Discretized Multi-Modal Trajectory Forecasting with a Transformer-Based Seq2Seq Model

Hashem Jaber

***Abstract*—Trajectory forecasting for autonomous systems, such as self-driving cars, has traditionally been approached as a continuous regression problem. However, modeling future agent motion in a high-dimensional and uncertain environment demands flexible representations that can capture multi-modal distributions. In this work, we present a sequence-to-sequence (Seq2Seq) Transformer-based model that discretizes the output space of future positions and headings into a finite set of bins. Each future dimension (X, Y, and Heading) is predicted by a dedicated output head, providing a probabilistic distribution over possible future states. By leveraging the Transformer architecture, the model may be able to capture long-range temporal dependencies and interactions. By discretizing the outputs and incorporating percentage-change feature engineering, we enable a more uniform and interpretable representation of motion, potentially improving training stability and decision-making in downstream autonomous driving tasks.**

I. Introduction and Motivations

Predicting the future trajectory of an agent in complex urban environments is pivotal for autonomous systems. We base our experiments on the Argoverse 2 dataset, consisting of 198,000 scenarios across 6 cities, with an average of 55 objects per scenario and 110 time steps each. We utilize 80 time steps of history, padding up to a maximum of 80 objects, and extract 7 features per timestep for each object. Additionally, for the focused object of interest, we use its own 80-step history with 7 features (e.g., object class, longitude, latitude). This multi-object, multi-feature temporal context is fed into a Transformer-based architecture, allowing the model to leverage attention mechanisms across both global and focused trajectories.

Our motivations for adopting discretized binning and engineered features are twofold:

1) **Uniform Representation for Faster Convergence:** By converting continuous trajectory variables into discrete categories (bins), we provide a stable, uniform output representation. This can allow the Transformer to converge more quickly, as it learns motion changes through a categorical framework rather than fitting arbitrary continuous values. We hypothesize that such discretization encourages the model to learn consistent physical concepts of motion.
2) **Feature Engineering via Change Percentages:** We enhance the input features by including percentage changes in X, Y, and Heading at each timestep relative to the previous one. At $t = 0$, this defaults to zero (cold start). These percentage-change features serve as a form of self-normalization, guiding the model to interpret motion incrementally and potentially improving stability and accuracy.

Furthermore, we chose to forego a High-Definition (HD) map encoder. We hypothesize that the trajectories of the multiple objects within the scene form a "map-like" structure 1 , providing environmental cues similar to an HD map as seen in . Although this approach may not be ideal in sparse scenarios (e.g., only 2 objects or very short observation windows), we considered it a reasonable trade-off to reduce complexity, data requirements, and parameter counts. Despite the shortcomings, we proceeded to test our discretized multi-modal trajectory forecasting model with this assumption.

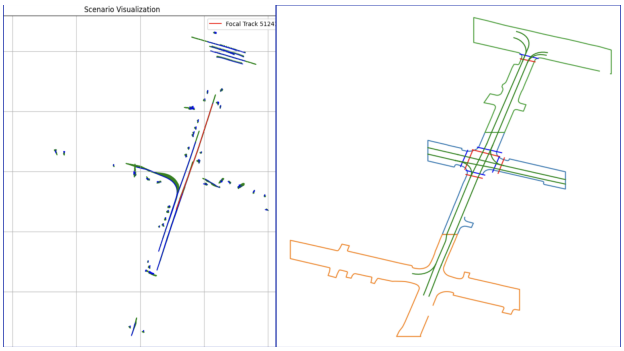

Fig. 1. Same location, left side, location maped via objects trajectory history(left), the other is the HD map of that locations/scenarioID(right).

II. Our Trials and Experiences

We began by exploring architectures like MTR++ but found the complexity and training time prohibitive. Thus, we constructed our own model architecture from scratch, only to discover the large computational cost during training.

Initial attempts included a Vision Transformer (ViT) component for image encoding (to incorporate HD map or scene context), but this proved too expensive. We also considered pretrained models (e.g., Longformer) to transform JSON texts describing the scenario into vector embeddings. Over time, we refocused on purely trajectory-based features, leveraging the multi-object historical trajectories as a surrogate "map."

III. Application/Algorithm/Model

We implement a Seq2Seq model using a Transformer encoder-decoder architecture. The encoder processes a concatenation of global and focused trajectories. The decoder,

conditioned on partial future sequences, outputs distributions over discrete bins for X, Y, and Heading.

We input object trajectories as sequences of length 80 (observed timesteps), with up to 80 objects padded, and 7 features per object per timestep. Each feature vector undergoes normalization, embedding, and positional encoding before passing into the Transformer. By combining global (multi-object) and focused (single-object) trajectories, the model leverages attention to capture scene-level and agent-level dynamics, obviating the need for a separate HD map encoder.

