LATENT VARIABLE SEQUENTIAL SET TRANSFORMERS FOR JOINT MULTI-AGENT MOTION PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robust multi-agent trajectory prediction is essential for the safe control of robotic systems. A major challenge is to efficiently learn a representation that approximates the true joint distribution of contextual, social, and temporal information to enable planning. We propose Latent Variable Sequential Set Transformers which are encoder-decoder architectures that generate scene-consistent multi-agent trajectories. We refer to these architectures as "AutoBots". The encoder is a stack of interleaved temporal and social multi-head self-attention modules which alternately perform equivariant processing across the temporal and social dimensions. The decoder can produce either the trajectory of one ego-agent or a distribution over the future trajectories for all agents in the scene. For the single-agent prediction case, our model achieves top results on the global nuScenes vehicle motion prediction leaderboard, and produces strong results on the Argoverse vehicle prediction challenge. In the multi-agent setting, we evaluate on TrajNet to showcase the model's socially-consistent predictions. We also demonstrate our model on general sequences of sets and provide illustrative experiments modelling the sequential structure of the multiple strokes that make up symbols in the Omniglot data. A distinguishing feature of our method is that all models are trainable on a single desktop GPU (1080 Ti) in under 48h.

1 INTRODUCTION

Many problems require processing complicated hierarchical compositions of sequences and sets. For example, multiple choice questions are commonly presented as a query (e.g. natural language) and an unordered set of options (Kembhavi et al., 2017; Richardson et al., 2013). Machine learning models that perform well in this setting should be insensitive to the order in which the options are presented. Similarly, for motion prediction tasks (Biktairov et al., 2020; Sadeghian et al., 2018) where the inputs are agent trajectories evolving over time and outputs are future agent trajectories, models should be insensitive to agent ordering.

In this work, we focus on the generative modelling of sequences and sets and demonstrate this method on a variety of different motion forecasting tasks. Suppose we have a set of $M \in \mathbb{N}$ sequences $\mathbb{X} = \{(x_1, \ldots, x_K)_1, \ldots, (x_1, \ldots, x_K)_M\}$ each with $K \in \mathbb{N}$ elements. Allowing also that $\mathbb{X}$ evolves across some time horizon T, we denote this sequence of sets as $\mathbf{X} = (\mathbb{X}_1, \ldots, \mathbb{X}_T)$.

This problem setting often requires the model to capture high-order interactions and diverse futures. Generally, solutions are built with auto-regressive sequence models which include t steps of context. In our approach here, to better model the multi-modal distribution over possible futures, we also allow to include a discrete latent variable Z, to create sequential models of sets of the form

$$P(\mathbb{X}_{t+1}|\mathbb{X}_t, \ldots, \mathbb{X}_1) = \sum_Z P(\mathbb{X}_{t+1}, Z|\mathbb{X}_t, \ldots, \mathbb{X}_1).$$

In this work, we propose this type of parameterized conditional mixture model over sequences of sets, and call this model a Latent Variable Sequential Set Transformer (affectionately referred to as "AutoBot"). We evaluate AutoBot on *nuScenes* (Caesar et al., 2020) and *Argoverse* (Chang et al., 2019), two autonomous driving trajectory prediction benchmarks, on *TrajNet* (Sadeghian et al., 2018), a pedestrian trajectory forecasting benchmark, and *Omniglot* (Lake et al., 2015), a dataset of hand-drawn characters that we use for predicting strokes. All of these provide empirical validation of

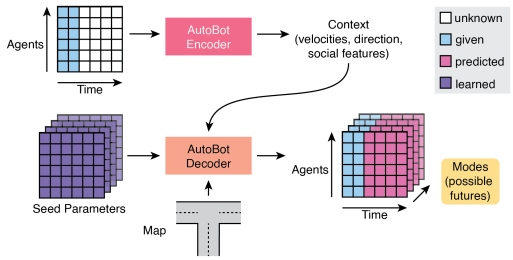

Figure 1: **Overview:** Our model generates multiple possible futures for agents based on several input timesteps. The input trajectories are encoded into a context tensor that captures their respective behavior and interaction with surrounding agents. While Transformer-based architectures are often autoregressive, we decode all future steps at once. This is accomplished via learnable seed parameters. Our decoder transforms these, together with the map and the context into the possible future trajectories that we call "modes".

our model's ability to perform robust multi-modal prediction. Specifically, we make the following contributions:

- We propose a novel approach for modelling sequences of set structured continuous variables, and extend this to a latent-variable formulation to capture multi-modal distributions.
- We provide a theoretical analysis highlighting our model's permutation sensitivity with respect to different components of the input data structures.
- We validate our method with strong empirical results on diverse tasks, including the large-scale *nuScenes* and *Argoverse* datasets for autonomous driving (Caesar et al., 2020; Chang et al., 2019), the multi-lingual *Omniglot* dataset of handwritten characters (Lake et al., 2015), and the *TrajNet* pedestrian trajectory prediction task (Sadeghian et al., 2018).

With this submission, we included an implementation of our method in a Jupyter notebook for easy reproduction and to demonstrate that the architecture is performant enough to train on a desktop GPU.

2 BACKGROUND

We now review several components of the Transformer (Vaswani et al., 2017) and Set Transformer(Lee et al., 2019) architectures, mostly following the notation found in their manuscripts. For additional background, see their works and Appendix A.

Multi-Head Self Attention (MHSA) can be thought of as an information retrieval system, where a query is executed against a key-value database, returning values where the key matches the query best. While (Vaswani et al., 2017) defines MHSA on three tensors, for convenience, we input a single set-valued argument. Internally, MHSA then performs intra-set attention by casting the input set $\mathbb{X}$ to query, key and value matrices and adding a residual connection, $\text{MHSA}(\mathbb{X}) = \mathbf{X} + \text{MHSA}(\mathbf{X}, \mathbf{X}, \mathbf{X})$.

Multi-Head Attention Blocks (MAB) resemble the encoder proposed in Vaswani et al. (2017), but lack the position encoding and dropout (Lee et al., 2019). Specifically, they consist of the MHSA operation described in Eq. 8 (appendix) followed by a row-wise feed-forward neural network (rFFN), with residual connections and layer normalization (LN) (Ba et al., 2016) after each block. Given an input set $\mathbb{X}$ containing n_x elements of dimension d, and some conditioning variables $\mathbb{C}$ containing n_c elements of dimension d, the MAB can be described by the forward computation,

$$\text{MAB}(\mathbf{X}) = \text{LN}(\mathbf{H} + \text{rFFN}(\mathbf{H})), \quad \text{where} \quad \mathbf{H} = \text{LN}(\mathbf{X} + \text{MHSA}(\mathbf{X}, \mathbf{X}, \mathbf{X})). \tag{1}$$

Multi-head Attention Block Decoders (MABD) were also introduced in Vaswani et al. (2017), and were used to produce decoded sequences. Given input matrices $\mathbf{X}$ and $\mathbf{Y}$ representing sequences, the decoder block performs the following computations:

$$\begin{aligned} \text{MABD}(\mathbf{Y}, \mathbf{X}) &= \text{LN}(\mathbf{H} + \text{rFFN}(\mathbf{H})) \\ \text{where} \quad \mathbf{H} &= \text{LN}(\mathbf{H}' + \text{MHSA}(\mathbf{H}', \mathbf{X}, \mathbf{X})) \\ \text{and} \quad \mathbf{H}' &= \text{LN}(\text{MHSA}(\mathbf{Y})) \end{aligned} \tag{2}$$

Our model, described throughout the following section, makes extensive use of these functions.

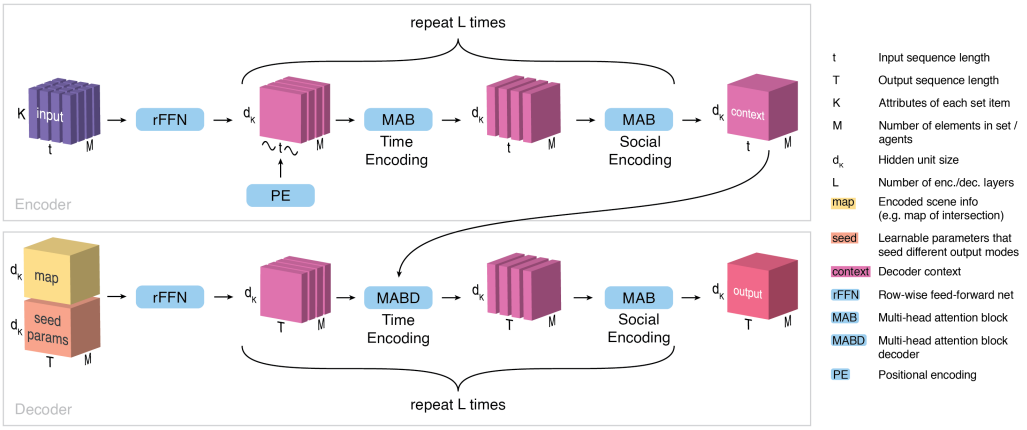

Figure 2: **Architecture Overview.** Our model takes as input a tensor of dimension K, M, t. A row-wise feed-forward network (rFFN) is applied to each row along the $t \times M$ plane transforming vectors of dimension K to d_K. After adding positional encoding (PE) to the t axis, the encoder passes the tensor through L repeated layers of multi-head attention blocks (MAB) that are applied to the time axis (time encoding) and the agent axis (social encoding) before outputting the context tensor. In the decoder, the encoded map and the learnable seed parameters tensor are concatenated and passed through an rFFN before being passed through L repeated layers of a multi-head attention block decoder (MABD) along the time axis (using the context from the encoder) followed by a MAB along the agent axis. This figure shows the process for predicting one possible future trajectory ("mode"). When multiple modes are predicted, each mode has its own learnable seed parameters and the decoder is rolled out once for each mode.

3 Latent Variable Sequential Set Transformers

Latent Variable Sequential Set Transformers is a class of encoder-decoder architectures that process sequences of sets (see Fig 2). In this section, we describe the encoding and decoding procedures, the latent variable mechanism, and the training objective. Additionally, we provide proof of the permutation equivariance of our model in Appendix B, and implementation details in Appendix C as well as a discussion of the special case of AutoBot-Ego, which is similar to AutoBots but predicts a future sequence for only one element in a scene.

