Spatiotemporal Transformer for 2D Time Series Sequence Forecasting

Calvin Janitra Halim*
Katsuya Kosukegawa*
Kazuhiko Kawamoto*
calvinjh@chiba-u.jp
katsuya_kosukegawa@chiba-u.jp
kawa@faculty.chiba-u.jp
Chiba University
Chiba-shi, Chiba, Japan

ABSTRACT

Two-dimensional (2D) spatiotemporal sequence prediction has long been essential in gaining understanding of real-world phenomena. Existing solutions to this problem have mostly been derivations of methods based on convolutional long short-term memory (ConvLSTM). However, attention-based networks, such as Transformers, are becoming popular as the standard method in natural language processing tasks, because of their ability to model long-term dependencies, unlike LSTM. Although Transformers have been successfully used in time series prediction, their application to spatiotemporal sequences remains limited. Here, we propose a novel Transformer that models both spatial and temporal features by utilizing attention and self-attention mechanisms to model 2D features from a timestep with respect to self and other timestep features, without any convolution neural network- or recurrent neural network-based modifications. In the experiments, the model shows improvements over the standard Transformer, LSTM, and its seq2seq variants. It also exhibits a performance comparable to that of ConvLSTM in both point prediction and quantile regression settings.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Artificial intelligence**.

KEYWORDS

time series forecasting, transformer, spatiotemporal sequence forecasting, self-attention, quantile regression

ACM Reference Format:
Calvin Janitra Halim, Katsuya Kosukegawa, and Kazuhiko Kawamoto. 2021. Spatiotemporal Transformer for 2D Time Series Sequence Forecasting. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Singapore, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

*All authors contributed equally to this research.

Singapore '21, August 14–18, 2021, Singapore, Singapore

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

1.1 Background and Proposal

2D time series sequence forecasting tasks predict future sequences of spatiotemporal data, i.e., data comprising spatial features ordered through time. The primary examples of such sequences are physical phenomena. The task of predicting spatiotemporal sequence data is vital in gaining deeper understanding of various natural phenomena, as successful predictions allow us to prepare for future events.

Most of the approaches to the task of spatiotemporal sequence prediction has been based of a combination of long-short term memory (LSTM) and convolutional neural networks (CNNs), called convolutional LSTM (ConvLSTM) [26]. The inherent approximation capabilities of this model enabled a predictive power that effectively utilizes spatial and temporal features of a sequence, as demonstrated by its derivative and enhanced models achieving state-of-the-art performance in the field [14, 27, 34–36]. However, it is known that even though LSTM was created to mitigate the "vanishing gradient" problem of RNN, in practice, LSTM still suffers from a weakness in capturing long-term dependencies. In addition, according to Luo et al. [21], stacked CNN layers in ConvLSTM provide a smaller receptive field than theoretically possible. Recent approaches such as [19] have attempted to mitigate this using self-attention layers to replace the default memory cell inside ConvLSTM to better record and encode the long-range temporal and spatial dependencies. However, the general LSTM bottleneck remains.

On the other hand, the Transformer model [32] has benefited from a surge in popularity after its breakthrough in the field of natural language processing (NLP), which was previously dominated by LSTM-based approaches [13, 24, 38]. It follows a structure called sequence-to-sequence (seq2seq), popularized by [5] and [30], which uses the encoder model to encode the hidden representation of the input sequence and the decoder model to output predictions conditioned on the hidden representation output by the encoder model. Along with this structure, they also employ multi-headed (and scaled) dot-product attention layers [3, 11, 22] and feed-forward layers. The enhanced performance of Transformer is supported by combination of seq2seq structure and attention layers mitigating

the vanishing gradient problem. This has evidently led to Transformer being used in many forms of NLP models and has achieved the best performance in their respective tasks, especially models such as BERT and its variants [8, 23, 29, 42]. Additionally, there have also been movements in the field of computer vision to adopt Transformer, and recent work proves that the model shows comparable performance and even surpasses some of the major CNN-based models, showing the effectiveness of the model in various tasks even outside NLP [4, 9, 31]. Moreover, recent research has explored Transformer in application to more general time series forecasting tasks [2, 18, 37], with notably promising results.

The effectiveness of transformer models can be traced back to their usage of self-attention layers. Self-attention layers allow global dependencies between input and output to be leveraged more fully. In addition, the omission of recurrent structure allows the model to be parallelized among multiple GPU cores, increasing its training efficiency. Although Transformer has been recently yielding good results in time series data, application to spatiotemporal data remains limited. We note that this is due to the Transformer catering only to multivariate data at maximum.

With the aim of solving the problem mentioned above, we propose an extension of Transformer leveraging spatiotemporal sequence data more effectively by extending the self and normal attention layer to calculate not only spatial attention but also calculate spatial attention with respect to other timestep data, as the general Transformer can extract attention features only in one dimension, either temporal [32] or spatial [9]. The spatial attention is then averaged to create spatiotemporal attention for each timestep to extract features that make use of both spatial and temporal features. We note that this model is novel in the field, as it extracts high-level features from both spatial and temporal dimensions by primarily relying on only attention and feed-forward layers, without any use of RNN or CNN-based models. In summary, the main contributions of our study are as follows.

- We propose a model called Spatiotemporal Transformer aiming to extend the capabilities of vanilla Transformer to spatiotemporal data by utilizing both spatial and temporal self and normal attention layers between timesteps.
- We also describe how our model can be trained to generate both point and quantile predictions of spatiotemporal sequence data, acting as deterministic and probabilistic forecasting settings, respectively.
- In an experiment conducted on both moving MNIST and real-world precipitation data with deterministic and probabilistic forecasting settings, we show that our model performed better than original Transformer and LSTM as well as LSTM-based seq2seq and attentional variants, showing that our approach is more effective in learning spatiotemporal data.
- We also show that the model gives a comparable performance to ConvLSTM, showing its potential as a future state-of-the-art in the field of 2D-gridded spatiotemporal forecasting.

1.2 Related Work

To the best of our knowledge, we are the first to propose a spatiotemporal sequence model utilizing only the Transformer paradigm without any integrated CNN or LSTM-based enhancements. Approaches such as [10, 40, 41, 43] utilize spatial and temporal self-attention layers to predict spatiotemporal sequences, but we note that their models all apply some form of CNN or LSTM to the network structure or divide the self-attention layer into spatial and temporal layers. Moreover, all of the prior approaches focus on predicting graph time series data, especially traffic data, but not 2D-gridded spatiotemporal sequence data, which is the primary data structure we want to tackle in this paper.

One particular method not targeted for time series prediction proposed a spatiotemporal transformer, which is similar to our model, incorporated a self-attention calculating mechanism as a single layer [1] for a 3D human motion prediction task. However, further inspection reveals that their model divides the self-attention layer into spatial and temporal sublayers, similar to the abovementioned related work. Instead of dividing the layer into spatial and temporal attention components, we utilize the structure proposed by Dosovitskiy et al. [9], and calculate spatial attention not only from the current timestep but also from other timesteps, resulting in attention weights spanning both the spatial and temporal axes.

1.3 Paper Structure

The remainder of this paper is structured as follows. The formulation of the spatiotemporal sequence forecasting task is described in Section 2. The details of our proposal are described in Section 3, while its training and forecasting flow are presented in 4. We then describe the experimental setup, results, and the implications in Section 5. We finally present our conclusions in Section 6.

