Autobots: Latent Variable Sequential Set Transformers

Roger Girgis$^{1,2}$ Florian Golemo$^{2,3}$ Felipe Codevilla$^{2,4}$ Jim Aldon D’Souza$^5$
Martin Weiss$^{1,2}$ Samira E. Kahou$^{2,6,7,8}$ Felix Heide$^{5,9}$ Christopher Pal$^{1,2,3,8}$

$^1$Polytechnique Montréal, $^2$Mila, Quebec AI Institute $^3$ElementAI / Service Now,
$^4$Independent Robotics, $^5$Algolux, $^6$McGill University, $^7$École de technologie supérieure,
$^8$Canada CIFAR AI Chair, $^9$Princeton University. Correspondence: roger.girgis@gmail.com
https://fgolemo.github.io/autobots/

Abstract

Robust multi-agent trajectory prediction is essential for the safe control of robots and vehicles that interact with humans. Many existing methods treat social and temporal information separately and therefore fall short of modelling the joint future trajectories of all agents in a socially consistent way. To address this, we propose a new class of Latent Variable Sequential Set Transformers which autoregressively model multi-agent trajectories. We refer to these architectures as “AutoBots”. AutoBots model the contents of sets (e.g. representing the properties of agents in a scene) over time and employ multi-head self-attention blocks over these sequences of sets to encode the sociotemporal relationships between the different actors of a scene. This produces either the trajectory of one ego-agent or a distribution over the future trajectories for all agents under consideration. Our approach works for general sequences of sets and we provide illustrative experiments modelling the sequential structure of the multiple strokes that make up symbols in the Omniglot data. For the single-agent prediction case, we validate our model on the NuScenes motion prediction task and achieve competitive results on the global leaderboard. In the multi-agent forecasting setting, we validate our model on TrajNet. We find that our method outperforms physical extrapolation and recurrent network baselines and generates scene-consistent trajectories.

1 Introduction

Many problems require processing complicated hierarchical compositions of sequences and sets. For example, multiple choice questions are often presented as compositions of ordered (e.g. natural language) and unordered data (the choices). Similarly, 3D-point cloud segmentation [27] requires the model to be invariant to rotations and orderings of input points, but sensitive to the ordering of point-specific attributes and time. Motion prediction [3, 22] may also require modelling an ordered time series containing information about multiple (unordered) agents.

In this work, we are interested in the generative modelling of sequences and sets. Suppose we have a set of $M \in \mathbb{N}$ sequences $X = \{ (x_1, \ldots, x_K)_1, \ldots, (x_1, \ldots, x_K)_M \}$ each with $K \in \mathbb{N}$ elements, which may consist of a mix of real valued scalars and other types of information such as categorical variables. Allowing also that $X$ evolves across some time horizon $T$, we denote this sequence of sets as $X = (X_1, \ldots, X_T)$. Predicting the evolution of sets is a general problem that may include complex interactions between elements and divergent futures due to aleatoric uncertainty.

Therefore, 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

Preprint. Under review.
We propose a model for hierarchies of sets and sequences. We analyze the theoretical implications of combining permutation variant and invariant architectures, and examine practical applications. A key application setting, shown in this figure, displays a scene involving moving objects structured as a hierarchy of sets and sequences. At each timestep, we have a collection of objects, where each object is represented by a sequence of scalar-valued properties. To generate diverse possible future scenes, we propose a model which is permutation invariant across certain dimensions and computes discrete modes for the next timestep for either a single object or all objects.

also allow to include a discrete latent variable $Z$, to create sequential models of sets of the form

$$P(X_{t+1}|X_t, \ldots, X_1) = \sum_Z P(X_{t+1}, Z|X_t, \ldots, X_1).$$

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 [6], an autonomous driving trajectory prediction benchmark, and on TrajNet [22], a pedestrian trajectory forecasting benchmark. Both include time-evolving sets of agents with highly dynamical states, and provide empirical validation of 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 validate our method with empirical results on diverse tasks, including a novel particle dataset, the multi-lingual Omniglot dataset of handwritten characters [14], a pedestrian trajectory prediction dataset TrajNet [22], and the large-scale nuScenes dataset for autonomous driving [6].