3.1 Encoder: Input Sequence Representation

AutoBot takes as an input a sequence of sets, $\mathbf{X}_{1:t} = (\mathbb{X}_1, \ldots, \mathbb{X}_t)$, which in the motion prediction setting may be viewed as the state of a scene evolving over t timesteps. Let us describe each set as having M elements (or agents) with K attributes (e.g. x/y position). To process the social and temporal information the encoder applies the following two transformations, repeating the procedure L times. Appendix B.2 provides a proof of equivariance with respect to the agent dimension for this encoder.

First, the AutoBots encoder injects temporal information into the sequence of sets using a sinusoidal positional encoding function PE(.) (Vaswani et al., 2017). For this step, we view the data as a collection of matrices, $\{\mathbf{X}_0, \ldots, \mathbf{X}_M\}$, that describe the temporal evolution of agents. The encoder processes temporal relationships between sets using a MAB, performing the following operation on each set where $m \in \{1, \ldots, M\}$:

$$\mathbf{S}_m = \text{MAB}(\text{rFFN}(\mathbf{X}_m))$$

where rFFN(.) is an embedding layer projecting each element in the set to the size of the hidden state d_K. Next we process temporal slices of $\mathbf{S}$, retrieving sets of agent states $\mathbb{S}_\tau$ at some timestep τ and is processed using $\mathbb{S}_\tau = \text{MAB}(\mathbb{S}_\tau)$. These two operations are repeated L times to produce a **context tensor** $\mathbf{C} \in \mathbb{R}^{(d_K, M, t)}$ summarizing the entire input scene.

3.2 DECODER: MULTIMODAL SEQUENCE GENERATION

The goal of the decoder is to generate temporally and socially consistent predictions in the context of multi-modal data distributions. To generate $c \in \mathbb{N}$ different predictions for the same input scene, the AutoBot decoder employs c tensors of learnable seed parameters $\mathbf{Z} \in \mathbb{R}^{(d_K, M, T)}$ where T is the prediction horizon. Additional contextual information (such as a rasterized image of the environment) is encoded using a convolutional neural network to produce a vector of features $\mathbf{m}_i$ where $i \in \{1, ..., c\}$. In order to provide the contextual information to all future timesteps and to all set elements, we copy this vector along the M and T dimensions, producing the tensor $\mathbf{M}_i \in \mathbb{R}^{(d_K, M, T)}$. Each tensor $\mathbf{Z}_i$ is concatenated with $\mathbf{M}_i$ along the d_K dimension, as shown on the left side of Figure 2 (bottom). This tensor is then processed using a rFFN(.) to produce the tensor $\mathbf{H} \in \mathbb{R}^{(d_K, M, T)}$.

We begin to decode by processing the time dimension, conditioning on the encoder's output $\mathbf{C}$, and the encoded latent variables and environmental information in $\mathbf{H}$. The decoder processes each of m agents in $\mathbf{H}$ separately with an MABD layer:

$$\mathbf{H}'_m = \text{MABD}(\mathbf{H}_m, \mathbf{C}_m)$$

where $\mathbf{H}'$ is the output tensor that encodes the future time evolution of each element in the set independently. In order to ensure that the future scene is socially consistent between set elements, we process each temporal slice of $\mathbf{H}'$, retrieving sets of agent states $\mathbb{H}'_\tau$ at some future timestep τ where each element $\boldsymbol{h}'_\tau \in \mathbb{H}_\tau$ is processed by a MAB layer. Essentially, the MAB layer here performs a per-timestep attention between all set elements, similar to the Set Transformer (Lee et al., 2019).

These two operations are repeated L times to produce a final output tensor for mode i. The decoding process is repeated c times with different learnable seed parameters Z_i and additional contextual information $\mathbf{m}_i$. The output of the decoder is a tensor $\mathbf{O} \in \mathbb{R}^{(d_K, M, T, c)}$ which can then be processed element-wise using a neural network $\phi(.)$ to produce the desired output representation. In several of our experiments, we generate trajectories in (x, y) space, and as such, ϕ produces the parameters of a bivariate Gaussian distribution.

3.3 TRAINING OBJECTIVE

Given a dataset $\mathcal{D} = \big\{(\mathbb{X}_1, \ldots, \mathbb{X}_T)\big\}_{i=1}^{N}$ consisting of N sequences, each having T sets, our goal is to maximize the likelihood of the future trajectory of all elements in the set given the input sequence of sets, i.e., $\max p(\mathbf{X}_{t+1:T}|\mathbf{X}_{1:t})$. In order to simplify notation, we will henceforth refer to the ego-agent's future trajectory by $\mathcal{Y} = \mathbf{X}_{t+1:T}$. As discussed above, AutoBot employs discrete latent variables, which allows us to compute the log-likelihood exactly. The gradient of our objective function can then be expressed as follows:

$$\begin{aligned} \nabla_\theta \mathcal{L}(\theta) &= \nabla_\theta \log p_\theta(\mathcal{Y}|\mathbf{X}_{1:t}) = \nabla_\theta \log \Big(\sum_Z p_\theta(\mathcal{Y}, Z|\mathbf{X}_{1:t}) \Big) \\ &= \sum_Z p_\theta(Z|\mathcal{Y}, \mathbf{X}_{1:t}) \nabla_\theta \log p_\theta(\mathcal{Y}, Z|\mathbf{X}_{1:t}) \end{aligned} \tag{3}$$

As previously discussed in Tang and Salakhutdinov (2019), computing the posterior likelihood $p_\theta(Z|\mathcal{Y}, \mathbf{X}_{1:t})$ is difficult in general and varies with θ. As discussed in Bishop (2006), one can introduce a distribution over the latent variables, $q(Z)$, that is convenient to compute. With this distribution, we can now decompose the log-likelihood as follows:

$$\log p_\theta(\mathcal{Y}|\mathbf{X}_{1:t}) = \sum_Z q(Z) \log \frac{p_\theta(\mathcal{Y}, Z|\mathbf{X}_{1:t})}{q(Z)} + D_{KL}(q(Z)||p_\theta(Z|\mathcal{Y}, \mathbf{X}_{1:t})), \tag{4}$$

where $D_{KL}(.||.)$ is the Kullback-Leibler divergence between the approximating posterior and the actual posterior. A natural choice for this approximating distribution is $q(Z) = p_{\theta_{old}}(Z|\mathcal{Y}, \mathbf{X}_{1:t})$, where θ_{old} corresponds to AutoBot's parameters before performing the parameter update. With this choice of $q(Z)$, the objective function can be re-written as

$$\begin{aligned} \mathcal{L}(\theta) &= Q(\theta, \theta_{old}) + \text{const.} \\ Q(\theta, \theta_{old}) &= \sum_Z p_{\theta_{old}}(Z|\mathcal{Y}, \mathbf{X}_{1:t}) \log p_\theta(\mathcal{Y}, Z|\mathbf{X}_{1:t}) \\ &= \sum_Z p_{\theta_{old}}(Z|\mathcal{Y}, \mathbf{X}_{1:t}) \big\{ \log p_\theta(\mathcal{Y}|Z, \mathbf{X}_{1:t}) + \log p_\theta(Z|\mathbf{X}_{1:t}) \big\}. \end{aligned} \tag{5}$$

where the computation of $p_\theta(Z|\mathbf{X}_{1:t})$ is described in Appendix C.2. Note that the posterior $p_{\theta_{old}}(Z|\mathcal{Y}, \mathbf{X}_{1:t})$ can be computed exactly in the model we explore here since AutoBot operates with discrete latent variables. Therefore, our final objective function becomes to maximize $Q(\theta, \theta_{old})$ and minimize $D_{KL}(p_{\theta_{old}}(Z|\mathcal{Y}, \mathbf{X}_{1:t})||p_\theta(Z|\mathbf{X}_{1:t}))$.

As mentioned, for our experiments we use a function ϕ to produce a bivariate Gaussian distribution for each agent at every timestep. We found that to ensure that each mode's output sequence does not have high variance, we introduce a mode entropy (ME) regularization term to penalize large entropy values of the output distributions,

$$L_{\text{ME}} = \lambda_e \max_Z \sum_{\tau=t+1}^{T} H(p_\theta(\mathbb{X}_\tau|Z, \mathbf{X}_{1:t})). \tag{6}$$

As we can see, this entropy regularizer penalizes only the mode with the maximum entropy. Our results in Section D.1 show the importance of this term in learning to cover the modes of the data.

4 EXPERIMENTS

We now demonstrate the capability of AutoBots to model sequences of set-structured data across several domains. In subsection 4.1 we show strong performance on the competitive motion-prediction dataset nuScenes (Caesar et al., 2020), a dataset recorded from a self-driving car, capturing other vehicles and the intersection geometry. In subsection 4.2, we demonstrate our model on the Argoverse autonomous driving challenge. In subsection 4.3, we show results on TrajNet (Sadeghian et al., 2018), a benchmark for trajectory prediction that captures pedestrians, bicycles, and skaters from an overhead drone. Finally in subsection 4.4, we show the that our model is a general method that can be used to generate characters conditioning across strokes, and verify that AutoBot-Ego is permutation-invariant across an input set. See Appendix D.1, for an illustrative toy experiment that highlights our model's ability to capture discrete latent modes.

4.1 NUSCENES, A REAL-WORLD DRIVING DATASET

In this section, we present AutoBot-Ego's results on the nuScenes dataset. The goal of this benchmark dataset is to predict the ego-agent's future trajectory (6 seconds at 2hz) given the past trajectory (up to 2 seconds at 2hz). Additionally, the benchmark provides access to all neighbouring agents' past trajectories and a birds-eye-view RGB image of the road network in the vicinity of the ego-agent. Figure 3 shows three example predictions produced by AutoBot-Ego trained with $K = 10$. We observe that the model learns to produce trajectories that are in agreement with the road network and covers most possible futures directions. We also note that for each direction, AutoBot-Ego assigns different modes to different speeds to more efficiently cover the possibilities.