2 SPATIOTEMPORAL SEQUENCE FORECASTING TASK

2.1 Point Prediction

In this section, we define the task of deterministic/point forecast of spatiotemporal sequences. We first define a 2D-gridded spatial observation of an event with K measurements, M rows, and N columns as a tensor $X \in \mathbb{R}^{K \times M \times N}$. The K measurements mean that there could be several values at the same point in time and space. A common example here would be the RGB channel in a pixel frame when performing video prediction. In this paper, we restrict our evaluation to observations with only one measurement per point in space and time, that is, $K = 1$ for ease of evaluation. This results in the observation being a 2D matrix rather than a tensor. One can then define observations taken within the first and subsequent timesteps T (inclusive) as $X_{1:T}$. This notation is used throughout.

Here, we follow [26] in defining the task of predicting the forecast. Given past observations $X_{1:T}$ as input, predicting a future sequence from a timestep $T+1$ to $T+\Delta T$ can be defined as follows.

$$\hat{X}_{T+1:T+\Delta T} = \underset{X_{T+1:T+\Delta T}}{\arg\max}\, p(X_{T+1:T+\Delta T} | X_{1:T}) \tag{1}$$

The equation defines the forecasts as the values in which the conditional probability of the future sequence is the highest given the past observations.

2.2 Probabilistic Prediction

Depending on the task description, the forecasting task might not be defined as selecting the most plausible sequence, but rather as

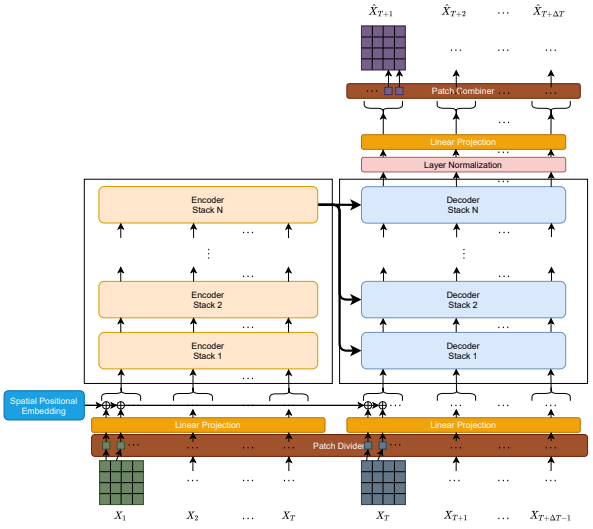

Figure 1: Spatiotemporal Transformer. 2D inputs are first divided into patches and projected to features with d_{model} length. Spatial positional encoding is then added, and the resulting sequence is input to the encoder and decoder. The outputs of the encoder are processed by the decoder to calculate the attention between the target sequence and extracted high-level input sequence features. Finally, the decoder outputs are normalized and projected into the original patch size and recombined to construct the forecasts.

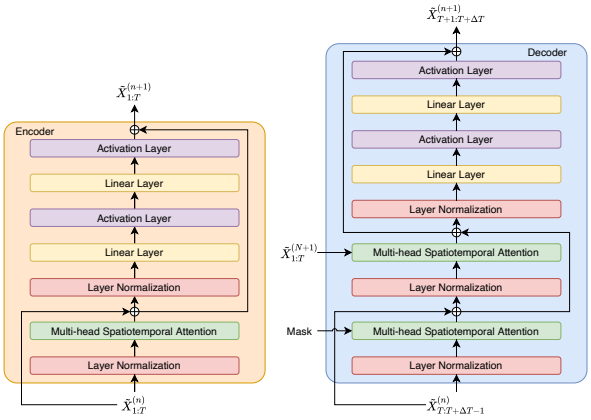

Figure 2: Encoder and decoder structure.

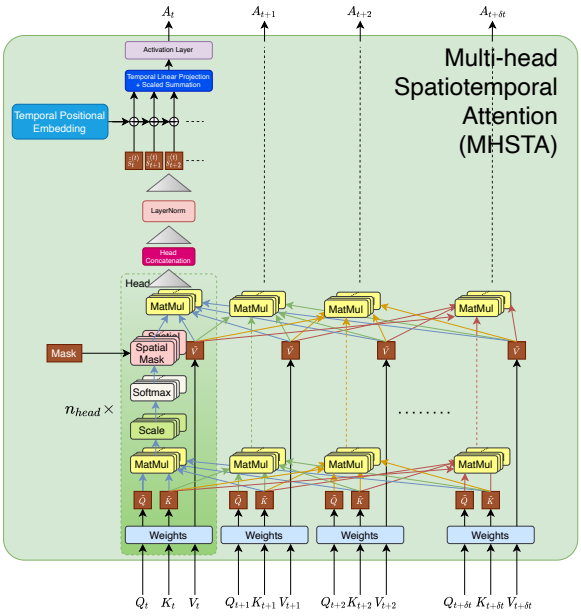

Figure 3: Multi-head Spatiotemporal Attention Module

generating a set of plausible sequences. The primary approach we are taking here is to consider the problem as a quantile regression problem. We follow the formulation given in Rodrigues et al. [25], with the first step being the prediction of the quantile values in each forecast grid cell. The quantile level of the values lies between 0 and 1 (i.e., quantile level is denoted as $q \in [0, 1]$). Multiple quantile levels can be predicted, but typically, two quantile levels, namely the upper quantile q_u and lower quantile q_l, are sufficient to predict the range of values the forecast can take, with a probability $\alpha = q_u - q_l$, in which $q_u > q_l$. Here, α is also known as the confidence level of the prediction interval. This can be derived easily from the definition of the quantiles themselves. It is also easily known that different quantile levels will yield varying confidence in prediction intervals.

Thus, by reformulating Equation 1, we can define the equation to calculate the quantile levels as follows.

$$\hat{X}_{T+1:T+\Delta T,q} = F_q(X_{1:T}), q \in Q, \tag{2}$$

where Q is a set of quantile values $Q \in [0, 1]$, and F_q is a function defining the model to calculate q-quantile values. Our aim is to find the set of functions $\{F_q\}_{q \in Q}$. In the quantile regression models introduced in this paper, the functions are estimated by one model trained to minimize the quantile loss summed for each quantile level. This will be elaborated further in the next section.

3 SPATIOTEMPORAL TRANSFORMER

3.1 Model Summary

We propose Spatiotemporal Transformer as an extension to the existing Transformer model to capture both spatial and temporal dependencies in a single stack. The structure of the model is inspired by Vision Transformer (ViT) model by Dosovitskiy et al. [9], in which the Transformer model takes ordered patches of an image as input to extract features (akin to CNNs) for image classification tasks. The main idea of our proposal is to extend this notion to extract 2D spatial features for each timestep using self-attention, and to extract the spatial correlation from other timesteps using a conventional attention mechanism. These attention maps are then stacked temporally for each timestep, producing "spatiotemporal" maps, which are then projected using a linear layer, averaged, and

passed through an activation layer. Before projection, we add an additional temporal embedding to encode the temporal position in each attention map. We then follow the seq2seq structure introduced in the original Transformer model [32] and replace the multi-head attention module with a multi-head version of the proposed spatiotemporal attention mechanism in both the encoder and decoder. The use of spatiotemporal attention allows us to maintain two spatial dimensions in the model. The process flow of the model is shown in Figure 1. We describe the proposed model in detail in the following section.

