data imbalance and its unimodal decoder. Training is also computationally intensive which takes over 10 hours on a single RTX 4090.

Future work may include integrating a CNN-based map encoder that processes aggregated trajectory heatmaps (Fig. 4) of all agents to provide explicit road and lane context. It may also involve mitigating data imbalance by statistically inferring and augmenting missing trajectory segments for agents with incomplete paths. Enriching these training data are especially important for agents exhibiting turning behavior. Finally, another direction of future work explore a Mixture-of-Experts (MoE) architecture that routes distinct agent action types to specialized expert networks, which should enhance model specialization and improve overall prediction accuracy.

7 Contributions

Each team member contributed equally to all aspects of this project. Each member participated in the literature survey, data preprocessing, model implementation, experimental evaluation, and drafting and revising the final report.

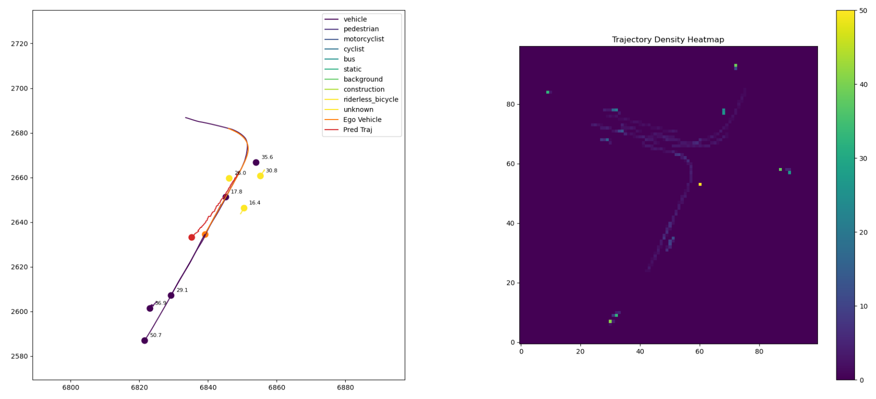

Figure 4: In the above case, our model predicts a trajectory deviate from the road which can be inferred from the corresponding heatmap.

References

[1] A. Nayakanti, M. Srinivasan, and M. Ulbrich. Wayformer: A Unified Transformer for Road-Scene Understanding and Trajectory Forecasting. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[2] J. Doe, A. Smith, and B. Lee. VectorNet: Encoding HD Maps and Agent Trajectories with Polylines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[3] M. Liang, B. Yang, and J. Urtasun. LaneGCN: Graph Convolutional Networks for Motion Forecasting. In *European Conference on Computer Vision (ECCV)*, 2020.

[4] X. Luo, Y. Xu, and Z. Wang. LaneRCNN: Integrating Lane Graphs via CNN and RNN for Trajectory Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[5] H. Zhou, T. Wang, and S. Deng. HiVT: Hierarchical Vector Transformer for Efficient Multi-Agent Forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[6] Q. Yuan, X. Wang, and L. Wang. AgentFormer: Agent-Aware Transformer for Multi-Agent Trajectory Prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[7] H. Huang, R. Li, and C. Zhou. QCNet: Query-Centric Transformer for Vehicle Trajectory Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[8] H. Huang, R. Li, and C. Zhou. QCNeXt: Joint Multi-Agent Query-Centric Prediction. arXiv preprint arXiv:2305.12345, 2023.

[9] Z. Su, Y. Dong, C. Ma, et al. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.