We compare our results on this benchmark to the state-of-the-art in Table 1. **Min ADE (5)** and **(10)** are the average of pointwise L2 distances between the predicted trajectory and ground truth over 5 and 10 most likely predictions respectively. A prediction is classified as a *miss* if the maximum pointwise L2 distance between the prediction and ground truth is greater than 2 meters. **Miss Rate Top-5 (2m)** and **Top-10 (2m)** are the proportion of misses over all agents, where for each agent, 5 and 10 most likely predictions respectively are evaluated to check if they're misses. **Min FDE (1)** is the L2 distance between the final points of the prediction and ground truth of the most likely prediction averaged over all agents. **Off Road Rate** is the fraction of predicted trajectories that are not entirely contained in the drivable area of the map. Our model was trained on a single Nvidia GTX 1080 Ti for 3 hours.

AutoBot-Ego obtains the best overall Min ADE out of 5 predictions ($K = 5$). This demonstrates our model's exceptional ability to capture multi-modal distributions over future trajectories. In addition, AutoBot-Ego excelled in maintaining predictions on the road, showing that the model has effectively used its capacity to attend to map information. In order to generate ten predictions from five modes, at test time, we create combinations between the latent modes one-hot vector by summing one-hot vectors. When we employ a model with ten modes $K = 10$, we see that AutoBot-Ego achieves slightly better results than WIMP (Khandelwal et al., 2020) on the Min ADE (10) score, and outperforms it on the MinADE (5) metric. However, WIMP outperforms AutoBot-Ego on the Miss Rate metric, which we hypothesize is due to WIMP's better map representation using polyline

Model	Min ADE (5)	Min ADE (10)	Miss Rate Top-5 (2m)	Miss Rate Top-10 (2m)	Min FDE (1)	Off Road Rate
Noah_prediction	1.59	1.37	0.69	0.62	9.23	0.08
CXX	1.63	1.29	0.69	0.60	8.86	0.08
LISA(MHA_JAM)	1.81	1.24	0.59	0.46	8.57	0.07
Trajectron++	1.88	1.51	0.70	0.57	9.52	0.25
CoverNet	2.62	1.92	0.76	0.64	11.36	0.13
Physics Oracle	3.70	3.70	0.88	0.88	9.09	0.12
WIMP	1.84	1.11	**0.55**	**0.43**	8.49	0.04
GOHOME	1.42	1.15	0.57	0.47	**6.99**	0.04
AutoBot-Ego (K=10)	1.43	1.05	0.66	0.45	8.66	0.03
AutoBot-Ego (ensemble)	**1.37**	**1.03**	0.62	0.44	8.19	**0.02**

Table 1: **Quantitative Results on the nuScenes dataset**. Other methods: LISA (Messaoud et al., 2020); Trajectron++ (Salzmann et al., 2020); CoverNet (Phan-Minh et al., 2020); Physics Oracle (Caesar et al., 2020); WIMP (Khandelwal et al., 2020); GOHOME (Gilles et al., 2021)

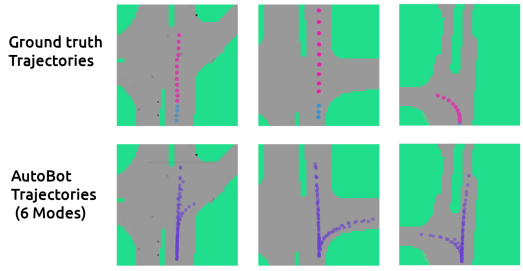

Figure 3: **NuScenes Trajectory Prediction**. Top row: birds-eye-view of road network with ground truth trajectory data where given trajectory is in cyan and held-out trajectory information is in pink. Bottom row: diverse trajectories generated by the different modes of AutoBot-Ego. The model generates trajectories that adhere to the road network and captures distinct possible futures.

attention, a key contribution of their approach. We leave it to future work to explore if AutoBots with polyline road network representations can yield improved results. Note, that in the appendix C, we present the computational complexity of AutoBot-Ego, which shows a *2x inference speedup* over WIMP. Finally, we note that like all other models, AutoBot-Ego struggles to predict the probability of the correct mode, which results in poor performance of the Min FDE (1).

4.2 ARGOVERSE RESULTS

Table 2 shows the results of AutoBot-Ego on the Argoverse dataset test set. While our method does not achieve top scores, it uses much less computation, less data augmentation and no ensembling compared to alternatives. Further, we compare quite well to the 2020 Argoverse competition winner, a model called "Jean". In addition, we note that the Argoverse challenge does not restrict the use of additional training data and that many of the top entries on the leaderboard have no publicly available papers or codebases. Many methods are therefore optimized for Argoverse by means of various data augmentations. For example, Scene Transformer (Ngiam et al., 2021) used a data augmentation scheme that includes agent dropout and scene rotation which we have not, and the current top method and winner of the 2021 challenge (QCraft) makes use of ensembling.

We believe that a fair comparison of the performance between neural network methods should ensure that each architecture is parameter-matched and the training is FLOP-matched. While the Argoverse leaderboard does not require entrants to report their parameter or FLOP count, we report available public statistics to enable comparison here. First, Scene Transformer (Ngiam et al., 2021) reports that their model trains in 3 days on TPUs and TPU-v3s have a peak performance of 420 TeraFLOPS, which means their model could have consumed up to 3 days $\times$ 24 hours $\times$ 420 TFLOPS, or 108,000,000 TFLOPS. On the other hand, AutoBot-Ego was trained on a Nvidia 1080ti for 10 hours, consuming roughly 396,000 TFLOPs, or 0.4% as much. Jean's method (Mercat et al., 2020) consumes roughly 8,600,000 TFLOPs (training our method uses 5% as much compute). While the method currently topping the leaderboard has not revealed their model details, the fact that they used ensembles very likely requires significantly more compute than our method.

Model	**Min ADE (↓)**	**Min FDE (↓)**	**Miss Rate (↓)**	**DAC (↑)**
AutoBot-Ego (Valid Set)	0.73	1.10	0.12	-
AutoBot-Ego (Test Set)	0.89 (top-5)	1.41 (top-5)	0.16 (top-6)	0.9886 (top-3)
Jean (2020 winner)	0.97	1.42	0.13	0.9868
WIMP	0.90	1.42	0.17	0.9815
TNT	0.94	1.54	0.13	0.9889
LaneGCN	0.87	1.36	0.16	0.9800
TPCN	0.85	1.35	0.16	0.9884
mmTransformer	0.84	1.34	0.15	0.9842
GOHOME	0.94	1.45	**0.105**	0.9811
Scene Transformer	**0.80**	**1.23**	0.13	**0.9899**

Table 2: **Quantitative Results on the Argoverse dataset test set**. Other methods that had an associated paper: Jean (Mercat et al., 2020); WIMP (Khandelwal et al., 2020); TNT (Zhao et al., 2020a); LaneGCN (Liang et al., 2020); TPCN (Ye et al., 2021); mmTransformer (Liu et al., 2021); GOHOME (Gilles et al., 2021); Scene Transformer (Ngiam et al., 2021).

4.3 TRAJNET, A PEDESTRIAN MOTION PREDICTION DATASET

We further validate AutoBots on a pedestrian trajectory prediction dataset TrajNet (Sadeghian et al., 2018) where we are provided with the state of all agents for the past eight timesteps and are tasked with predicting the next 12 timesteps. In this experiment, we apply our general architecture (AutoBot) to a multi-agent scene forecast situation, and ablate various the social components in the decoder only, and decoder and encoder.

Model	**Number Of Collisions**	**Scene-level Min ADE(5)**	**Scene-level Min FDE(5)**
AutoBot-AntiSocial	474	0.571	1.02
AutoBot-Ego	466	0.602	1.07
AutoBot	**326**	**0.471**	**0.833**

Table 3: TrajNet ablation studies for a multi-agent forecasting scenario. We evaluate the impact of using the extra social data as a set and the actual multi-agent modelling on this context. We found that AutoBot is able to cause fewer collisions between agents compared to variants without the social attention.

In Table 3, we present three variants of the model. The first, AutoBot-AntiSocial corresponds to a model without the intra-set attention functions in the encoder and decoder (see Figure 2). The second variant, AutoBot-Ego, only sees the past social context and has no social attention during the trajectory generation process. The third model corresponds to the general architecture, AutoBot. All models were trained with $K = 5$ and we report scene-level error metrics as defined by (Casas et al., 2020). Using the social data with the multi agent modelling in the decoder greatly reduces the number of collisions between predicted agent futures. Furthermore, we can see that AutoBot significantly outperforms its ablated counterparts on generating scenes that match ground-truth, as observed by the reduced scene-level minADE and minFDE. We expect this theoretically since the ego-centric formulation makes the independence assumption that the scene's future evolution can be decomposed as the product of the individual agent's future motion in isolation, as remarked in (Casas et al., 2020). AutoBot does not make this assumption as we condition between agents at every timestep.

Figure 4 shows an example of multi-agent trajectory generation using the three different approaches for predicting the scene evolution. We present four of the five modes generated by the model for this example input scene. One can observe that the only model achieving scene-consistent predictions across all modes is AutoBot . Interestingly, we also observe that the different modes correspond to different directions of motion, highlighting the utility of using modes. In fact, the modes in AutoBot condition the entire scene future evolution, producing alternative realistic multi-agent futures.

4.4 OMNIGLOT, STROKE-BASED IMAGE GENERATION

To demonstrate the general efficacy of our approach to sequence generation, we demonstrate AutoBots on a character completion, a diverse alternative to automotive and pedestrian trajectory prediction.

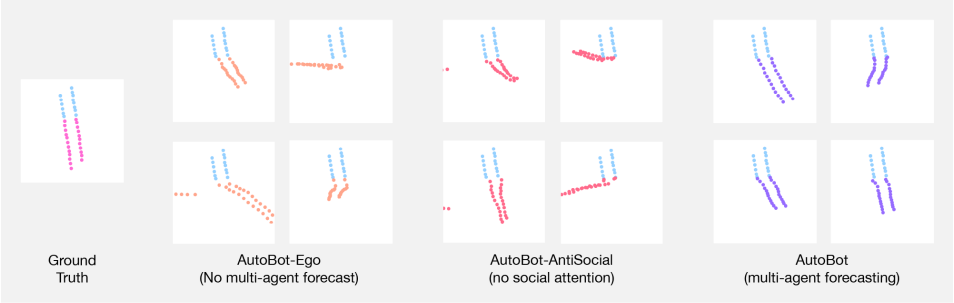

Figure 4: **TrajNet qualitative results.** In this example, we see two agents moving together in the past (cyan) and future (pink, left). We compare how different variants of AutoBots make predictions in this social situation. AutoBot-Ego and AutoBot-AntiSocial produce some modes containing collisions or unrealistic trajectories, while AutoBot, which has intra-set attention during decoding, achieves consistent trajectories.