3.2 Model Description

We first formulate the components of Spatiotemporal Transformer. Following the previous Section 2, let $X_{1:T}$ be past observations and $X_{T+1:T+\Delta T}$ be future observations. We then define past observations as an input sequence input to an encoder model, and define future observations as a target sequence, input to a decoder model. During training, the encoder model receives input from the past observations. However, the decoder model receives shifted future observations including the immediate last timestep from past observations, namely observations from T to $T + \Delta T - 1$ ($X_{T:T+\Delta T-1}$). Based on this data, the decoder model outputs predictions $\hat{X}_{T+1:T+\Delta T}$. As with the original transformer, we mask the first attention module of the decoder to prevent the model from "peeking" to the future when generating predictions. Explanation of the forecasting flow can be found in Section 4.2. Both the encoder and decoder structures are shown in Figure 2. Moreover, the equations describing the pre-encoder and decoder layers, encoder layer, and decoder layer can be referred to in the appendix.

3.2.1 Encoder. In the encoder model, we first input the target sequence in a patch divider layer. The patch divider layer divides each timestep's 2D grid feature into patches of equal width and height. The resulting patches are then flattened and run through a linear projection layer, which projects the patches into features of d_{model} dimension, which is the main dimensionality of the model. A learnable positional embedding is added to each projected patch for each timestep to encode the patch position relative to the original 2D grid features, following Dosovitskiy et al. [9].

The core of the model is layer stacks of multi-head spatiotemporal attention layers and feed-forward layers. The input is first normalized before being input to the spatiotemporal attention layer. The spatiotemporal attention layer calculates not only the scaled-dot product spatial self-attention of each timestep, but also the spatial attention with respect to other timesteps, which are then concatenated head-wise and temporally for each timestep. This is finally utilized to extract spatiotemporal features of the input. The output of this component is then added to the previously normalized input, forming a residual connection.

Finally, it is renormalized and run through the feed-forward network, refining the features. This process is repeated N times with N layers, following the original Transformer [32]. The final output of the encoder model is then input to the decoder model's intra-attention layer, which is also adapted from the original Transformer. In the vein of the seq2seq model, this is structured so that the encoded features are passed to the decoder with minimal information loss.

3.2.2 Decoder. Similar to the encoder model, the shifted future observations $X_{T:T+\Delta T-1}$ are passed to preprocessing layers consisting of a patch divider layer, a linear projection layer, and a positional embedding. However, the main structure is modified with a masking process on the multi-head spatiotemporal attention layer and a new intra-attention layer before the feed-forward network. The additional mask input on the first attention layer prevents future observations from affecting the attention calculation of current observation, maintaining the regular information flow (i.e., preventing the model from cheating in predicting the observation of the current timestep). The intra-attention layer is the same as the multi-head spatiotemporal attention layer, with the keys and values (context) changed to encoder outputs. The resulting features are then passed through the feed-forward layer to extract the features. As with the encoder model, each layer is equipped with layer normalization and residual connections. In addition, this combination of layers is stacked N times to extract higher-level features. The last output of the stack is then put into a projection layer to return the dimension to the original patch size, which is then combined to generate 2D grid features equal to the original size as forecasts.

3.2.3 Multi-head Spatiotemporal Attention. The multi-head spatiotemporal attention mechanism is the main component of our proposed model. In this segment, the features from the inputs are projected into query, key, and value tensors. These inputs are projected along each timestep with shared weights and biases among the timesteps. After that, for each timestep, we calculate the self-attention (query, key, value coming from the same timestep) and normal attention with respect to other timesteps (with a query from the current timestep and key-value pairs from other timesteps). These attentions have the form of scaled dot-product attention. The resulting features from these attentions are stacked temporally according to the order of the key-value pair timestep. For the intra-attention in the decoder, the key-value pairs are extracted from the encoder's immediately previous outputs. The attentions essentially extract spatial dependencies within the queries, keys, and values, in which the combination of normal attentions can be considered as also taking the temporal dependencies together, thus the name of spatiotemporal attention. We also employ the multi-headed paradigm to the mechanism during the dot-product attention calculation, hence the multi-head name. We describe this component using the following equations. Let $Q, K, V \in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{model}}}$ be the query, key, and value, $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$ be the weights, t, p, i be temporal, patch, and head index, respectively, then

(Query, key and value) for every t, p and i

$$\tilde{Q}_{t,i} = Q_t W_i^Q, \tilde{Q} \in \mathbb{R}^{T \times n_{\text{head}} \times n_{\text{patch}} \times d_{\text{head}}}, \quad (3)$$

$$\tilde{K}_{t,i} = K_t W_i^K, \tilde{K} \in \mathbb{R}^{T \times n_{\text{head}} \times n_{\text{patch}} \times d_{\text{head}}}, \quad (4)$$

$$\tilde{V}_{t,i} = V_t W_i^V, \tilde{V} \in \mathbb{R}^{T \times n_{\text{head}}, \times n_{\text{patch}} \times d_{\text{head}}}. \quad (5)$$

(Attention score) For every t_1, t_2, i

$$S_{t_1,t_2,i} = \text{softmax}\Big(\frac{\tilde{Q}_{t_1,i}\tilde{K}_{t_2,i}^\top}{\sqrt{d_{head}}}\Big)\tilde{V}_{t_2,i}, S \in \mathbb{R}^{T \times T \times n_{\text{head}} \times n_{\text{patch}} \times d_{\text{head}}}, \quad (6)$$

where $d_{\text{head}} = \frac{d_{\text{model}}}{n_{\text{head}}}$ with n_{head} is the number of heads and $t_1 = t, t_2 = t$ is the first original temporal axis and an additional

temporal axis within the original temporal axis. The masked version of this attention mechanism in the decoder model essentially blocks out the future values during the attention calculation. We do this by forcefully setting the attention scores from future timesteps to zero-valued matrices. In contrast to the original Transformer, setting the values to extremely small numbers or negative infinity before the softmax layer in the attention layer does not work. This can be inferred directly from the softmax equation itself, as it results in values equal to $\frac{1}{d_{\text{model}}}$.

After calculating the attention features, we concatenate the heads together and normalize the outputs. We then add learnable temporal embedding, which is shared over encoder and decoder models on the second temporal axis. Finally, we merge the attention features over the second temporal axis to produce high-level spatiotemporal features for each timestep by running them through a fully connected layer and adding them together before passing the sum to an activation layer. Before passing the input through the fully connected layer, we scaled the values with the reciprocal of the timestep length, essentially trying to calculate the temporal average of the projected outputs. The process is expressed as follows. Let $\text{TE} \in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{model}}}$ be temporal embedding, W_{t_2}, b_{t_2} be the weights and biases of the projection layer for t_2, then

(Multi-head concatentation) for every t_1, t_2

$$\begin{aligned} \tilde{S}_{t_1,t_2} &= \text{Concatenate}(S_{t_1,t_2,1}, S_{t_1,t_2,2}, \ldots, S_{t_1,t_2,n_{\text{head}}}), \\ \tilde{S} &\in \mathbb{R}^{T \times T \times n_{\text{patch}} \times d_{\text{model}}} \end{aligned} \tag{7}$$

(Temporal embedding) For every t_1

$$\tilde{\tilde{S}}_{t_1} = \text{LayerNorm}(\tilde{S}_{t_1}) + \text{TE}, \tilde{\tilde{S}} \in \mathbb{R}^{T \times T \times n_{\text{patch}} \times d_{\text{model}}} \tag{8}$$

(Projection) For every t_1 and p

$$\begin{aligned} \text{MHSTA}_{t_1,p} &= \text{Act}\Big(\sum_{t_2=1}^{T} \Big(W_{t_2} \Big(\frac{\tilde{\tilde{S}}_{t_1,t_2,p}}{T} \Big) + b_{t_2} \Big) \Big), \\ \text{MHSTA} &\in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{model}}} \end{aligned} \tag{9}$$

where Act is the activation layer. The full process flow of this attention module is shown in Figure 3.

