

Long-Range Transformers for Dynamic Spatiotemporal Forecasting

Jake Grigsby
University of Virginia
jcg6dn@virginia.edu

Zhe Wang
University of Virginia
zw6sg@virginia.edu

YanJun Qi
University of Virginia
yanjun@virginia.edu

Abstract

Multivariate Time Series Forecasting focuses on the prediction of future values based on historical context. State-of-the-art sequence-to-sequence models rely on neural attention between timesteps, which allows for *temporal* learning but fails to consider distinct *spatial* relationships between variables. In contrast, methods based on graph neural networks explicitly model variable relationships. However, these methods often rely on predefined graphs and perform separate spatial and temporal updates without establishing direct connections between each variable at every timestep. This paper addresses these problems by translating multivariate forecasting into a “spatiotemporal sequence” formulation where each Transformer input token represents the value of a single variable at a given time. Long-Range Transformers can then learn interactions between space, time, and value information jointly along this extended sequence. Our method, which we call Spacetimeformer, achieves competitive results on benchmarks from traffic forecasting to electricity demand and weather prediction while learning fully-connected spatiotemporal relationships purely from data.

1 Introduction

Multivariate forecasting attempts to predict future outcomes based on historical context and has direct applications to many domains, including science, policy, and business. Jointly modeling a set of variables allows us to interpret dependency relationships that provide additional context or early warning signs of changes in future behavior. Time Series Forecasting (TSF) models typically deal with a small number of variables with long-term *temporal* dependencies that require historical recall and distant forecasting. This is commonly handled by encoder-decoder sequence-to-sequence (seq2seq) architectures based on recurrent networks or one-dimensional convolutions [4]. Current state-of-the-art TSF models substitute classic seq2seq architectures for neural-attention-based mechanisms [62] [24]. However, these models represent the value of multiple variables per timestep as a single input token¹. This lets them learn “temporal attention” amongst timesteps but can ignore the distinct *spatial*² relationships that exist between variables. A temporal attention network is shown in Figure 1b.

¹An *input token* is one element of a Transformer’s input sequence. Attention mechanisms (Sec 2.1) learn representations by sharing information between tokens.

²In many real-world forecasting tasks, variables in close proximity influence one another or exhibit similar patterns. The term “spatial” refers to such relationships between multiple variables generally.

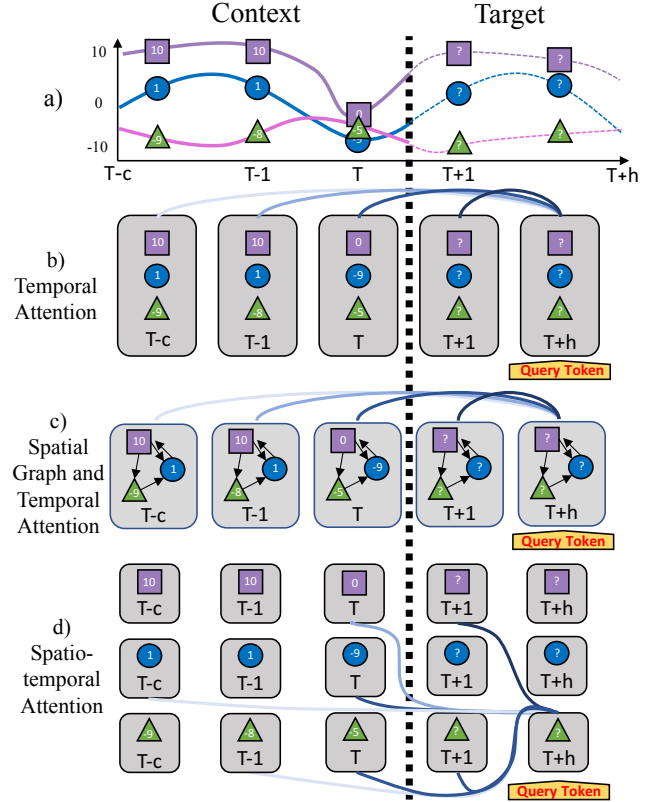


Figure 1: Attention in multivariate forecasting. (a) A three-variable sequence with three context points (left of the dashed vertical black line) and two target points that need predictions (right of the dashed line). (b) Temporal attention where each token contains all three variables. Darker blue lines between grey tokens represent increasing attention. (c) Temporal attention extended to have spatial connections within each token, which may be defined by known variable graphs (shown in black). (d) Spatiotemporal attention with all variables at each timestep as a separate token.

In contrast, forecasting models based on Graph Neural Networks (GNNs) explicitly model spatial dependencies. These methods typically involve alternating spatial and temporal layers in a deep network. Earlier work performs spatial learning with a graph convolution and temporal learning with a recurrent or convolutional sequence layer [25] [57]. Recently it has become more common to implement one or both of the spatial and temporal components with an attention-based message passing system [55] [35] [8] [37]. This process extends the temporal attention network with spatial graph-based learning amongst the variables of each token (Figure

1c). While these methods help address the spatial shortcoming of classic TSF sequence models, they struggle with at least one of three common problems:

- (1) They require a **predefined adjacency matrix** representing the ground-truth relationship between input variables.
- (2) They perform **separate temporal and spatial updates**. This can make it more difficult to share knowledge between different timesteps of different variables (Figure 2).
- (3) Their decoder architectures are designed for short-term spatial predictions and **struggle with long-term temporal forecasting**.

This paper proposes a simple approach that addresses all three problems, creating a unified method for multivariate forecasting that can be applied to problems with long-term temporal dependencies and complex spatial relationships. Let N be the number of variables we are predicting and L be the sequence length in timesteps. We flatten multivariate inputs of shape (L, N) into long sequences of shape $(L \times N)$ where each input token isolates the value of a single variable at a given timestep. The resulting input allows Transformer architectures to learn attention networks across both space and time jointly, creating a “spatiotemporal attention” mechanism. Spatiotemporal attention addresses problems (1) and (2) by creating a direct path between every token at every timestep and by learning spatial relationships purely from data (Figure 1d). An encoder-decoder Transformer architecture addresses problem (3) by enabling accurate and self-consistent long-term predictions. While the method is conceptually simple, its implementation presents significant engineering challenges due to the runtime and memory requirements of the Transformer attention mechanism at long sequence lengths. We utilize recent advances in efficient long-range attention mechanisms to generate competitive results in various domains spanning the spatiotemporal spectrum from traffic forecasting to electricity production and weather prediction.