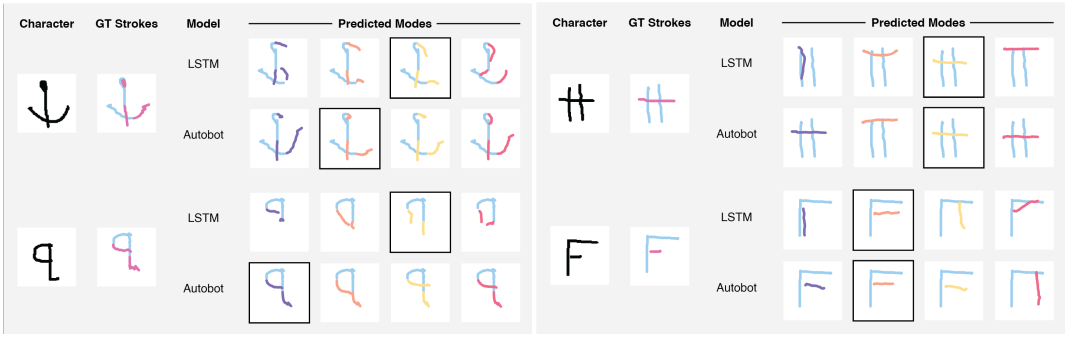

Figure 5: **Omniglot qualitative results.** *Left: stroke completion task.* We show examples of characters generated using AutoBot and an LSTM baseline. The first two columns show the ground-truth image of the character and the corresponding ground-truth strokes. In this task, the models are provided with the first half of all strokes (cyan) and are tasked with generating the other half (pink). The four other columns show the generated future strokes, one for each latent mode. We can see that AutoBot produces more consistent and realistic characters compared to the LSTM baseline. *right: character completion task.* We show two examples of characters completed using AutoBot-Ego and an LSTM baseline. The first two columns show the ground-truth image of the character as before. In this case, the models are provided with two complete strokes and are tasked with generating a new stroke to complete the character. We observe that AutoBot-Ego generates more realistic characters across all modes, given ambiguous input strokes.

We set up two different tasks on the Omniglot dataset (Lake et al., 2015): (a) **stroke-completion task** in which the model is presented with the first half of each stroke and has to complete all of them in parallel and (b) **character-completion task** in which the model is given several full strokes and has to draw the last stroke of the character. In task (a), the model is trained on the entire Omniglot dataset (training set), in which strokes are encoded as sequential sets of points, and we are comparing the performance of AutoBot qualitatively against that of an LSTM baseline with discrete latents (similar to the ones in AutoBot) on a held-out test set. With the first task, we illustrate that our model is flexible enough to learn to legibly complete characters from a variety of different languages. In task (b), we train AutoBot-Ego only on characters "F" and "H" to demonstrate that our discrete latent variable captures plausible character completions given the context of the other strokes, e.g. a character "F" can be completed as an "H" when the top stroke is missing. The results of these experiments can be found in Fig. 5. Implementation details and additional results can be found in Appendix section D.3.

5 Related Work

Processing set structured data. Models of set-valued data (i.e. unordered collections) should be permutation-invariant (see Appendix B for formal definitions), and be capable of processing sets with arbitrary cardinality. Canonical forms of feed forward neural networks and recurrent neural network models do not have these properties (Lee et al., 2019). One type of proposed solution is to integrate pooling mechanisms with feed forward networks to process set data (Zaheer et al., 2017; Su et al., 2015; Hartford et al., 2016). This is refereed as social pooling in the context of motion prediction

(Alahi et al., 2016; Deo and Trivedi, 2018; Lee et al., 2017). Attention-based models (Lee et al., 2019; Casas et al., 2020; Yuan et al., 2021; Ngiam et al., 2021; Zhao et al., 2020b) or graph-based models (Zhao et al., 2020a; Messaoud et al., 2020; Li et al., 2020; Gao et al., 2020) have performed well in some set processing tasks that require modelling high-order element interactions. In contrast, we provide a joint single pass attention-based approach for modelling more complex data-structures composed by sets **and** sequence, while also addressing how to generate diverse and heterogeneous data-structures. Although those properties exist in concurrent motion prediction papers (Tang and Salakhutdinov, 2019; Casas et al., 2020; Yuan et al., 2021; Ngiam et al., 2021), our proposed method is targeting a more general set of problems while still getting competitive results in motion prediction benchmarks at a fraction of other models' compute.

Diverse set generation. Generation of diverse samples using neural sequence models is an important part of our model and a well-studied problem. For example, in Bowman et al. (2015) a variational (CVAE) approach is used to construct models that condition on attributes for dialog generation. The CVAE approach has also been integrated with transformer networks (Lin et al., 2020; Wang and Wan, 2019). For motion prediction diverse response generation is fundamental (Lee et al., 2017; Deo and Trivedi, 2018), more recently, (Casas et al., 2020) use a variational mechanism where each agent has their own intentions encoded. The most similar strategy to ours was developed by (Tang and Salakhutdinov, 2019) and used in recent other works (Suo et al., 2021; Casas et al., 2020). This strategy allows conditional generation of sequences of sets using discrete latent variables. We built on this method by using one joint transformer conditioned on the latent variables and added a maximum entropy likelihood loss that can improve the variability of the generated outcomes.

Sequential state modelling. We are interested in data that can be better represented as a sequence. The main example of this is language modelling (Vaswani et al., 2017; Devlin et al., 2018), but also includes sound (Kong et al., 2020), video (Becker et al., 2018), and others (Kant et al., 2020; Yang et al., 2021). For motion prediction, prior work has largely focused on employing RNNs (LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Cho et al., 2014)) to model the input and/or output sequence (Alahi et al., 2016; Lee et al., 2017; Deo and Trivedi, 2018; Tang and Salakhutdinov, 2019; Messaoud et al., 2020; Casas et al., 2020; Li et al., 2020). With its recent success in natural language processing, Transformers (Vaswani et al., 2017) have been adopted by recent approaches for motion forecasting (Giuliari et al., 2021; Yu et al., 2020; Liu et al., 2021). In contrast to the aforementioned models, AutoBots produces representations that jointly encode both the time and social dimensions by treating this structure as a sequence of sets. This provides reuse of the model capacity and a lightweight approach. Concurrent to our method, Yuan et al. (2021), also jointly encodes social and time dimensions but flattens both of those axis into a single dimension and also restricts itself to only the motion prediction problem.

6 CONCLUSION

In this paper, we propose the Latent Variable Sequential Set Transformer to model time-evolution of sequential sets using discrete latent variables. We make use of the multi-head attention block to efficiently perform intra-set attention, to model the time-dependence between the input and output sequence, and to predict the prior probability distribution over the latent modes. We validate our AutoBot-Ego by achieving competitive results on the nuScenes benchmark for ego-centric motion forecasting in the domain of autonomous vehicles. We further demonstrate that AutoBot can model diverse sequences of sets that adhere to social conventions. We validate that since our model can attend to the hidden state of all agents during the generative process, it produces scene-consistent predictions on the TrajNet dataset when compared with ego-centric forecasting methods. Our approach is limited to operate in a setting where a perceptual pipeline is providing structured sets representing the other entities under consideration. If our approach is used in a robot or autonomous car, any errors or unknown entities not captured by sets used as input to the algorithm would not be modelled by our approach. Memory limitations may also limit the number of sets that can be processed at any time. In future work, we plan to extend this line of work to broader and more interpretable scene-consistency. Further, we would like to investigate the combination of a powerful sequence model like AutoBot with a reinforcement learning algorithm to create a data-efficient model-based RL agent for multi-agent settings.

REFERENCES

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Stefan Becker, Ronny Hug, Wolfgang Hubner, and Michael Arens. Red: A simple but effective baseline predictor for the trajnet benchmark. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

Yuriy Biktairov, Maxim Stebelev, Irina Rudenko, Oleh Shliazhko, and Boris Yangel. PRANK: motion prediction based on ranking. *CoRR*, abs/2010.12007, 2020. URL https://arxiv.org/abs/2010.12007.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer; 1st ed. 2006, 2006.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

H. Caesar, Varun Bankiti, A. Lang, Sourabh Vora, Venice Erin Liong, Q. Xu, A. Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. *arXiv preprint arXiv:2007.12036*, 2020.

Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Sławomir Bąk, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8740–8749, 2019.

Kyunghyun Cho, B. V. Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *ArXiv*, abs/1406.1078, 2014.

Nachiket Deo and M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1549–15498, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output forfuture motion estimation. *arXiv preprint arXiv:2109.01827*, 2021.

Francesco Giuliari, Irtiza Hasan, M. Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10335–10342, 2021.

Jason S Hartford, James R Wright, and Kevin Leyton-Brown. Deep learning for predicting human strategic behavior. In *NIPS*, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

Yash Kant, Dhruv Batra, Peter Anderson, Alexander Schwing, Devi Parikh, Jiasen Lu, and Harsh Agrawal. Spatially aware multimodal transformers for textvqa. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 715–732. Springer, 2020.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5376–5384, 2017. doi: 10.1109/CVPR.2017.571.

Siddhesh Khandelwal, William Qi, Jagjeet Singh, Andrew Hartnett, and Deva Ramanan. What-if motion prediction for autonomous driving, 2020.

Qiuqiang Kong, Yong Xu, Wenwu Wang, and Mark D Plumbley. Sound event detection of weakly labelled data with cnn-transformer and automatic threshold optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2450–2460, 2020.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

N. Lee, W. Choi, Paul Vernaza, C. Choy, P. Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2165–2174, 2017.

Lingyun Luke Li, Bin Yang, Ming Liang, Wenyuan Zeng, Mengye Ren, Sean Segal, and Raquel Urtasun. End-to-end contextual perception and prediction with interaction transformer. *arXiv preprint arXiv:2008.05927*, 2020.

