MissFormer: (In-)attention-based handling of missing observations for trajectory filtering and prediction

Stefan Becker[1], Ronny Hug[1], Wolfgang Huebner[1], Michael Arens[1], and Brendan T. Morris[2]

[1] Fraunhofer IOSB*, Ettlingen, Germany
{firstname.lastname}@iosb.fraunhofer.de
www.iosb.fraunhofer.de
[2] University of Nevada, Las Vegas, USA
brendan.morris@unlv.edu

Abstract. In applications such as object tracking, time-series data inevitably carry missing observations. Following the success of deep learning-based models for various sequence learning tasks, these models increasingly replace classic approaches in object tracking applications for inferring the objects' motion states. While traditional tracking approaches can deal with missing observations, most of their deep counterparts are, by default, not suited for this.
Towards this end, this paper introduces a *transformer*-based approach for handling missing observations in variable input length trajectory data. The model is formed indirectly by successively increasing the complexity of the demanded inference tasks. Starting from reproducing noise-free trajectories, the model then learns to infer trajectories from noisy inputs. By providing missing tokens, binary-encoded missing events, the model learns to in-attend to missing data and infers a complete trajectory conditioned on the remaining inputs. In the case of a sequence of successive missing events, the model then acts as a pure prediction model. The abilities of the approach are demonstrated on synthetic data and real-world data reflecting prototypical object tracking scenarios.

Keywords: Transformer · Trajectory Data · Missing Input Data · Filtering · Trajectory Prediction · Missing Observations

1 Introduction & Related Work

One crucial task for autonomous systems is estimating the agents' motion states based on observations. Following the success of deep learning-based models in various sequence processing tasks, like speech recognition [7, 12] and caption generation [10, 37], these models are successfully utilized for trajectory prediction.

* Fraunhofer IOSB is a member of the Fraunhofer Center for Machine Learning.

In trajectory prediction applications, deep learning-based approaches are increasingly replacing classic approaches due to their ability to capture better contextual cues from the static (e.g., obstacles; *scene cues*) or dynamic environment (e.g., other objects in the scene; *social cues*)[29]. Commonly used approaches for encoding object motions rely on *recurrent neural networks* (RNNs) [1, 14], *temporal convolution networks* (TCNs) [2, 24], or *transformers* [11, 30]. The reader is referred to these surveys [28, 29, 20] for a comprehensive overview of current deep learning-based approaches for trajectory prediction. Since these models have the ability to consider *social cues* and *scene cues*, the focus of most research is how to incorporate these cues better. Although this research direction offers the strongest performance boost, problems such as missing observations are partly ignored or mainly addressed with data imputation and omitting the missing data [32]. To be more specific, this applies to trajectory prediction relying on observation extracted from an agent's trajectory as basic input (e.g., positions). Only this type of prediction problem is considered here and is referred to as *trajectory cues*-based prediction in the remainder. For example, RNNs are designed to receive input data in every step and therefore are by default not suited to deal with missing inputs. In contrast, *transformers* offer an alternative to the step-by-step processing in the form of the underlying attention mechanisms in combination with positional encoding. In general, data imputation means to substitute the missing values with methods like interpolation [21] or spline fitting [8] which results in a process where imputation and prediction models are separated [6]. Thus, only suboptimal results are achieved since the model does not effectively explore the missing pattern. The simplest strategy for omitting is to remove samples in which a value is missing. While for RNNs this may work for training but cannot be applied during inference, omitting the missing value can be applied with *transformers*. Giuliari et al. [11] suggested omitting data with a *transformer* model for trajectory prediction as an advantage compared to RNN-based models. They analyzed the effect of omitting the last observations of a fixed-length input sequence. Alternatively, and in particular for RNNs, the problem can be modeled with marked missing values. A missing value can be masked and explicitly excluded, or the model can be encouraged to learn that a specific value represents the missing observation (*missing tokens*)[5]. Most approaches are for healthcare applications [34] or in the field of speech recognition [25]. In the field of trajectory prediction, Becker et al. [3] introduced an RNN-based full temporal filtering cycle for motion state estimation to better deal with missing observations. The Kalman filter-inspired model learns to weigh between its short-term predictions and observations enriched with missing tokens. In cases of missing inputs, the model entirely relies on predictions. Due to the recursive incorporation of new observations, deep Kalman models can be adapted similarly.

