TRAJGRU-ATTENTION-ODE: NOVEL SPATIOTEMPORAL PREDICTIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

To perform the long-term spatiotemporal sequence prediction (SSP) task with irregular time sampling assumptions, we build the sequence-to-sequence models based on the Trajectory Gated Recurrent Unit (TrajGRU) network and our proposed deep learning modules. First, we design a novel attention mechanism, namely Motion-based Attention (MA), and insert it into the TrajGRU network to create the TrajGRU-Attention model. In particular, the TrajGRU-Attention model can alleviate the impact of the vanishing gradient, which leads to the blurry effect in the long-term predictions and handle both regularly sampled and irregularly sampled time series. Second, leveraging the advances in Neural Ordinary Differential Equation (NODE) technique, we proposed the TrajGRU-Attention-ODE model, which can be applied in continuous-time applications. To evaluate the performance of the proposed models, we select three available spatiotemporal datasets based on the complexity level, including the MovingMNIST, MovingMNIST++, and KTH Action. Our models outperform the state-of-the-art NODE model and generate better results than the standard TrajGRU model for SSP tasks with different circumstances of time sampling.

1 INTRODUCTION

Spatiotemporal Sequence Predictive (SSP) learning has been an essential research topic and plays a critical role in many real-world applications, attracting widespread attention in recent years. In real life, spatiotemporal data often occur around us whenever data are collected across space and time (Wang et al., 2020). Overall, SSP learning refers to the approaches used for studying and processing these data for practical tasks ranging from many daily practical applications (e.g., motion prediction (Wu et al., 2017), traffic flow prediction (Luo et al., 2019)) to many global-scale applications (e.g., precipitation and weather hazard nowcasting (Shi et al., 2015; 2017)).

With SSP learning tasks, researchers have studied and proposed various machine learning algorithms and advanced deep learning models to capture the spatial and temporal properties (Fang et al., 2021). Typically, there are two types of time sampling in temporal analysis: regular time sampling (i.e., the time interval is assumed to be constant) and irregular time sampling (i.e., the time interval is considered arbitrary). Although some sequence models usually apply a standard time sampling variable and use the concept of frame rate to sample the data, there are numerous scenarios in which this assumption is not practical. First, not all spatiotemporal datasets follow this sampling type; some research areas are interested in data recorded at arbitrary time steps, such as medical, economics, and climate modelling (Foley, 2010; Kanakidou et al., 2005). Besides discrete series, some spatiotemporal data in continuous space are applied for continuous-time physical models (Zhong et al., 2019), biology applications (Pinckaers & Litjens, 2019) and telecommunications (Huai et al., 2022). Second, an observed time series can lose data points at some time because of hardware errors, noise, and outside effects on machines. Furthermore, handling time series at arbitrary time steps can help systems be flexibly used for numerous applications and save time and resources.

Recent studies proposed approaches to solve complicated SSP tasks with fixed frame rates, and these models are designed to address the vanishing gradient issue (Shi et al., 2015; 2017; Wang et al., 2017; Su et al., 2020). Other works used the concept of irregular time sampling in continuous-time video generation and prediction (Chen et al., 2018; Yildiz et al., 2019; De Brouwer et al., 2019).

However, those continuous-time models, as described in Section 2, experimented with relatively simple datasets; thus, they are not practical for real-world applications with complicated datasets.

Based on those observations, we propose two novel sequence-to-sequence predictive models which can work effectively with the overall SSP task with different time sampling assumptions: TrajGRU-Attention and TrajGRU-Attention-ODE. In the two models, we presented a novel attention module, namely Motion-based Attention (MA), which can be flexibly dropped into a Recurrent Neural Network (RNN) to boost memory ability and help the model handle irregularly sampled time series. In the TrajGRU-Attention-ODE, besides the MA modules, we design a novel RNN, namely Trajectory Gated Recurrent Unit integrating Ordinary Differential Equation techniques (TrajGRU-ODE), which can perform continuous-time spatiotemporal prediction tasks. According to extensive experiments on various video datasets, the proposed models can generate predictions properly with high accuracy. Furthermore, the proposed models outperform state-of-the-art ODE-based approaches for SSP tasks with different circumstances of time sampling.

2 Related Work

Convolutional Recurrent Neural Networks (ConvRNNs). ConvRNNs are proposed based on Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). Convolutional Long-short Term Memory (ConvLSTM) (Shi et al., 2015), and Convolutional Gated Recurrent Unit (ConvGRU) (Ballas et al., 2015; Siam et al., 2017) are two prior variants in this group. Based on the fundamental idea of the learnable spatial transformer module, Shi et al. (2017) proposed the Trajectory Gated Recurrent Unit (TrajGRU) model that can learn the location-variant structure for the recurrent connection and capture the spatiotemporal corrections effectively. Wang et al. (2017; 2018; 2022) proposed PredRNN models with novel ConvRNN layers for the SSP task. Essentially, the PredRNN cell is the extended version of the ConvLSTM unit containing dual memories: the temporal and spatial memory states. As a result, the recurrent layer can store more knowledge of the observed series, and the system can alleviate the vanishing gradient problem. In addition, the ConvLSTM and ConvGRU models are considered first-order Markovian models because their memory cell takes only one previous hidden state at a time to generate the spatiotemporal representation state (Soltani & Jiang, 2016). This causes an intrinsic difficulty in learning spatiotemporal correlations in the long-term forecasting task. Therefore, Su et al. (2020) extended the ConvLSTM to a higher-order network that can combine several previous states and feed them into the memory mechanism at a time step, namely Convolutional Tensor-Train Long Short-Term Memory (Conv-TT-LSTM).

Neural Ordinary Differential Equations (NODEs). For the SSP task, most of the recent sequence models follow the regular time sampling assumption. Therefore, the learning ability of these models is limited to specific datasets with a constant time gap between adjacent data frames. As a result, these models cannot perform irregularly sampled or continuous-time prediction tasks. A temporary solution to this problem is to preprocess the sequences by removing some intermediate data points of irregularly sampled sequences. However, this will significantly impact the training and testing processes of the whole system because some data are ignored, especially with sparse datasets. Consequently, the preprocessing step is not optimal for the given task. In that situation, NODEs have been proposed as promising methods for irregularly sampled and continuous-time prediction problems. The group of ODE-based models is a new family of deep neural network models (Chen et al., 2018; 2020). In the SSP field, the evolution of a phenomenon can be represented by differential equations, and these equations can be solved by mathematical methods. Based on this observation, ODE-based models have been designed, and specific neural networks have been proposed to implement the differential equations describing the changing processes of representation states in the latent field. For instance, Rubanova et al. (2019) proposed a generative latent function continuous-time model (Latent-ODE), considered one of the first NODEs implemented for time series prediction tasks. Yildiz et al. (2019) proposed Deep generative second-order ODEs with Bayesian neural networks (ODE2VAE) for autoencoding irregularly sampled time series. Although the ODE-based models above can handle irregularly sampled data and learn the continuous-time state dynamics, their performances were investigated with relatively simple datasets, such as image rotation and bouncing ball (Rubanova et al., 2019; Yildiz et al., 2019; De Brouwer et al., 2019). Park et al. (2021) proposed a continuous-time video generation model using the ODE technique, namely Vid-ODE, which shows some promising performances with real-world videos. Vid-ODE essentially combined the standard ConvGRU network, NODE techniques, and the pixel-level composition technique.

3 MODELS

3.1 PROBLEM STATEMENT

With the given spatiotemporal sequence input, the main objective of the SSP task is to forecast the output sequence at future time steps. To do that, sequence-to-sequence models are typically implemented with an encoding-decoding structure. Simply put, the encoder receives, processes the input sequence, and sends practical information to the decoder, generating output predictions. In this work, the main SSP problem formulation can be expressed as

$$\begin{cases} \mathbf{X} = (\mathbf{X}_{input}, \mathbf{X}_{target}) = (\mathbf{X}(t_1), ..., \mathbf{X}(t_N); \mathbf{X}(t_{N+1}), ..., \mathbf{X}(t_{N+K})), \\ \mathbf{X}_{prediction} = f_{decoder}(f_{encoder}(\mathbf{X}_{input})), \\ min(f_{loss}(\mathbf{X}_{prediction}, \mathbf{X}_{target})), \end{cases} \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{S \times C \times H \times W}$ indicates the examined sequence, consisting of S data frames at time steps $\{t_1, t_2, ..., t_S\}$; $\mathbf{X}_{input} \in \mathbb{R}^{N \times C \times H \times W}$, $\mathbf{X}_{target} \in \mathbb{R}^{K \times C \times H \times W}$ are the input and target subsequences, respectively (i.e., $S = N + K$); $\mathbf{X}_{prediction} \in \mathbb{R}^{K \times C \times H \times W}$ presents the output prediction of the model; $f_{encoder}$, $f_{decoder}$ represent the encoder and the decoder of the model, respectively; f_{loss} indicates the main loss function, as illustrated in Appendix A.2.

3.2 ENCODING-DECODING STRUCTURE

In this work, a model architecture is developed from the stacked style. In the stacked ConvLSTM architecture (Shi et al., 2015), multiple recurrent layers are stacked: the lowest-level layer is the bottom layer, and the highest-level one is the top layer. During the recurrent computation of a specific layer, each recurrent neural unit's output hidden state is sent to the higher-level layer's neural unit. Therefore, the neural unit of the top layer at a specific time step might forget some practical information coming from the neural unit of the bottom layer at the previous time step when the system uses a high number of stacked layers (Wang et al., 2017). Consequently, the final representation state of each recurrent layer in the encoder cannot contain sufficient knowledge of a long-range series; thus, the decoder cannot learn the spatiotemporal features effectively.