2 Related Work

Models of set-valued data should be permutation invariant (see Appendix A 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 [15]. One type of proposed solution is to integrate pooling mechanisms with feed forward networks to process set data [30, 24, 11]. Additionally, attention-based models have performed well in some set processing tasks that require modelling high-order element interactions [15, 7]. However, neither approach provides a straightforward approach for modelling more complex data-structures of sets and sequence, nor do they address how to generate these diverse and heterogeneous data-structures.

Generation of diverse samples using neural sequence models is a well-studied problem. For example, in Bowman et al. [5] 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 [18, 28], and Tang and Salakhutdinov [25] have recently proposed a model for image generation using discrete variables. Building on these works, we propose a method for the conditional generation of sequences of sets using discrete latent variables and show how a maximum entropy likelihood loss can improve the variability of the generated outcomes in this setting.

An exciting application of conditional generative models is trajectory or motion prediction, which can be formulated as the problem of predicting a multimodal distribution over sequences of sets, where each element in the set corresponds to an agent’s state. There are two main dimensions that need to be addressed by multi-agent motion forecasting models. The first dimension corresponds to how one models the sequence of states of an agent. Prior work has largely focused on employing RNNs (LSTMs [12] or GRUs [8]) to model the input and/or output sequence [1, 16, 9, 25, 20, 7, 17]. With
its recent success in natural language processing, Transformers [26] have been adopted by recent approaches for motion forecasting [10, 29, 19]. The second dimension is concerned with encoding the social interactions between agents. Early work employed a social pooling layer [1, 9, 16] to model the interaction between neighbouring agents’ representations, while more recent approaches employed graph-based attention or self-attention [7, 31, 20, 17]. Most prior approaches encode the time dimension separately from the social dimension. We argue that this limits the model’s representational capacity since it does not have the social dependence during the sequence encoding phase. By leveraging the Transformer’s attention mechanism, AutoBots produce representations that jointly encode both the time and social dimensions by treating this structure as a sequence of sets.

3 Background

We now review several components of the Transformer [26] and Set Transformer [15] architectures, mostly following the notation found in their manuscripts. Notice that a set \( \mathcal{A} \) may be cast (by imposing a random ordering) to a matrix \( A \) if every element shares the same dimension and type.

**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. Given \( n_q \) query vectors \( Q \in \mathbb{R}^{n_q \times d_q} \), and key-value pairs \( (K \in \mathbb{R}^{n_k \times d_k} \) and \( V \in \mathbb{R}^{n_v \times d_v} \), a single attention head executes \( Q \) by performing the computation:

\[
\text{Attn}(Q, K, V) = \text{softmax}(QK^T)V, \tag{1}
\]

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. 1 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

\[
\text{MHSA}(Q, K, V) = \text{concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \quad \text{where} \quad \text{head}_i = \text{Attn}(QW^Q_i, KW^K_i, VW^K_i), \tag{2}
\]

and where \( W^Q_i, W^K_i \) and \( W^K_i \) along with the output projection matrix \( W^O \) are the learnable projection matrices of each attention head in the MHSA. Note that \( W^Q_i, W^K_i \) and \( W^K_i \) project the initial set of inputs to smaller dimensionality. For example, \( W^Q_i \in \mathbb{R}^{d_q \times d_q} \) projects the original queries with dimension \( d_q \) to \( d_q^M \) which is typically chosen to be \( d_q/h \). In this work, we also “overload” the MHSA function to accept a single set-valued argument instead of three matrices. Then we perform intra-set attention by casting the input set \( X \) to query, key and value matrices and adding a residual connection, \( \text{MHSA}(X) = X + \text{MHSA}(X, X, X) \).