2 Background and Related Work

2.1 Long-Range Transformers

The Transformer [45] is a deep learning architecture for sequence-to-sequence prediction that is widely used in natural language processing (NLP) [13]. Transformers operate on sequences of d -dimensional vectors $X \in \mathbb{R}^{L_x \times d}$, $Z \in \mathbb{R}^{L_z \times d}$, where L_x and L_z are sequence lengths and X can be equal to Z . The primary component of the model is the attention mechanism that updates the representation of tokens in X with information from Z . Tokens in Z are mapped to *key* vectors with learned parameters W^K , while tokens in X generate *query* vectors with W^Q . The dot-product similarity between query and key vectors is re-normalized to determine the attention matrix $A(X, Z) \in \mathbb{R}^{L_x \times L_z}$:

$$A(X, Z) = \text{softmax} \left(\frac{W^Q(X)(W^K(Z))^T}{\sqrt{d}} \right) \quad (1)$$

This paper will focus on Transformers and attention as a form of message passing along a dynamically generated adjacency matrix [20] [29] [58], where $A(X, Z)[i, j]$ denotes the strength of the connection between the i th token in X and the j th token in Z . The information passed along the edges of this graph are *value* vectors

of sequence Z , generated with parameters W^V . We create a new representation of sequence X according to the attention-weighted sum of $W^V(Z)$:

$$\text{Attention}(X, Z) \in \mathbb{R}^{L_x \times d} := A(X, Z)W^V(Z) \quad (2)$$

In an encoder-decoder Transformer, a stack of consecutive encoder layers observe a context sequence and perform self-attention between that sequence and itself ($\text{Attn}(Z, Z)$), leading to an updated representation of the context sequence Z' . A stack of decoder layers process the target sequence X and alternate between self-attention ($\text{Attn}(X, X)$) and cross-attention with the output of the encoder ($\text{Attn}(X, Z')$). Each encoder and decoder layer also includes normalization, fully connected layers, and residual connections applied to each token [45] [53]. The output of the last decoder layer is passed through a linear transformation to generate a sequence of predictions.

Because Eq (2) involves matching each query to the keys of the entire sequence, its runtime and memory use grows quadratically with the length of its input. This becomes a significant obstacle in long-input domains where attention might otherwise be a promising solution, including document classification [34], summarization [30], and protein sequence analysis [23]. As a result, the research community has raced to develop and evaluate variants for longer sequences [43]. Many of these methods introduce heuristics to sparsify the attention matrix. For example, we can attend primarily to adjacent input tokens [24], select global tokens [17], increasingly distant tokens [56] [10] or a combination thereof [58] [60]. While these methods are effective, their inductive biases about the structure of the trained attention matrix are not always compatible with tasks outside of NLP.

Another approach looks to approximate attention in sub-quadratic time while retaining its flexibility [46] [62] [54] [64]. Particularly relevant to this work is the Performer [11]. Performer approximates attention in linear space and time with a kernel of random orthogonal features and enables the long-sequence approach that is central to our work.

2.2 Temporal Forecasting

Time Series Forecasting in practice is often focused on business/finance applications and utilizes relatively simple but reliable statistical methods [6] [44]. Analogous to the trend in areas like computer vision and NLP, hand-crafted approaches have begun to be replaced by deep learning techniques [42]. This is particularly true in multivariate forecasting, where the relationships between multiple time series challenge traditional models’ ability to scale [27] [4]. Deep learning approaches to TSF are generally based on a seq2seq framework in which a *context window* of the c most recent timesteps is mapped to a *target window* of predictions for a time horizon of h steps into the future. More formally: given a sequence of time inputs (x_{T-c}, \dots, x_T) and variable(s) (y_{T-c}, \dots, y_T) up to time T , we output another sequence of variable values $(\hat{y}_{T+1}, \dots, \hat{y}_{T+h})$ corresponding to our predictions at future timesteps $(x_{T+1}, \dots, x_{T+h})$.

The most common class of deep TSF models are based on a combination of Recurrent Neural Networks (RNNs) and one-dimensional convolutions (Conv1Ds) [5] [42] [39] [22]. More related to the proposed method is a recent line of work on attention mechanisms

that aim to overcome RNNs’ autoregressive training and difficulty in interpreting long-term patterns [19] [24] [59] [47] [33]. Notable among these is the Informer [62] - a general encoder-decoder Transformer architecture for TSF. Informer takes the sequence timestamps $(x_{T-c}, \dots, x_{T+h})$ and embeds them in a higher dimension d . The variable values - with zeros replacing the unknown target sequence $(y_{T-c}, \dots, y_T, \mathbf{0}_{T+1}, \dots, \mathbf{0}_{T+h})$ - are mapped to the same dimension d . The time (x) and variable (y) components of the sequence sum to create the input sequence $Z \in \mathbb{R}^{(c+h) \times d}$. The encoder processes the sub-sequence $Z[0 \dots c]$ while the decoder observes the target sequence along with a small overlap of the context sequence with length s called the *start token*: $Z[c - s \dots c + h]$.

Zhou et al. [62] observe that attention scores have a long tail distribution in which many tokens have a near-negligible impact on one-another. Their ProbSparse attention uses a sampling method to reduce memory by only computing the most important attention connections. ProbSparse attention is applied to self-attention while the decoder uses standard cross-attention with the encoder output sequence. It is important to note that each token in Z contains a projection of all N variables in the value sequence of (y_{T-c}, \dots, y_T) , in contrast with popular Transformer domains like NLP in which each token represents just one unified idea (e.g., a single word) [41]. The message-passing graph that results from attention over a multivariate sequence learns patterns across time while representing multiple variables per token (Figure 1b). This setup forces the variables within each token to receive the same amount of information from other timesteps, despite the fact that variables may have distinct patterns or relationships to each other. Ideally, we would have a way to model these kinds of variable relationships.