Based on the above observations, we design a novel architecture combining the stacked style, zigzag connection, and reverse layers at the decoder (see Figure 1). To alleviate the vanishing gradient effect, we take the idea of the spatiotemporal memory flow of Wang et al. (2017) to design a zigzag connection, which creates a shortcut to communicate between the top and bottom recurrent layers. This zigzag shortcut can boost the model's memory ability by increasing the network's depth (i.e., the connection between units of top and bottom layers allows the model to generate more representations between two timestamps). Furthermore, this architecture enables the model to use more stacked intermediate layers. Consequently, it improves the system's overall performance. At the encoder, the operations of neural units when applying the zigzag connection can be formulated as

$$\begin{aligned} \mathbf{H}_{zz}(t_i) &= leakyRELU\left(\mathbf{W}_{upsampling} * \mathbf{H}_{top}(t_i)\right), \\ \mathbf{Z}_{zz}(t_i) &= \sigma\left(\mathbf{W}_h * \mathbf{H}_{top}(t_i) + \mathbf{W}_{zz} * \mathbf{H}_{zz}(t_i)\right), \\ \mathbf{H}^{new}_{bottom}(t_i) &= \mathbf{H}_{bottom}(t_i) + \mathbf{Z}_{zz}(t_i) \circ \mathbf{H}_{zz}(t_i), \end{aligned} \tag{2}$$

and vice versa at the decoder, the information flows are reversed so that we have the following equations for the zigzag connection

$$\begin{aligned} \mathbf{H}_{zz}(t_i) &= leakyRELU\left(\mathbf{W}_{downsampling} * \mathbf{H}_{bottom}(t_i)\right), \\ \mathbf{Z}_{zz}(t_i) &= \sigma\left(\mathbf{W}_h * \mathbf{H}_{top}(t_i) + \mathbf{W}_{zz} * \mathbf{H}_{zz}(t_i)\right), \\ \mathbf{H}^{new}_{top}(t_i) &= \mathbf{H}_{top}(t_i) + \mathbf{Z}_{zz}(t_i) \circ \mathbf{H}_{zz}(t_i). \end{aligned} \tag{3}$$

Here, $\mathbf{Z}_{zz}(t_i)$ indicates the update gate at the time step t_i, which decides how much information of the zigzag state $\mathbf{H}_{zz}(t_i)$ will be obtained; $\mathbf{W}_{upsampling}$ and $\mathbf{W}_{downsampling}$ refer to the learnable parameters of the up-sampling and down-sampling layers, respectively; $\mathbf{W}_h$, $\mathbf{W}_{zz}$ refer to the learnable parameters of the convolutional layers applied to extract the features of the hidden states. Consequently, the new state $\mathbf{H}^{new}_{bottom}(t_i)$ at the encoder and $\mathbf{H}^{new}_{top}(t_i)$ at the decoder can be obtained. This work divides the model building blocks into two groups: the central and connection building blocks.

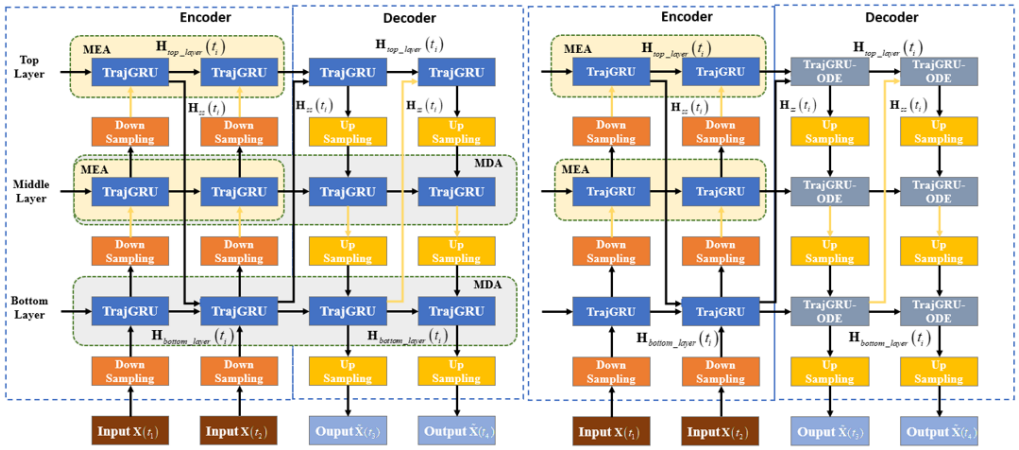

Figure 1: **(Left)** Illustration of TrajGRU-Attention model. **(Right)** Illustration of TrajGRU-Attention-ODE model. In the figure, the encoder and decoder use three layers to predict two future frames $\tilde{\mathbf{X}}(t_3), \tilde{\mathbf{X}}(t_4)$ given two input frames $\mathbf{X}(t_1), \mathbf{X}(t_2)$.

In the group of connection blocks, besides up-sampling and down-sampling blocks, we present MA modules: Motion-based Encoding Attention (MEA) and Motion-based Decoding Attention (MDA) modules used in the TrajGRU-Attention model. In the TrajGRU-Attention-ODE, the MDA modules are not used, and TrajGRU blocks are replaced with the TrajGRU-ODE blocks.

3.3 TRAJGRU-ATTENTION

Typically, the current representation state can be computed using the current data input and the previous hidden state at each timestamp when using the traditional recurrent computation. Since the updating computation at each neural cell takes into account the short-term dependencies of two adjacent data frames, the long-term dependencies between the current input and all previous series are not effectively captured (Vaswani et al., 2017). This issue leads to the vanishing gradient problem reducing the reconstruction quality, especially with long-range examined sequences.

In the time domain, the evolution and changing processes over time of the spatiotemporal phenomena intrinsically involve the motion characteristics, which are mathematically described in terms of position, velocity, acceleration, and time variables. Therefore, we leverage the characteristic motion analysis as the key idea to build the MA module. The main objective of this mechanism is to help the system look back on prior knowledge and pay more attention to valuable parts based on the motion characteristics of the observed hidden states and input frames. Moreover, by learning the representation states of the motions over time, the system is capable of handling data points sampled at arbitrary given timestamps. In this work, the MA module is built as a connection building block which can be flexibly inserted into different RNNs. According to the encoding-decoding model structure, there are also two attention mechanisms: the MEA and MDE. These mechanisms create the parallelization computation inside the recurrent operation of central blocks (e.g., TrajGRU).

Motion-based Encoding Attention (MEA) mechanism. From the side of the encoder, the MEA mechanism is proposed for performing the parallelization computations on the input sequence. The entire step-by-step of applying the MEA is described as follows.

First, at the time step t_Q of the input sequence, we derive the formulas used to compute additional states representing the motion characteristics. By assuming the set of input frames $\{\mathbf{X}(t_i)\}_{i\leq Q, i\in\mathbb{N}}$ (i.e., the set length is Q) as the position information, we can define the set of velocity and acceleration states as follows

$$\{\mathbf{V}(t_i), \mathbf{A}(t_i)\}_{i\leq Q, i\in\mathbb{N}} : \begin{cases} \mathbf{V}(t_i) = f_V\left(\frac{\mathbf{X}(t_i)-\mathbf{X}(t_{i-1})}{t_i - t_{i-1}}\right), \\ \mathbf{A}(t_i) = f_A\left(\frac{\mathbf{V}(t_i)-\mathbf{V}(t_{i-1})}{t_i - t_{i-1}}\right), \\ \mathbf{A}(t_1) = \mathbf{V}(t_1) = \mathbf{X}(t_1). \end{cases} \tag{4}$$

In Equation 4, f_V, f_A indicate distinct networks: each includes a convolutional layer with a kernel size of (1×1) and an average pooling layer. Thus, the sets $\{\mathbf{V}(t_i), \mathbf{A}(t_i)\}_{i \leq Q, i \in \mathbb{N}}$ can be represented by tensors $\mathbf{V}_{t_Q}$ and $\mathbf{A}_{t_Q}$, respectively, with the same shape of $(Q \times B \times C \times 1)$ (where Q denotes the set length, B denotes the batch size, C denotes the number of channels of the state). The set of time steps $\mathbf{T}_{t_Q}$ is reshaped from $(Q \times B \times 1)$ to $(Q \times B \times C \times 1)$ by replicating the values.

Second, based on the computations of Luong et al. (2015), the formulation computing the correlation between the current input data $\mathbf{X}(t_Q)$ with its previous frames $\{\mathbf{X}(t_i)\}_{i \leq Q, i \in \mathbb{N}}$, is written as

$$\mathbf{C}_{t_Q} = Avg\left(\tanh\left(\mathbf{W}_{a1} * \mathbf{X}(t_Q) + \mathbf{W}_{a2} * \{\mathbf{X}(t_i)\}_{i \leq Q, i \in \mathbb{N}}\right)\right), \tag{5}$$

where $\mathbf{W}_{a1}$, $\mathbf{W}_{a2}$ represent the learnable weights of convolutional layers; Avg denotes the average pooling layer. In Equation 5; the term $\mathbf{W}_{a1} * \mathbf{X}(t_Q)$ is added to each element in the set of tensors $\mathbf{W}_{a2} * \{\mathbf{X}(t_i)\}_{i \leq Q, i \in \mathbb{N}}$; then they are stacked together. Consequently, four tensors $\mathbf{V}_{t_Q}$, $\mathbf{A}_{t_Q}$, $\mathbf{T}_{t_Q}$, $\mathbf{C}_{t_Q}$ are obtained with the same shape of $(Q \times B \times C \times 1)$, presenting four keys of the attention. In particular, those keys are expected to help the model make a decision about how much attention to pay to each input data frame. Furthermore, since those keys contain information on the changing rates (i.e., the evolution) regardless of regular or irregular sampling assumptions, the model can learn and recognize spatiotemporal representations at specified time steps. In the next step, based on those keys, we compute the alignment scores, which indicate the collection of weights to assign to each element in the set of input frames $\{\mathbf{X}(t_i)\}_{i \leq Q, i \in \mathbb{N}}$

$$\mathbf{S}_{t_Q} = \mathrm{softmax}\left(f_{combine}\left(\mathbf{V}_{t_Q}, \mathbf{A}_{t_Q}, \mathbf{T}_{t_Q}, \mathbf{C}_{t_Q}\right)\right), \tag{6}$$

where $f_{combine}$ represents the linear layer with the number of input channels being $(4 \times C)$ and the number of output channels being (C). Finally, we assign this score $\mathbf{S}_{t_Q}$ to the set of input frames $\{\mathbf{X}(t_i)\}_{i \leq Q, i \in \mathbb{N}}$ to generate the attention state and the new input at time step t_Q

$$\begin{aligned} \mathbf{H}^a(t_Q) &= \mathbf{S}_{t_Q} \circ \{\mathbf{X}(t_i)\}_{i \leq Q, i \in \mathbb{N}}, \\ \mathbf{X}^{new}(t_Q) &= \mathbf{W}_{1\times1} * \{\mathbf{X}(t_Q), \mathbf{H}^a(t_Q)\}, \end{aligned} \tag{7}$$

where "$\circ$" denotes the Hadamard product, "$*$" denotes the convolutional operator; $\mathbf{W}_{1\times1}$ represents the learnable weights of the convolutional layers with a kernel size of (1×1). Consequently, the new input intrinsically obtains the knowledge of data frames stretching from the beginning timestamp t_1 to the current instant t_Q. Therefore, the MEA helps the encoder possesses a long-term memory and an ability to learn the irregularly sampled time series.