Ming Liang, Binh Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. *ArXiv*, abs/2007.13732, 2020.

Zhaojiang Lin, Genta Indra Winata, Peng Xu, Zihan Liu, and Pascale Fung. Variational transformers for diverse response generation. *arXiv preprint arXiv:2003.12738*, 2020.

Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. *ArXiv*, abs/2103.11624, 2021.

Jean Mercat, Thomas Gilles, Nicole Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Gil. Multi-head attention for joint multi-modal vehicle motion forecasting. In *IEEE International Conference on Robotics and Automation*, 2020.

Kaouther Messaoud, Nachiket Deo, Mohan M Trivedi, and Fawzi Nashashibi. Multi-head attention with joint agent-map representation for trajectory prediction in autonomous driving. *arXiv*, pages arXiv–2005, 2020.

Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified multi-task model for behavior prediction and planning. *arXiv preprint arXiv:2106.08417*, 2021.

Tung Phan-Minh, E. Grigore, F. Boulton, Oscar Beijbom, and Eric M. Wolff. Covernet: Multimodal behavior prediction using trajectory sets. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14062–14071, 2020.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1020.

Amir Sadeghian, Vineet Kosaraju, Agrim Gupta, Silvio Savarese, and Alexandre Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *arXiv preprint*, 2018.

Tim Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *ArXiv*, abs/2001.03093, 2020.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021.

Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, pages 15424–15434, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

T. Wang and Xiaojun Wan. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *IJCAI*, 2019.

Zhengyuan Yang, Yijuan Lu, Jianfeng Wang, Xi Yin, Dinei Florencio, Lijuan Wang, Cha Zhang, Lei Zhang, and Jiebo Luo. Tap: Text-aware pre-training for text-vqa and text-caption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8751–8761, 2021.

Maosheng Ye, Tongyi Cao, and Qifeng Chen. Tpcn: Temporal point cloud networks for motion forecasting. *ArXiv*, abs/2103.03067, 2021.

Cunjun Yu, Xiao Ma, J. Ren, H. Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *ECCV*, 2020.

Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

Hang Zhao, Jiyang Gao, T. Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Y. Shen, Yuning Chai, C. Schmid, Congcong Li, and Dragomir Anguelov. Tnt: Target-driven trajectory prediction. *ArXiv*, abs/2008.08294, 2020a.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer, 2020b.

A BACKGROUND

We now review several components of the Transformer (Vaswani et al., 2017) and Set Transformer(Lee et al., 2019) architectures, mostly following the notation found in their manuscripts.

Multi-Head Self Attention (MHSA) as proposed in (Vaswani et al., 2017) is a function defined on n_q query vectors $\mathbf{Q} \in \mathbb{R}^{n_q \times d_q}$, and key-value pairs ($\mathbf{K} \in \mathbb{R}^{n_k \times d_q}$ and $\mathbf{V} \in \mathbb{R}^{n_k \times d_v}$). With a single attention head, the function resolves the queries $\mathbf{Q}$ by performing the computation:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\mathbf{Q}\mathbf{K}^\top)\mathbf{V}, \tag{7}$$

where d_q is the dimension of the query and key vectors and d_v is the dimension of the value vector. MHSA consists of h attention heads performing the operation shown in Eq. 7 with h linear projections of keys, queries, and values. The final output is a linear projection of the concatenated output of each attention head. These computations can be expressed as

$$\begin{aligned} \text{MHSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O, \quad \text{where} \\ \text{head}_i &= \text{Attn}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \end{aligned} \tag{8}$$

and where $\mathbf{W}_i^Q$, $\mathbf{W}_i^V$ and $\mathbf{W}_i^K$ along with the output projection matrix $\mathbf{W}^O$ are the learnable projection matrices of each attention head in the MHSA. Note that $\mathbf{W}_i^Q$, $\mathbf{W}_i^V$ and $\mathbf{W}_i^K$ project the initial set of inputs to smaller dimensionality. For example, $\mathbf{W}_i^Q \in \mathbb{R}^{d_q \times d_q^M}$ projects the original queries with dimension d_q to d_q^M which is typically chosen to be d_q/h. This operation has useful applications in learning a representation of an input set where all its elements interact with each other.

An interesting application of MHSA is to perform self-attention on an input set (Lee et al., 2019). Given an input set X, one can perform intra-set attention by using X as the queries, keys and values and having a residual connection, i.e., $\text{MHSA}(\mathbf{X}) = X + \text{MHSA}(\mathbf{X}, \mathbf{X}, \mathbf{X})$. This operation has useful applications in learning a representation of an input set where all its elements interact with each other.

Multi-head Attention Block Decoders (MABD) were also introduced in Vaswani et al. (2017), and were used to produce decoded sequences. Given an input matrix $\mathbf{X}$ and an output matrix $\mathbf{Y}$ representing sequences, the decoder block performs the following computations:

$$\begin{aligned} \text{MABD}(\mathbf{Y}, \mathbf{X}) &= \text{LN}(\mathbf{H} + \text{rFFN}(\mathbf{H})) \\ \text{where} \quad \mathbf{H} &= \text{LN}(\mathbf{H}' + \text{MHSA}(\mathbf{H}', \mathbf{X}, \mathbf{X})) \\ \text{and} \quad \mathbf{H}' &= \text{LN}(\text{MHSA}(\mathbf{Y})) \end{aligned} \tag{9}$$

During training, MABD can be trained efficiently by shifting Y backward by one (with a start token at the beginning) and computing the output with one forward pass. During testing where one does not have access to Y, the model would then generate the future autoregressively. In order to avoid conditioning on the future during training, the $\text{MHSA}(Y)$ operation in MABD employs time-masking. This would prevent the model from accessing privileged future information during training.

B PERMUTATION EQUIVARIANCE OF THE MODEL

In this section, we prove the permutation equivariance of the Latent Variable Sequential Set Transformer with respect to the dimension M. To do so, we analyze the model's response to permutation along this dimension of the input tensor. This tensor, $\mathbf{X}$, is of dimension (K, M, t), where $K, M, t \in \mathbb{N}$. The dimensions t and K are ordered, i.e. if we index into the tensor along dimension M (we denote this operation as $\mathbf{X}_m$ where $m \in \{1, \ldots, M\}$) then we retrieve a (K, t)-dimensional matrix. However, the M dimension is unordered. This implies that when we index by another dimension, for example t, we retrieve an invertible multiset of vectors, denoted $\mathbb{X}_\tau = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ where $\tau \in \{1, \ldots, t\}$ and $\boldsymbol{x}_m \in \mathbb{X}_\tau$ are K-dimensional.

Our proof that our model is permutation equivariant on M demonstrates a clear difference with the properties of similar models. For example, the model proposed in Set Transformers (Lee et al., 2019) features a permutation equivariant encoder combined with a permutation *invariant* decoder. And while Deep Sets (Zaheer et al., 2017) provides a theoretical framework for designing neural network

models that operating on sets, they do not directly address the processing of hetereogeneous data structures as we do here.

The rest of the section will proceed in the following manner:

1. Provide definitions and lemmas of the mathematical structures we will use in our proofs.
2. Show the permutation equivariance of the AutoBot encoder on M.
3. Show the permutation equivariance of the AutoBot decoder on M.

For clarity, we re-state the motivation for proving permutation equivariance in this particular model. In tasks like motion prediction, we may not be able to uniquely identify moving objects (perhaps due to occlusion or similar visual features). Therefore, we may wish to represent these objects as a collection of states evolving over time. Readers may benefit from this perspective on $\mathbf{X}$ as a time-evolving (t) collection of M objects with K attributes. We described $\mathbb{X}_\tau$ as a multiset for generality, as it permits repeated elements. However, in motion prediction tasks it might be safe to assume that no two distinct objects will share the same attribute vector. By proving that our model is equivariant with respect to M, we ensure that it will not waste capacity on spurious order-specific features. The inductive bias for this type of task differs from other settings like machine translation where word order is meaningful.

B.1 Definitions and Lemmas

A permutation equivariant function is one that achieves the same result if you permute the set then apply the function to the permuted set, as you would achieve if you had applied the function to the set then applied the same permutation to the output set. We provide a formal description in Def B.1.

Definition B.1 (Permutation Equivariance)**.** *Let $\mathbb{X}$ be the set of n vectors $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$, and S_n the group of permutations of n elements $\pi : \mathbb{X} \to \mathbb{X}$. Suppose we have a function $f : \mathbb{X} \to \mathbb{X}$, we call f* ***Permutation Equivariant*** *iff $f(\pi(\mathbb{X})) = \pi(f(\mathbb{X}))$.*

One of the central components of the Sequential Set Transformer is the Multi-Head Self Attention function (MHSA : $\mathbb{X} \to \mathbb{X}$) originally introduced in (Vaswani et al., 2017). MHSA computes an updated representation for one input vector in a set of input vectors by processing its relation with a collection of other vectors. Without the explicit addition of positional information to the content of these vectors, this operation is *invariant* to input order. MHSA is a content-based attention mechanism, and updates the contents of a *single* input vector based on the contents of the others. However, if we define $\mathbf{f}_{\text{MHSA}}$ as the function applying MHSA to update *each* input vector, then we have created a permutation equivariant function, described in Lemma B.1.

Lemma B.1 ($\mathbf{f}_{\text{MAB}}$ is Permutation Equivariant on M)**.** *Let $\mathbb{X}$ be a set of D-dimensional vectors $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_M\}$, and $\mathbf{f}_{\text{MAB}} : \mathbb{R}^{D\times M} \to \mathbb{R}^{D\times M}$ be a function that applies* MAB *(as defined in Equation 1 of the main text) to each $\boldsymbol{x}_m \in \mathbb{X}$. Then $\mathbf{f}_{\text{MAB}}(\mathbb{X}_\tau)$ is permutation equivariant because $\forall \pi \in S_m$, $\mathbf{f}_{\text{MAB}}(\pi(\mathbb{X})) = \pi(\mathbf{f}_{\text{MAB}}(\mathbb{X}))$.*

Having discussed the preliminaries, we now show the permutation equivariance of AutoBot's encoder.