In this paper, we further explore the ability of *transformer* networks to handle missing observations. Compared to the work of Giuliari et al. [11], we utilize a modified encoder-only *transformer* model and provide missing tokens. Thus, the model is encouraged to learn specific placeholder values representing the missing

observations. We analyze to what extent the combination of the underlying attention mechanisms with the positional encoding is able to then handle missing inputs along a variable length trajectory. Further, our model is not primarily designed as a prediction model, but in contrast, the model is formed indirectly by successively increasing the complexity of the demanded inference tasks. Starting from reproducing noise-free trajectories, the model then learns to infer trajectories from noisy inputs. The model outputs a full trajectory despite only being given partly observed trajectory data. Thus, for a sequence of successive missing events, the model then acts as a mere prediction model. The analysis of the model ability is performed under controlled conditions using synthetic data. For a comparison to other prediction models, the commonly used, publicly available *BIWI* [27] and *UCY* [22] datasets are used.

In the following, a brief formalization of the problem and a description of the proposed *transformer* model are provided in section 2. The achieved results are presented in section 3. Finally, a conclusion is given in section 4.

2 MissFormer

The goal is to devise a model that can successfully infer the trajectory of a tracked agent conditioned on *trajectory cues* (e.g., positions, headings, velocities) with missing observations. Trajectory prediction is formally stated as follows. Given an input sequence $\mathcal{X}$ of consecutively observed positions $\vec{x}^k = (p_x^k, p_y^k)$ (or other *trajectory cues*) at time step k along a trajectory, the task is to generate predictions for future positions $\{\vec{x}^{k+1}, \vec{x}^{k+2}, \ldots\}$ Here, we adapt the formal description as follows. Given a sequence of noisy, potentially missing observations $\tilde{\vec{x}}^k$, the task is to estimate the noise-free positions of the trajectory $\vec{x}^k$. So, $\tilde{\vec{x}}^k$ is a realization of $\vec{x}^k$ despite the fact the inputs of a *transformer* are deterministic. In case the observations are noise-free, the task is to reproduce the trajectories. Although this might sound trivial, there exists no commonly accepted standards on encoding trajectory data in a deep learning model [15]. When noise is present, the task is filtering. In addition to learning an adequate representation, the model needs to compensate input noise. If observations are missing at the end of the input sequence, the model acts as a prediction model and still infers the complete trajectory. Disregarding *scene* and *social cues*, trajectory prediction is here divided into different inference tasks with increasing complexity which the model can learn successively.

Input/output: For an agent, the *transformer* network outputs the complete trajectory $\{\vec{x}^1, \ldots \vec{x}^k\}$ up to time step k conditioned on $\{\tilde{\vec{x}}^1, \ldots \tilde{\vec{x}}^k\}$. To encourage the model to learn that a specific value represents missing, a binary-coded missing pattern is provided. The missing token is chosen as $(\vec{0}^k, 1^k)$ for a missing observation and, respectively, $(\tilde{\vec{x}}^k, 0^k)$ for a default input, where $k \in \{1, k_{max}\}$. The adapted input is embedded onto a higher d_{model}-dimensional space by means of a linear mapping $\vec{e}^k = \text{EMB}(\tilde{\vec{x}}^k; \vec{\Theta}_e)$. Accordingly, the output of the *transformer* model is re-mapped to the 2-dimensional coordinate system. Since *transformers*

contain no recurrence and no convolution, information about the position in the sequence must be injected. *Positional encodings* are added to the input embeddings in accordance to the original *transformer* [35]. The *positional encodings* have the same dimension d_{model} as the embeddings, so that both can be summed up. Hence, the embedded input is time-stamped at time step k by adding a *positional encoding* vector PE^k. Following [35], sine and cosine functions of different frequencies are used to define $PE^k = \{PE_{k,d}\}_{d=1}^{d_{\text{model}}}$ with

$$PE_{k,d} = \begin{cases} \sin\left(\frac{k}{10000^{d/d_{\text{model}}}}\right) & \text{for } d \text{ even} \\ \cos\left(\frac{k}{10000^{d/d_{\text{model}}}}\right) & \text{for } d \text{ odd} \end{cases} . \tag{1}$$

The time step k corresponds to the position in the sequence and d is the dimension. Each dimension of the positional encoding varies in time according to a sinusoid of different frequencies, from 2π to $10000 \cdot 2\pi$. That way, unique timestamps for sequences of up to 10000 elements are ensured.