**Multi-Head Attention Blocks (MAB)** resemble the encoder proposed in Vaswani et al. [26], but lack the position encoding and dropout [15]. Specifically, they consist of the MHSA operation described in Eq. 2 followed by row-wise feed-forward neural network (rFFN), with residual connections and layer normalization (LN) [2] after the application of each function. Given an input set \( X \) containing \( n_x \) elements of dimension \( d \), and some conditioning variables \( C \) containing \( n_c \) elements of dimension \( d \), the MAB can be described by the forward computation,

\[
\text{MAB}(X, C) = \text{LN}(H + \text{rFFN}(H)), \quad \text{where} \quad H = \text{LN}(X + \text{MHSA}(X, C, C)). \tag{3}
\]

**Multi-head Attention Block Decoders (MABD)** were also introduced in Vaswani et al. [26], and were used to produce decoded sequences. Given an input matrix \( X \) and an output matrix \( Y \) representing sequences, the decoder block performs the following computations:

\[
\text{MABD}(Y, X) = \text{LN}(H + \text{rFFN}(H)) \quad \text{where} \quad H = \text{LN}(H' + \text{MHSA}(H', X, X)) \tag{4}
\]

and \( H' = \text{LN}(\text{MHSA}(Y)) \)

Our model, described throughout the following section, makes extensive use of these functions.
Our model takes as input a tensor \( X \) of dimension \( K, M, t \). A row-wise feed-forward network (rFFN) is applied to each column of data along the \( t \times M \) plane transforming the \( K \) elements into \( d_K \). The last timestep is used as a seed for the output trajectory and concatenated into the decoder tensor. In the encoder, the data tensor is transformed by a multi-head self-attention (MHSA) layer, and after positional encoding (PE) on the \( t \) axis, a multi-head attention block (MAB) into the “context” tensor, which is used to condition the multi-head attention block decoder (MABD) for each item in the output sequence. In the decoder, the output tensor is concatenated with the last timestep, the embedding of the discrete latent, and the embedding of the map/environment, all along the \( d_K \) axis before being fed through the rFFN (again, along the \( \tau \times M \) plane). The resulting tensor is positioned encoded (again, along the \( \tau \) axis), and fed through the conditional MABD and into another MHSA layer. This results in an update to the output tensor and this process is repeated for every timestep of the output sequence length.

4 Latent Variable Sequential Set Transformers

Latent Variable Sequential Set Transformers are a neural encoder-decoder model for sequences of sets. The following section describes the encoding procedure, Sub-section 4.2 describes the decoder and latent variable mechanism, Sub-section 4.3 describes the training objective and loss function, and finally Sub-section 4.4 describes the special case of predicting a single element of future sets (called AutoBot-Ego). We provide additional theoretical details in Appendix A and implementation details in Appendix B. An architecture overview can be seen in Fig.2.

4.1 Encoder: Input Sequence Representation

Here, we describe how AutoBot processes an input sequence of sets, \( X_{1:t} = (X_1, \ldots, X_t) \), where each set has \( M \) elements with \( K \) attributes. We process social relationships between set elements using MHSA function which performs the following operation on each set where \( \tau \in \{1, \ldots, t\} \):

\[
S_\tau = MHSA(e(X_\tau))
\]

where \( e(\cdot) \) is an embedding layer projecting each element in the set to the size of the hidden state \( d_K \). \( S_\tau \) is the encoded set, where each element \( s \in S_\tau \) may contain information about all other elements.

To inject temporal information into the sequence of sets \( S = (S_1, \ldots, S_t) \), we construct a sinusoidal positional encoding function called \( PE(\cdot) \) [26]. We apply PE to the sequence of sets \( PE(S) \) such that each vector-valued element \( s \in S_\tau \) is modified in the same way. Because this operation is performed uniformly across set elements, it does not introduce an ordering to the set.