B.2 AutoBot Encoder Properties

Theorem B.2 (AutoBot Encoder is Permutation Equivariant on M)**.** *Let $\mathbf{X}$ be a tensor of dimension (K, M, t) where $K, M, t \in \mathbb{N}$, and where selecting on t retrieves an invertible multiset of sequences, denoted $\mathbb{X}_\tau = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_M\}$, where $\tau \in \{1, \dots, t\}$ and $\boldsymbol{x}_m \in \mathbb{X}_\tau$ are K-dimensional. The AutoBot encoder, as defined in Section 3.1, is permutation equivariant on M.*

Proof. What follows is a direct proof of the permutation equivariance of the AutoBot encoder. The encoder is a composition of three functions, rFFN: $\mathbb{R}^{K\times t} \to \mathbb{R}^{d_K\times t}$, $\mathbf{f}_{\text{MAB}} : \mathbb{R}^{d_k\times M} \to \mathbb{R}^{d_k\times M}$, and $\mathbf{f}_{\text{MAB}} : \mathbb{R}^{d_K\times t\times M} \to \mathbb{R}^{d_K\times t\times M}$.

First, we show that the embedding function rFFN is equivariant on M. Let rFFN represent the application of a function e: $\mathbb{R}^K \to \mathbb{R}^{d_K}$ to each element of a set $\mathbb{X}_\tau$ where $\tau \in \{1, \dots, t\}$. Then rFFN is equivariant with respect to M by Definition B.1. Specifically, we can see that the function rFFN satisfies the equation $\forall \pi \in S_M$, rFFN$(\pi(\mathbb{X}_\tau)) = \pi(\text{rFFN}(\mathbb{X}_\tau))$. In the AutoBot encoder, the

function rFFN is applied to each set $\mathbb{X}_\tau$, and because it is permutation equivariant on M for each, the application to each set represents a permutation equivariant transformation of the entire tensor.

Next, we show that the function $\mathbf{f}_{\text{MAB}}$, which applies a Multi-Head Attention Block (MAB) to each set, $(\mathbb{X}_1, \ldots, \mathbb{X}_t)$, is equivariant on M. $\mathbf{f}_{\text{MAB}}$ is permutation equivariant with respect to the M dimension of $\mathbf{X}$ because MAB is permutation equivariant on each set $\mathbb{X}_\tau$ by Lemma B.1.

Finally, we define a function $\mathbf{f}_{\text{MAB}}$ which applies a positional encoding along the time dimension of $\mathbf{X}$ then a Multi-head Attention Block (MAB) to each matrix in $\mathbf{X}$, $\{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$. $\mathbf{f}_{\text{MAB}}$ is permutation equivariant with respect to the M dimension of $\mathbf{X}$ because the collection of matrices $\{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$ is an unordered set, and the uniform application of a function to each element of this set satisfies permutation equvariance.

A composition of equivariant functions is equivariant, so the encoder is equivariant on M. □

B.3 AutoBot Decoder Properties

We now provide a direct proof the the permutation equivariance of the AutoBot decoder on M. The decoder is an auto-regressive function initialized with a seed set $\mathbb{X}_t$ containing M vectors of dimension d_k, each concatenated with a vector of latent variables and other context of dimension $2d_k$. For $\tau \in \{1, \ldots, T-t\}$ iterations, we concatenate the output of the decoder with the seed, the previous decoder output, and the latents to produce a tensor of dimension $(3d_K, M, \tau)$. In particular, the decoder is a composition of three functions, rFFN_{dec}: $\mathbb{R}^{3d_K \times \tau} \rightarrow \mathbb{R}^{d_K \times \tau}$, $\mathbf{f}_{\text{MAB}} : \mathbb{R}^{d_k \times M} \rightarrow \mathbb{R}^{d_k \times M}$, and $\mathbf{f}_{\text{MABD}} : \mathbb{R}^{d_K \times M \times \tau} \rightarrow \mathbb{R}^{d_K \times M \times \tau}$.

Theorem B.3 (AutoBot Decoder is Permutation Equivariant on M)**.** *Given an invertible multiset $\mathbb{X}_t$ with M vector-valued elements, and an auto-regressive generator iterated for $\tau \in \{1, \ldots, T-t\}$, the AutoBot decoder, formed by a composition of three functions, rFFN_{dec}: $\mathbb{R}^{3d_K \times M} \rightarrow \mathbb{R}^{d_K \times M}$, $\mathbf{f}_{\text{MAB}} : \mathbb{R}^{d_k \times M} \rightarrow \mathbb{R}^{d_k \times M}$, and $\mathbf{f}_{\text{MABD}} : \mathbb{R}^{d_K \times \tau \times M} \rightarrow \mathbb{R}^{d_K \times \tau \times M}$, is equivariant on M.*

Proof. First, we must establish that the function rFFN_{dec}: $\mathbb{R}^{3d_K \times M} \rightarrow \mathbb{R}^{d_K \times M}$ is equivariant with respect to M. The AutoBot decoder applies rFFN_{dec} to each set $\mathbb{X}_\tau$ where $\tau \in \{1, \ldots, T-t\}$, transforming the M vectors of dimension $3d_K \rightarrow d_k$. Because rFFN_{dec} represents a uniform application of e_{dec} to each element of the set, we can see that it satisfies the definition of permutation equivariance, specifically that $\forall \pi \in S_M$, $\text{rFFN}_{dec}(\pi(\mathbb{X}_\tau)) = \pi(\text{rFFN}_{dec}(\mathbb{X}_\tau))$.

Next, we take the function $\mathbf{f}_{\text{MABD}}$, which first applies a function $\text{PE} : \mathbb{R}^{d_k, M, \tau} \rightarrow \mathbb{R}^{d_k, M, \tau}$ to $\mathbf{X}$ adding position information along the temporal dimension, then applies $f_{\text{MABD}} : \mathbb{R}^{d_K \times \tau} \rightarrow \mathbb{R}^{d_K \times \tau}$ to each matrix $\{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$. $\mathbf{f}_{\text{MABD}}$ is permutation equivariant with respect to the M dimension of $\mathbf{X}$ because the collection of matrices $\{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$ is an unordered set, and the uniform application of a function to transform each element independently does not impose an order on this set.

Similar to the final step of the previous proof, we see that a function $\mathbf{f}_{\text{MAB}}$ applying a multi-head self-attention block (MAB) to each set $\{\mathbb{X}_{t+1}, \ldots, \mathbb{X}_\tau\}$ is equivariant on M because MAB is permutation equivariant on each set (see Lemma B.1).

The composition of equivariant functions is equivariant, so the decoder is equivariant. □

C Model Details

C.1 Ego-Centric Forecasting

AutoBot can be used to predict the future trajectories of all agents in a scene. However, in some settings we may wish to forecast the future trajectory of only a single agent in the set. We call this variant "AutoBot-Ego". Referring back to the encoder presented in Section 3.1, instead of propagating the encoding of all elements in the set, we can instead select the state of that element produced by the intra-set attention, $\boldsymbol{s} \in \mathbb{S}_\tau$. With this, our encoder proceeds with the rest of the computation identically to produce the tensor $\mathbf{C}^m_{1:t}$, which is an encoding of the ego element's history conditioned on the past sequence of all elements.

AutoBot-Ego's decoder proceeds autoregressively as presented in the previous section, with one exception. Since AutoBot-Ego only generates the future trajectory of an ego element, we do not

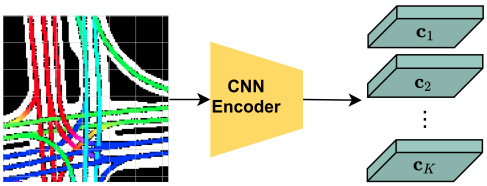

Figure 6: **AutoBots Context Encoder.** Example of the birds-eye-view road network provided by the Nuscenes dataset. Our CNN encoder produces a volume which we partition equally into K modes. This allows each generated trajectory to be conditioned on a different representation of the input image.

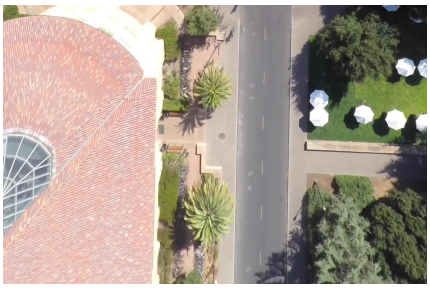

Figure 7: **TrajNet Map Example.** Example birds-eye-view map provided by the TrajNet benchmark. The footage was captured by a drone and pedestrians and bicycles are freely moving through the scene.

perform intra-set attention in the decoder. The objective function is updated to compute the likelihood of one element's future given the entire input sequence. That is, in AutoBot-Ego, $\mathcal{Y} = \mathbf{X}^m_{t+1:T}$.

C.2 Expectation Maximization Details

Our model also computes the distribution $P(Z|\mathbf{X}_{1:t})$ of discrete random variables. To achieve this, we employ c learnable vectors (one for each mode) concatenated into the matrix $\mathbf{P}$, which behave like seeds to compute distribution over modes prior to observing the future. The distribution is generated by performing the computations

$$p(Z|\mathbf{X}_{1:t}) = \text{softmax}(\text{rLin}(\mathbf{F})), \quad \text{where} \quad \mathbf{F} = \text{MABD}(\mathbf{P}_{1:c}, \mathbf{C}),$$

where rLin is a row-wise linear projection layer to a vector of size c.

C.3 Context Encoding Details

In the nuScenes dataset, we are provided with a birds-eye-view 63×63 RGB image of the road network, as shown in Figure 6. In the TrajNet dataset, we are also provided with a birds-eye-view of the scene, as shown in Figure 7. We process this additional context using a 4 layer convolutional neural network (CNN) which encodes the map information into a volume of size $7 \times 7 \times (7 * K)$ where K is the number of modes. We apply a 2D dropout layer with a rate of 0.3 on this output volume before processing it further. As our model employs discrete latent variables, we found it helpful to divide this volume equally among all K modes, where each mode receives a flattened version of the $7 \times 7 \times 7$ volume. As described in Section 3.2, this context is concatenated with the embedding of the mode's one-hot encoding and the current hidden state during the sequence generation process. Intuitively, each generated trajectory is conditioned on a different representation of the context.