MissFormer: Both the encoder and the decoder of a *transformer* are composed of a stack of identical layers consisting of two sub-layers. Firstly, an attention module, and secondly, a feed-forward fully-connected module. Around each sub-layer, a residual connection followed by layer normalization is employed. Here, we only use the encoder and directly map the encoded state to an entire estimated trajectory instead of an auto-regressive generation with a decoder. Contrary to using a step-by-step processing of RNNs or convolution, *transformers* rely entirely on self-attention to compute representations of its input and output. The attention function used by *transformers* is the so-called scaled dot-product attention. The inputs consists of *queries* and *keys* of dimension d_k, and *values* of dimension d_v packed into matrices Q, K and V. The attention layer is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\intercal}{\sqrt{d_k}})V \tag{2}$$

So, the attention layer computes a dot product of the *query* with all *keys*, divided by d_k, and followed by a softmax function to obtain the weights on the values. Multi-head attention performs several attention functions in parallel, yielding to d_v-dimensional outputs. These values are concatenated before projected to the final value. The *transformer* uses multi-head attention in different ways, whereas for an encoder-only architecture, solely the self-attention layer in the encoder is important. For more details, we refer to [35]. In a self-attention layer of the encoder, all of the *keys*, *values* and *queries* come from the same place, in this case the output of the previous layer in the encoder. On a high level, attention can be seen as routing of information. Thus, each position in the encoder can attend to all positions of the previous encoder layer or rather over all positions in the input sequence. The encoder creates a representation given the observation sequence resulting in the memory - the encoder state. In an encoder-decoder set-up, the encoder state is used to generate a *key* and *value*

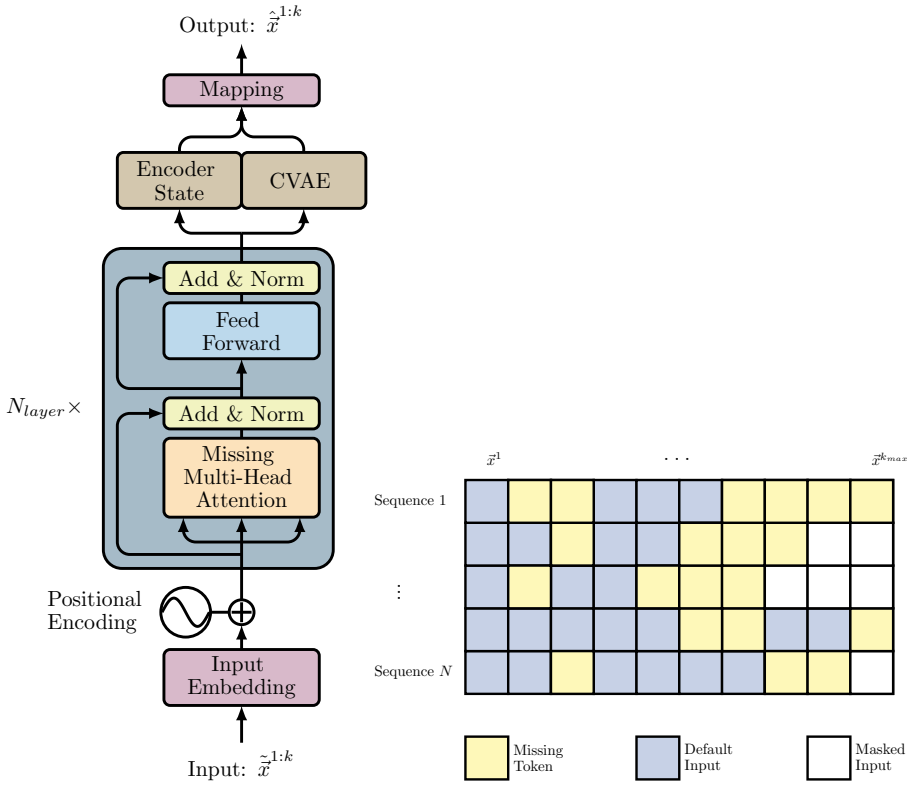

Fig. 1: (Left) Visualization of the adapted *transformer* network - MissFormer. (Right) Visualization of the adapted input data for the missing multi-head attention.

passed to the decoder. Since no new observations are provided to the decoder, we only used the encoder part. For every new observation, the extended input sequence is given to the model. As described, the model infers a sequence with a similar length to the input sequence. Our focus is on how well the described attention mechanism can route information to deal with missing observations and reconstruct complete, noise-free trajectories from the data. Further, the model is encouraged to still produce a meaningful representation and in-attend to useless placeholder values in the input sequence. The adapted *transformer* model, referred to as MissFormer, together with modified input data with missing tokens, is visualized in Figure 1.

On the left, the MissFormer with the missing self-attention sub-layer is shown. On the right, the *missing tokens* are highlighted in yellow and default inputs are highlighted in blue. The Missformer is trained by minimizing the L2-loss in the form of the mean squared error between the ground truth trajectories and the estimated trajectories. Exemplary, the encoded state is combined with

a conditional *variational auto encoder*(VAE) [18] for producing multiple outputs and capturing the uncertainty of the estimation. Thus, the *evidence-based lower bound* ELBO term is added as a second factor to the loss function (see for example [4]). However, capturing the multi-modality of trajectory prediction is out of the scope of this paper. For example, this component can be replaced by *flow*-based models [9] or $\mathcal{N}$-*curve* models [17].