Motion-based Decoding Attention (MDA) mechanism. From the perspective of the decoder, MDA is proposed for making parallel computations on the collection of hidden state sequences instead of the input sequences in the case of the MEA. As described previously, the encoder's recurrent layers take the attention states generated by implementing the MEA on the input series to create the output hidden form for each timestamp. Since the decoder uses the representation states from the encoder to make predictions, the same attention mechanism is applied to the set of hidden states in the decoder. Therefore, the entire attention operations are straightforward, and the information flows are distributed following a common recurrent computation.

Overall, a MA block has a set of learnable parameters coming from five convolutional layers (i.e., f_V, f_A, $\mathbf{W}_{a1}$, $\mathbf{W}_{a2}$, $\mathbf{W}_{1\times1}$) and one linear layer (i.e., $f_{combine}$). The details of MA module settings for the entire model are illustrated in Appendix A.3. Finally, the TrajGRU-Attention model is constructed by combining the TrajGRU networks and the MA module setting (see Figure 1).

3.4 TRAJGRU-ATTENTION-ODE

In the TrajGRU-Attention-ODE structure, the TrajGRU and MDA blocks are replaced by the TrajGRU-ODE network. This neural network implements an internal loop to perform the recurrent computation like many conventional RNNs. However, instead of just using the pair of the input $\mathbf{X}(t_i)$ at the time step t_i and the previous hidden state $\mathbf{H}(t_{i-1})$, a new intermediate state $\mathbf{D}(t_i)$ is defined, namely the difference representation; then, it is inserted into the recurrent computation alongside this pair. In particular, this state is computed using the couple of latent continuous-time states $\{\mathbf{H}_{ode}(t_{i-1}), \mathbf{H}_{ode}(t_i)\}$ which is generated by the ODE solver component. Hence, this state can emphasize the distinctness and the evolution of data from time step t_{i-1} to t_i. Eventually, the

formulation to compute $\mathbf{D}(t_i)$ can be rewritten as follows

$$\begin{aligned}
&\{\mathbf{H}_{ode}(t_i)\}_{i=1,...,T}\} = \text{ODEsolver}(f_{ode}(\mathbf{H}_{ode}, \mathbf{W}_\theta, t), \mathbf{H}_{ode}(t_0), \{t_0, ..., t_T\}, \text{method}_{\text{ODE}}), \\
&\mathbf{D}_{t_i} = \mathbf{W}^{ode}_{combine} * \{\mathbf{H}_{ode}(t_{i-1}), \mathbf{H}_{ode}(t_i)\},
\end{aligned} \tag{8}$$

where $\mathbf{W}^{ode}_{combine}$ denotes the convolutional layer with a kernel size of 1×1. More specifically, we will examine a simple task to generate T unseen data frames at T random timestamps using the given initial knowledge. First, the integrated ODE solver module is defined by a neural network that played a role in the ODE function and a numerical integration method (Chen et al., 2018). In this work, an ODE function is represented by a sequence of convolutional and tanh activation layers. Then, this ODE solver will take the initial state to generate the set of continuous-time hidden states $\{\mathbf{H}_{ode}(t_0), ..., \mathbf{H}_{ode}(t_T)\}$ in the latent space at the expected timestamps from t_0 to t_T.

Although those states obtain the continuous-time dynamics, the ODE function (i.e., the ODE neural network) of the ODE solver module is insufficient to capture important spatial features and reconstruct the resulting outcomes with high sharpness simultaneously, especially when the sequence data contain different complicated spatiotemporal properties. Therefore, these latent continuous-time states are incorporated with the hidden states generated by the extended TrajGRU layer to represent learnable features more effectively. According to Equation 8, the state $\mathbf{D}(t_i)$ at time step t_i, allows the system to foresee how the examined state can evolve at this time step. After that, the extended TrajGRU will combine this state with other knowledge processed by the gated mechanism to reconstruct the resulting output. Therefore, the TrajGRU-ODE block (see Figure 2) can simultaneously learn the sequence's continuous-time and spatiotemporal dynamics. As a result, this technique can be applied to handle the irregularly sampled time series and perform continuous-time prediction tasks. The detailed operation formulations of the TrajGRU-ODE unit can be written as follows

$$\begin{aligned}
&\mathbf{U}(t_i), \mathbf{V}(t_i) = \gamma\left(\mathbf{X}(t_i), \mathbf{H}(t_{i-1})\right), \\
&\mathbf{Z}(t_i) = \sigma\left(\mathbf{W}_{xz} * \mathbf{X}(t_i) + \sum_{l=1}^{L} \mathbf{W}_{hz,l} * warp(\mathbf{H}(t_{i-1}), \mathbf{U}(t_i, l), \mathbf{V}(t_i, l)) + \mathbf{B}_z\right), \\
&\mathbf{R}(t_i) = \sigma\left(\mathbf{W}_{xr} * \mathbf{X}(t_i) + \sum_{l=1}^{L} \mathbf{W}_{hr,l} * warp(\mathbf{H}(t_{i-1}), \mathbf{U}(t_i, l), \mathbf{V}(t_i, l)) + \mathbf{B}_r\right), \\
&\tilde{\mathbf{H}}(t_i) = \tanh\left(\mathbf{W}_{xh} * \mathbf{X}(t_i) + \mathbf{R}(t_i) \circ \sum_{l=1}^{L} \mathbf{W}_{hh,l} * warp(\mathbf{H}(t_{i-1}), \mathbf{U}(t_i, l), \mathbf{V}(t_i, l)) + \mathbf{B}_h\right), \\
&\mathbf{H}_{TrajGRU}(t_i) = \mathbf{Z}(t_i) \circ \mathbf{H}(t_{i-1}) + (1 - \mathbf{Z}(t_i)) \circ \tilde{\mathbf{H}}(t_i), \\
&\mathbf{Z}_{ode}(t_i) = \sigma\left(\mathbf{W}_{hzode} * \mathbf{H}_{TrajGRU}(t_i) + \mathbf{W}_{dzode} * \mathbf{D}(t_i) + \mathbf{B}_{zode}\right), \\
&\mathbf{R}_{ode}(t_i) = \sigma\left(\mathbf{W}_{hrode} * \mathbf{H}_{TrajGRU}(t_i) + \mathbf{W}_{drode} * \mathbf{D}(t_i) + \mathbf{B}_{rode}\right), \\
&\tilde{\mathbf{H}}_{ode}(t_i) = \tanh\left(\mathbf{W}_{hhode} * \mathbf{H}_{TrajGRU}(t_i) + \mathbf{R}_{ode}(t_i) \circ (\mathbf{W}_{dhode} * \mathbf{D}(t_i)) + \mathbf{B}_{hode}\right), \\
&\mathbf{H}(t_i) = \mathbf{Z}_{ode}(t_i) \circ \mathbf{H}_{TrajGRU}(t_i) + (1 - \mathbf{Z}_{ode}(t_i)) \circ \tilde{\mathbf{H}}_{ode}(t_i).
\end{aligned} \tag{9}$$

From the perspective of the TrajGRU operation, $\mathbf{X}(t_i) \in \mathbb{R}^{C_{in} \times H_{in} \times W_{in}}$, $\mathbf{H}(t_{i-1}) \in \mathbb{R}^{C_h \times H_h \times W_h}$ indicate the input data at the time step t_i and the previous hidden state, respectively; γ represents the neural network generating the continuous flow fields $\mathbf{U}(t_i)$, $\mathbf{V}(t_i)$ with L different connection links in the transition; $warp$ represents a function used to combine the sets of flow fields and the hidden state to generate an intermediate state with approximated location information; $\mathbf{Z}(t_i)$, $\mathbf{R}(t_i)$ indicate the update, reset gates, respectively. $\tilde{\mathbf{H}}(t_i)$ represents the memory state, which is modified according to the reset gate and information flows at each time step; $\mathbf{H}_{TrajGRU}(t_i)$ represents the output hidden state of the standard TrajGRU unit. In terms of the extended gates, the states $\mathbf{H}_{TrajGRU}(t_i)$ and $\mathbf{D}(t_i)$ will be the input data and the hidden state, respectively; $\mathbf{Z}_{ode}(t_i)$, $\mathbf{R}_{ode}(t_i)$ denote the additional update and reset gates; $\tilde{\mathbf{H}}_{ode}(t_i)$ represents the new memory state; eventually, the final output state $\mathbf{H}(t_i)$ contains both continuous-time and spatiotemporal features. The system has collections of learnable and trainable parameters for the optimizing process: $\mathbf{W}_k$, $\mathbf{B}_k$. Finally, "$*$", as previously, denotes the convolution operator; "$\circ$" denotes the Hadamard product; σ refers to the sigmoid function; $tanh$ refers to the tanh function.

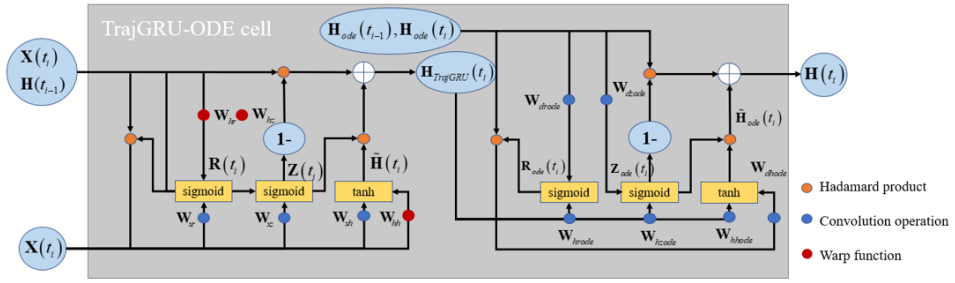

Figure 2: TrajGRU-ODE unit.

Overall, a TrajGRU-ODE network includes a TrajGRU network (described in Appendix A.1), three extended gates represented by six convolutional layers, an ODE network represented by two pairs of convolutional and $\tanh$ layers, and other activation layers. From the model structure and architecture perspective, the TrajGRU-Attention-ODE is designed in the same way as the TrajGRU-Attention (see Figure 1). The main difference between these two models is that the TrajGRU-ODE blocks replace the combination of TrajGRU blocks and MDA modules as the central blocks (i.e., central layers) in the decoder. Since the TrajGRU-ODE blocks themselves can generate continuous hidden states and reconstruct the outcomes at the expected time steps, the MEA modules are implemented at the encoder to generate the initial representation states.