3.2.4 Feed-forward Network. The feed-forward network here is a direct application of the original Transformer's feed-forward network, with the exception of an additional activation layer added to the final output.

4 TRAINING AND FORECASTING FLOW

4.1 Training Procedure

The training procedure for our model is described as follows. All data available in the dataset was split into inputs for the encoder and outputs for the decoder. As mentioned above, we assign past observations $X_{1:T}$ for the encoder and shifted future observations $X_{T:T+\Delta T-1}$ for decoder. The model output is the forecasted values $X_{T+1:T+\Delta T}$. We then calculate the loss between the forecasted values and the ground truth of the data according to the following deterministic and quantile regression objective. Note that the objective is calculated over batches of the dataset (mini-batch training).

4.1.1 Deterministic. For deterministic forecasting, the models are trained to minimize the mean squared error (MSE) loss between ground truth and prediction over time and 2D spatial dimensions. This objective is also applied to the baseline models in the experiments.

4.1.2 Quantile Regression. In quantile regression, instead of predicting a conditional mean, we try to predict multiple conditional quantiles. The advantage of predicting multiple quantiles comes in the form of being able to generate prediction intervals. Having a prediction interval allows us to estimate the ranges in which future values are expected to fall into; this is very important in estimating the uncertainty of the predictions. For example, predicting 10th and 90th percentile yields a confidence interval of $\alpha = 90\% - 10\% = 80\%$ from the definition of the quantile itself.

While training, we try to minimize the tilted loss, also known as quantile loss. To produce the quantiles, we modify the final layer of the model with multiplies of the original hidden size. With this modification, we regularize the model to have the same latent space from which to generate multiple quantiles. Similar to the deterministic objective, in the experiments, we follow the same procedure for every baseline model.

4.2 Forecasting Flow

We describe our method for forecasting the values given the following types of models. Note, however, that in the following experiment, we focus on evaluating baselines' and our model's forecasting capability using the one-step-ahead method.

4.2.1 Deterministic. Compared to the more popular LSTM, as Transformer models do not have a recurrent structure, it is harder to predict the forecast in a recursive manner, which is the standard in the field. However, it is not entirely impossible to do so. In the experiments of this paper, during training and the evaluation, Transformer models assume that we already have the observations $X_{1:T+\Delta T-1}$, which will be divided into input sequence $X_{1:T}$ and target sequence $X_{T:T+\Delta T-1}$ (again, note the temporally right-shifted target sequence). We can then run the model, and in principle, the model will generate forecasts for $\hat{X}_{T+1:T+\Delta T}$. In the experiment, we utilized the forecasts generated with this method. Even though this is unrealistic, as it is impossible to have access to future observations in advance, the masked attention in the decoder prevents the model from inferring the forecast using future values, which keeps the model's autoregressive characteristic and provides a fair comparison with other baseline models. The model uses only past observations to infer each timestep's forecast. In a real-world application, one can perform a further one-step-ahead prediction by shifting the input and target sequence by one timestep into the future to produce forecasts for $X_{T+\Delta T+1}$ and repeat accordingly, essentially acting as a normal autoregressive model.

When the available observations are insufficient, one can fill the unavailable observations with 0 tensors, as they will be masked by the model, preventing contaminated forecasts from being generated. Additionally, multi-step-ahead forecasts can be performed by applying the same procedure but replacing future inputs with the previous prediction.

4.2.2 Quantile Regression. Generating forecasts with quantile regression versions of the models is essentially the same as the deterministic predictions, with the key difference that the models generate multiple quantiles instead of point forecasts.

5 EXPERIMENTS AND EVALUATION

In this section, we explain the experimental setup and the evaluation metrics used to compare the proposed model to baseline models. We then present the evaluation results of the models and discuss the implications of the results. Further details of each metric's equation, dataset, preprocessing, and model and training parameters are described in the appendix.

5.1 Experiment Detail and Metrics

Here, we aim to evaluate the performance of Spatiotemporal Transformer compared to the baselines. We divide the evaluation criteria into point predictions and probabilistic forecasts. For point predictions, we evaluate the models with two metrics: MSE and mean absolute error (MAE). The metrics here basically calculate the average of the corresponding type of error per grid in the data. In this paper, we report all error metrics in their normalized forms. The details of normalization are explained in the appendix. The metrics here were averaged over all data with timesteps of 10.

For probabilistic forecasts, we evaluate the results using quantiles generated by each model. We use the quantile loss (QL) as described in the previous section. Here, we train the models to generate quantiles in the 10th, 50th (median), and 90th percentiles, with which we can calculate a prediction interval of 80%. In addition, we also evaluated the quantiles on two additional metrics, interval coverage percentage (ICP) and mean interval length (MIL), following Rodrigues et al. [25]. ICP gives the percentage of the real observation that falls into the prediction interval, which in this case is preferably around 80%. On the other hand, MIL calculates the average length of the prediction interval generated by the models. A large interval length is usually associated with a large ICP. However, the desired result is a model providing broad coverage while maintaining a small MIL, as a model can obtain broader coverage of observation by enlarging its MIL, which means that the prediction interval given by model with large MIL is sub-optimal compared to models with tighter MIL [25].

5.2 Datasets

We evaluated the models on the Moving MNIST dataset [28], representing an evaluation of a video prediction task. The Moving MNIST dataset consists of videos of two moving handwritten digits reminiscent of the MNIST dataset [17]. In this video sequences, digits move in one direction until they reach the boundary, and then reflect off the image frame. Overlapping and occlusion occur when the digits move past each other. These dynamics make the dataset elementary and sufficiently suitable as a toy dataset for spatiotemporal model evaluations. Indeed, it has been used in several state-of-the-art research works.

The second dataset, the CPC Merged Analysis of Precipitation (CMAP) data [39] was used to evaluate the models on their prediction capabilities when faced with chaotic dynamics. We obtained the data from the official website (`https://psl.noaa.gov/`) of the National Oceanic and Atmospheric Administration (NOAA)/Oceanic and Atmospheric Research (OAR)/Earth System Research Laboratories (ESRL), Physical Sciences Laboratory (PSL), Boulder, Colorado, USA. The CMAP data contains rain gauge measurements of the global gridded precipitation rate (mm/day). We believe that this data poses a substantial challenge for evaluating spatiotemporal sequences due to its chaotic dynamics and the relatively long gap between measurements. Note that the data being evaluated here is single-channel, focusing on the evaluation of data with single-channeled spatiotemporal sequences.

Table 1: Deterministic prediction results on Moving MNIST dataset. Bolded and italicized numbers denote best performance excl. ConvLSTM and incl. ConvLSTM, respectively.