3 Evalution & Analysis

This section consists of an evaluation of the proposed MissFormer. The evaluation concerns with verifying the approach's overall viability in situations with missing observations in variable-length trajectory data. One part of the evaluation is done with synthetically generated data because, firstly, reference models can not handle mission observation by default. Secondly, current pedestrian trajectory data sets do not consider this aspect. For a comparison to other approaches, the publicly available *BIWI* [27] and *UCY* [22] datasets are used according to the common practice of fully observed input data.

3.1 Synthetic Data

The synthetic data consists of diverse trajectories covering different types of prototypical object motion present in trajectory datasets [16]. The generated trajectories include the motion patterns of constant velocity, curvilinear motion, acceleration and deceleration motion. For generating synthetic trajectories of a basic object motion on a ground plane, random agents are sampled from a uniform distribution of speeds ($\mathcal{U}(5.0^{m}/_{s}, 10.0^{m}/_{s})$). The frame rate is set to $1fps$. The heading direction is sampled from $\mathcal{U}(0°, 360°)$ with a change of heading during a sampling period also sampled from uniform distribution of $\mathcal{U}(-20°, 20°)$. The de- and acceleration during a sampling period is sampled from $\mathcal{U}(-0.8^{m}/_{s^2}, 1.5^{m}/_{s^2})$. Missing events are drawn from a Bernoulli distribution $\mathcal{B}(\cdot, \cdot)$. The positional observation noise is assumed to follow a zero-mean Gaussian distribution $\mathcal{N}(0^{m}/_{s}, (\cdot^{m}/_{s})^2)$. The evaluation set includes always 5000 samples. The number of training samples is varied thought-out the experiments. The models have been implemented using *Pytorch* [26]. For training, an ADAM optimizer variant [19, 23] with a learning rate of 0.001 is used.

In order to emphasize some statements, parts from the first experiments are summarized in Table 1. For comparison of the different trained MissFormer models, the average displacement error (ADE) is calculated as the average L2 distance between the estimated positions and the ground truth positions. It should be noted that the model directly infers positions. Typically, only velocities or rather offsets are predicted and the last observation is used as a reference point. Since the amount of variation for offsets is lower compared to positions and the range in the data is more limited, less modeling effort and less data is required for model training (see from example [2]). Because the last observation is affected by noise or even missing, this practice is not applicable. Thus, here

Task: Encoding / Reconstruction								
input	output	obs.	pred.	#samples	#epochs	noise	missing	ADE σ_{ADE}
pos.	pos.	8-20	✗	1000	1000	✗	✗	0.067 0.013
off.	pos.	8-20	✗	1000	1000	✗	✗	0.061 0.012
pos.	pos.	8-20	✗	1000	1000	✗	$\mathcal{B}(0.1, 0.9)$	0.377 0.403
off.	pos.	8-20	✗	1000	1000	✗	$\mathcal{B}(0.1, 0.9)$	0.175 0.154
pos.	pos.	8-20	✗	3000	1000	✗	$\mathcal{B}(0.1, 0.9)$	0.138 0.074
off.	pos.	8-20	✗	3000	1000	✗	$\mathcal{B}(0.1, 0.9)$	0.155 0.079
pos.	pos.	8-20	✗	3000	3000	✗	✗	0.030 0.020
off.	pos.	8-20	✗	3000	3000	✗	✗	0.039 0.013
pos.	pos.	8-20	✗	3000	3000	✗	$\mathcal{B}(0.1, 0.9)$	0.087 0.060
off.	pos.	8-20	✗	3000	3000	✗	$\mathcal{B}(0.1, 0.9)$	0.095 0.065
pos.	pos.	8-20	✗	4000	4000	✗	✗	0.028 0.015
pos.	pos.	8-20	✗	4000	4000	✗	$\mathcal{B}(0.1, 0.9)$	0.081 0.015
off.	pos.	8-20	✗	4000	4000	✗	✗	0.031 0.014
off.	pos.	8-20	✗	4000	4000	✗	$\mathcal{B}(0.1, 0.9)$	0.084 0.014
Task: De-Noising / Filtering								
input	output	obs.	pred.	#samples	#epochs	noise	missing	ADE σ_{ADE}
pos.	pos.	8-20	✗	4000	4000	$\mathcal{N}(0, 1^2)$	✗	0.126 0.049
pos.	pos.	8-20	✗	4000	4000	$\mathcal{N}(0, 1^2)$	$\mathcal{B}(0.1, 0.9)$	0.165 0.071
offs.	pos.	8-20	✗	4000	4000	$\mathcal{N}(0, 1^2)$	✗	0.148 0.055
offs.	pos.	8-20	✗	4000	4000	$\mathcal{N}(0, 1^2)$	$\mathcal{B}(0.1, 0.9)$	0.222 0.137
Task: Prediction								
input	output	obs.	pred.	#samples	#epochs	noise	missing	ADE σ_{ADE}
pos.	pos.	8-14	6-12	4000	4000	$\mathcal{N}(0, 1^2)$	✗	0.809 0.514
pos.	pos.	8-14	6-12	4000	4000	$\mathcal{N}(0, 1^2)$	$\mathcal{B}(0.1, 0.9)$	0.920 0.422
offs.	pos.	8-14	6-12	4000	4000	$\mathcal{N}(0, 1^2)$	✗	1.186 0.583
offs.	pos.	8-14	6-12	4000	4000	$\mathcal{N}(0, 1^2)$	$\mathcal{B}(0.1, 0.9)$	1.221 0.734