4 Experiments

4.1 Implementation

This section implements the TrajGRU-Attention and TrajGRU-Attention-ODE models with three video datasets: the MovingMNIST, MovingMNIST++, and KTH Action. The detailed configuration of the proposed models is presented in Appendix A.5. For the training process, we use the combination of Mean Square Error (MSE), Mean Absolute Error (MAE), and Structural Similarity Index Measure (SSIM), as the loss function, as presented in Appendix A.2. To compare the performance of the proposed model, we implement the ConvGRU, TrajGRU, and Vid-ODE as the baseline models whose configurations are described in Appendix A.1.

4.2 Datasets

For data preprocessing, each examined sequence is divided into two subsequences (i.e., the input and the target), and then the sampling methods (i.e., regular and irregular sampling) are applied to each subsequence. The detailed information on sampling assumptions is presented in Appendix A.4.

MovingMNIST. Essentially, the MovingMNIST illustrates the movements of the handwriting MNIST digits in 64×64 frames. For data configuration, all data frames consist of two random MNIST digits; the input is ten frames long, the target is ten frames long, and the time interval between adjacent frames depends on the sampling assumption. The training set contains 10000 sequences, the validation set contains 2000 sequences, and the testing set contains 5000 sequences.

MovingMNIST++. Compared to the MovingMNIST dataset, each frame contains three MNIST digits, and more complicated motion patterns for the MNIST digits are added to all frames, such as random rotations, scale changes, and illumination changes (Shi et al., 2017). However, similar to the previous dataset, the same preprocessing and sampling methods are used to generate the input and target sequences for the training, validation, and testing sets.

KTH Action. KTH Action is a video database describing six types of human actions: walking, jogging, running, boxing, hand waving, and clapping (Schuldt et al., 2004). For data preprocessing, we use 255 videos for training and 144 videos for validating and testing. Similar to the case of the MovingMNIST, the input and target lengths are ten, and the two time-sampling methods are also applied for the SSP task.

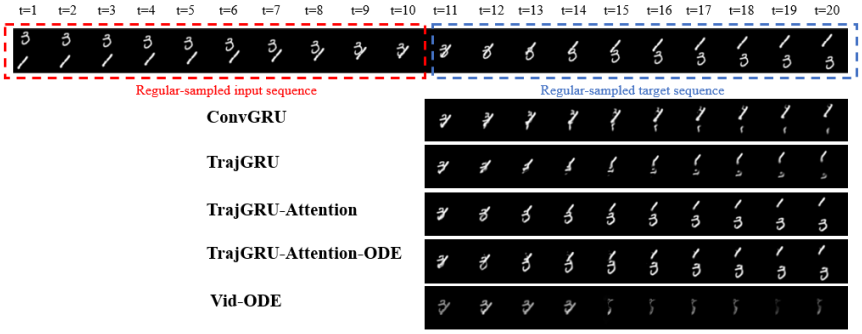

Figure 3: Results using the MovingMNIST with regular-sampled input and target sequences.

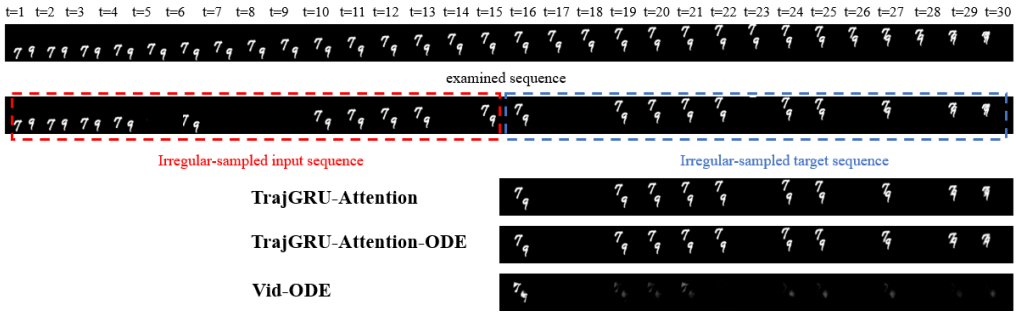

Figure 4: Results using the MovingMNIST with irregular-sampled input and target sequences.

4.3 Results and Comments

To test the models' performance, we use four evaluation metrics: MSE, MAE, SSIM, and Peak-Signal-to-Noise Ratio (PSNR) (Hore & Ziou, 2010), as described in Table 1. In addition, some prediction results using the MovingMNIST are shown in Figures 3, 4, and prediction results using the KTH Action are shown in Figures 5, 6.

Based on the experiments, the results show that the two proposed models achieve better performance than the baseline models. In particular, the TrajGRU-Attention and TrajGRU-Attention outperform the Vid-ODE for the irregularly sampled time series assumptions. Moreover, regularly sampled sequence prediction results demonstrate that the MA modules can alleviate the blurry effect caused by the vanishing gradient issue. Overall, the TrajGRU-Attention yields the best performance in the regular sampling assumption, and the TrajGRU-Attention-ODE achieves the best results when the data is collected at arbitrary timestamps. Additional comparisons are provided in Appendix A.6.

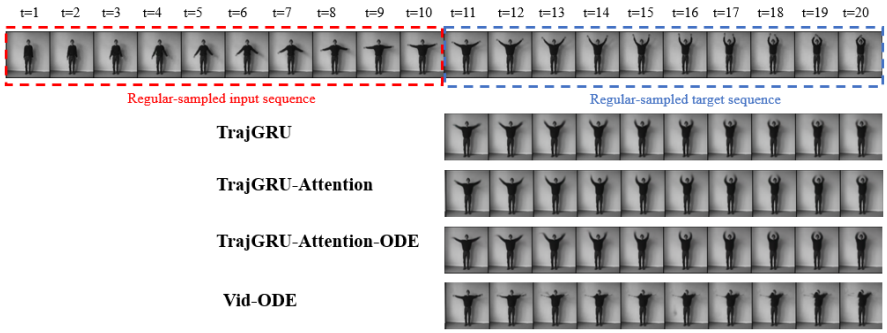

Figure 5: Results using the KTH Action with regular-sampled input and target sequences.

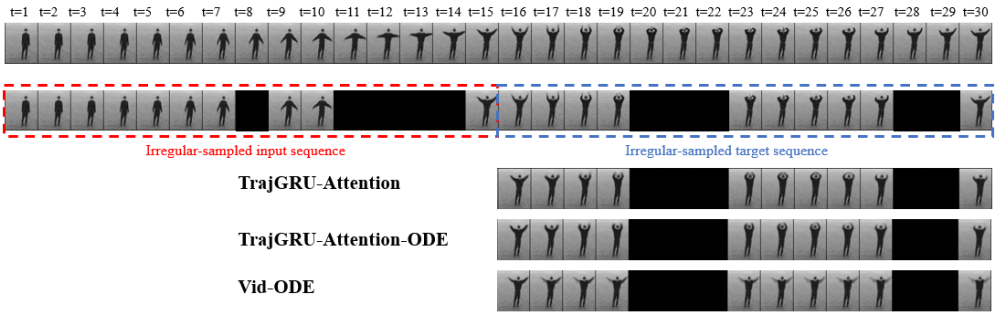

Figure 6: Results using the KTH Action with irregular-sampled input and target sequences.

Datasets	Model	MSE (↓) $\times 10^4$	MAE(↓) $\times 10^4$	SSIM (↑) $\times 10^2$	PSNR(↑)
MovingMNIST (regular sampling)	ConvGRU	80.72	178.21	92.38	21.35
	TrajGRU	58.41	141.57	94.32	22.94
	Vid-ODE	319.31	507.19	70.78	15.01
	TrajGRU-Attention	**39.62**	**110.03**	**96.06**	**24.97**
	TrajGRU-Attention-ODE	41.55	115.33	95.78	24.62
MovingMNIST (irregular sampling)	Vid-ODE	356.27	503.87	69.67	14.53
	TrajGRU-Attention	**77.29**	**181.88**	**91.89**	**21.54**
	TrajGRU-Attention-ODE	91.01	203.01	90.59	20.75
MovingMNIST++ (regular sampling)	ConvGRU	107.88	298.11	75.44	19.83
	TrajGRU	92.409	262.75	78.25	20.13
	Vid-ODE	369.32	494.41	72.12	14.36
	TrajGRU-Attention	**78.35**	**226.17**	**82.35**	**21.15**
	TrajGRU-Attention-ODE	90.73	260.53	78.51	20.53
MovingMNIST++ (irregular sampling)	Vid-ODE	378.31	500.56	61.93	14.26
	TrajGRU-Attention	112.15	307.61	75.72	19.73
	TrajGRU-Attention-ODE	**111.33**	**305.14**	**76.02**	**19.95**
KTH Action (regular sampling)	TrajGRU	22.737	183.33	87.98	27.29
	Vid-ODE	35.61	224.27	84.67	25.23
	TrajGRU-Attention	**20.84**	**175.74**	**89.43**	**27.59**
	TrajGRU-Attention-ODE	21.05	174.49	89.759	27.67
KTH Action (irregular sampling)	Vid-ODE	44.91	263.63	83.72	24.39
	TrajGRU-Attention	29.59	210.27	87.06	26.05
	TrajGRU-Attention-ODE	**28.64**	**209.35**	**87.27**	**26.16**

Table 1: Values of evaluation metrics for the models.

5 Conclusion

This paper proposes two sequence-to-sequence models named TrajGRU-Attention and TrajGRU-Attention-ODE that can handle irregularly sampled spatiotemporal data. Overall, the proposed models can perform the overall SSP task with different datasets and outperform the baseline models. Despite some promising results of our models, there is still room for further improvement in our system. Furthermore, the combination of attention mechanisms and NODE is very promising since the TrajGRU-Attention-ODE can perform the SSP task effectively in some instances.

6 ETHICS STATEMENT

We acknowledge that all co-authors of this work have read and committed to adhering to the ICLR Code of Ethics

7 REPRODUCIBILITY STATEMENT

We strive to ensure the reproducibility of the experiment results. The entire implementation details are described in Appendix A.1, Appendix A.2, Appendix A.3, Appendix A.5 and Appendix A.6. Finally, the source code will be made publicly available.

REFERENCES

Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Neural spatio-temporal point processes. *arXiv preprint arXiv:2011.04583*, 2020.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Advances in neural information processing systems*, 32, 2019.