Model	MSE (x10^2)	MAE (x10^2)
SpaTrans (Ours)	**1.243±0.0502**	***3.049±0.0818***
Transformer	2.051±0.0029	4.659±0.0124
LSTM	2.461±0.0068	6.259±0.0071
LSTM Seq2Seq	2.455±0.0062	6.174±0.0151
LSTM Seq2Seq + Attn	2.460±0.0061	6.200±0.0089
ConvLSTM	*1.242±0.0037*	3.090±0.0134

5.3 Model and Training Setup

For comparison baselines, we employ Transformer, LSTM with linear projection, the seq2seq version of the former, and the seq2seq version with additional scaled dot-product attention between the encoder and decoder hidden states. Further, we also employ ConvLSTM for comparison with state-of-the-art method. For LSTM and Transformer, we flattened the 2D spatial data into 1D data, as they only accept 1D sequences. We also modified Transformer's FFN layer to have a final activation layer, after our Spatiotemporal Transformer model.

Note that we replace all the activation layers other than softmax, tanh, and sigmoid in the models with GeLU activation layers [12]. For Transformer-based models, we also changed the hidden size of the middle linear layer of FFNs to double that of d_{model}. We also added dropout layers to the models for additional regularization. We used AdamW as the optimizer [20].

Additionally, we trained each model five times and then averaged all metrics with respect to those runs. We also report the standard errors given by the variability of the runs for each metric. We employ early stopping with a tolerance of ten epochs (stop the training if there is no improvement over ten epochs), with the best performing model in validation used to evaluate performance in testing, whose performance is reported in the next section. The code for our model and experiment can be found here [1].

5.4 Evaluation Results

In this section, we present the results of the conducted experiments.

5.4.1 Moving MNIST. The evaluation results in the deterministic point forecasts category are reported in Table 1. The evaluation results for the probabilistic forecasting task are presented in Table

[1] https://anonymous.4open.science/r/84428b05-d9ca-4647-8970-622f97bc42c5/

Table 2: Probabilistic prediction results on Moving MNIST dataset. Bolded and italicized numbers denote best performance excl. ConvLSTM and incl. ConvLSTM, respectively.

Model	MSE (x10^2)	MAE (x10^2)	QL (x10^2)	ICP (80%)	MIL (80%)
SpaTrans (Ours)	***3.390±0.3401***	**4.589±0.2727**	***5.165±0.5326***	0.947±0.0231	0.186±0.0251
Transformer	4.212±0.0001	4.932±0.0004	12.723±0.0146	0.019±0.0000	0.199±0.0003
LSTM	4.217±0.0001	4.913±0.0008	6.082±0.1298	0.829±0.0059	0.192±0.0058
LSTM Seq2Seq	4.218±0.0001	4.906±0.0013	6.645±0.0146	0.912±0.0107	0.200±0.0043
LSTM Seq2Seq + Attn	4.217±0.0002	4.910±0.0005	6.455±0.1618	0.900±0.0110	0.198±0.0052
ConvLSTM	3.695±0.5235	*4.440±0.4553*	6.474±1.0091	0.912±0.0050	0.027±0.0148

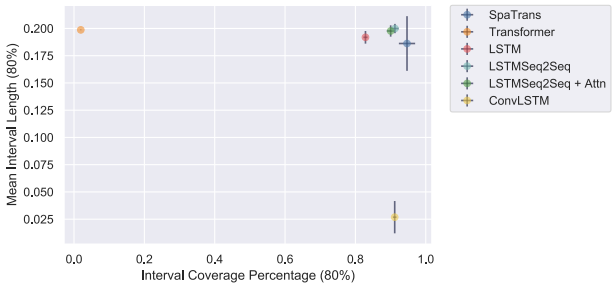

Figure 4: ICP-MIL plot on Moving MNIST dataset. Black lines indicates error bar visualized using standard error.

Table 3: Deterministic prediction results on CMAP dataset. Bolded and italicized numbers denote best performance excl. ConvLSTM and incl. ConvLSTM, respectively.

Model	MSE (x10^4)	MAE (x10^3)
SpaTrans (Ours)	**3.220±0.0041**	**8.957±0.0508**
Transformer	3.460±0.0058	9.289±0.0227
LSTM	3.610±0.0144	9.529±0.0270
LSTM Seq2Seq	3.967±0.1213	10.340±0.2205
LSTM Seq2Seq + Attn	3.456±0.0050	9.374±0.0059
ConvLSTM	*3.208±0.0045*	*8.866±0.0322*

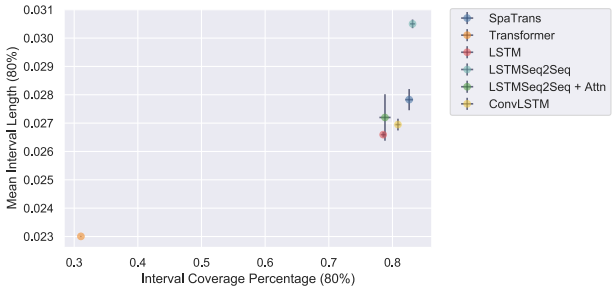

Figure 5: ICP-MIL plot on CMAP dataset. Black lines indicates error bar visualized using standard error.

2. We divided the categories of the models using horizontal lines for easier categorization. We also plotted ICP with respect to MIL for a better understanding of the reported results, which is shown in Figure 4.

As shown in the tables, in the point prediction category, our Spatiotemporal Transformer model managed to give the lowest

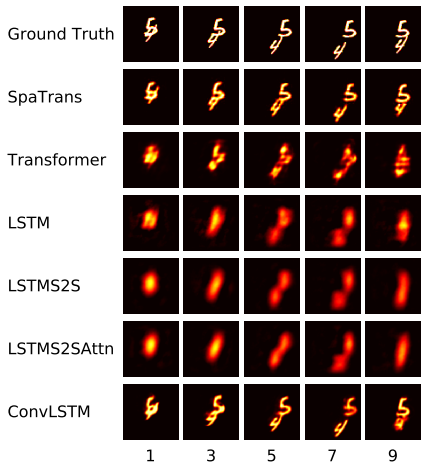

Figure 6: Visualization on moving MNIST prediction result conditioned on previous 10 past frames (not shown). Shown here is the one-step ahead prediction on the 1st, 3rd, 5th, 7th, and 9th frames.

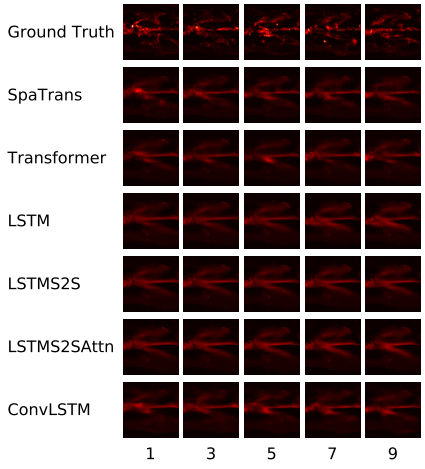

Figure 7: Visualization on CMAP prediction result conditioned on previous 10 past frames (not shown). Shown here is the one-step ahead prediction on the 1st, 3rd, 5th, 7th, and 9th frames.

MSE and MAE compared to other model categories, beating vanilla

Table 4: Probabilistic prediction results on CMAP dataset. Bolded and italicized numbers denote best performance excl. ConvLSTM and incl. ConvLSTM, respectively.