Table 1: Results for a comparison between several trained MissFormer models for different inference tasks. The inference tasks are reconstruction, filtering and prediction. Reconstruction: In case the observations is noise-free, the task is to reproduce the trajectories. Filtering: Here positional observation noise is added and the model has to filter out this noise to generate noise-free trajectories. Prediction: Future object locations are inferred from noisy, observed trajectories.

the outputs of the model are positions, and the inputs are varied between using positions or using offsets to infer positions by path integration.

The results show that the MissFormer is able to successfully in-attend to the missing tokens and successfully only uses the remaining inputs for conditioning. The difference between the model's estimate without missing observation and a missing probability $\mathcal{B}(0.1, 0.9)$ is very low. Of course, there is a drop in performance which can best be seen when looking at the reconstruction task.

But the model can there basically learn a trivial solution of the identity of the input. Even when a trivial solution exists for using positions as inputs, the error does not drop to zero. Positional trajectory data can be seen as some sort of an increasing trend that cannot be fully captured by using non-linear activation functions. However, deep networks can achieve outputs greater than the bound of single activation functions, but they can saturate at minimum or maximum values, particularly for trending input data. For all inference tasks, but in particular for reconstruction, the model requires enough variation and training time to handle missing data. At first, the model simply reproduces placeholder values as outputs. By increasing the number of provided samples, the MissFormer starts to better route the information to compensate for outages. When the model has to additionally compensate for noise, the difference between missing and no-missing decreases. Here, the MissFormer has to generalize and filter out the noise. Thus, input identity mapping cannot be applied. When switching to a prediction task by replacing the last inputs with missing tokens, the difference between the models' estimates of fully observed and missing data decreases further.

In the experiments for all tasks, using positions as inputs works slightly better. The increased modeling effort is compensated by providing more variation during training. Since the error by using path integration for estimating the true position is propagated, the result is comprehensible. Further, in the context of a *dynamical* system, only observing offsets is an unobservable system where it is impossible to identify the initial condition uniquely. Thus, the error in the first positional estimate cannot be compensated. However, without missing observations and low positional observation noise, choosing offsets over positions has shown superior results on public trajectory prediction benchmarks ([2]).

For these experiments only one attention head ($N_{\text{head}} = 1$) and one attention layer ($N_{\text{layer}} = 1$) is used. Firstly, this allows a better understanding of the resulting attention because the attention filter directly shows what input information is used to encode a current trajectory. Secondly, compared to a *nature language processing* (NLP) or a vision task, single trajectory processing requires no attention to several aspects of the input data (e.g., a second attention filter on context information in the background). Some exemplary estimates from the MissFormer with corresponding attention filters are depicted in Figure 2. The values of the attention filter are color-coded (0 ▬→ ▬ max.). Here, results for the prediction tasks are shown where the last inputs are purposely missing tokens. The time steps where the input data is missing are marked with a cross and missing input indexes are shown above the attention filters. The input length k varies between 8 and 14 for a maximum length of $k_{max} = 20$ with a missing probability of $\mathcal{B}(0.1, 0.9)$.

The shown examples highlight several things. Firstly and most importantly, it can be seen that the models learned to in-attend to missing observation and to encode the trajectories based on the other inputs. Secondly, that the attention filters do not necessarily follow the typical look of high values along the diagonal as in an *NLP* task. This can be explained by the fact that there are many