2.3 Spatial-Temporal Forecasting

Multivariate TSF involves two axes of complexity: the forecast sequence’s duration, L , and the number of variables N considered at each timestep. As N grows, it becomes increasingly important to model the relationships between each variable. The multivariate forecasting problem can be reformulated as the prediction of a sequence of graphs, where at each timestep t we have a graph $\mathcal{G}_t = (N, E)$ consisting of N variable nodes connected by E edges. Given a context sequence of graphs $(\mathcal{G}_{T-c}, \dots, \mathcal{G}_T)$, we must predict the node values of the target sequence $(\mathcal{G}_{T+1}, \dots, \mathcal{G}_{T+h})$.

Graph Neural Networks (GNNs) are a category of deep learning techniques that aim to explicitly model variable relationships as interactions along a network of nodes [48]. One approach to this is the spatial Graph Convolutional Network (GCN) [7] [32] which aggregates information over a variable’s neighborhood of connected nodes much like a standard Conv1D aggregates information over neighboring elements in a sequence. In recent years, GNN components have become a popular solution to the problem of modeling the impact of node edges within the graphs of multivariate sequences. Earlier work uses GCN layers to pass information amongst the variables of each timestep while using standard sequence learning architectures (e.g., RNNs [25], Conv1Ds [57], dilated Conv1Ds [50]) to adapt those representations across time. More recent work has extended this formula by replacing the spatial or temporal learning mechanisms (or both) with attention modules

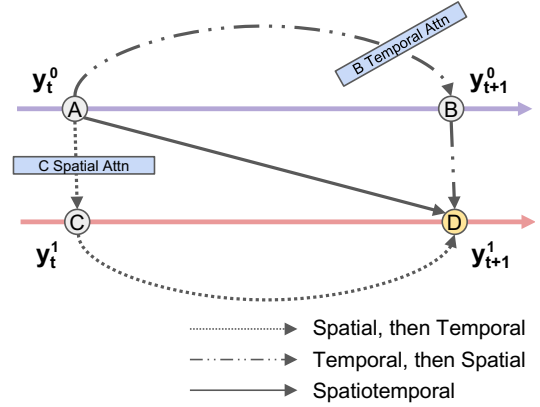


Figure 2: “Spatial and Temporal” vs. Spatiotemporal Attention. We depict a two variable time series with purple (top) and red (bottom) variable sequences. Tokens corresponding to timesteps t and $t + 1$ have been labeled A, B, C, D for notational simplicity. Imagine that the information at token A is important to the representation we want to produce at token D. Alternating temporal and spatial attention layers force this information to take an indirect route through the attention mechanism of an unrelated token (e.g., B or C). The A→D path is then dependent on the similarity of B and A or C and A, as well as any other tokens involved in B’s temporal attention or C’s spatial attention. In contrast, true spatiotemporal attention creates a direct path A→D with no risk of information loss.

much like Eq. (2) [8] [55] [37] [35] [52]. Temporal attention with intra-token spatial graph learning is depicted in Figure 1c. The exact implementation details vary and can be quite domain specific, but models in this category have at least one of three shortcomings:

- (1) They require a **predefined adjacency matrix** representing the ground-truth relationship between input variables. These hardcoded graphs are used to define the graph convolution or graph self-attention component. There are many GNN problems (e.g. traffic forecasting) where relationships are based on physical distance or connectivity and adjacency matrices are somewhat easy to define. However, this makes it difficult to solve more abstract multivariate prediction problems where relationships are unknown or time-dependent. Exceptions include MTGNN [49] and Graph WaveNet [50] - which can learn (or adapt) adjacency matrices from data. Also included are methods that use spatial RNN-based attention [3] without GNN components [41] [16].
- (2) They perform **separate temporal and spatial updates**. This can make it more difficult to share knowledge between different timesteps of different variables, because it forces information to take an indirect route through the attention mechanism of an unrelated token. This issue is explained in more detail in Figure 2. Exceptions include concurrent work: TraverseNet [52] and STJGCN [61].

- (3) Their decoder architectures are **not designed for long-term temporal forecasting**. For example, the model includes a large spatial-temporal attention encoder for the context sequence and a one-layer convolutional output. While this is sufficient for the short time horizons considered in common GNN problems, it can make it more difficult to apply these methods to highly temporal domains that require distant predictions. Exceptions include methods with both spatial and temporal attention mechanisms, which generally opt for a true encoder-decoder architecture [61] [8] [35].

We categorize the most important related work according to these three methodological differences in Table 1.

Method	Attention Type	Predefined Spatial Graph	Decoder Attention
Informer [62]	Temporal	✗	✓
TFT [26]	Temporal	✗	✓
Traffic Trans. [8]	Spatial + Temporal	✓	✓
STTN [55]	Spatial + Temporal	✓	✗
ST-GRAT [35]	Spatial + Temporal	✓	✓
TPA-LSTM [41]	Spatial + Temporal	✗	✓
STAM [16]	Spatial + Temporal	✗	✓
MTGNN [49]	Spatial + Temporal	✗	✗
TraverseNet [52]	Spatiotemporal*	✓	✗
STJGCN [61]	Spatiotemporal*	✗	✓
Spacetimeformer	Spatiotemporal	✗	✓

Table 1: Attention-based multivariate forecasting related work categorized by attention type (Fig 2) and the requirement of a hard-coded variable (spatial) graph. * [52] [61] are concurrent works motivated by full spatiotemporal attention. However, both approximate this effect by restricting it to a fixed time horizon [52] and mixing several of those horizon lengths together [61]. Long-Range Transformers are the primary engineering advancement that let us skip these approximations.