Model	MSE (x10^4)	MAE (x10^3)	QL (x10^3)	ICP (80%)	MIL (x10^2, 80%)
SpaTrans (Ours)	**3.667±0.1131**	***8.588±0.0902***	8.620±0.1349	0.826±0.0061	2.783±0.0374
Transformer	4.523±0.0006	9.565±0.0002	12.317±0.0047	0.311±0.0002	2.300±0.0011
LSTM	3.718±0.0070	8.629±0.0086	***8.529±0.0117***	0.785±0.0038	2.659±0.0095
LSTM Seq2Seq	4.517±0.0098	9.481±0.0066	9.593±0.0418	0.832±0.0041	3.050±0.0153
LSTM Seq2Seq + Attn	3.827±0.1776	8.711±0.1942	8.652±0.2181	0.788±0.0088	2.720±0.0820
ConvLSTM	*3.618±0.0142*	8.589±0.0375	8.602±0.0573	0.809±0.0040	2.695±0.0206

Transformer and the whole LSTM variant baseline. We also note that it had a comparable performance with ConvLSTM, albeit with slightly higher MSE. This shows the improvement that our spatiotemporal attention mechanism brings to the vanilla Transformer.

On the probabilistic side, our Spatiotemporal Transformer again thrives on all error metrics (MSE, MAE, and QL) compared to other baselines. As with the deterministic result, it outperforms Transformer and the whole LSTM model family by quite a significant margin, also giving a competitive result compared with ConvLSTM (slightly worse results on MSE and MAE, but outperforming ConvLSTM on QL). This again shows that the improvement Spatiotemporal Transformer brings is not limited to deterministic point prediction but also encompasses probabilistic prediction, in this case, quantile regression.

Shifting our focus to the ICP and MIL results, we see that all models' ICP values cover more than the prediction interval, which is 80%. This is to be expected, as the moving MNIST dataset consists of comparably non-chaotic dynamics and easy-to-guess movements. Interestingly, the vanilla Transformer model seems to stand out here, giving extremely bad coverage over the forecasts. This might be because of its inability to model probabilistic quantiles, as there is only a linear layer responsible for producing all quantiles at the output layer. Another possibility is that the model is more challenging to train than other models, falling into a suboptimal region of the training curve easily, reflected by its comparably large QL error. This favorably shows that by extending the model to Spatiotemporal Transformer, its capability of modeling probabilistic quantiles is significantly enhanced.

Looking at the ICP and MIL plot, we can clearly see that models other than Transformer congregate on the plot's high-coverage, low-interval length region. These results agree with our earlier deduction that all models give reasonable prediction intervals over the forecasts, even though the coverage is moderately higher than the expected 80% because of the low dynamics of the dataset. We can also see that Transformer provides inadequate coverage even though its MIL is within the acceptable range.

Focusing on the high-coverage, low-interval region of the plot, we see that ConvLSTM gives the best ICP to MIL ratio owing to its proximity to 80% prediction coverage and its reasonably tight MIL. The Spatiotemporal Transformer also gives a high coverage percentage compared to ConvLSTM, with a lower MIL than other LSTM variants, again confirming its superiority over LSTMs and vanilla Transformer on both deterministic and probabilistic forecasting tasks.

5.4.2 CMAP. The evaluation results in the deterministic point forecasts category are reported in Table 3. The evaluation results for the probabilistic forecasting task are presented in Table 4. Similar to the previous dataset, we also plotted the ICP with respect to MIL, which is shown in Figure 5.

The results seem to indicate that the trend on the Moving MNIST dataset translates well to real-world data with chaotic dynamics. On point prediction, our Spatiotemporal Transformer again outperformed all baselines on both the MSE and MAE metric, albeit with a slightly worse performance than ConvLSTM. Nonetheless, it shows that the spatiotemporal attention mechanism extends well, even to real-world data. One thing to note here is that, as with the Moving MNIST dataset, Transformer outperforms LSTM models in the deterministic category. This reinforces the strong inference power given by the attention mechanism [37].

We also observe the same trend in the probabilistic forecast results, with our model giving the best performance over MSE and MAE, except QL, in which LSTM takes the lead by a small margin. Our model also gives comparable performance compared to ConvLSTM. This shows the robustness of the Spatiotemporal Transformer even on quantile forecasting of real-world dynamics. We note that all of the models (except Transformer) managed to better capture the 80% prediction interval than the Moving MNIST dataset, with most of them hovering around 80% coverage. We observe again the trend of Transformer performing suboptimally with high errors and low coverage, confirming its characteristic of being harder to train on quantile regression tasks.

Focusing on the region where models other than vanilla Transformer congregate, even though LSTM takes the lead on QL metric, we note that LSTM and its seq2seq attention variant fails to reach the 80% prediction interval, unlike other models, giving subpar performance. On the contrary, ConvLSTM provides the tightest bound on the interval with exceptionally good prediction intervals compared to the other models. The Spatiotemporal Transformer was second only to ConvLSTM, echoing its robustness in not only the deterministic forecasting task, but also the task of quantile prediction compared to the Transformer and LSTM baselines.

5.5 Discussion

The experiment results clearly demonstrate that the combination of spatial attention inspired by Vision Transformer [9] and temporal attention composed of conventional attention and self-attention mechanisms shows significant improvement over the standard Transformer and LSTM baselines in both deterministic and probabilistic spatiotemporal forecasting settings. Moreover, the model

also outperforms Transformer on ICP and MIL metrics, exhibiting its robustness in generating plausible prediction intervals and capturing real forecasts well.

However, the model is not without limitations, as we can note that the model struggles to surpass ConvLSTM, albeit giving close results. There are several reasons why the model is still underperforming. Over the forecasts, qualitative observations (Figures 6 and 7) show that Spatiotemporal Transformer cannot capture spatial dependencies of patch edges when generating the forecast in the output layer. Some forecasts produced by this model have an obvious non-continuous disconnection between patches. This is because the output layer is managed by only a single layer over all patches in all timesteps. We believe that changing the output layer to encompass all patches can solve this problem, and indeed it does solve the problem of patch edge spatial dependencies on our preliminary experiments. However, the resulting MSE and MAE are suboptimal compared to the "single output layer for all patches" solution. We believe that this is a significant limitation to be addressed, and we seek to pursue this future work. However, the resulting visualization on Moving MNIST data shows that the model is very capable of producing sharp images that rival ConvLSTM.

Increasing the patch size in our model might increase the performance, but it comes at the cost of computational complexity. We believe that this aspect merits improvement. We envision an approach leveraging existing computational upgrades of Transformers [6, 16, 33] to be applied to the model, which would position it better in terms of application feasibility and environmental impact. Nevertheless, the results seem to favor Spatiotemporal Transformer, and in the long run, improvements on the model might lead it to surpass even the state-of-the-art methods in the field of spatiotemporal sequence forecasting.

6 CONCLUSION

This paper has introduced an extension of Transformer named Spatiotemporal Transformer utilizing a spatiotemporal attention mechanism to effectively model spatiotemporal sequence data. Experimental results show that it outperformed all baselines and performed comparably to ConvLSTM, on which all current state-of-the-art models are based, showing its potential as a new contender in the field of spatiotemporal sequence forecasting. As with NLP and computer vision tasks, it is expected that further work on the model can possibly push the performance to rival current state-of-the-art methods.