Wei Fang, Yupeng Chen, and Qiongying Xue. Survey on research of rnn-based spatio-temporal sequence prediction algorithms. *Journal on Big Data*, 3(3):97, 2021.

AM Foley. Uncertainty in regional climate modelling: A review. *Progress in Physical Geography*, 34(5):647–670, 2010.

Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pp. 2366–2369. IEEE, 2010.

Jianzhu Huai, Yuan Zhuang, Yukai Lin, Grzegorz Jozkow, Qicheng Yuan, and Dong Chen. Continuous-time spatiotemporal calibration of a rolling shutter camera-imu system. *IEEE Sensors Journal*, 22(8):7920–7930, 2022.

Maria Kanakidou, JH Seinfeld, SN Pandis, Ian Barnes, Franciscus Johannes Dentener, Maria Cristina Facchini, Rita Van Dingenen, Barbara Ervens, ANCJSE Nenes, CJ Nielsen, et al. Organic aerosol and global climate modelling: a review. *Atmospheric Chemistry and Physics*, 5 (4):1053–1123, 2005.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Xianglong Luo, Danyang Li, Yu Yang, and Shengrui Zhang. Spatiotemporal traffic flow prediction with knn and lstm. *Journal of Advanced Transportation*, 2019, 2019.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Sunghyun Park, Kangyeol Kim, Junsoo Lee, Jaegul Choo, Joonseok Lee, Sookyung Kim, and Edward Choi. Vid-ode: Continuous-time video generation with neural ordinary differential equation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 2412–2422, 2021.

Hans Pinckaers and Geert Litjens. Neural ordinary differential equations for semantic segmentation of individual colon glands. *arXiv preprint arXiv:1910.10470*, 2019.

Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pp. 55–69. Springer, 1998.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pp. 32–36. IEEE, 2004.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. *Advances in neural information processing systems*, 30, 2017.

Mennatullah Siam, Sepehr Valipour, Martin Jagersand, and Nilanjan Ray. Convolutional gated recurrent networks for video segmentation. In *2017 IEEE international conference on image processing (ICIP)*, pp. 3090–3094. IEEE, 2017.

Rohollah Soltani and Hui Jiang. Higher order recurrent neural networks. *arXiv preprint arXiv:1605.00064*, 2016.

Jiahao Su, Wonmin Byeon, Jean Kossaifi, Furong Huang, Jan Kautz, and Anima Anandkumar. Convolutional tensor-train lstm for spatio-temporal learning. *Advances in Neural Information Processing Systems*, 33:13714–13726, 2020.

Quang-Khai Tran and Sa-kwang Song. Computer vision in precipitation nowcasting: Applying image quality assessment metrics for training deep neural networks. *Atmosphere*, 10(5):244, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Senzhang Wang, Jiannong Cao, and Philip Yu. Deep learning for spatio-temporal data mining: A survey. *IEEE transactions on knowledge and data engineering*, 2020.

Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. *Advances in neural information processing systems*, 30, 2017.

Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and S Yu Philip. Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *International Conference on Machine Learning*, pp. 5123–5132. PMLR, 2018.

Yunbo Wang, Haixu Wu, Jianjin Zhang, Zhifeng Gao, Jianmin Wang, Philip Yu, and Mingsheng Long. Predrnn: A recurrent neural network for spatiotemporal predictive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. *Advances in Neural Information Processing Systems*, 30, 2017.

Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. Ode2vae: Deep generative second order odes with bayesian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, 2019.

A APPENDIX

A.1 BASELINE MODEL CONFIGURATION

In this work, three models are implemented as the baseline models: the ConvGRU, TrajGRU (Shi et al., 2017), and Vid-ODE (Park et al., 2021). As mentioned earlier, both regular and irregular time-sampling assumptions will be examined for the datasets; thus, ConvGRU and TrajGRU models cannot be used in some cases. For a fair comparison, we use some common hyperparameters of the ConvGRU, TrajGRU and the proposed models. With the Vid-ODE, we use the model structure, optimizer, and loss function, described in the Vid-ODE paper ((Park et al., 2021)). We also implemented this model based on their original code. For the training, validation, and testing processes, we use some hyperparameters like the number of epochs, the number of iterations, and the initial learning rate, similar to the proposed models' hyperparameters. Our simulation results and the learning curves in Appendix A.6 show that the number of epochs used is sufficient. In addition, the configurations of the ConvGRU and TrajGRU models are illustrated in Table 2, and the Vid-ODE model parameters are described in Table 3.

ConvGRU. The main formulas of the ConvGRU network can be expressed as

$$\begin{aligned}
\mathbf{Z}(t_i) &= \sigma\left(\mathbf{W}_{xz} * \mathbf{X}(t_i) + \mathbf{W}_{hz} * \mathbf{H}(t_{i-1}) + \mathbf{B}_z\right), \\
\mathbf{R}(t_i) &= \sigma\left(\mathbf{W}_{xr} * \mathbf{X}(t_i) + \mathbf{W}_{hr} * \mathbf{H}(t_{i-1}) + \mathbf{B}_r\right), \\
\tilde{\mathbf{H}}(t_i) &= \tanh\left(\mathbf{W}_{xh} * \mathbf{X}(t_i) + \mathbf{R}(t_i) \circ \mathbf{W}_{hh} * \mathbf{H}(t_{i-1}) + \mathbf{B}_h\right), \\
\mathbf{H}(t_i) &= \mathbf{Z}(t_i) \circ \mathbf{H}(t_{i-1}) + (1 - \mathbf{Z}(t_i)) \circ \tilde{\mathbf{H}}(t_i),
\end{aligned} \tag{10}$$

where $\mathbf{X}(t_i)$ and $\mathbf{H}(t_{i-1})$ represent the input at timestamp t_i and previous hidden state, respectively; $\mathbf{Z}(t_i)$ and $\mathbf{R}(t_i)$ represent the update gate and reset gate, respectively, regulating the information flows; $\tilde{\mathbf{H}}(t_i)$ represents the memory state storing new information of the recent input data; "$*$" indicates convolution operator and "$\circ$" denotes Hadamard product; "σ" " refers to the Sigmoid activation function, "$\tanh$" refers the tanh activation function. For learnable parameters, a ConvGRU cell contains six convolutional layers ($\mathbf{W}_{xz}, \mathbf{W}_{hz}, \mathbf{W}_{xr}, \mathbf{W}_{hr}, \mathbf{W}_{xh}, \mathbf{W}_{hh}$).

TrajGRU. The main formulas of the TrajGRU network can be expressed as

$$\begin{aligned}
&\mathbf{U}(t_i), \mathbf{V}(t_i) = \gamma\left(\mathbf{X}(t_i), \mathbf{H}(t_{i-1})\right), \\
&\mathbf{Z}(t_i) = \sigma\left(\mathbf{W}_{xz} * \mathbf{X}(t_i) + \sum_{l=1}^{L} \mathbf{W}_{hz,l} * warp(\mathbf{H}(t_{i-1}), \mathbf{U}(t_i, l), \mathbf{V}(t_i, l))) + \mathbf{B}_z\right), \\
&\mathbf{R}(t_i) = \sigma\left(\mathbf{W}_{xr} * \mathbf{X}(t_i) + \sum_{l=1}^{L} \mathbf{W}_{hr,l} * warp(\mathbf{H}(t_{i-1}), \mathbf{U}(t_i, l), \mathbf{V}(t_i, l)) + \mathbf{B}_r\right), \\
&\tilde{\mathbf{H}}(t_i) = \tanh\left(\mathbf{W}_{xh} * \mathbf{X}(t_i) + \mathbf{R}(t_i) \circ \sum_{l=1}^{L} \mathbf{W}_{hh,l} * warp(\mathbf{H}(t_{i-1}), \mathbf{U}(t_i), \mathbf{V}(t_i)) + \mathbf{B}_h\right), \\
&\mathbf{H}(t_i) = \mathbf{Z}(t_i) \circ \mathbf{H}(t_{i-1}) + (1 - \mathbf{Z}(t_i)) \circ \tilde{\mathbf{H}}(t_i),
\end{aligned} \tag{11}$$

where γ represents the neural network generating the continuous flow fields $\mathbf{U}(t_i)$, $\mathbf{V}(t_i)$ with L different connection links in the transition; $warp$ represents a function used to combine the sets of flow fields and the hidden state to generate an intermediate state with approximated location information; $\mathbf{Z}(t_i)$, $\mathbf{R}(t_i)$ represent the update gate and reset gate, respectively, regulating the information flows; $\tilde{\mathbf{H}}(t_i)$ represents the memory state storing new information of the recent input data; "$*$" indicates convolution operator and "$\circ$" denotes Hadamard product; "σ" " refers to the Sigmoid activation function, "$\tanh$" refers the tanh activation function. For learnable parameters, a TrajGRU cell consists of six convolutional layers ($\mathbf{W}_{xz}, \mathbf{W}_{hz}, \mathbf{W}_{xr}, \mathbf{W}_{hr}, \mathbf{W}_{xh}, \mathbf{W}_{hh}$) and a spatial transformer module γ containing three convolutional layers. Figure 7 illustrates the overall operations of the ConvGRU and TrajGRU networks, and Figure 8 presents the common structure for the two models.

Vid-ODE. Vid-ODE essentially combined the standard ConvGRU network, NODE techniques, and the pixel-level composition technique (Park et al., 2021). From the structure perspective, this model used an encoding-decoding structure for the video generation task. The encoder is constructed by

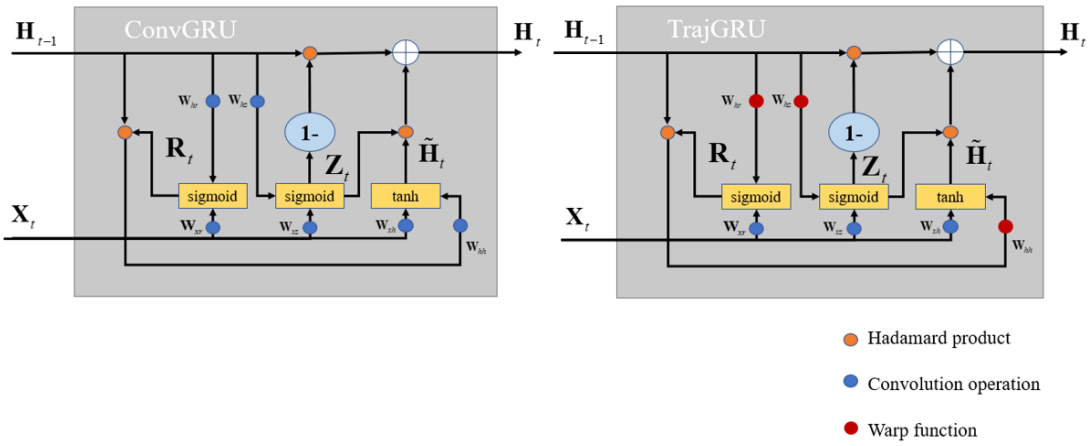

Figure 7: **(Left)** ConvGRU unit. **(Right)** TrajGRU unit.