3 Method: Spatiotemporal Forecasting with Transformers

3.1 Spatiotemporal Sequences

Our goal is to create an attention-based spatial-temporal forecasting method that addresses the three issues listed in Sec 2.3. We begin by taking a step back from the GNN literature and build upon the Informer-style [62] encoder-decoder Transformer framework that has been shown to make accurate long-term temporal predictions. As discussed in Sec 2.2, Informer generates d -dimensional embeddings of the sequence $((x_{T-c}, y_{T-c}), \dots, (x_T, y_T), (x_{T+1}, 0_{T+1}), \dots, (x_{T+h}, 0_{T+h}))$, with the result expressed in matrix form as $Z \in \mathbb{R}^{(c+h) \times d}$. If we modify the token embedding input sequence by flattening each multivariate y_t vector into N scalars with a copy of its timestamp x_t , we get a new sequence: $((x_{T-c}, y_{T-c}^0), \dots, (x_{T-c}, y_{T-c}^N), \dots, (x_T, y_T^0), \dots, (x_T, y_T^N), (x_{T+1}, 0_{T+1}^0), \dots, (x_{T+1}, 0_{T+1}^N), \dots, (x_{T+h}, 0_{T+h}^0), \dots, (x_{T+h}, 0_{T+h}^N))$. Embedding this longer sequence results in a

$Z' \in \mathbb{R}^{N(c+h) \times d}$. When we pass Z' through a Transformer the attention matrix (Eq. (1)) $A(Z', Z') \in \mathbb{R}^{N(c+h) \times N(c+h)}$ then represents a spatiotemporal graph with a direct path between every variable at every timestep (Figure 2). Just as importantly, we are now learning spatial relationships that do not rely on predefined variable graphs and can change dynamically according to the time and variable values of the input data. This concept is depicted in Figure 1d. There are two important questions left to answer:

- (1) How do we embed this sequence so that the attention network parameters (W^Q, W^K, W^V) can accurately interpret the information in each token?
- (2) Can we multiply the sequence length by a factor of N and still find a way to scale to real-world problems?

3.2 Spatiotemporal Embeddings

After flattening the input variables, we need to allow Transformers to correctly interpret the variable each token originates from as well as its time and scalar value. Transformers are permutation invariant, meaning they cannot interpret the order of input tokens by default. This is fixed by adding a *position embedding* to the tokens - usually with a fixed sinusoidal pattern [45]. In our flattened representation, blocks of N consecutive tokens should have the same position. However, a fixed positional embedding only preserves relative order. Relative position is all that matters in the NLP context, but in a time series problem the input sequence takes place in a specific window of time that may be relevant to long-term seasonal patterns. For example, it is important to know the relative order of three weeks of temperature observations, but it is also important to know that those three weeks take place in late spring when temperatures are typically rising. We achieve this by learning positional embeddings based on a Time2Vec layer [21]. Time2Vec passes a representation of absolute time (e.g., the calendar datetime) through sinusoidal patterns of learned offsets and wavelengths. This helps represent periodic relationships that are relevant to accurate predictions. We also append a relative order index to the time input values³. The concatenated time series value and time embedding are then projected to the input dimension of the Transformer model with a feed-forward layer. We refer to the resulting output as the “value & time embedding.”

Now that the model can distinguish between timesteps, it also has to tell the difference between the variables at each of those timesteps. We add a “variable embedding” to each token. The variable embedding indicates the time series from which each token originates. This is implemented as a standard embedding layer that maps the integer index of each series to a higher-dimensional representation as is common for word embeddings in NLP. Note that this means the variable representations are randomly initialized and learned end-to-end during training.

A final consideration is the representation of missing time series values, including null values in the dataset as well as the target tokens that the model is tasked to predict. The encoder observes the context sequence in which the only missing entries are null

³This is helpful when the data are taken at close intervals such that most entries of the resulting Time2Vec output vectors are identical. In the extreme case, it serves as a fail-safe for datasets that do not provide distinct timestamps for each datapoint, e.g., when all observations occur on the same day but the hour and minute values are not recorded.

values in the dataset, which we fill with zeros. However, the decoder deals with unknown values that need prediction in addition to the start token of the last few timesteps of the context sequence (Sec 2.2). As mentioned in Sec 3.1, missing target sequence tokens are also filled in with zeros. Because variable values near zero can be common in normalized datasets, we resolve ambiguity between genuine context/start-token values of zero and missing tokens with a binary “given embedding”. The value & time, variable, and given embeddings sum to create the encoded sequence. A visual overview of the embedding process is shown in Appendix Figure 5.

3.3 Long-Range Transformers for Real-World Forecasting

After the spatiotemporal embedding, the sequence is ready to be fed through a standard Transformer encoder-decoder. Our method does not assume access to predefined variable relationships or even that those relationships remain fixed over time; the embedding represents the spatiotemporal forecasting problem in a raw format that puts full faith in the learning power of the Transformer architecture. The cost of this conceptual simplicity is the practical engineering challenge of training Transformers at extreme sequence lengths. The embedding in Sec 3.2 converts a sequence-to-sequence problem of length L into a new one of length LN . Standard Transformer layers have a maximum sequence length of less than 1,000 - and on hardware aside from the highest-end GPUs/TPUs we are limited to something closer to ~ 500 . The spatiotemporal sequence lengths considered in our experimental results exceed 26,000. Clearly, additional optimization is needed to make these problems feasible.

Scaling with Fast-Attention. However, we are fortunate that there is a rapidly-developing field of research in long-sequence Transformers (Sec 2.1). We modify open-source implementations of fast-attention mechanisms in an effort to scale our model. Informer [62] proposes ProbSparse attention. While this method is more scalable than vanilla attention, we are only able to reach sequences between 1k-2k tokens, mostly because it is only applicable to self-attention modules. Instead, we opt for the Performer FAVOR+ attention [11] - a linear approximation of softmax⁴ attention using a random kernel method. The experiments in this paper are performed on four 8 – 12GB GPUs - significantly less memory than the maximum currently available (e.g., larger clusters of V100s, TPUs). This constraint is likely to be less relevant as hardware development continues to shift towards intensive deep learning. However, we find that a multi-GPU Performer setup is enough to tackle all but the longest spatiotemporal sequences.

Scaling with Initial and Intermediate Conv1Ds. In the longest-sequence problems, we can also look for ways to learn shorter representations of the input to save memory. This is only applicable to the context window encoder, but in practice the context window is typically the bottleneck. We can learn to summarize the sequence with strided one-dimensional convolutions; we use the proper stride and padding to cut sequence length in half per Conv1D layer. “Initial” convolutions are applied to the value & time embedding while the variable embedding tokens are repeated for half their usual length. “Intermediate” convolutions occur between

attention layers and are a key component of the Informer [62] architecture. However, our embedding creates an undesirable emphasis on the order of the flattened variable tokens. We address this by passing each variable’s full sequence through the convolution independently (halving their length) and then recombining them into the longer spatiotemporal sequence. We prefer to learn the full-size spatiotemporal graph when it fits in memory. Therefore none of the reported results use intermediate convolutions; the longest dataset (AL Solar) model uses one initial convolution.