Finally to encode each agent’s history, we instead view \( S \) as a collection of matrices, \( \{S_0, \ldots, S_M\} \), where each matrix contains attributes over time. For each \( S_m \) we apply \( L \) MAB encoding layers. Each MAB encoder performs the computation \( MAB(S_m, S_m) \) which has the effect of processing temporal change across the socially aggregated state vectors. The output of this procedure is a context tensor \( C \) of dimension \( (d_K, M, t) \) that represents the encoded socio-temporal data-structure.
4.2 Decoder: Multimodal Sequence Generation

AutoBot encourages diverse predictions using $K$ discrete latent variables to condition the sequence generation process. We represent the different modes with $K$ one-hot binary vectors $Z$ where $\sum Z = 1$.

The decoding process begins similarly to the encoding process. First, we select the set $X_t$ containing the final state of the input tensor, and embed each element using a linear layer, $h_{e}(X_t)$. Next, the decoder concatenates a linear projection of the mode’s one-hot vector, $z = e_{\theta}(Z)$ with each vector $h \in \mathbb{H}_t$. If additional context $e$ is provided (e.g., a map of a road network), these features are concatenated with the two other embeddings, $\text{concat}(h, z, e)$, then fed into a feed-forward network and summed with a sinusoidal positional encoding.

Next, these concatenated agent state vectors are passed through $L$ MABD layers. Up to this point, the future of each element in the set would be generated independently from all other elements. To relate them, AutoBot employs an MHSA layer, performing multi-head attention between all elements of the set at the current timestep. We repeat this process auto-regressively for $\tau \in \{t, \ldots, T\}$, i.e.,

\[ H_{t-\tau+1} = \text{MHSA}(\text{MABD}(\text{PE}(H'_{t,\tau}, C_{1:t})^T), \text{rFFN}(\text{concat}(h, z, e))|h \in \mathbb{H}_t), \]

and $H'_{t,\tau}$ is the sequence $(\mathbb{H}'_{t,1}, \ldots, \mathbb{H}'_{t,\tau})$. Finally, a function $\phi$ is used to convert the generated output sequence to the correct representation. In several of our experiments, we generate trajectories in $(x, y)$ space, and as such, $\phi$ produces the parameters of a bivariate Gaussian distribution.

Notice that the decoder generates trajectories in an autoregressive fashion, instead of using teacher forcing. We found that with teacher forcing, the model does not learn to produce trajectories that adhere to the current mode. Instead, the model produces identical trajectories from which one would need to sample in order to produce diverse futures trajectories. This is undesirable since it removes control over the generated trajectories, and reduces the likelihood of reproducing the same trajectories.

Our model also computes the distribution $P(Z|X_{1:t})$ of discrete random variables using the structure shown in Appendix B. To achieve this, we employ $K$ learnable vectors concatenated into the matrix $F$, which behave like seeds. The distribution is generated by performing the computations,

\[ p(Z|X_{1:t}) = \text{softmax}(\text{rLin}(F)), \quad \text{where} \quad F = \text{MABD}(P_{1:K}, C_{1:t}), \]

and where $\text{rLin}$ is a row-wise linear projection layer to a vector of size $K$.

4.3 Training Objective

Given a dataset $D = \{(X_1, \ldots, X_T)\}_{i=1}^N$ 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., $\max p(X_{t+1:T}|X_{1:t})$. In order to simplify notation, we will henceforth refer to the ego-agent’s future trajectory by $Y = X_{t+1:T}$. 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:

\[
\nabla_{\theta} L(\theta) = \nabla_{\theta} \log p_{\theta}(Y|X_{1:t}) = \nabla_{\theta} \log \left( \sum_Z p_{\theta}(Y, Z|X_{1:t}) \right) \\
= \sum_Z p_{\theta}(Z|Y, X_{1:t}) \nabla_{\theta} \log p_{\theta}(Y, Z|X_{1:t})
\]

(5)