REFERENCES

[1] Emre Aksan et al. 2020. A Spatio-temporal Transformer for 3D Human Motion Prediction. (2020). arXiv:2004.08692 [cs.CV]
[2] Alexander Alexandrov et al. 2020. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research* 21, 116 (2020), 1–6.
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.
[4] Irwan Bello. 2021. Lambda Networks: Modeling long-range Interactions without Attention. In *ICLR*.
[5] Kyunghyun Cho et al. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.
[6] Krzysztof Choromanski et al. 2021. Rethinking Attention with Performers. In *ICLR*.
[7] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. 2020. On the Relationship between Self-Attention and Convolutional Layers. In *ICLR*.
[8] Jacob Devlin et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.
[9] Alexey Dosovitskiy et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.
[10] Shen Fang et al. 2019. GSTNet: Global Spatial-Temporal Network for Traffic Flow Prediction. In *IJCAI*.
[11] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. (2014). arXiv:1410.5401 [cs.NE]
[12] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). (2016).
[13] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *ACL*.
[14] Xu Jia et al. 2016. Dynamic Filter Networks. In *NeurIPS*.
[15] Eugenia Kalnay et al. 1996. The NCEP/NCAR 40-Year Reanalysis Project. *Bulletin of the American Meteorological Society* 77, 3 (1996), 437 – 472.
[16] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *ICLR*.
[17] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/
[18] Bryan Lim et al. 2019. Temporal fusion transformers for interpretable multi-horizon time series forecasting. (2019). arXiv:1912.09363 [stat.ML]
[19] Zhihui Lin et al. 2020. Self-Attention ConvLSTM for Spatiotemporal Prediction. In *AAAI*.
[20] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *ICLR*.
[21] Wenjie Luo et al. 2016. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. In *NeurIPS*.
[22] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.
[23] Matthew Peters et al. 2018. Deep Contextualized Word Representations. In *NAACL*.
[24] Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to Generate Reviews and Discovering Sentiment. (2017). arXiv:1704.01444 [cs.LG]
[25] Filipe Rodrigues and Francisco C. Pereira. 2020. Beyond Expectation: Deep Joint Mean and Quantile Regression for Spatiotemporal Problems. *IEEE Transactions on Neural Networks and Learning Systems* 31 (2020), 5377–5389.
[26] Xingjian Shi et al. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NeurIPS*.
[27] Xingjian Shi et al. 2017. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. (2017). arXiv:1706.03458 [cs.CV]
[28] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. 2015. Unsupervised Learning of Video Representations using LSTMs. In *ICML*.
[29] Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning. In *ICML*.
[30] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*.
[31] Hugo Touvron et al. 2020. Training data-efficient image transformers & distillation through attention. (2020). arXiv:2012.12877 [cs.CV]
[32] Ashish Vaswani et al. 2017. Attention is All you Need. In *NeurIPS*.
[33] Sinong Wang et al. 2020. Linformer: Self-Attention with Linear Complexity. (2020). arXiv:2006.04768 [cs.LG]
[34] Yunbo Wang et al. 2017. PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs. In *NeurIPS*.
[35] Yunbo Wang et al. 2018. PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning. In *ICML*.
[36] Yunbo Wang et al. 2019. Memory in Memory: A Predictive Neural Network for Learning Higher-Order Non-Stationarity From Spatiotemporal Dynamics. In *CVPR*.
[37] Neo Wu et al. 2020. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. (2020). arXiv:2001.08317 [cs.LG]
[38] Yonghui Wu et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. (2016). arXiv:1609.08144 [cs.CL]
[39] Pingping Xie and Phillip A. Arkin. 1997. Global Precipitation: A 17-Year Monthly Analysis Based on Gauge Observations, Satellite Estimates, and Numerical Model Outputs. *Bulletin of the American Meteorological Society* 78, 11 (1997), 2539 – 2558.
[40] Mingxing Xu et al. 2020. Spatial-Temporal Transformer Networks for Traffic Flow Forecasting. (2020). arXiv:2001.02908 [eess.SP]
[41] Cunjun Yu et al. 2020. Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction. In *ECCV*.
[42] Zhengyan Zhang et al. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *ACL*.
[43] Lei Zhu, Tianrui Li, and Shengdong Du. 2019. TA-STAN: A Deep Spatial-Temporal Attention Learning Framework for Regional Traffic Accident Risk Prediction. In *IJCNN*.

A APPENDIX

A.1 Model Equations

A.1.1 Pre-Encoder and Decoder Layers. The pre-encoder and decoder layers can be summarized with the following equations.

$$X_{1:T}^{(\text{patched})} = \text{PatchDivider}(X_{1:T}) \in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{patch}}}, \tag{10}$$

$$X_{1:T,1:n_{\text{patch}}}^{(\text{projected})} = W_p X_{1:T,1:n_{\text{patch}}}^{(\text{patched})} + b_p \in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{model}}}, \tag{11}$$

$$\tilde{X}_{1:T}^{(1)} = X_{1:T}^{(\text{projected})} + \text{PE} \in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{model}}}, \tag{12}$$

where n_{patch} and d_{patch} are the number of patches and flattened dimension of each patch, respectively, $W_p \in \mathbb{R}^{d_{\text{model}} \times d_{\text{patch}}}$ and $b_p \in \mathbb{R}^{d_{\text{model}}}$ are respectively the trainable weight and bias of the linear projection layer, and $\text{PE} \in \mathbb{R}^{n_{\text{patch}} \times d_{\text{model}}}$ is positional embedding.

A.1.2 Encoder Layer. The following equations show the process of each encoder layer stack.

$$A_{1:T}^{(n)} = \tilde{X}_{1:T} + \text{MHSTA}(\text{LayerNorm}_1(\tilde{X}_{1:T}^{(n)}, \tilde{X}_{1:T}^{(n)}, \tilde{X}_{1:T}^{(n)})), \in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{model}}} \tag{13}$$

$$\tilde{X}_{1:T}^{(n+1)} = A_{1:T} + \text{FFN}(\text{LayerNorm}_2(A_{1:T}^{(n)})) \in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{model}}}, \tag{14}$$

where $\text{MHSTA}(Q, K, V)$ is the multi-head spatiotemporal attention function with query, key, and value inputs, and $\text{FFN}(X)$ is the feed-forward layer. $n = 1, 2, \ldots, N$ is the index of the stack. Note that every stack has independent attention, normalization, and feed-forward layers (weights are not shared among stacks).