Local and Global Architecture. We empirically find that learning an attention layer over a longer global sequence eliminates a helpful local bias that hurts performance in problems with large N . We address this with additional “local” attention modules in each encoder and decoder layer. Each variable sequence attends to itself first before attending to global tokens. Note that this does not mean we are simplifying the global attention layer by separating temporal and spatial attention, as is common in the methods discussed above in addition to Video Transformers [2] and other long-range NLP architectures [64] [60]. Rather, every token attends to every token in its own sequence and then to every token in the entire spatiotemporal global sequence. We use a Pre-Norm architecture [53] and BatchNorm normalization rather than the standard LayerNorm and other alternatives in the literature. Additional implementation details are discussed in Appendix B. A one-layer encoder-decoder architecture is depicted in Figure 3.

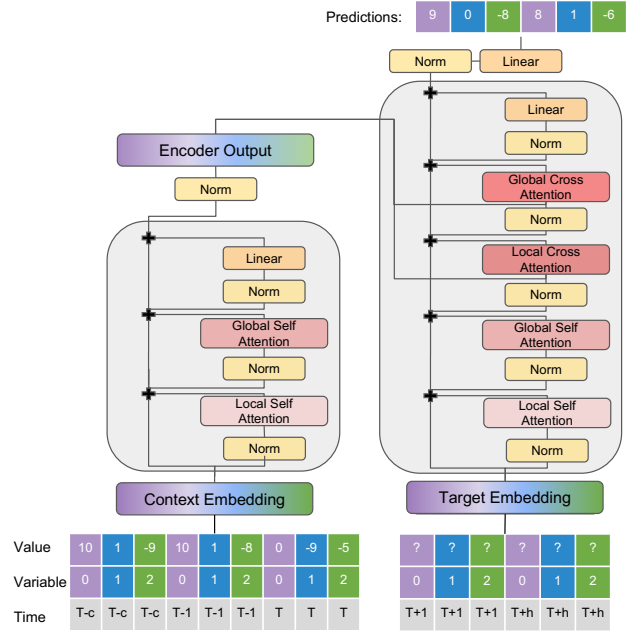


Figure 3: The Spacetimeformer architecture for joint spatiotemporal learning applied to the sequence shown in Figure 1a. This architecture creates a practical implementation of the spatiotemporal attention concept in Figure 1d. The start token and “given” embedding values have been left out for simplicity.

⁴In practice we find that the generalized ReLU kernel variant is slightly more memory efficient and use that as our default attention mechanism.

Output and Loss Functions. The final feed-forward layer of the decoder generates a two-dimensional vector that parameterizes the mean and standard deviation of a Normal distribution. This sequence of distributions is folded back into the original input format of (x_t, \hat{y}_t) , and the timesteps corresponding to predictions on the start tokens are discarded. We can then optimize a variety of forecasting loss functions, depending on the particular problem and the baselines we are comparing against. Explanations of common loss functions can also be found in Appendix B.

Because of all our additional engineering, we avoid direct comparisons to Informer although it is a key inspiration and closely related work. Instead, our most direct baseline is a version of our model that follows Informer’s temporal attention framework while reusing our implementation details. The sequence is never flattened into distinct spatial tokens; the value & time embedding input includes all the time series values concatenated together as normal, and there is no variable embedding. There is no concept of “local vs. global” in this setup, so we skip the local attention modules. We typically compensate for parameter count differences with additional encoder/decoder layers. We refer to this temporal-only ablation as the Temporal model.

We nickname our full model the Spacetimeformer for clarity in experimental results.

4 Experiments

Our experiments are designed to answer the following questions:

- (1) Is our model competitive with sequence-to-sequence methods on highly temporal tasks that require long-term forecasting?
- (2) Is our model competitive with graph-based methods on highly spatial tasks, despite not having access to a predefined variable graph?
- (3) Does the spatiotemporal sequence embedding let Transformers learn meaningful spatial relationships?

We evaluate our method on five datasets, including traffic forecasting, weather prediction, solar energy production, and a simple toy example. Descriptions of each task can be found in Appendix A, and a summary of key information is provided in Table 2. We compare against representative methods from the TSF and GNN literature in addition to ablations of our model. These include the previously mentioned MTGNN model and Temporal ablation, as well as a basic autoregressive linear model (Linear AR), a standard encoder-decoder LSTM without attention, and a RNN/Conv1D-based LSTNet [22]. See Appendix B.2 for details. Hyperparameter information for Spacetimeformer models can be found in Appendix B.1. All results average at least three random seeds, with additional runs collected when variance is non-negligible. All models use the same training loop, dataset code, and evaluation process to ensure a fair comparison. We also add Time2Vec information to baseline inputs when applicable because it has been shown to improve the performance of a variety of sequence models [21].

4.1 Toy Example

We begin with a toy dataset inspired by [41] consisting of a multivariate sine-wave sequence with strong inter-variable dependence. More details can be found in Appendix A. Several ablations of our

	Variables (N)	Length (L)	Size (Timesteps)
NY-TX Weather	6	800	569,443
Toy Dataset	20	160	2,000
AL Solar	137	192	52,560
Metr-LA Traffic	207	24	34,272
Pems-Bay Traffic	325	24	52,116

Table 2: Dataset Summary.

	Temporal	Temporal (Deeper)	Temporal (Deeper & Full Attn)	ST Local	ST Local (Deeper)	Spacetimeformer
<i>MSE</i>	0.006	0.010	0.005	0.021	0.014	0.003
MAE	0.056	0.070	0.056	0.104	0.090	0.042
RRSE	0.093	0.129	0.094	0.180	0.153	0.070

Table 3: Toy Dataset Results. We indicate the loss function metric (here, MSE) with italics in subsequent tables.

method are considered. Temporal modifies the spatiotemporal embedding as discussed at the end of Sec 3.3. ST Local skips the global attention layers but includes spatial information in the embedding. The “Deeper” variants attempt to compensate for the additional parameters of the local+global attention architecture of our full method. All models use a small Transformer model and optimize for MSE. The final quarter of the time series is held out as a test set; a similar test split is used in all experiments [9]. The results are shown in Table 3. The Temporal embedding is forced to compromise its attention over timesteps in a way that reduces predictive power over variables with such distinct frequencies. Standard (“Full”) attention fits in memory with the Temporal embedding but is well approximated by Performer. Our method learns an uncompromising spatiotemporal relationship among all tokens to generate the most accurate predictions by all three metrics.