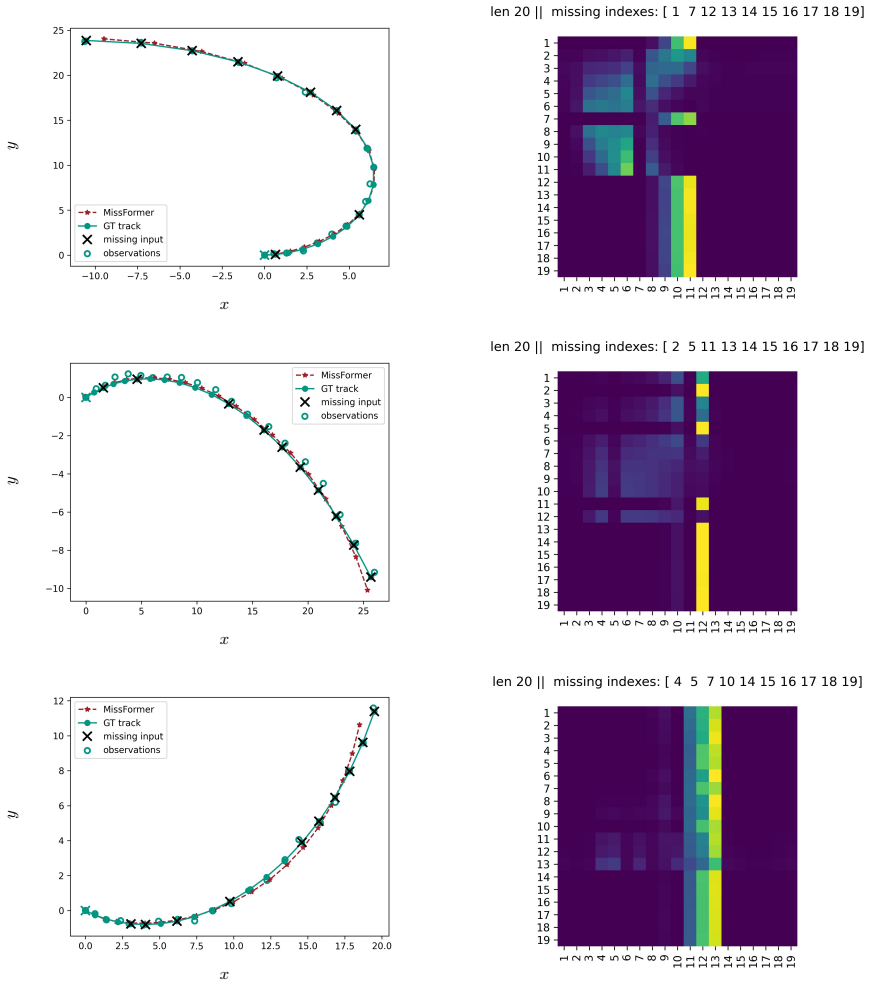

Fig. 2: (Left) Some exemplary estimates from the MissFormer with missing observations. The time steps where the input data is missing are marked with an cross. (Right) Corresponding attention filters. The attention filter values are color-coded (0 ■ → ■ max.)

possibilities for trajectory generation from the given inputs although solutions with predominantly high values along the diagonal can also result. Further, the examples show how the MissFormer filters out the noise and estimates relatively smooth outputs. And lastly they demonstrate, how prediction is done mainly relying on the last inputs, which corresponds to the common assumption that the last observations mainly influence motion. In summary, when providing the MissFormer with enough variation in terms of diversity and amount of training

data, the model can handle missing observations and ignore placeholder values provided as missing tokens.

3.2 Real-World Data

For real-world data and comparing the model to recent reference models, the publicly available *BIWI* [27] and *UCY* [22] datasets are used. Combined, these datasets contain five sequences from an oblique view capturing scenes with pedestrians in a shopping street and on a university campus. As mentioned before, most reference models cannot handle mission observation and this problem is not considered in their corresponding evaluation. For the sake of completeness and in order to compare the approach to others, we follow the common trajectory prediction protocol. So, evaluation is done by leaving one-out cross-validation for the 5 sequences. For conditioning, a fixed-length, fully observed trajectory of 8 points ($3.2s$) is provided before predicting 12 points ($4.8s$) into the future. The average displacement error (ADE) and the final displacement error (FDE) are used as error metrics. The ADE is defined as the average L2 distance between ground truth and the prediction over all predicted time steps and the FDE is defined as the L2 distance between the predicted final position and the actual final position.