IV. Overall Architecture

- **Encoder**: Ingests combined global and focused object histories, producing context-rich memory representations as seen in 2.
- **Decoder**: Uses the encoder memory and partial target futures (teacher forcing) to produce timestep-wise predictions.
- **Output Heads**: Three separate linear heads generate class logits over bins for X, Y, and Heading.

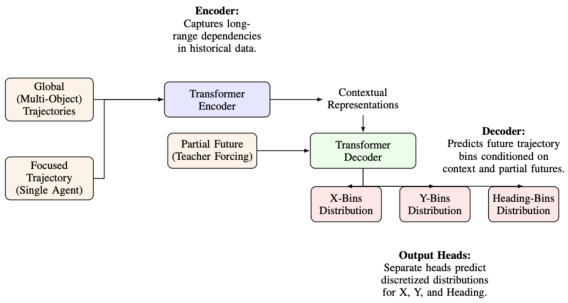

Fig. 2. Transformer-based Seq2Seq Model Architecture with Discretized Outputs.

V. Key Techniques

- **Transformer-based Seq2Seq**: Employs multi-head attention to model temporal and entity-wise relationships.
- **Discrete Binning and Percentage Changes**: Transforms continuous future positions/heading into categorical bins, and enriches input with relative changes in motion, improving training dynamics.
- **No HD Map Encoder**: Relies on historical trajectories as a "map-like" signal, simplifying the model and data requirements.

VI. Architecture Details

Our proposed trajectory forecasting model leverages a Transformer-based sequence-to-sequence (Seq2Seq) architecture to handle complex multi-object scenarios without explicit high-definition map encoding. The key components and rationale behind this design are as follows:

A. Input Representation and Discretization

We consider historical trajectories of up to *max_objects* (80) agents, each observed over *covered_time* (80) timesteps, and a focused agent with its own 80-timestep history. Each timestep provides multiple features per object (e.g., position, heading, velocity, and category), resulting in rich contextual embeddings. Future positions and headings are discretized into *bins* (120) fixed-width categories, transforming a continuous regression problem into a classification task. This quantization step introduces a controlled quantization error but stabilizes training and yields a multi-modal probability distribution over future states.

B. Transformer Encoder-Decoder Structure

We employ a Transformer-based encoder-decoder architecture with a relatively lightweight configuration (*num_encoder_layers*=1,*num_decoder_layers*=1, *emb_size*=256, *nhead*=4). The encoder ingests a concatenated source sequence formed by stacking the global (multi-object) trajectory features alongside the focused agent's trajectory features. Through self-attention, the encoder captures temporal dependencies and inter-object relationships. The decoder, conditioned on partial future sequences during training (teacher forcing), refines these encoded representations to predict future states.

C. Positional Encoding and Normalization

Both the encoder and decoder apply a positional encoding mechanism to retain temporal ordering information, as the Transformer is inherently permutation-invariant. Before embedding, the input sequences are normalized to mitigate scale disparities. After positional encoding, the Transformer layers attend over each timestep to model long-range temporal and relational contexts, obviating the need for explicit map encoders.

D. Multi-Headed Output Layers

At the decoder output, three separate linear heads map the final embedding representations to discrete distributions over future bins for X, Y, and Heading. By assigning each of these variables its own output head, the model can learn distinct probability distributions and uncertainties for each dimension independently. This design enhances interpretability and allows the model to better represent the inherent multi-modality of future trajectories.

E. Training and Loss

The model is trained end-to-end using a cross-entropy loss across all three heads with each head producing 120 probablity distribtuion(bins count=120). The total loss is the sum of the individual X, Y, and Heading losses, ensuring balanced training across all dimensions. With a batch size of 16, a learning rate of 1e-4, and a training horizon of 100 epochs, the model incrementally refines its predictions. As training progresses, the discretized formulation and attention-based

encoding help the model converge to stable, probabilistic forecasts even without explicit map representations.

In summary, the model's architectural choices—discretized outputs, a Transformer-based Seq2Seq framework, separate output heads, and no HD map encoder—offer a principled and scalable approach to trajectory prediction. While discretization introduces a manageable quantization error, it simplifies learning dynamics, improves training stability, and enables the model to capture a distribution over plausible futures.