4.2 Long (L) Temporal Study: Temperature Prediction

We evaluate each model on a custom dataset of temperature values collected from the ASOS Weather Network. We use three stations in central Texas and three in eastern New York to create two almost unrelated spatial sub-graphs. Temperature values are taken at one-hour intervals, and we investigate the impact of sequence length by predicting 40, 80, and 160 hour target sequences. The results are shown in Table 4. Our spatiotemporal embedding scheme provides the most accurate forecasts, and its improvement over the Temporal method appears to increase over longer sequences where the temporal attention compromise may be more relevant. LSTM is the most competitive baseline. This is not surprising given that this problem lies towards the TSF end of the spatiotemporal spectrum, where LSTMs are a common default. The success of MTGNN is more surprising because we are using it at sequence lengths far longer than considered in the original paper [49]. However, its convolution-only output mechanism begins to struggle at the 80 and 160 hour lengths.

	Linear AR	LSTM	MTGNN	Temporal	Spacetimeformer
40 hours					
MSE	18.84	14.29	13.32	13.29	<u>12.49</u>
MAE	3.24	2.84	2.67	2.67	<u>2.57</u>
RRSE	0.40	0.355	0.34	0.34	<u>0.33</u>
80 hours					
MSE	23.99	18.75	19.27	19.99	<u>17.9</u>
MAE	3.72	3.29	3.31	3.37	<u>3.19</u>
RRSE	0.45	<u>0.40</u>	0.41	0.41	<u>0.40</u>
160 hours					
MSE	28.84	22.11	24.28	24.16	<u>21.35</u>
MAE	4.13	3.63	3.78	3.77	<u>3.51</u>
RRSE	0.50	<u>0.44</u>	0.46	0.46	<u>0.44</u>

Table 4: NY-TX Weather Results.

4.3 Spatial-Temporal Study: Solar Power and Traffic Prediction

4.3.1 AL Solar. We turn our focus to problems on the GNN end of the spatiotemporal spectrum where N approaches or exceeds L . The AL Solar dataset consists of solar power production measurements taken at 10-minute intervals from 137 locations. We predict 4-hour sequences and the results are shown in Table 5. Having both large N and L leads to an extremely long spatiotemporal sequence length of 26,304, but a strided convolution over each variable sub-sequence in the context window (Sec 3.3) yields a more manageable final length of 14,796. Spacetimeformer is significantly more accurate than the TSF baselines. We speculate that this is due to an increased ability to forecast unusual changes in power production due to weather or other localized effects. MTGNN learns similar spatial relationships, but its temporal predictions are not as accurate.

	Linear AR	LSTNet	LSTM	MTGNN	Temporal	Spacetimeformer
MSE	14.3	15.09	14.35	11.40	14.3	<u>8.96</u>
MAE	2.29	2.08	2.02	1.76	1.85	<u>1.43</u>

Table 5: AL Solar Results.

4.3.2 Traffic Prediction. Next, we experiment on two datasets common in GNN research. The Metr-LA and Pems-Bay datasets consist of traffic speed readings at 5 minute intervals, and we forecast the conditions for the next hour. For these experiments we include results directly from the literature to get a better comparison with GNN-based spatial models that use predefined road graphs. The results are listed in Table 6. Our method clearly separates itself from TSF models and enters the performance band of dedicated GNN methods on both datasets. However, Spacetimeformer is more competitive on Metr-LA than Pems-Bay. Metr-LA has 118 fewer variable nodes, resulting in a spatiotemporal sequence length of 4,968 compared to Pems-Bay’s 7,800. However, GPU memory constraints prevent us from continuing to scale our Pems-Bay model and reduce validation scores. Advanced hardware and/or a more efficient attention implementation would allow for more model parameters. We consider it an interesting direction for future work to see how the performance of spatiotemporal attention scales with model size, and whether this would lead to performance gains on datasets like Pems-Bay.

4.3.3 Spatiotemporal Attention Patterns. Our method relies on approximate attention mechanisms that never explicitly compute the attention matrix (Eq. (1)). However, we use a trick described in the Performer appendix [11] to visualize attention patterns. Standard Temporal Transformers learn sliding patterns that resemble convolutions. These wave-like patterns are the result of Time2Vec information in the input. Our method learns distinct connections between variables - this leads to attention diagrams that tend to be structured in “variable blocks” due to the way we flatten our input sequence. Figure 4 provides an annotated example for the NY-TX weather dataset. If viewing a digital copy of this paper, the reader may be able to zoom in and see the wave-like temporal patterns that exist within each “block” of variable attention. Our method learns spatial attention over blocks of variable inputs and convolution-style temporal attention over the timesteps within each variable. Some attention heads convincingly recover the ground-truth relationship between input variables in the NY-TX weather dataset. GNN-level performance in highly spatial tasks like traffic forecasting supports a similar conclusion for more complex graphs.

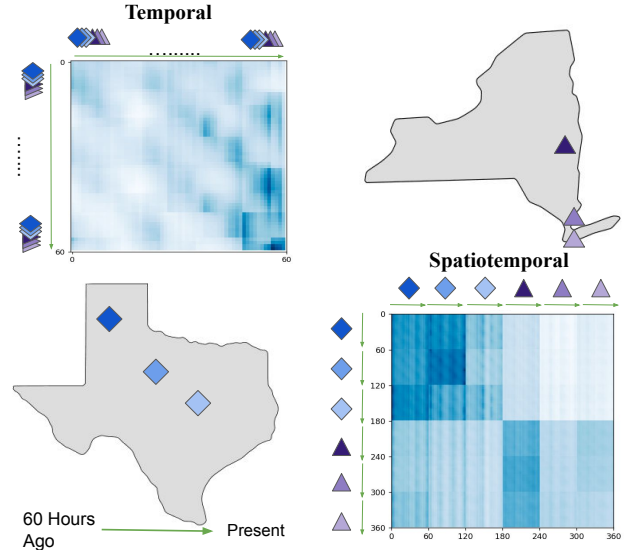


Figure 4: Discovering Spatial Relationships from Data: We forecast the temperature at three weather stations in Texas (lower left, blue) and three stations in New York (upper right, purple). Temporal attention models stack all six time series into one input variable and attend across time alone (upper left). By flattening the input and adding a spatial component to the existing time embedding, our method recovers the correct spatial relationship between the variables along with the strided temporal relation (lower right); nearby stations look to each other for relevant information about future weather patterns and ignore the stations hundreds of miles away. (Darker blue \rightarrow more attention).