As previously discussed in Tang and Salakhutdinov [25], computing the posterior likelihood $p_{\theta}(Z|Y, X_{1:t})$ is difficult in general and varies with $\theta$. As discussed in Bishop [4], 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:

\[
\log p_{\theta}(Y|X_{1:t}) = \sum_Z q(Z) \log \frac{p_{\theta}(Y, Z|X_{1:t})}{q(Z)} + D_{KL}(q(Z)||p_{\theta}(Z|Y, X_{1:t})),
\]

(6)

where $D_{KL}(||)$ is the Kullback-Leibler divergence between the approximating posterior and the actual posterior. A natural choice for this approximating distribution is $q(Z) = p_{\theta_{\text{std}}}(Z|Y, X_{1:t})$, where $\theta_{\text{std}}$ corresponds to AutoBot’s parameters before performing the parameter update. With this
We now demonstrate the capability of AutoBots to model sequences of set-structured data across with different turning rates and speeds, as shown in Figure 3 (top row). In the second row of Figure 3, we show strong performance on the competitive motion-prediction dataset NuScenes [6], a dataset 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 perform intra-set attention in the decoder. The objective function is updated to compute the likelihood of the ego element's history conditioned on the past sequence of all elements. We call this variant "AutoBot-Ego". Referring back to the encoder presented in Section 4.1, 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, \(s \in S_m\). With this, our encoder proceeds with the rest of the computation identically to produce the tensor \(C_{1:t}^{m}\), 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 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, \(Y = X_{t+1:T}^{m}\).

### 5 Experiments

We now demonstrate the capability of AutoBots to model sequences of set-structured data across several domains. In sub-section 5.1, we present our results on a small particle dataset to highlight our model's ability to capture discrete latent modes and express them as separate trajectories in a very simple setting. Then in sub-section 5.2, we use AutoBots to generate characters conditioning across strokes, and verify that AutoBot-Ego is permutation invariant across an input set. In sub-section 5.3 we show strong performance on the competitive motion-prediction dataset NuScenes [6], a dataset recorded from a self-driving car, capturing other vehicles and the intersection geometry. Finally, in sub-section 5.4 we show results on TrajNet [22], a benchmark for trajectory prediction that captures pedestrians, bicycles, and skaters from an overhead drone.

#### 5.1 Small Synthetic Non-linear Particle Accelerator Dataset

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 3 (top row). In the second row of Figure 3, 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 3 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 choice of \(q(Z)\), the objective function can be re-written as

\[
\mathcal{L}(\theta) = Q(\theta, \theta_{old}) + \text{const.}
\]

\[
Q(\theta, \theta_{old}) = \sum_Z p_{\theta_{old}}(Z|Y, X_{1:t}) \log p_{\theta}(Y, Z|X_{1:t})
\]

\[
= \sum_Z p_{\theta_{old}}(Z|Y, X_{1:t}) \left\{ \log p_{\theta}(Y|Z, X_{1:t}) + \log p_{\theta}(Z|X_{1:t}) \right\}.
\]

(7)

Note that the posterior \(p_{\theta_{old}}(Z|Y, X_{1:t})\) 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 \(D_{KL}(p_{\theta}(Z|Y, X_{1:t})||p_{\theta}(Z|X_{1:t}))\).

The datasets we tackle are concerned with predicting the \((x, y)\) coordinates of agents in a social setting. As discussed above, we perform this by outputting a bivariate Gaussian distribution at every timestep. 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,

\[
L_{ME} = \lambda_e \max_Z \sum_{t=t+1}^{T} H(p_{\theta}(X_t|Z, X_{1:t})).
\]

(8)

As we can see, this entropy regularizer penalizes only the mode with the maximum entropy. Our results in Section 5.1 show the importance of this term in learning to cover the modes of the data.
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.

### 5.2 Omniglot, Stroke-based Image Generation

In this section, we evaluate AutoBots on a character stroke completion as a diverse alternative task. We set up two different tasks on the Omniglot dataset [14]: (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 hope to illustrate that our model is not specific to traffic forecasting and can learn to complete characters from a variety of different languages legibly. In task (b), we train AutoBot-Ego only on characters “F”, “H”, and “π” to demonstrate that our discrete latent variable captures plausible character completions given the context of the other strokes, e.g. a letter “F” can be completed as an “H” when the top stroke is missing. The results of these experiments can be found in Fig. 4. Implementation details and additional results can be found in appendix section C.2.