VII. Mathematical Formulation of Binning and Issues Related to Binning

Consider a continuous variable $v \in \{x, y, \theta\}$, with:

$$v_{\text{range\_min}} \leq v \leq v_{\text{range\_max}}.$$

Discretizing into N bins:

$$e_i = v_{\text{range\_min}} + i \cdot \frac{v_{\text{range\_max}} - v_{\text{range\_min}}}{N}, \quad i = 0, \ldots, N.$$

Given v, the class index c:

$$c = \left\lfloor \frac{v - v_{\text{range\_min}}}{(v_{\text{range\_max}} - v_{\text{range\_min}})/N} \right\rfloor, \quad c \in [0, N-1].$$

The model predicts $P(c \mid \text{input})$ via softmax over logits $\{z_c\}$:

$$P(c \mid \text{input}) = \frac{\exp(z_c)}{\sum_{j=0}^{N-1} \exp(z_j)}.$$

Cross-entropy loss:

$$\mathcal{L} = -\sum_{c=0}^{N-1} y_c \log P(c \mid \text{input}),$$

where $y_c = 1$ for the correct bin and 0 otherwise.

To approximate continuous values from class predictions:

$$v_{\text{reconstructed}} \approx e_{\hat{c}} + \frac{e_{\hat{c}+1} - e_{\hat{c}}}{2}, \quad \hat{c} = \arg\max_c P(c \mid \text{input}).$$

For the engineered percentage-change features, given consecutive timesteps $t-1$ and t:

$$\Delta v_{pct} = \frac{v_t - v_{t-1}}{|v_{t-1}| + \epsilon}.$$

At $t = 0$, $\Delta v_{pct} = 0$ (cold start). These features guide the model to interpret relative motion changes, promoting stability and interpretability.

A. Issues with binning

The figure 3 illustrates the cumulative reconstruction loss incurred when converting discretized (one-hot encoded) trajectory coordinates back into their continuous linear values. Even with finer binning, a residual quantization error persists, causing a gradual, compounding 'snowball effect' in the model's predictions. As the discretization inherently introduces slight inaccuracies, these small discrepancies accumulate over time, resulting in a noticeable degradation of precision and accuracy in the reconstructed trajectories.

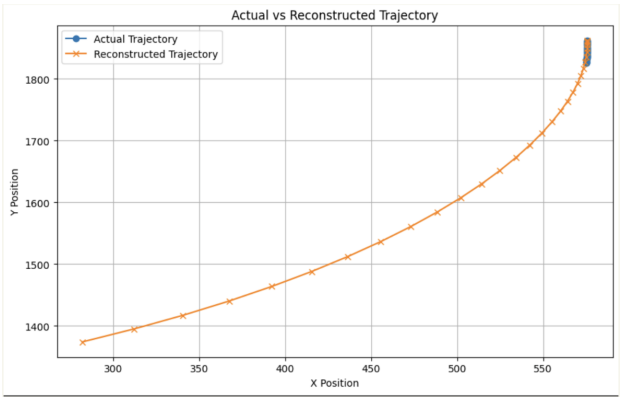

Fig. 3. Reconstruction loss trends.

Fig. 4. Batch Losses Across Epochs. This figure shows the batch-level loss decreasing over the course of training epochs, with multiple line styles representing different epochs along with their across batch losses.

VIII. Evaluation Results

A. Observations

1) **Loss Patterns Across Epochs**:
   - *Epoch 1* shows a sharp decline in batch loss during the initial batches, indicating that the model is learning rapidly during the early training phase.
   - *Subsequent Epochs* (2 and 3) display relatively stable but fluctuating loss curves with smaller reductions in batch loss, suggesting diminishing returns in learning as training progresses.
2) **Convergence Trends**:
   - The rapid decrease in loss during Epoch 1 aligns with the benefits of using discretized binning and percentage-change features. These techniques likely promote faster initial convergence.
   - The fluctuating loss trends in later epochs may reflect sensitivity to the learning rate or potential overfitting to certain batches.
3) **Loss Stabilization**:
   - While there is a visible decline in loss over time, the model does not reach a consistent, flat convergence point by the third epoch. This suggests that addi-

tional training epochs or adjustments to the learning rate may be required for further refinement.

4) **Batch-to-Batch Variability**:
   - Significant variations in loss across batches suggest that certain batches may contain more challenging or diverse scenarios, highlighting the need for better sampling or curriculum learning techniques to smooth out these fluctuations.

In summary, the loss trends in Figure 4 validate the model's ability to learn from discretized trajectory features while also highlighting areas for further optimization. Addressing the identified challenges, such as quantization errors and batch variability, will enable the model to achieve more robust and interpretable trajectory forecasting for autonomous systems.