4.3.4 Node Classification. During our development process, we became interested in the amount of spatial information that makes it through the encoder and decoder layers. We created a way to

	Time Series Models				GNN Models					
	Linear AR	LSTM	LSTNet	Temporal	DCRNN	MTGNN	<i>ST-GRAT*</i> [35]	<i>Graph WaveNet*</i> [50]	<i>Traffic Trans.*</i> [8]	Spacetimeformer
Metr-LA										
MAE	4.28	3.34	3.36	3.14	3.03	2.92	3.03	3.09	2.83	2.73
MSE	91.57	48.19	44.23	41.88	37.88	39.07	39.77	39.84	<u>32.82</u>	35.14
MAPE	11.11	9.89	9.53	8.9	8.27	8.46	8.25	8.42	7.70	<u>7.51</u>
Pems-Bay										
MAE	2.24	2.41	2.03	2.37	1.59	1.75	1.62	1.63	<u>1.53</u>	1.73
MSE	27.62	25.49	19.17	23.52	13.69	15.23	13.85	13.87	<u>13.25</u>	14.37
MAPE	4.98	5.81	4.68	5.52	3.61	4.08	3.65	3.67	<u>3.49</u>	3.85

Table 6: Traffic Forecasting Results. *Models listed in italics use results directly from the cited literature and are averages of the 3, 6, and 12 step statistics that are commonly reported on these datasets.

measure this by adding a softmax classification layer to the encoder and decoder output sequences. This layer interprets each token and outputs a prediction for the time series variable it originated from. Importantly, we detach this classification loss from the forecasting loss our model is optimizing; only the classification layer is trained on this objective. Classification accuracy begins near zero but spikes upwards of 99% in all problems within a few hundred gradient updates due to distinct (randomly initialized) variable embeddings that are well-preserved by the residual Pre-Norm Transformer architecture. In TSF tasks like NY-TX weather, this accuracy is maintained throughout training. However, in more spatial tasks like Metr-LA, accuracy begins to decline over time. Because this decline corresponds to increased forecasting accuracy, we interpret this as a positive indication that related nodes are grouped with similar variable embeddings that become difficult for a single-layer classification model to distinguish.

4.3.5 Ablations. Finally, we perform an ablation experiment on the NY-TX Weather and Metr-LA Traffic datasets. The results are shown in Table 7. The relative importance of space and time embedding information changes based on whether the problem is more temporal like NY-TX Weather or spatial like traffic forecasting. While global spatiotemporal attention is key to state-of-the-art performance on Metr-LA, it is interesting to see that local attention on its own is a competitive baseline. We investigate this further by removing both global attention and the variable embeddings that differentiate traffic nodes, which leads to the most inaccurate predictions. This implies that the local attention graph is adapting to each node based on the static spatial information learned by variable embeddings.

5 Conclusion and Future Directions

This paper has presented a unified method for multivariate forecasting based on the application of Long-Range Transformer architectures to elongated spatiotemporal input sequences. Our approach jointly learns temporal and spatial relationships to achieve competitive results on long-sequence time series forecasting, and can scale to high dimensional spatial problems without relying on a predefined graph. We see at least two promising directions for future development. First, Long-Range Transformers are an active area of research; smaller memory footprints and increased computational resources are likely to improve performance and open up applications of this method to larger problems. Second, it is not clear that

	MAE	Classification Acc. (%)
NY-TX Weather ($L = 200, N = 6$)		
Full Spatiotemporal	<u>2.57</u>	<u>99</u>
No Local Attention	2.62	<u>99</u>
No Variable Embedding	2.66	45
Temporal Embedding/Attention	2.67	-
No Value Embedding	3.83	<u>99</u>
No Time Embedding	4.01	<u>99</u>
Metr-LA Traffic ($L = 24, N = 207$)		
Full Spatiotemporal	<u>2.73</u>	<u>58</u>
No Global Attention	3.00	46
No Time Embedding	3.11	50
Temporal Embedding/Attention	3.14	-
No Local Attention	3.27	54
No Variable Embedding	3.29	2
No Global Attention, No Variable Emb.	3.48	1

Table 7: Ablation Results.

we are fully utilizing the dynamic graph capabilities of our method. Our model learns variable graphs across time and space jointly without assuming a fixed spatial relationship. However, GNN methods that do make this assumption have similar forecasting accuracy. It may be that adapting the relationship between variables over time has limited benefit in the benchmark datasets discussed. For example, traffic datasets contain some situations where adaptive graphs are beneficial (e.g. when intersection relationships change due to accidents or other road closures), however there are probably applications where this kind of temporal graph adjustment is more essential. The proposed method and future improvements may be more beneficial in these domains.

References

- [1] ARAABI, A., AND MONZ, C. Optimizing transformer for low-resource neural machine translation, 2020.
- [2] ARNAB, A., DEGHANI, M., HEIGOLD, G., SUN, C., LUČIĆ, M., AND SCHMID, C. Vivit: A video vision transformer, 2021.
- [3] BAHDANAU, D., CHO, K., AND BENGIO, Y. Neural machine translation by jointly learning to align and translate, 2016.
- [4] BENIDIS, K., RANGAPURAM, S. S., FLUNKERT, V., WANG, B., MADDIX, D., TURKMEN, C., GASTHAUS, J., BOHLKE-SCHNEIDER, M., SALINAS, D., STELLA, L., ET AL. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240* (2020).
- [5] BOROVYKH, A., BOHTE, S., AND OOSTERLEE, C. W. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691* (2017).