C.4 Implementation and Training details

We implemented our model using the Pytorch open-source framework. Table 4 shows the values of parameters used in our experiments across the different datasets. The MAB encoder and decoder blocks in all parts of AutoBots use dropout. We train the model using the Adam optimizer with an initial learning rate. For the Nuscenes dataset, we anneal the learning rate every 10 epochs for the first 20 epochs by a factor of 2, followed by annealing it by a factor of 1.33 every 10 epochs for the next 30 epochs. In all datasets, we found it helpful to clip the gradients to a maximum magnitude of 0.5. For the Nuscenes experiments, our model takes approximately 80 epochs to converge, which corresponds to approximately 3 hours of compute time on a single Nvidia Geforce GTX 1080Ti GPU, using approximately 2 GB of VRAM. For the TrajNet and Omniglot datasets, our model takes

Parameter	Description	Toy	OmniGlot	Nuscenes	TrajNet
d_k	Hidden dimension throughout all parts of the model.	64	128	128	128
Learning Rate	Learning rate used in Adam Optimizer.	1e-4	5e-4	7.5e-4	7.5e-4
Batch size	Batch size used during training.	6	64	64	64
K	Number of discrete latent variables.	10	4	5 or 10	5
λ_e	Entropy regularization strength.	3.0	1.0	40.0	5.0
Dropout	Amount of dropout used in MHSA functions.	0.0	0.2	0.3	0.3
L	Number of stacked MAB in encoder and MABD in decoder.	1	1	2	2
H	Number of attention heads in all MHSAs.	8	8	8	8

Table 4: Hyperparameters used for AutoBots across all four datasets.

approximately 100 epochs which corresponds to a maximum 2 hours of compute time on the same resources. Table 5 highlights AutoBot-Ego's inference speed compared to WIMP, highlighting its potential to be employed in real-world applications.

	Input Steps	Output Steps	No. of Entities	Inference Rate
WIMP	$O(n)$	$\mathbf{O(n^2)}$	$O(n)$	$\sim$12Hz
AutoBot-Ego	$\mathbf{O(1)}$	$\mathbf{O(n^2)}$	$\mathbf{O(1)}$	$\sim$**25Hz**

Table 5: Time complexity and inference rate (nuScenes).

D AUTOBOTS ADDITIONAL RESULTS

D.1 ANALYSIS OF ENTROPY REGULARIZATION

This dataset contains the trajectories of a single particle having identical pasts but diverse futures, with different turning rates and speeds, as shown in Figure 8 (top row). In the second row of Figure 8, we show that our model trained without entropy regularization, $\lambda_e = 0.0$, can cover all modes. However, the short trajectory variants are all represented using only one of the modes (2nd row, 3rd column), as shown by the growth of the uncertainty ellipses as the trajectory moves forward. This shows that AutoBots learn to increase the variance of the bivariate Gaussian distribution in order to cover the space of trajectories. The third row of Figure 8, shows the learned trajectories for this variant trained with $\lambda_e = 3.0$. As we can see, the entropy regularization penalizes the model if the uncertainty of bivariate Gaussians is too large. By pushing the variance of the output distribution to a low magnitude, we restrict the model such that the only way it can achieve high likelihood is to minimize the distance between the mean of the bivariate distributions and ground-truth trajectories.

Figure 8: **Influence of entropy loss on particle experiment.** The input trajectory (cyan, only shown in top row) is identical in all cases. The model trained without an entropy loss term (middle row) covers all modes with high variance, while the model with moderate entropy regularization (bottom row) is able to learn the modes with low variance and without overlap.

Ablation Study - Importance of Entropy Loss Term. In Table 6, we show an ablation study to evaluate the importance of the entropy component on training the model for nuScenes. Again, we observe that for multi-modal predictions, enforcing low entropy on the predicted bivariate distributions has a great positive impact on the results, with an increased λ_e resulting in better performance. This was observed on all evaluated metrics with the exception of the min FDE, which was poor regardless of the model. Furthermore, we can see from Table 1 that, in general, all prior models have a difficulty obtaining strong performance on the min FDE (1) metric, with the best value having an overall error of 8.49 meters. This may be due to the difficulty of predicting the correct mode when one only has access to two seconds of the past sequence.

λ_e	**Min ADE (5)**	**Min ADE (10)**	**Miss Rate Top-5 (2m)**	**Miss Rate Top-10 (2m)**	**Min FDE (1)**	**Off Road Rate**
0	1.75	1.29	0.67	0.58	9.55	0.10
1	1.71	1.22	0.65	0.57	9.03	0.08
5	1.70	1.14	0.62	0.53	9.11	0.05
30	1.75	1.14	0.63	0.54	9.51	0.04
40	1.72	1.11	0.60	0.51	9.01	0.04

Table 6: **NuScenes Quantitative Results by Entropy Regularization**. We study AutoBot-Ego's performance when varying the entropy regularization term on nuScenes, and see that increased entropy improves model performance (particularly Min ADE 10 and Off Road Rate) up to a limit.

D.2 TRAJNET ADDITIONAL RESULTS

We refer the reader to Figures 9, 10, 11 and 12 where we show some additional qualitative results of AutoBots' variants compared to AutoBot on the TrajNet dataset. These results highlight the effectiveness of the sequential set attention mechanism in the decoding process of our method and how important it is to predict all agents in parallel compared to a 1-by-1 setup.

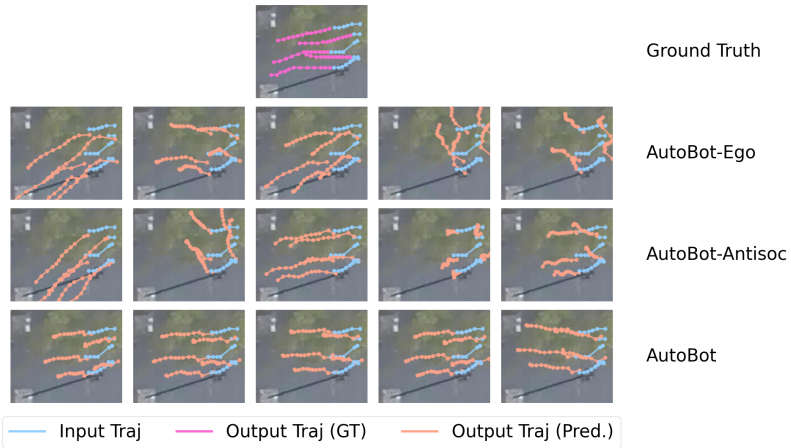

Figure 9: **TrajNet qualitative results 1/4.** Example scene with multiple agents moving together. These trajectories are plotted over the birds-eye-view image of the scene where we zoom into interesting trajectories. We can see that only AutoBot produces trajectories that are realistic in a group setting across all modes.

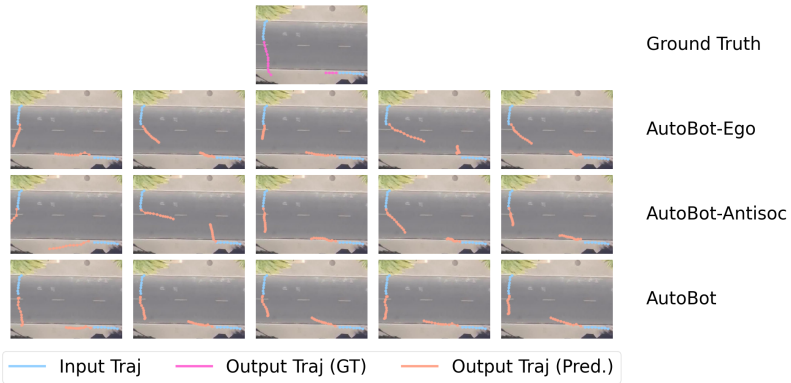

Figure 10: **TrajNet qualitative results 2/4.** Example scene with two agents moving separately in a road setting. We want to highlight this interesting scenario where some modes of AutoBots-Solo and AutoBots-AntiSocial results in trajectories that lead into the road, while AutoBot seems to produces trajectories more in line with the ground-truth, and lead the agent to cross the road safely.

D.3 Omniglot Additional Details and Results

The LSTM baseline used in our Omniglot experiments have a hidden size of 128 and use $K = 4$ latent modes. The input is first projected into this space using an embedding layer, which is identical to the one used in AutoBot. All input sequences of sets are encoded independently using a shared LSTM encoder. During the decoding phase, we autoregressively generate the next stroke step using an LSTM decoder. In order to ensure consistency between the predicted future strokes, inspired by Social LSTM (Alahi et al., 2016), the LSTM baseline employs an MHSA layer at every timestep operating on the hidden state of the different strokes making up the character. Similar to AutoBot, we concatenate a transformation of the one-hot vector representing the latent mode with the socially encoded hidden state at every timestep. The output model is identical to the one used AutoBot, generating a sequence of bivariate Gaussian distributions for each stroke. We performed a hyperparameter search on the learning rate and found the optimal learning rate to be $5e-4$. Furthermore, as with all our other experiments, we found it helpful to employ gradient clipping during training.

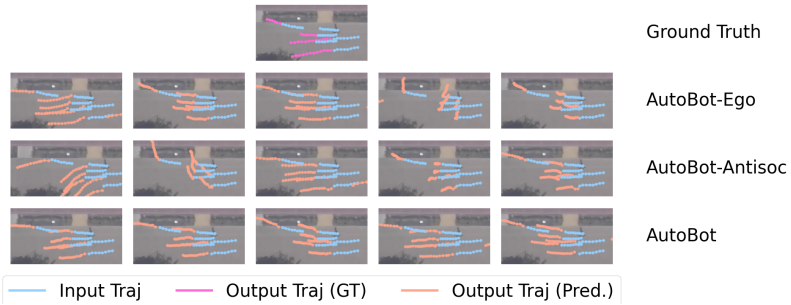

Figure 11: **TrajNet qualitative results 3/4.** Additional example scenes with multiple agents moving together. Again, we wish to highlight the advantage of modelling the scene jointly, which is evident by the results of AutoBot.