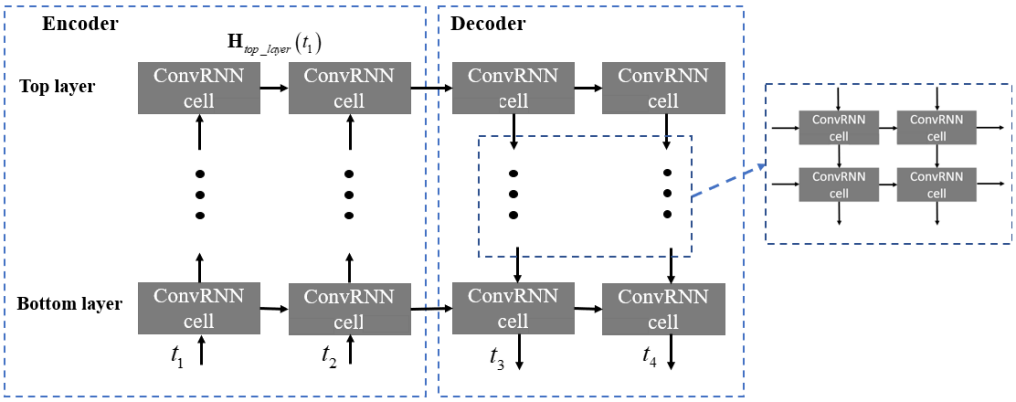

Figure 8: Encoding-decoding structure for the ConvGRU and TrajGRU models.

the ConvGRU layer and a specified ODE neural network which helps capture the irregular sampling intervals; the decoder is constructed by an ODE solver and the pixel-level composition technique.

A.2 Training Process

Loss function. Generally, the Mean Absolute Error (MAE) and Mean Square Error (MSE) are the commonly used loss functions for sequence prediction tasks. However, many prior studies have mentioned the drawbacks when using these error functions for the training process. Since they compute the global errors between the output and ground truth during the process, the system might ignore some local errors. Consequently, the sharpness of the resulting images decreased significantly, especially with long-term sequence prediction problems. Therefore, some related research proposed other loss functions instead of using only a simple MAE or MSE (Tran & Song, 2019). Therefore, we propose the combination of the MAE, MSE, and the Structural Similarity Index Measure (SSIM) as the loss function to guide the training process (Hore & Ziou, 2010). With the SSIM metric, we can somewhat alleviate the blurry effect by modelling the distortions in the loss function. The main formula of the proposed loss function can be derived as

$$\begin{cases} L_{MSE} = \frac{\sum_{i=1}^{K}(\mathbf{X}_{target}(t_i) - \mathbf{X}_{prediction}(t_i))^2}{K}, \\ L_{MAE} = \frac{\sum_{i=1}^{K}|\mathbf{X}_{target}(t_i) - \mathbf{X}_{prediction}(t_i)|}{K}, \\ L_{SSIM} = 1 - SSIM(\mathbf{X}_{target}, \mathbf{X}_{prediction}), \\ f_{loss} = aL_{MSE} + bL_{MAE} + cL_{SSIM}. \end{cases} \tag{12}$$

TrajGRU	**ConvGRU**
Common hyperparameters	
3 down-sampling blocks (encoder): (Conv($C_{dataset} \times 64 \times 3 \times 3$) ,leakyRELU) (Conv($64 \times 96 \times 3 \times 3$) ,leakyRELU) (Conv($96 \times 128 \times 3 \times 3$) ,leakyRELU) 4 up-sampling blocks (decoder): (Deconv($128 \times 96 \times 4 \times 4$) ,leakyRELU) (Deconv($96 \times 64 \times 4 \times 4$) ,leakyRELU) (Deconv($64 \times 32 \times 3 \times 3$) ,leakyRELU) Conv($32 \times 1 \times 1 \times C_{dataset}$)	
Central building blocks	
Encoder:	**Encoder**:
TrajGRU($C = (64, 64), K = 3, K_\gamma = 1, L = 9$)	ConvGRU($C = (64, 64), K = 3$)
TrajGRU($C = (96, 96), K = 3, K_\gamma = 1, L = 9$)	ConvGRU($C = (96, 96), K = 3$)
TrajGRU($C = (128, 128), K = 3, K_\gamma = 1, L = 9$)	ConvGRU($C = (128, 128), K = 3$)
Decoder:	**Decoder**:
TrajGRU($C = (64, 64), K = 3, K_\gamma = 1, L = 9$)	ConvGRU($C = (64, 64), K = 3$)
TrajGRU($C = (96, 96), K = 3, K_\gamma = 1, L = 9$)	ConvGRU($C = (96, 96), K = 3$)
TrajGRU($C = (128, 128), K = 3, K_\gamma = 1, L = 9$)	ConvGRU($C = (128, 128), K = 3$)
Total number of parameters	
4859181 (MovingMNIST), 4860367 (KTH Action).	3695553 (MovingMNIST), 3696739 (KTH Action).

Table 2: Model configuration of ConvGRU and TrajGRU.

Here, $\mathbf{X}_{target}$, $\mathbf{X}_{prediction}$ represent the target and output prediction sequences, respectively; $a = 1, b = 1, c = 0.05$ denote the weights of the three elements of the loss function. The weights of the MSE and MAE are equal and more significant than the SSIM because MSE and MAE are supported to be the main elements manipulating the global errors, and the SSIM is the supportive factor impacting the local errors. The comparison between different loss functions is shown in Table 4

Based on Table 4, our proposed loss function helps the model generate the results with the best evaluation metrics, so we choose this loss function as the standard function to train the models (i.e., ConvGRU, TrajGRU, ConvGRU-Attention, ConvGRU-Attention-ODE, TrajGRU-Attention, and TrajGRU-Attention-ODE).

Optimizer. This work uses the Adam optimizer to train all the models because of its stability and adaptive features (Kingma & Ba, 2014). In addition, the training processes of all models used the early stopping mechanism to obtain the final results. The initial learning rate is 10^{-4}, which is decayed by 0.99 after every epoch. For all models, we set the maximum number of epochs to 25. In addition, the early stopping mechanism is applied to obtain the final results and the corresponding set of trainable parameters (Prechelt, 1998).

A.3 Motion-based Attention Setting

In the TrajGRU-Attention model, the MEA blocks are used at the encoder, and the MDE blocks are used at the decoder to handle the irregularly sampled time series. Moreover, the model can deal with long-range sequence prediction with these attention mechanisms. However, if the MEA and MDE modules are integrated into all central blocks of the model, there will be considerable computational complexity with a massive set of trainable parameters. Therefore, we just insert those attention modules on specific layers to balance the overall performance and the complexity.

Encoder:
4 down-sampling blocks: (Conv($64 \times 64 \times 3 \times 3$), BatchNorm, RELU),
(Conv($64 \times 128 \times 4 \times 4$), BatchNorm, RELU),
(Conv($128 \times 256 \times 4 \times 4$), BatchNorm, RELU),
(Conv($256 \times 512 \times 4 \times 4$), BatchNorm, RELU),
ConvGRU($C = (512, 512), K = 3$).
ODE network: (Conv($512 \times 128 \times 3 \times 3$), Tanh),
(Conv($256 \times 256 \times 3 \times 3$), Tanh),
(Conv($256 \times 256 \times 3 \times 3$), Tanh),
(Conv($256 \times 256 \times 3 \times 3$), Tanh),
(Conv($256 \times 256 \times 3 \times 3$), Tanh).
ODE method: Euler.
Decoder:
ODE network: (Conv($512 \times 128 \times 3 \times 3$), Tanh),
(Conv($256 \times 256 \times 3 \times 3$), Tanh),
(Conv($256 \times 256 \times 3 \times 3$), Tanh), (Conv($256 \times 256 \times 3 \times 3$), Tanh),
(Conv($256 \times 256 \times 3 \times 3$), Tanh).
4 up-sampling blocks: (Bilinear(scale=2), Conv($512 \times 256 \times 3 \times 3$),BatchNorm,RELU),
(Bilinear(scale=2), Conv($256 \times 128 \times 3 \times 3$),BatchNorm,RELU),
(Bilinear(scale=2), Conv($128 \times 64 \times 3 \times 3$),BatchNorm,RELU),
(Bilinear(scale=2), Conv($64 \times 32 \times 3 \times 3$),BatchNorm,RELU), Conv($32 \times C_{dataset} \times 3 \times 3$),
ODE method: Euler.
GAN network:
Generator: Encoder and Decoder.
Discriminator: (Conv($1 \times 64 \times 4 \times 4$), InstanceNorm, leakyRELU),
(Conv($64 \times 128 \times 4 \times 4$), InstanceNorm, leakyRELU),
(Conv($128 \times 256 \times 4 \times 4$), InstanceNorm, leakyRELU),
(Conv($256 \times 512 \times 4 \times 4$), InstanceNorm, leakyRELU),
Conv($512 \times 64 \times 4 \times 4$).
Total number of parameters: 60060164 (MovingMNIST), 60067078 (KTH Action).

Table 3: Vid-ODE model configuration.

Datasets/Model	**Loss function**	**MSE** (↓)	**MAE**(↓)	**SSIM** (↑)	**PSNR**(↑)
MovingMNIST TrajGRU	L_{MSE}	61.21	176.23	92.01	22.35
	$L_{MSE} + L_{MAE}$	69.22	167.43	92.03	22.06
	$\mathbf{f}_{loss}$ **(Equation 12)**	61.93	**153.34**	**93.89**	**22.73**

Table 4: Comparison between different loss functions of the TrajGRU model when using the MovingMNIST.

To choose the optimal attention mechanism settings, the TrajGRU-Attention model is tuned using the MovingMNIST data in the scenario where the numbers of attention modules vary and the other hyperparameters are constant. In addition, the training process is described in Appendix A.2.