### 5.3 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). Furthermore, we have access to all neighbouring agents’ past trajectories. In our experiments, we limited the number of agents to the ten nearest neighbours at the final timestep in the input sequence. The benchmark also provides a birds-eye-view RGB image of the road network in the vicinity of the ego-agent. This allows the model to attend to the different possible roads.

We compare our quantitative 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.

Considering those metrics, we obtained the best overall Min ADE out of 5 predictions when we employ a model with only five modes ($K = 5$). In addition, we also excelled in maintaining predictions on the road, showing that the model has learned to correctly use the 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 [13] 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 attention, a task-specific approach. We leave it to future work to explore if AutoBots with
Figure 4: **Omniglot qualitative results.** Left: stroke completion task. We show three examples of characters generated using AutoBot and an LSTM baseline. The two first 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 (blue) 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 three 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 task, 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.

| 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**     |
| AutoBot-Ego (K=10)     | **1.54**    | **1.10**     | 0.68                 | 0.50                  | 9.12        | **0.04**     |
| AutoBot-Ego (K=5)      | **1.46**    | 1.25         | 0.70                 | 0.53                  | 8.99        | **0.03**     |

Table 1: **Quantitative Results on the nuScenes dataset.** Other methods: LISA [20]; Trajectron++ [23]; CoverNet [21]; Physics Oracle [6]; WIMP [13]

polyline road network representations can yield improved results. Note, that in the appendix B, we present the computational complexity of AutoBot-Ego, which show 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).

Figure 5 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.

**Ablation Study - Importance of Entropy Loss Term.** In Table 2, 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 $\lambda_e$ 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.

| \( \lambda_c \) | 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 2: 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.

5.4 TrajNet, A Pedestrian Motion Prediction Dataset

We further validate AutoBots on a pedestrian trajectory prediction dataset TrajNet [22] 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 investigate the contribution of various components of our model.

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 (Figures ??, ??). 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 \( K = 5 \).

Figure 6 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 realities.

Although all methods have a similar Min ADE (5) on the ego-agent’s future trajectory, using the social data with the multi agent modelling in the decoder greatly reduces the number of collisions between predicted agent futures. We argue that producing a collision free prediction is more aligned with reality than an exact reproduction of the ground truth data. 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 [7]. AutoBot does not make this assumption as we condition between agents at every timestep.

| Model              | Min ADE (5) | Number Of Collisions |
|--------------------|-------------|----------------------|
| AutoBot-AntiSocial | 0.338       | 474                  |
| AutoBot-Ego        | 0.356       | 466                  |
| AutoBot            | 0.343       | 326                  |

Table 3: 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 while non-social methods fit the data better (lower Min ADE), AutoBot is able to cause fewer collisions between agents.
Figure 6: 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 compare 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.

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. Autonomous robot and car technology could be used by hostile actors for undesirable applications and there are many detailed discussions underway involving the technology community and broader members of society addressing the many aspects of these issues. 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.

Acknowledgments and Disclosure of Funding

We thank NSERC, Algolux, Mila and the MEI as well as CIFAR for their support under the AI Chairs and Catalyst programs.

References

[1] 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.

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

[3] 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.

[4] C. M. Bishop. Pattern Recognition and Machine Learning. 2006.
[5] 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.

[6] 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.

[7] 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.

[8] Kyunghyun Cho, B. V. Merrienboer, Çağlar 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.

[9] 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.

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

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

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

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

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

[15] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosioruk, 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.

[16] N. Lee, W. Choi, Paul Vernaza, C. Choy, P. Torr, and Mannmohan 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.

[17] 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.

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

[19] Yicheng Liu, Jinghui Zhang, Liangji Fang, Qinshong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. *ArXiv*, abs/2103.11624, 2021.