A.1.3 Decoder. The following equations describe the main structure of the decoder model.

$$A_{T:T+\Delta T-1}^{(n)} = \tilde{X}_{T:T+\Delta T-1} + \text{MaskedMHSTA}(\text{LayerNorm}_1(\tilde{X}_{T:T+\Delta T-1}^{(n)}, \tilde{X}_{T:T+\Delta T-1}^{(n)}, \tilde{X}_{T:T+\Delta T-1}^{(n)})), \tag{15}$$

$$\tilde{A}_{T:T+\Delta T-1}^{(n)} = \tilde{A}_{T:T+\Delta T-1}^{(n)} + \text{MHSTA}(\text{LayerNorm}_2(A_{T:T+\Delta T-1}^{(n)}, \tilde{X}_{1:T}^{(N+1)}, \tilde{X}_{1:T}^{(N+1)})), \tag{16}$$

$$\tilde{X}_{T:T+\Delta T-1}^{(n+1)} = \tilde{A}_{T:T+\Delta T-1}^{(n)} + \text{FFN}(\text{LayerNorm}_3(\tilde{A}_{T:T+\Delta T-1}^{(n)})). \tag{17}$$

At the last stack output

$$\hat{X}_{T+1:T+\Delta T} = \text{PatchCombiner}(W_f \text{LayerNorm}_f(\tilde{X}_{T:T+\Delta T-1,1:n_{\text{patch}}}^{(N+1)}) + b_f), \in \mathbb{R}^{T \times h \times w} \tag{18}$$

where $A_{T:T+\Delta T-1}^{(n)}, \tilde{A}_{T:T+\Delta T-1}^{(n)}, \tilde{X}_{T:T+\Delta T-1}^{(n+1)} \in \mathbb{R}^{T \times n_{\text{patch}} \times d_{\text{model}}}$ are the intermediate outputs, $\hat{X}_{T+1:T+\Delta T}$ are the forecasts. In addition, note that $\tilde{X}_{1:T}^{(N+1)}$ is the final output from the encoder. The $\text{MaskedMHSTA}(Q, K, V)$ is the masked version of multi-head spatiotemporal attention, $W_f \in \mathbb{R}^{d_{\text{patch}} \times d_{\text{model}}}, b_f \in \mathbb{R}^{d_{\text{patch}}}$ are the weights and biases of the final projection layer, and $\text{PatchCombiner}(X)$ rearranges the patches back into full 2D grid features.

A.2 Metric and Loss Equations

A.2.1 Mean Squared Error. The mean squared error is defined as follows.

$$\text{MSE} = \frac{1}{\Delta T \times M \times N} \sum_{t=1}^{\Delta T} \sum_{i=1}^{M} \sum_{j=1}^{N} (X_{t,i,j} - \hat{X}_{t,i,j})^2, \tag{19}$$

where $\Delta T, M, N$ is the prediction length number of rows, and of columns of the data, respectively, and $X, \hat{X}$ are the ground truth and prediction values.

A.2.2 Mean Absolute Error. The equation for mean absolute error is defined as follows.

$$\text{MAE} = \frac{1}{\Delta T \times M \times N} \sum_{t=1}^{\Delta T} \sum_{i=1}^{M} \sum_{j=1}^{N} |X_{t,i,j} - \hat{X}_{t,i,j}| \tag{20}$$

A.2.3 Quantile Loss. The definition of quantile loss is given as follows.

$$QL = \frac{1}{n_q \times \Delta T \times M \times N} \sum_{k=1}^{n_q} \sum_{t=1}^{\Delta T} \sum_{i=1}^{M} \sum_{j=1}^{N} (\max(q_k(X_{t,i,j} - \hat{X}_{t,i,j}), (q_k - 1)(X_{t,i,j} - \hat{X}_{t,i,j}))), \tag{21}$$

where n_q is the amount of quantiles and $q_k \in [0, 1]$ with $k = 1, 2, \ldots, n_q$ are the quantiles.

A.3 Dataset Details and Preprocessing

A.3.1 Moving MNIST. The dataset's video size was 64 × 64 pixels and consisted of videos with a length of 20 frames. We divided the frames into two groups of ten frames, with the first group assigned as input to the models and the second as the video to be predicted. We also normalized the data to the range [0, 1] with min-max scaling, with a minimum value of 0 and a maximum value of 255 (8-bit greyscale channel).

A.3.2 CMAP. We chose the enhanced version of the data with missing values interpolated using data assimilation methods [15, 39]. We chose the pentad-valued data from 1979/01 to 2017/01 (accessed on 2020/12/01) among several versions of the data available on the website. This created a larger dataset to work with, specifically, a total of 2774 timesteps. As input to the model, we generated overlapping sequences of 20 frames, with the former ten frames being input and the rest (also ten frames) as the target. As with the Moving MNIST dataset, we normalized the data to [0, 1] with min-max scaling. The range of the scaling is [0, 170]. Moreover, we cropped each frame of the data from the original size of 72 × 144 to 72 × 72, cropping the center of the frames.

A.3.3 Dataset Splitting. We divided both datasets with a ratio of 0.64 : 0.16 : 0.2 for training, validation, and testing. This yielded 6400, 1600, and 2000 data for Moving MNIST and 1775, 444, and 555 data for CMAP data. In addition, for CMAP data, we paid extra attention so that the generated sequence does not overlap with different datasets (validation and testing data does not contaminate training data and vice-versa. The same applies to the validation and testing data.).

Table 5: Specifications of computing nodes. The CPUs used were Intel®Core CPUs, while the GPUs were Nvidia®GPUs.

	CPU	GPU	RAM
1st	i7-6850K @ 3.60GHz	TITAN RTX 24GB	32GB
2nd	i5-10600 @ 3.30GHz	GeForce RTX 3090 24GB	64GB
3rd	i5-10600 @ 3.30GHz	GeForce RTX 3090 24GB	64GB

Table 6: Optimizer parameters

Parameters	Values
Optimizer	AdamW
Learning rate (LR)	0.0001
β_1	0.9
β_2	0.999
LR decay	0.01
Batch size	12

Table 7: Model parameters. Our model is highlighted with boldface, where Ours is Spatiotemoporal Transformer, T is Transformer, L1 is LSTM, L2 is LSTM Seq2Seq, L3 is LSTM Seq2Seq with Attention, and CL is ConvLSTM. Input sequence length here means the sequence length of the output layer of LSTM and ConvLSTM. In other words, it means the sequence length of past information used by the models to forecast the next timestep.

	Ours	T	L1	L2	L3	CL
Patch size	8×8	-	-	-	-	-
d_{model}	72	72	-	-	-	-
n_{head}	9	9	-	-	-	-
Hidden size	-	-	72	72	72	-
Hidden channels	-	-	-	-	-	8 (all layers)
Linear output size	-	-	72	72	72	-
Kernel size	-	-	-	-	-	3×3
Input seq. length	-	-	10	-	-	10
n_{layer}	6	6	6	6	6	6
Dropout rate	0.2	0.2	0.2	0.2	0.2	0.2

A.4 Model and Training Parameters

All of the models were trained on computing nodes interchangeably, as shown in Table 5.

The parameters for each optimizer are listed in Table 6. We used the same parameters for all models as best we could to maintain fairness, as shown in Table 7.

Our reasoning for setting the parameters in all ConvLSTM models to ensure fairness with Transformer-based models was due to the fact shown by Cordonnier et al. [7], that CNN layers can be approximated by multi-head self-attention (MHSA) networks when they satisfy the following conditions.

(1) The MHSA network utilizes relative positional encoding of dimensionality higher than or equal to 3 ($d_{\text{pos}} \geq 3$).
(2) The square root of the number of heads in MHSA is equal to the kernel width and height of the CNN (kernel = $\sqrt{n_{\text{head}}} \times \sqrt{n_{\text{head}}}$).
(3) The number of channels in the CNN is equal to the head dimension or output dimension of the MHSA, (channel = $\min(d_{\text{head}}, d_{\text{out}})$), whichever is smaller.

We followed the condition closely when setting the corresponding ConvLSTM and Transformer models, except for the fact that we used absolute positional encoding instead of relative positional encoding. However, we believe that the conditions work well enough as a reference point for the relationship between the Transformer-based models and the ConvLSTM models.