Table 5 illustrates the resulting evaluation metrics of the TrajGRU-Attention model with different attention settings when training and evaluating with the MovingMNIST dataset. For instance, when the encoder uses the attention module at its central layer 3 and the decoder uses the attention module at its central layer 3, we abbreviate this setting as "en-a3-de-a3". Based on Table 5, the setting "en-a23-de-a21" shows the best results in all four metrics, so we chose this setting to design the TrajGRU-Attention and TrajGRU-Attention-ODE models, as presented in Appendix A.5.

A.4 Time Sampling Methods.

For the overall SSP task, each series is divided into two subsequences (i.e., the input and the target) and then the sampling methods are applied to each subsequence. We define two collections of timestamps representing the regular and irregular sampling assumptions, as shown in Table 6.

Setting	MSE (↓)	MAE(↓)	SSIM (↑)	PSNR(↑)
en-a3-de-a3	61.81	151.23	93.78	22.59
en-a2-de-a2	62.11	156.12	93.71	22.55
en-a1-de-a1	58.42	150.23	93.96	22.85
en-a23-de-a32	53.71	136.34	94.66	23.33
en-a23-de-a21	**49.63**	**131.32**	**94.94**	**23.73**
en-a123-de-a321	60.83	145.23	93.92	22.92

Table 5: Comparison between different settings of the TrajGRU-Attention model when using the MovingMNIST.

Sampling method	Regular	Irregular
Set size	1	100
Input interval	$[1, 10]$	$[1, 15]$
Target interval	$[11, 20]$	$[16, 30]$

Table 6: configuration of the two sampling methods

In Table 6, the set size represents the number of distinct collections of time steps, and the input and target time intervals denote the minimum and maximum time steps for each sampling method. In the experiments shown in Section 4, the input and target sequences have the same length of 10.

A.5 Proposed Model Configuration

Table 7 describes the two proposed models' overall configurations and important hyperparameters.

Generally, we use the TrajGRU network containing the convolutional layers that extract the image features of each data frame, as illustrated in Figures 2 and 7. In addition, for the hierarchical feature extraction, we insert a resampling block (i.e., an upsampling or a downsampling block) between two adjacent central blocks (i.e., central neural layers) for the SSP task to help the system capture spatial features with different levels. We build the up-sampling block using convolutional and LeakyRELU activation layers in the proposed models. In contrast, the down-sampling block is constructed by a pair of deconvolutional and LeakyRELU activation layers. As such, the shape of the output states can be easily adjusted by defining the hyper-parameters of the convolutional and deconvolutional layers (e.g., the kernel size, number of input and output channels, padding, stride, dilation).

TrajGRU-Attention	**TrajGRU-Attention-ODE**
Common hyperparameters	
3 down-sampling blocks (encoder): (Conv($C_{dataset} \times 64 \times 3 \times 3$) ,leakyRELU) (Conv($64 \times 96 \times 3 \times 3$) ,leakyRELU) (Conv($96 \times 128 \times 3 \times 3$) ,leakyRELU) Zigzag connection (encoder): (Deconv($128 \times 64 \times 4 \times 4$), Conv($64 \times 64 \times 3 \times 3$) , Conv($64 \times 64 \times 3 \times 3$)) 4 up-sampling blocks (decoder): (Deconv($128 \times 96 \times 4 \times 4$) ,leakyRELU) (Deconv($96 \times 64 \times 4 \times 4$) ,leakyRELU) (Deconv($64 \times 32 \times 3 \times 3$) ,leakyRELU) Conv($32 \times 1 \times 1 \times C_{dataset}$) Zigzag connection (decoder): (Conv($64 \times 128 \times 4 \times 4$), Conv($128 \times 128 \times 3 \times 3$) , Conv($128 \times 128 \times 3 \times 3$))	
Central building blocks	
Encoder:	
TrajGRU($C = (64, 64), K = 3, K_\gamma = 1, L = 9$)	TrajGRU($C = (64, 64), K = 3, K_\gamma = 1, L = 9$)
TrajGRU($C = (96, 96), K = 3, K_\gamma = 1, L = 9$)	TrajGRU($C = (96, 96), K = 3, K_\gamma = 1, L = 9$)
TrajGRU($C = (128, 128), K = 3, K_\gamma = 1, L = 9$)	TrajGRU($C = (128, 128), K = 3, K_\gamma = 1, L = 9$)
Decoder:	
TrajGRU($C = (64, 64), K = 3, K_\gamma = 1, L = 9$)	TrajGRU-ODE($C = (64, 64), K = 3, K_\gamma = 1,$ $L = 9, n_{ode} = 2, K_{ode} = 3, method_{ode} = Euler$)
TrajGRU($C = (96, 96), K = 3, K_\gamma = 1, L = 9$)	TrajGRU-ODE($C = (96, 96), K = 3, K_\gamma = 1$ $L = 9, n_{ode} = 2, K_{ode} = 3, method_{ode=Euler}$)
TrajGRU($C = (128, 128), K = 3, K_\gamma = 1, L = 9$)	TrajGRU-ODE($C = (128, 128), K = 3, K_\gamma = 1$ $L = 9, n_{ode} = 2, K_{ode} = 3, method_{ode} = Euler$)
Attention blocks	
Encoder:	
MEA(4 conv($C = (96, 96), K = 1$); 1 linear($C = (384, 96)$)	MEA(4 conv($C = (96, 96), K = 1$); 1 linear($C = (384, 96)$)
MEA(4 conv($C = (128, 128), K = 1$); 1 linear($C = (512, 128)$)	MEA(4 conv($C = (128, 128), K = 1$); 1 linear($C = (512, 128)$)
Decoder:	
MDA(4 conv($C = (96, 96), K = 1$); 1 linear($C = (384, 96)$)	
MDA(4 conv($C = (64, 64), K = 1$); 1 linear($C = (256, 64)$)	
Total number of parameters	
7258509 (MovingMNIST) 7259695 (KTH Action)	8844685 (MovingMNIST) 8845871 (KTH Action)

Table 7: Proposed Model Configuration.

A.6 ADDITIONAL COMPARISON BETWEEN MODELS

As shown in Table 1, the Vid-ODE can not perform the SSP task properly in some scenarios, especially with the MovingMNIST and MovingMNIST++. The reasons leading to the limitations of Vid-ODE's performance are the time sampling assumptions and the properties of the datasets (i.e., the MovingMNIST and MovingMNIST++). In the Vid-ODE paper, the authors used the concept of frame rate that can be fixed in the encoding and decoding parts (e.g., the encoder's frame rate is 1, and the decoder's frame rate is 0.5). Regarding the model structure, the composition mask and image difference techniques are ineffective when the time gap between adjacent frames is large and variable. Furthermore, the Vid-ODE decoder uses only an ODE solver component as the central layer to generate hidden states; thus, these states may not contain enough spatial features to reconstruct the output. On the other hand, the proposed models use multiple TrajGRU-ODE networks (the combination of TrajGRU networks and ODE solvers) at the TrajGRU-Attention-ODE model, and groups of TrajGRU networks and Motion-based Attention blocks at the TrajGRU-Attention model.

For a fair comparison, the Vid-ODE and proposed models are implemented with the KTH Action dataset for the SSP task where the input sequence has 5 frames, and the target sequence has 5 frames which is the setting used in Vid-ODE paper (Park et al. (2021)). Finally, the evaluation metrics are displayed in Table 8, and the prediction results are shown in Figure 9.

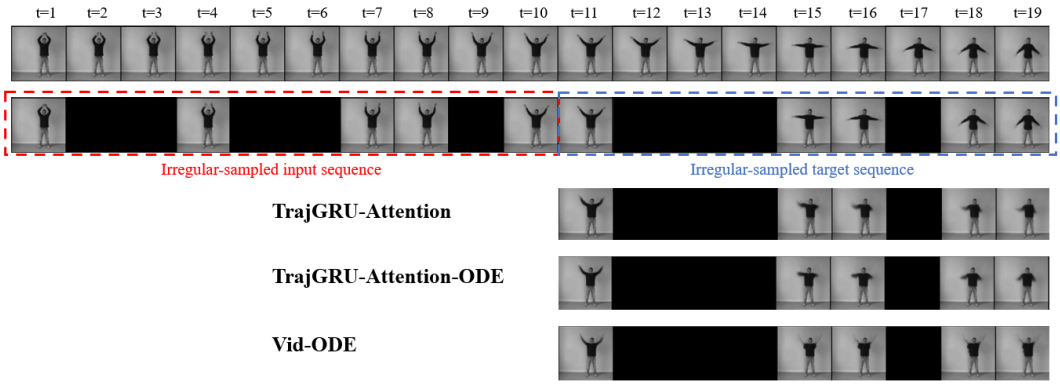

Figure 9: Results using the KTH Action with irregular-sampled input and target sequences. The input sequence contains 5 frames, and the target sequence contains 5 frames.

As shown in Table 8, the Vid-ODE model performs more effectively with short-range prediction; however, the proposed models generate better prediction results, especially with the irregular time sampling assumption.

To clarify the improvement of the proposed modules (i.e., the MA and TrajGRU-ODE network), we compare the proposed models (i.e., TrajGRU-Attention and TrajGRU-Attention-ODE) with the ConvGRU-Attention and ConvGRU-Attention-ODE models. Basically, the ConvGRU-Attention and ConvGRU-Attention-ODE use the ConvGRU networks illustrated in Table 2 as the central building blocks instead of the TrajGRU networks. These models are trained, validated, and tested with the MovingMNIST and MovingMNIST++ datasets. Finally, the evaluation metrics are displayed in Table 9, and the prediction results are shown in Figure 10.

As shown in Tables 9 and 1, the MA modules can improve the performance of the ConvRNNs (e.g., ConvGRU and TrajGRU) and the combinations of the ConvRNNs and ODE modules can allow the models to perform the SSP task with data frames at arbitrary time steps. Compared to the ConvGRU-Attention and ConvGRU-Attention-ODE, the proposed models can achieve better prediction results.

Datasets	Model	MSE (↓) $\times 10^4$	MAE(↓) $\times 10^4$	SSIM (↑) $\times 10^2$	PSNR(↑)
KTH Action (regular sampling)	Vid-ODE	16.41	156.02	87.58	29.28
	TrajGRU-Attention	**10.44**	**138.15**	**89.35**	**30.85**
	TrajGRU-Attention-ODE	10.71	138.17	89.32	30.78
KTH Action (irregular sampling)	Vid-ODE	33.28	217.46	85.29	25.59
	TrajGRU-Attention	28.33	199.29	87.49	26.46
	TrajGRU-Attention-ODE	**26.65**	**198.59**	**87.57**	**26.61**

Table 8: Values of evaluation metrics for the models: Vid-ODE, TrajGRU-Attention, TrajGRU-Attention-ODE.