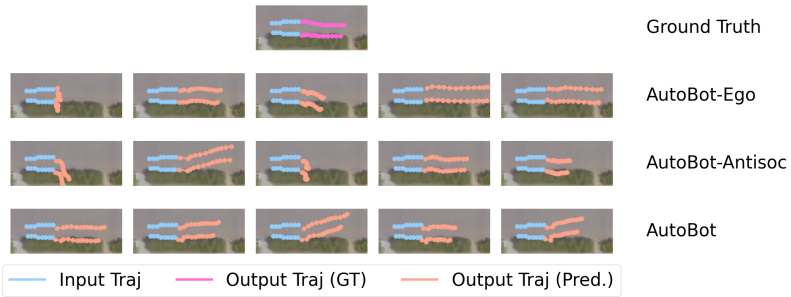

Figure 12: **TrajNet qualitative results 4/4.** Example scenes with two agents moving together. Again, we see that AutoBot produces trajectories consistent with the scene across all modes (e.g., not crashing into the bushes) and maintains the social distance between the walking agents.

We provide additional results on the two tasks defined in Section 4.4. Figure 13 shows additional successful AutoBot results on task 1 (completing multiple strokes) compared to an LSTM baseline equipped with social attention. These results highlight the effectiveness of sequential set transformers for generating consistent diverse futures given an input sequence of sets. Figure 14 shows examples where both models fail to correctly complete the character in task 1. Figure 15 compares AutoBot-Ego with the LSTM baseline on predicting a new stroke given the two first strokes of a character (task 2). These results highlight that although not all modes predict the correct character across all modes, all generated predictions are consistent with realistic characters (i.e., confusing “F" with “Π" or “H" with “Π").

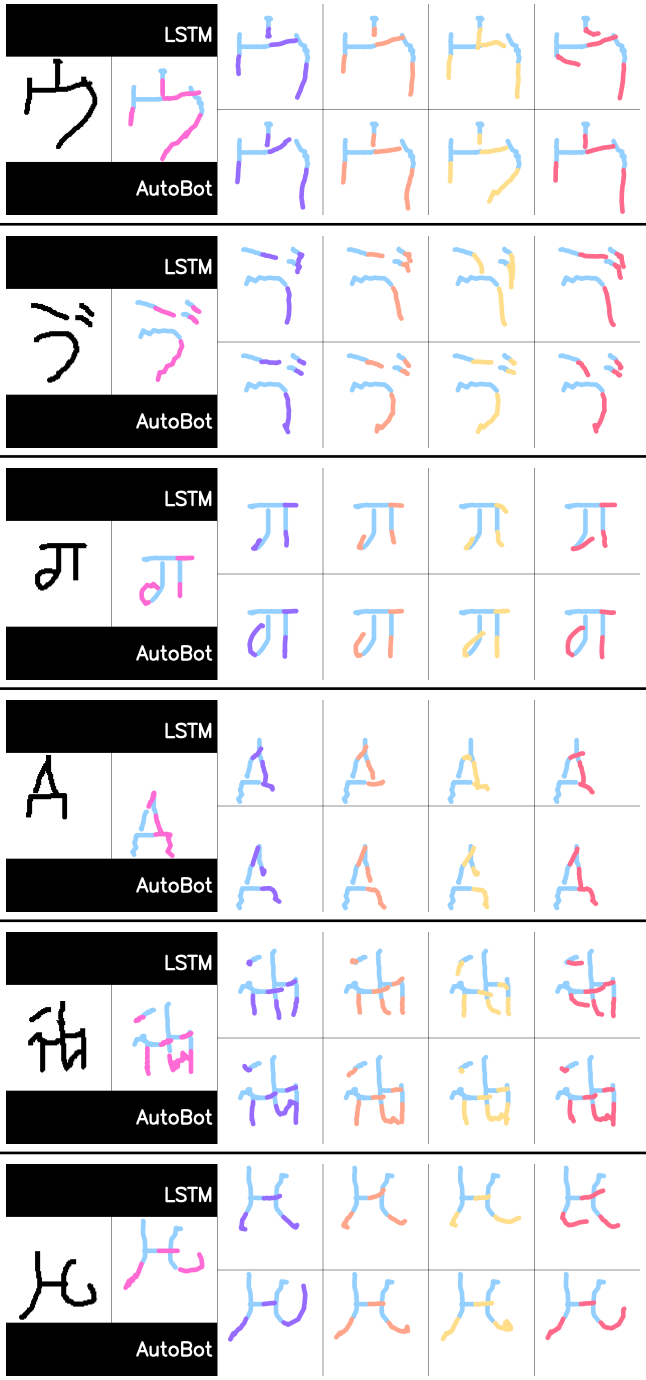

Figure 13: **Omniglot Task 1 Additional Results**. These are some additional random characters from the Omniglot stroke completion task. We can see that AutoBot produces plausible characters on the test set, while the LSTM struggles to create legible ones.

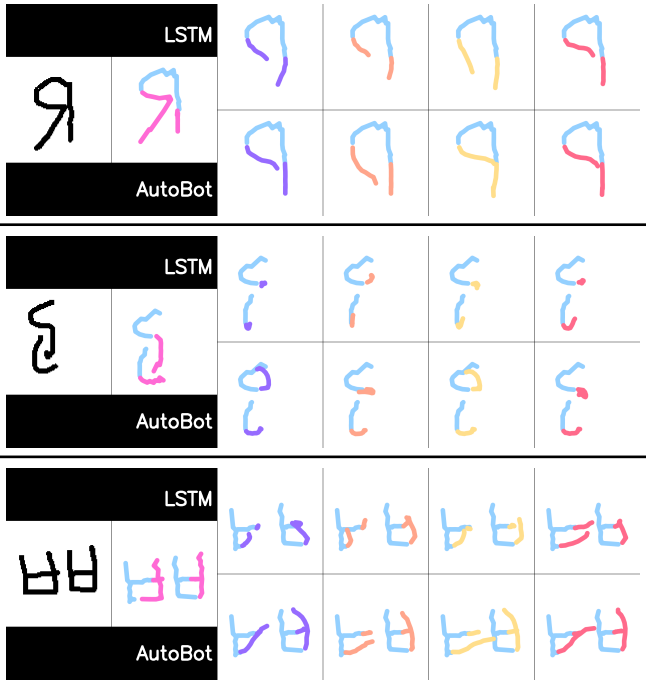

Figure 14: **Omniglot Task 1 Failure Cases**. There were some characters where AutoBot failed to learn the correct character completion. We can currently only speculate why that is the case. Our first intuition was that this might occur when there are 90 degree or less angles in the trajectory that is to be predicted but in Fig. 13, we can see that there are examples where this does not seem to be a problem. We believe that this might be due to a rarity of specific trajectories in the dataset but more investigation is required.

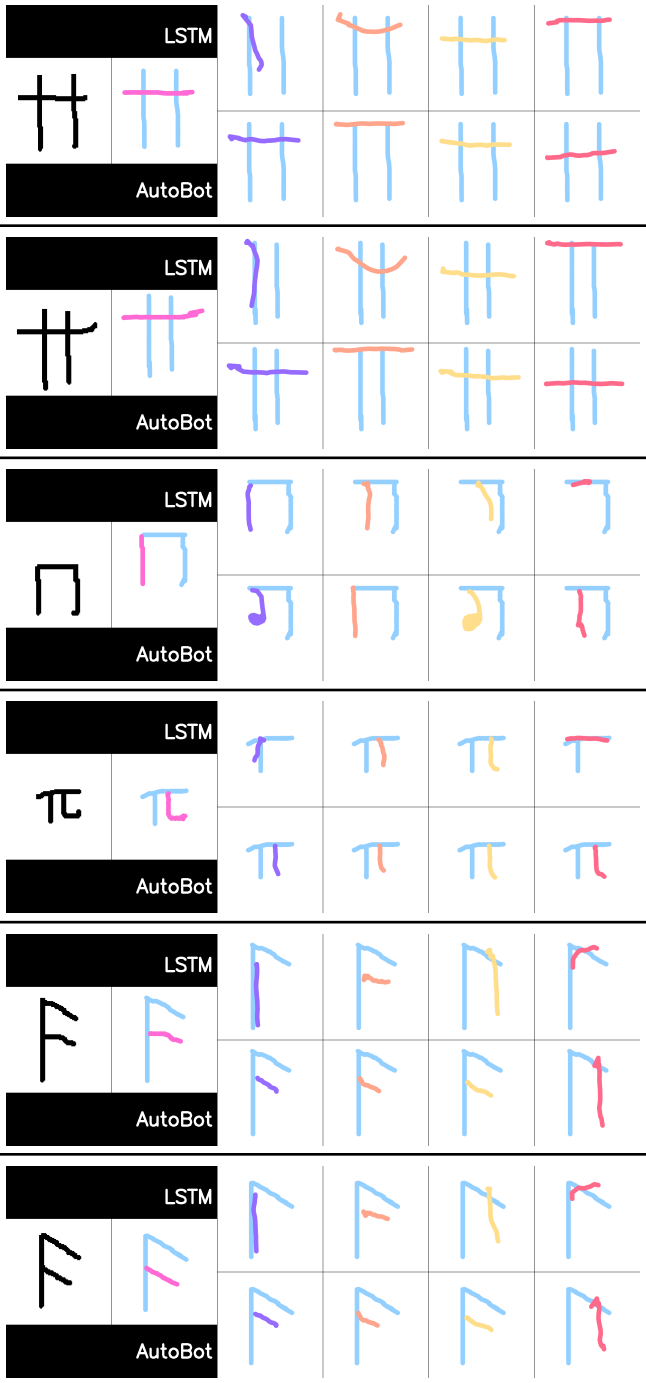

Figure 15: **Omniglot Task 2 Additional Results**. These are some additional random characters from the Omniglot character completion task. Again, we can see that AutoBot produces plausible characters on the test set, where different modes capture plausible variations (e.g. F to H and H to PI), while the LSTM struggles to capture valid characters in its discrete latents.