					ADE/FDE in meters					
Approach	cues			model type	dataset					average
	traj.	social	scene		*BIWI:ETH*	*BIWI:Hotel*	*UCY:Univ*	*UCY:Zara1*	*UCY:Zara2*	
Linear interpolation	✓	✗	✗	classic	1.33/2.94	0.39/0.72	0.82/1.59	0.62/1.21	0.77/1.48	0.79/1.59
LSTM	✓	✗	✗	RNN	1.09/2.94	0.86/1.91	0.61/1.31	0.41/0.88	0.52/1.11	0.70/1.52
GAN (Ind.)[13]	✓	✗	✗	RNN	1.13/2.21	1.01/2.18	0.60/1.28	0.42/0.91	0.52/1.11	0.74/1.54
Social-LSTM [1]	✓	✓	✗	RNN	1.09/2.35	0.79/1.76	0.67/1.40	0.47/1.00	0.56/1.17	0.72/1.54
Social-Att. [36]	✓	✓	✗	RNN	0.39/3.74	0.29/2.64	0.33/3.92	0.20/0.52	0.30/2.13	0.30/2.59
Trajectron++[31]	✓	✓	✓	RNN	0.50/1.19	0.24/0.59	0.36/0.89	0.29/0.72	0.27/0.67	0.34/0.84
TCN [24]	✓	✗	✗	TCN	1.04/2.07	0.59/1.17	0.57/1.21	0.43/0.90	0.34/0.75	0.59/1.22
TF [11]	✓	✗	✗	*transformer*	1.03/2.10	0.36/0.71	0.53/1.32	0.44/1.00	0.34/0.76	0.54/1.17
MissFormer (ours)	✓	✗	✗	*transformer*	0.99/1.94	0.36/0.89	0.51/1.29	0.43/0.89	0.34/0.74	0.53/1.15

Table 2: Results for a comparison between the Missformer and a selection of recent prediction models following the single trajectory deterministic protocol. The prediction is done for 12 time steps into the future conditioned on 8 observations. Results are partly taken from [11, 31, 24]

The results are summarized in Table 2. In the comparison, a collection of recent approaches is considered where in terms of models relying solely on *trajectory cues* at least one reference approach from the basic concepts of deep sequential trajectory processing and one classic approach is included (see column model type). The best performing models incorporate additional *scene cues* (e.g., semantic segmentation), *social cues* (e.g., interactions with other pedestrians) or both. When considering only *trajectory cues*, the MissFormer achieves a better or similar performance. Without any outage in conditioning trajectory, the *transformer* model of [11] and our MissFormer model are very similar.

Whereas Giuliari et al. utilize an encoder-decoder network with offsets as inputs and outputs, we use an encoder-only model with positional in- and outputs. To counter the lesser modeling effort of offset data, we pre-train the model on a diverse set of synthetically generated trajectories covering all types of prototypical pedestrian motion patterns. Therefore, the distribution and settings from section 3.1 for synthetic trajectory generation are adapted to match the underlying data better. For example, the frame rate is set to $2.5 fps$ and random agents are sampled from a Gaussian distribution according to a preferred pedestrian walking speed [33] ($\mathcal{N}(1, 38^{m}/_{s}, (0.37^{m}/_{s})^2)$). The model is pre-trained on a diverse set of 4000 synthetic trajectories for 4000 epochs. For the first half of training, the full trajectories are provided. Then, corresponding to the prediction length, the last inputs are replaced with missing tokens. Here, the number of heads and attention layer is set to 2 ($N_{\text{head}} = 2$, $N_{\text{layer}} = 2$) and the model dimension is set to $d_{\text{model}} = 256$. However, the achieved results for these datasets are very similar and the scope of this paper aims at further exploring the *transformers*' ability to deal with missing observations. On the *BIWI* [27] and *UCY* [22] datasets, there is no clearly best-performing individual *trajectory cues*-based model. Overall, different models partly require different concepts for improving their performance or overcoming shortcomings. The presented results show that *transformers* are a good choice for estimating trajectories and offer an built-in concept of dealing with missing inputs.

4 Conclusion

In this paper, a *transformer*-based approach for handling missing observations has been presented. The *transformers*' built-in attention mechanisms in combination with positional encoding is analyzed in terms of exploring the remaining inputs for inference with outages. By providing encoded missing information (*missing tokens*), the model is encouraged to learn that specific values represent missing. The presented results show that the model can in-attend to the placeholder values and successfully route the information from the remaining inputs to infer a full trajectory. The abilities of the approach are demonstrated on synthetic data and real-world data reflecting prototypical object tracking scenarios.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social LSTM: Human Trajectory Prediction in Crowded Spaces. In: Conference on Computer Vision and Pattern Recognition (CVPR). pp. 961–971 (2016) 2, 10
2. Becker, S., Hug, R., Hübner, W., Arens, M.: RED: A simple but effective Baseline Predictor for the TrajNet Benchmark. In: European Conference on Computer Vision (ECCV) Workshops. Springer International Publishing (2018) 2, 6, 8
3. Becker, S., Hug, R., Hübner, W., Arens, M., Morris, B.T.: Handling Missing Observations with an RNN-based Prediction-Update Cycle (2021) 2
4. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006) 6