Datasets	Model	MSE (↓) $\times 10^4$	MAE(↓) $\times 10^4$	SSIM (↑) $\times 10^2$	PSNR(↑)
MovingMNIST (regular sampling)	ConvGRU-Attention	73.63	168.64	92.99	21.81
	ConvGRU-Attention-ODE	64.85	154.51	93.71	22.39
	TrajGRU-Attention	**39.62**	**110.03**	**96.06**	**24.97**
	TrajGRU-Attention-ODE	41.55	115.33	95.78	24.62
MovingMNIST (irregular sampling)	ConvGRU-Attention	111.11	232.21	88.92	19.81
	ConvGRU-Attention-ODE	101.65	222.29	89.53	20.21
	TrajGRU-Attention	**77.29**	**181.88**	**91.89**	**21.54**
	TrajGRU-Attention-ODE	91.01	203.01	90.59	20.75
MovingMNIST++ (regular sampling)	ConvGRU-Attention	103.57	282.53	76.88	20.01
	ConvGRU-Attention-ODE	110.41	301.22	75.04	19.74
	TrajGRU-Attention	**78.35**	**226.17**	**82.35**	**21.15**
	TrajGRU-Attention-ODE	90.73	260.53	78.51	20.53
MovingMNIST++ (irregular sampling)	ConvGRU-Attention	134.24	340.69	72.37	18.92
	ConvGRU-Attention-ODE	139.59	351.55	71.26	18.74
	TrajGRU-Attention	112.15	307.61	75.72	19.73
	TrajGRU-Attention-ODE	**111.33**	**305.14**	**76.02**	**19.95**

Table 9: Values of evaluation metrics for the models: ConvGRU-Attention, ConvGRU-Attention-ODE, TrajGRU-Attention, TrajGRU-Attention-ODE.

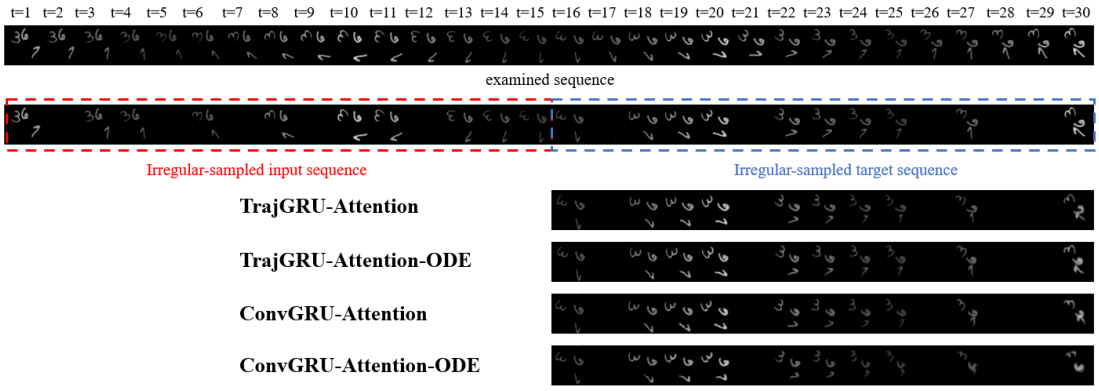

Figure 10: Results using the MovingMNIST++ with irregular-sampled input and target sequences.

In addition, the learning curves of the models used in Section 4, are shown in Figures 11, 12, 13, 14. They can evaluate models' learning ability. Overall, the TrajGRU-Attention and TrajGRU-Attention-ODE models achieve better performance during the training and validation processes.

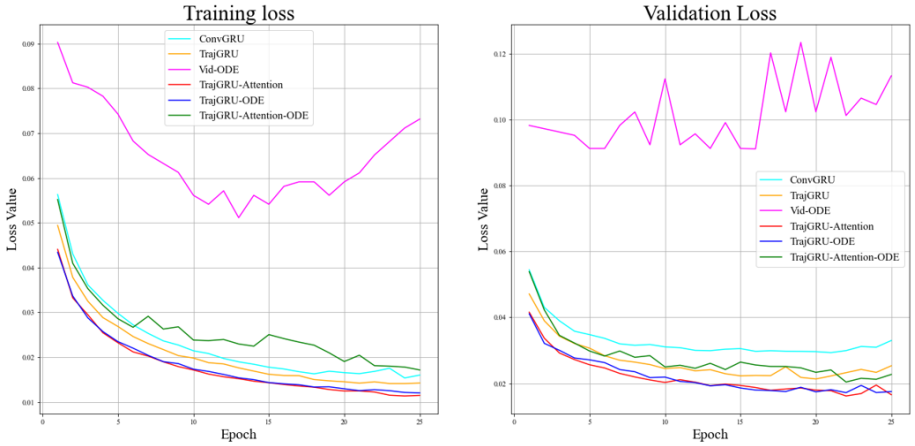

Figure 11: Learning curves of models using the MovingMNIST with regular sampling assumption at both the encoder and decoder: **(Left)** Training learning curve. **(Right)** Validation learning curve.

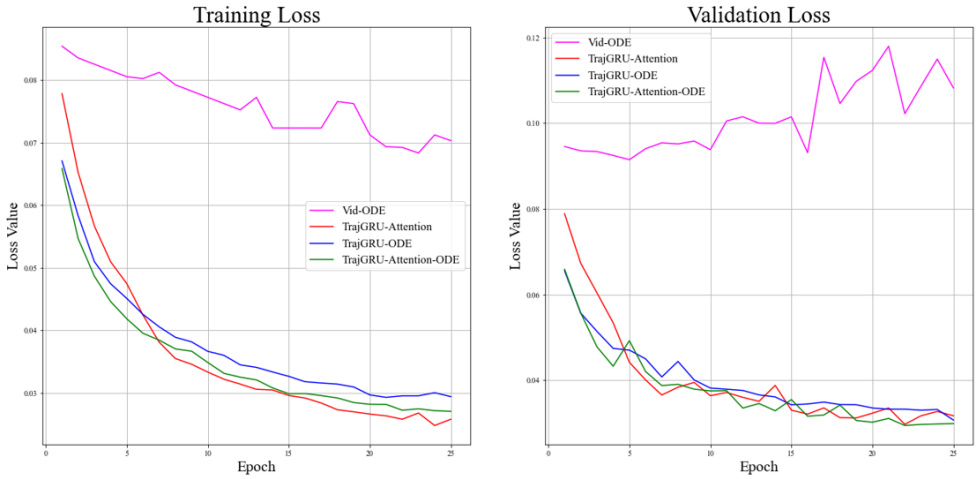

Figure 12: Learning curves of models using the MovingMNIST with irregular sampling assumption at both the encoder and decoder: **(Left)** Training learning curve. **(Right)** Validation learning curve.

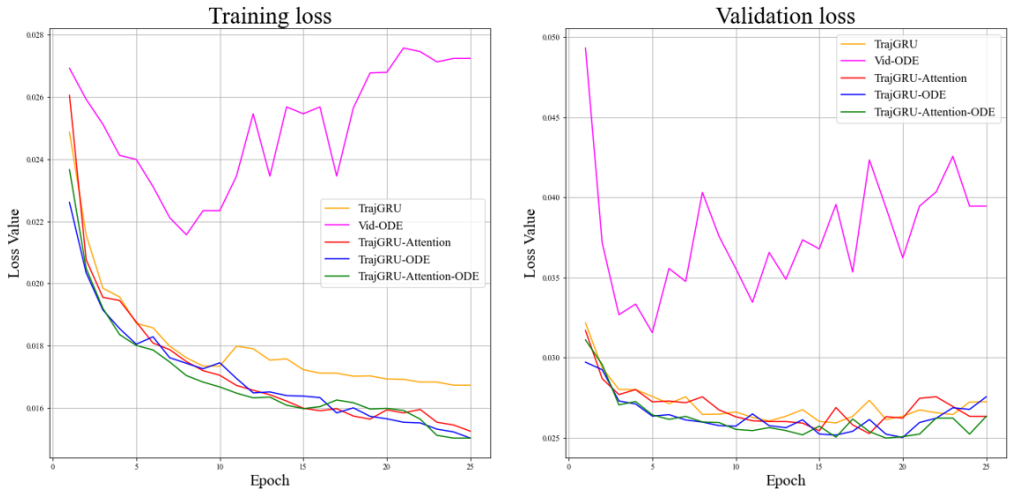

Figure 13: Learning curves of models using the KTH Action with regular sampling assumption at both the encoder and decoder: **(Left)** Training learning curve. **(Right)** Validation learning curve.

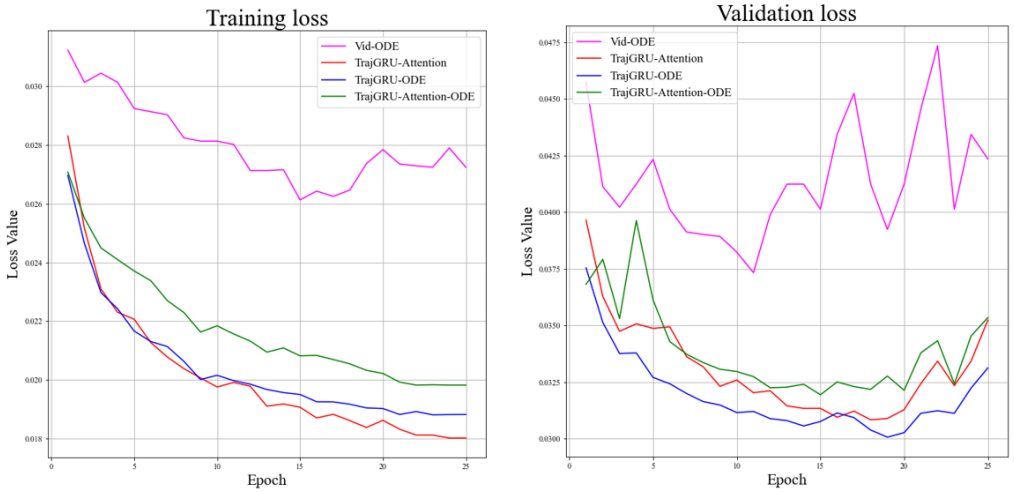

Figure 14: Learning curves of models using the KTH Action with irregular sampling assumption at both the encoder and decoder: **(Left)** Training learning curve. **(Right)** Validation learning curve.