[20] 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.

[21] 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.

[22] Amir Sadeghian, Vineet Kosaraju, Agrim Gupta, Silvio Savarese, and Alexandre Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *arXiv preprint*, 2018.
[23] 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.

[24] 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.

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

[26] 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.

[27] Renhao Wang, Marjan Albooyeh, and Siamak Ravanbakhsh. Equivariant networks for hierarchical structures. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 13806–13817. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/9efb1a59d7b58e69996cf0e32cb71098-Paper.pdf.

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

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

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

[31] 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, 2020.
A 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 $\mathcal{X}_\tau = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ where $\tau \in \{1, \ldots, t\}$ and $\mathbf{x}_m \in \mathcal{X}_\tau$ 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 [15] features a permutation equivariant encoder combined with a permutation invariant decoder. And while Deep Sets [30] provides a theoretical framework for designing neural network models that operating on sets, they do not directly address the processing of heterogeneous 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 $\mathcal{X}_\tau$ 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.

A.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 A.1).

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

One of the central components of the Sequential Set Transformer is the Multi-Head Self Attention function (MHSA $: \mathcal{X} \rightarrow \mathcal{X}$) originally introduced in [26]. 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 $f_{\text{MHSA}}$ as the function applying MHSA to update each input vector, then we have created a permutation equivariant function, described in Lemma A.1.

**Lemma A.1 ($f_{\text{MHSA}}$ is Permutation Equivariant on $M$).** Let $\mathcal{X}$ be a set of $D$-dimensional vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, and $f_{\text{MHSA}} : \mathbb{R}^{D \times M} \rightarrow \mathbb{R}^{D \times M}$ be a function that applies MHSA (as defined in Equation 2 of the main text) to each $\mathbf{x}_m \in \mathcal{X}$. Then $f_{\text{MHSA}}(\mathcal{X}_\tau)$ is permutation equivariant because $\forall \pi \in S_m, f_{\text{MHSA}}(\pi(\mathcal{X})) = \pi(f_{\text{MHSA}}(\mathcal{X})))$.

Having discussed the preliminaries, we now show the permutation equivariance of AutoBot’s encoder.
A.2 AutoBot Encoder Properties

**Theorem A.2** (AutoBot Encoder is Permutation Equivariant on \( M \)). Let \( X \) 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 \( \mathcal{X}_t = \{x_1, \ldots, x_M\} \), where \( \tau \in \{1, \ldots, M\} \) and \( x_m \in \mathcal{X}_\tau \) are \( K \)-dimensional. The AutoBot encoder, as defined in Section 4.1, 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, \( e: \mathbb{R}^{K \times t} \to \mathbb{R}^{dK \times t} \), \( f_{\text{MHSA}}: \mathbb{R}^{dK \times M} \to \mathbb{R}^{dK \times M} \), and \( f_{\text{MAB}}: \mathbb{R}^{dK \times t \times M} \to \mathbb{R}^{dK \times t \times M} \).

First, we show that the embedding function \( e \) is equivariant on \( M \). Let \( e \) represent the application of a function \( e: \mathbb{R}^K \to \mathbb{R}^{dK} \) to each element of a set \( X_\tau \) where \( \tau \in \{1, \ldots, M\} \). Then \( e \) is equivariant with respect to \( M \) by Definition A.1. Specifically, we can see that the function \( e \) satisfies the equation \( \forall \pi \in S_M, e(\pi(X_\tau)) = \pi(e(X_\tau)) \). In the AutoBot encoder, the function \( e \) is applied to each set \( X_\tau \), 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 \( f_{\text{MHSA}} \), which applies Multi-Head Self-Attention (MHSA) to each set, \((X_1, \ldots, X_M)\), is equivariant on \( M \). \( f_{\text{MHSA}} \) is permutation equivariant with respect to the \( M \) dimension of \( X \) because MHSA is permutation equivariant on each set \( X_\tau \) by Lemma A.1.