5. Brownlee, J.: Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future. Jason Brownlee (2017) 2
6. Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent Neural Networks for Multivariate Time Series with Missing Values. Scientific Reports (SREP) **8**(6085) (2018) 2
7. Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., Bengio, Y.: A Recurrent Latent Variable Model for Sequential Data. In: Advances in Neural Information Processing Systems (NeurIPS) (2015) 1
8. De Boor, C.: A practical guide to splines; rev. ed. Applied mathematical sciences, Springer, Berlin (2001) 2
9. Dinh, L., Krueger, D., Bengio, Y.: NICE: non-linear independent components estimation. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings (2015) 6
10. Donahue, J., Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term Recurrent Convolutional Networks for visual Recognition and Description. In: Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2015) 1
11. Giuliari, F., Hasan, I., Cristani, M., Galasso, F.: Transformer Networks for Trajectory Forecasting. In: International Conference on Pattern Recognition (ICPR) (2020) 2, 10
12. Graves, A., Mohamed, A., Hinton, G.: Speech recognition with deep recurrent neural networks. In: International Conference on Acoustics, Speech and Signal Processing. pp. 6645–6649 (2013) 1
13. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In: Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2018) 10
14. Hug, R., Becker, S., Hübner, W., Arens, M.: On the Reliability of LSTM-MDL Models for Pedestrian Trajectory Prediction. In: Representations, Analysis and Recognition of Shape and Motion from Imaging Data (RFMI). Savoie, France (2017) 2
15. Hug, R., Becker, S., Hübner, W., Arens, M.: A complementary trajectory prediction benchmark. In: ECCV Workshop on Benchmarking Trajectory Forecasting Models (BTFM) (2020) 3
16. Hug, R., Becker, S., Hübner, W., Arens, M.: Quantifying the complexity of standard benchmarking datasets for long-term human trajectory prediction. IEEE Access **9**, 77693–77704 (2021) 6
17. Hug, R., Hübner, W., Arens, M.: Introducing probabilistic bézier curves for n-step sequence prediction. Proceedings of the AAAI Conference on Artificial Intelligence **34**(06), 10162–10169 (Apr 2020) 6
18. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014) 6
19. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations (ICLR) (2015) 6
20. Kothari, P., Kreiss, S., Alahi, A.: Human Trajectory Forecasting in Crowds: A Deep Learning Perspective. arXiv preprint arXiv:2007.03639 (2020) 2
21. Kreindler, D., Lumsden, C.J.: The effects of the irregular sample and missing data in time series analysis. Nonlinear dynamics, psychology, and life sciences **10 2**, 187–214 (2006) 2

22. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Computer Graphic Forum **26**(3), 655–664 (2007) 3, 6, 10, 11
23. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2019) 6
24. Nikhil, N., Morris, B.: Convolutional neural network for trajectory prediction. In: The European Conference on Computer Vision (ECCV) Workshops (2018) 2, 10
25. Parveen, S., Green, P.: Speech Recognition with Missing Data using Recurrent Neural Nets. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 1189–1195. MIT Press (2002) 2
26. Paszke, A., et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 8024–8035. Curran Associates, Inc. (2019) 6
27. Pellegrini, S., Ess, A., Schindler, K., van Gool, L.: You'll never walk alone: Modeling social behavior for multi-target tracking. In: International Conference on Computer Vision (ICCV). pp. 261–268. IEEE (2009) 3, 6, 10, 11
28. Rasouli, A.: Deep Learning for Vision-based Prediction: A Survey. arXiv preprint arXiv:2007.00095 (2020) 2
29. Rudenko, A., Palmieri, L., Herman, M., Kitani, K.M., Gavrila, D.M., Arras, K.O.: Human Motion Trajectory Prediction: A Survey. The International Journal of Robotics Research (2020) 2
30. Saleh, K.: Pedestrian Trajectory Prediction using Context-Augmented Transformer Networks. arXiv preprint arXiv:2012.01757 (2020) 2
31. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. (eds.) Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVIII. Lecture Notes in Computer Science, vol. 12363, pp. 683–700. Springer (2020) 10
32. Schafer, J.L., Graham, J.W.: Missing data: Our view of the state of the art. Psychological Methods **7**(2), 147–177 (2002) 2
33. Teknom, K.: Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model. Ph.D. thesis, Tohoku University (2002) 11
34. Tresp, V., Briegel, T.: A Solution for Missing Data in Recurrent Neural Networks with an Application to Blood Glucose Prediction. In: International Conference on Neural Information Processing Systems (NeurIPS). pp. 971–977. MIT Press, Cambridge, MA, USA (1997) 2
35. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 5998–6008 (2017) 4
36. Vemula, A., Muelling, K., Oh, J.: Social attention: Modeling attention in human crowds. In: International Conference on Robotics and Automation (ICRA). pp. 1–7 (2018) 10
37. Xu, K., Ba, J., Kiros, R., K.Cho, Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 37, pp. 2048–2057. PMLR, Lille, France (2015) 1