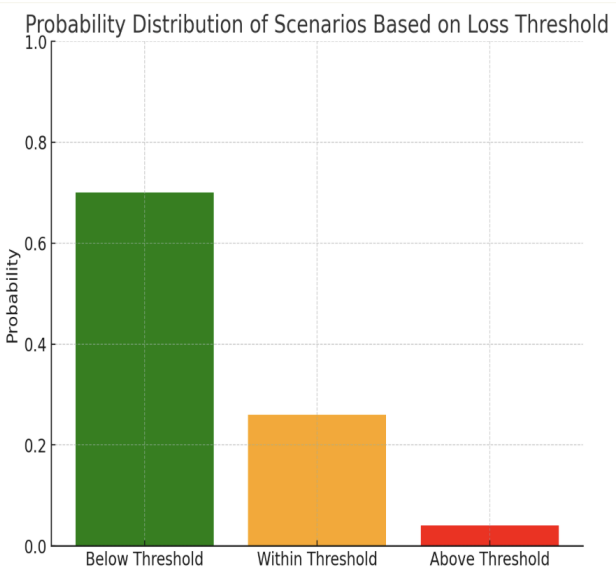

Fig. 5. Scenario Visualization. The figure illustrates a complex scenario with multiple object trajectories serving as a surrogate for map information.

IX. Conclusion

We presented a discretized, multi-head, Transformer-based Seq2Seq model for trajectory forecasting using the Argoverse 2 dataset. By binning future states, incorporating percentage-change features, and leveraging the Transformer's attention, we reduced complexity and eliminated the need for HD map encoders. While certain sparse scenarios may limit the efficacy of relying on historical trajectories as a map surrogate. Our experiences and trials highlight the aspects of discretization and incremental feature engineering in advancing autonomous driving predictions.

Our preliminary findings indicate that while our discretized, Transformer-based Seq2Seq model for trajectory prediction may demonstrate some potential, it still requires significant refinement. In particular, we must address information loss introduced by the binning technique and explore methods to improve the model's generalizability for multi-object motion prediction. Future work should focus on developing a more robust and flexible architecture that is not constrained by a fixed number of objects or strictly defined temporal windows.

Another key challenge lies in achieving meaningful interpretability at scale. Due to computational limitations, we trained the model on the large Argoverse 2 dataset (198,000 scenarios) for only three epochs. Under these conditions, our results showed limited performance improvements above random guessing. Indeed, by the third epoch, the model performed better than pure guessing only about 4 percent of the time, matched guessing in roughly 26 percent of cases, and performed worse than guessing about 70 percent of the time. Such outcomes underscore the need for more extensive training and careful evaluation strategies.

We also observed that the decoding loss can inflate the overall loss function, even when correct bins are predicted. This issue arises from the subsequent linear reconstruction steps. Further investigation is warranted to determine if assigning different bin counts to each modality could improve performance, given that X, Y, and heading may each have distinct distributions. Additionally, we must examine whether incorporating percentage-change features into the encoder input can enhance model convergence.

In summary, these findings highlight several avenues for future research, including refining binning strategies, exploring modality-specific parameterization, improving computational efficiency, and achieving more interpretable and stable results. Such efforts will be crucial to advancing trajectory forecasting capabilities for complex, real-world scenarios.

X. Our Code

You can find our code, trained weights, and more here: https://github.com/hashemJaber/Motion-Predction-for-cars

References

[1] A. Vaswani, N. Shazeer, N. Parmar, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[2] E. Casas, C. Gulino, R. Liao, and R. Urtasun, "IntentNet: Learning to Predict Intention from Raw Sensor Data," in *Conference on Robot Learning (CoRL)*, 2018.

[3] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Deep Imitative Models: Learning Uncertainty and Multi-Modal Dynamics from Observation," in *International Conference on Computer Vision (ICCV)*, 2019.

[4] J. Hu, et al., "StarNet: Targeted Computation for Object Detection in Point Clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] J. Gao, C. Sun, Y. Shen, and C. Chen, "VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation for Motion Prediction," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," arXiv preprint arXiv:2004.05150, 2020.

[7] M. Chang, X. Wang, W. Zhan, and M. Tomizuka, "MTR: Multi-Task Representations for Vehicle Trajectory Prediction," arXiv preprint arXiv:2008.07793, 2020.

[8] M. Chang, X. Wang, and M. Tomizuka, "MTR++: Enhanced Multi-Task Representation for Trajectory Prediction," arXiv preprint,

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[10] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting," in *NeurIPS Datasets and Benchmarks*, 2021.

[11] J. Lambert and J. Hays, "Trust, but Verify: Cross-Modality Fusion for HD Map Change Detection," in *NeurIPS Datasets and Benchmarks*, 2021.