Finally, we define a function \( f_{\text{MAB}} \) which applies a positional encoding along the time dimension of \( X \) then a Multi-head Attention Block (MAB) to each matrix in \( X \), \( \{X_1, \ldots, X_M\} \). \( f_{\text{MAB}} \) is permutation equivariant with respect to the \( M \) dimension of \( X \) because the collection of matrices \( \{X_1, \ldots, X_M\} \) is an unordered set, and the uniform application of a function to each element of this set satisfies permutation equivariance.

A composition of equivariant functions is equivariant, so the encoder is equivariant on \( M \). \( \square \)

A.3 AutoBot Decoder Properties

We now provide a direct proof of the permutation equivariance of the AutoBot decoder on \( M \). The decoder is an auto-regressive function initialized with a seed set \( X_0 \) 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, \( e_{\text{dec}}: \mathbb{R}^{3d_K \times \tau} \to \mathbb{R}^{dK \times \tau} \), \( f_{\text{MHSA}}: \mathbb{R}^{dK \times M} \to \mathbb{R}^{dK \times M} \), and \( f_{\text{MABD}}: \mathbb{R}^{dK \times M \times \tau} \to \mathbb{R}^{dK \times M \times \tau} \).

**Theorem A.3** (AutoBot Decoder is Permutation Equivariant on \( M \)). Given an invertible multiset \( \mathcal{X}_t \) 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, \( e_{\text{dec}}: \mathbb{R}^{3d_K \times M} \to \mathbb{R}^{dK \times M} \), \( f_{\text{MHSA}}: \mathbb{R}^{dK \times M} \to \mathbb{R}^{dK \times M} \), and \( f_{\text{MABD}}: \mathbb{R}^{dK \times M \times \tau} \to \mathbb{R}^{dK \times M \times \tau} \), is equivariant on \( M \).

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

Next, we take the function \( f_{\text{MABD}} \), which first applies a function \( \text{PE}: \mathbb{R}^{dK, M, \tau} \to \mathbb{R}^{dK, M, \tau} \) to \( X \) adding position information along the temporal dimension, then applies \( f_{\text{MABD}}: \mathbb{R}^{dK \times \tau} \to \mathbb{R}^{dK \times \tau} \) to each matrix \( \{X_1, \ldots, X_M\} \). \( f_{\text{MABD}} \) is permutation equivariant with respect to the \( M \) dimension of \( X \) because the collection of matrices \( \{X_1, \ldots, X_M\} \) 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 \( f_{\text{MHSA}} \) applying multi-head self-attention (MHSA) to each set \( \{X_{t+1}, \ldots, X_\tau\} \) is equivariant on \( M \) because MHSA is permutation equivariant on each set (see Lemma A.1).

The composition of equivariant functions is equivariant, so the decoder is equivariant. \( \square \)
B Model Details

B.1 Additional Context Encoding

In the NuScenes dataset, we are provided with a birds-eye-view 63 \times 63 RGB image of the road network, as shown in Figure 7. In the TrajNet dataset, we are also provided with a birds-eye-view of the scene, as shown in Figure 8. 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 \times 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 4.2, 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.

B.2 Implementation and Training details

| 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       |
| \(\lambda_e\)      | 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     |
| Number of stacked MAB in encoder | Number of stacked MAB in encoder and MABD in decoder. | 1   | 1        | 2        | 2       |
| Number of attention heads in all MHSA's. | 8   | 8        | 8        | 8       |

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

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 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        | $O(n)$       | $O(n^2)$        | $O(n)$         | $\sim$12Hz   |
| AutoBot-Ego | $O(1)$       | $O(n^2)$        | $O(1)$         | $\sim$25Hz   |

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

### C AutoBots Additional Results

#### C.1 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.

#### C.2 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 [1], 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 $5 \times 10^{-4}$. Furthermore, as with all our other experiments, we found it helpful to employ gradient clipping during training.

We provide additional results on the two tasks defined in Section 5.2. 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.