- [6] BOX, G. E., JENKINS, G. M., REINSEL, G. C., AND LJUNG, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [7] BRUNA, J., ZAREMBA, W., SZLAM, A., AND LECUN, Y. Spectral networks and locally connected networks on graphs, 2014.
- [8] CAI, L., JANOWICZ, K., MAI, G., YAN, B., AND ZHU, R. Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS* 24, 3 (2020), 736–755.
- [9] CERQUEIRA, V., TORGO, L., AND MOZETIČ, I. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning* 109, 11 (2020), 1997–2028.
- [10] CHILD, R., GRAY, S., RADFORD, A., AND SUTSKEVER, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509* (2019).
- [11] CHOROMANSKI, K., LIKHOSHERSTOV, V., DOHAN, D., SONG, X., GANE, A., SARLOS, T., HAWKINS, P., DAVIS, J., MOHIUDDIN, A., KAISER, L., BELANGER, D., COLWELL, L., AND WELLER, A. Rethinking attention with performers, 2021.
- [12] DETTMERS, T. Data dropout for wikitext-2 (tweet #1247998807494684672), 2020.
- [13] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [14] FALCON, WA, E. A. Pytorch lightning. *GitHub. Note: <https://github.com/PyTorchLightning/pytorch-lightning>* 3 (2019).
- [15] FAN, A., GRAVE, E., AND JOULIN, A. Reducing transformer depth on demand with structured dropout, 2019.
- [16] GANGOPADHYAY, T., TAN, S. Y., JIANG, Z., MENG, R., AND SARKAR, S. Spatiotemporal attention for multivariate time series prediction and interpretation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2021). IEEE, pp. 3560–3564.
- [17] GUO, Q., QIU, X., LIU, P., SHAO, Y., XUE, X., AND ZHANG, Z. Star-transformer. *arXiv preprint arXiv:1902.09113* (2019).
- [18] HUANG, X. S., PEREZ, F., BA, J., AND VOLKOV, M. Improving transformer optimization through better initialization. In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), H. D. III and A. Singh, Eds., vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 4475–4483.
- [19] IWATA, T., AND KUMAGAI, A. Few-shot learning for time-series forecasting. *arXiv preprint arXiv:2009.14379* (2020).
- [20] JOSHI, C. Transformers are graph neural networks. *The Gradient* (2020).
- [21] KAZEMI, S. M., GOEL, R., EGHBAI, S., RAMANAN, J., SAHOTA, J., THAKUR, S., WU, S., SMYTH, C., POUPART, P., AND BRUBAKER, M. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321* (2019).
- [22] LAI, G., CHANG, W.-C., YANG, Y., AND LIU, H. Modeling long- and short-term temporal patterns with deep neural networks, 2018.
- [23] LANCHANTIN, J., WEINGARTEN, T., SEKHON, A., MILLER, C., AND QI, Y. Transfer learning for predicting virus-host protein interactions for novel virus sequences. *bioRxiv* (2021).
- [24] LI, S., JIN, X., XUAN, Y., ZHOU, X., CHEN, W., WANG, Y.-X., AND YAN, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems* 32 (2019), 5243–5253.
- [25] LI, Y., YU, R., SHAHABI, C., AND LIU, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2018.
- [26] LIM, B., ARIK, S. O., LOEFF, N., AND PFISTER, T. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2020.
- [27] LIM, B., AND ZOHREN, S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A* 379, 2194 (2021), 20200209.
- [28] LIU, L., LIU, X., GAO, J., CHEN, W., AND HAN, J. Understanding the difficulty of training transformers, 2020.
- [29] LIU, P., CHANG, S., HUANG, X., TANG, J., AND CHEUNG, J. C. K. Contextualized non-local neural networks for sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 6762–6769.
- [30] MANAKUL, P., AND GALES, M. J. F. Long-span summarization via local attention and content selection, 2021.
- [31] NGUYEN, T. Q., AND SALAZAR, J. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895* (2019).
- [32] NIEPERT, M., AHMED, M., AND KUTZKOV, K. Learning convolutional neural networks for graphs, 2016.
- [33] ORESHKIN, B. N., CARPOV, D., CHAPADOS, N., AND BENGIO, Y. Meta-learning framework with applications to zero-shot time-series forecasting, 2020.
- [34] PAPPAGARI, R., ŻELASKO, P., VILLALBA, J., CARMIEL, Y., AND DEHAK, N. Hierarchical transformers for long document classification, 2019.
- [35] PARK, C., LEE, C., BAHNG, H., TAE, Y., JIN, S., KIM, K., KO, S., AND CHOO, J. St-grat: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Oct 2020).
- [36] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHAIN, N., ANTIGA, L., DESMAISON, A., KOPF, A., YANG, E., DEVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., BAI, J., AND CHINTALA, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.
- [37] PLIZZARI, C., CANNICI, M., AND MATTEUCCI, M. Skeleton-based action recognition via spatial and temporal transformer networks. *Computer Vision and Image Understanding* 208–209 (Jul 2021), 103219.
- [38] ROZEMBERCZKI, B., SCHERER, P., HE, Y., PANAGOPOULOS, G., RIEDEL, A., ASTEFANOAEI, M., KISS, O., BERES, F., LÓPEZ, G., COLLIGNON, N., AND SARKAR, R. Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models, 2021.
- [39] SALINAS, D., FLUNKERT, V., AND GASTHAUS, J. Deepar: Probabilistic forecasting with autoregressive recurrent networks, 2019.
- [40] SHEN, S., YAO, Z., GHOLAMI, A., MAHONEY, M. W., AND KEUTZER, K. Powernorm: Rethinking batch normalization in transformers, 2020.
- [41] SHIH, S.-Y., SUN, F.-K., AND LEE, H.-Y. Temporal pattern attention for multivariate time series forecasting. *Machine Learning* 108, 8 (2019), 1421–1441.
- [42] SMYL, S. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting* 36, 1 (2020), 75–85.
- [43] TAY, Y., DEHGHANI, M., ABNAR, S., SHEN, Y., BAHRI, D., PHAM, P., RAO, J., YANG, L., RUDER, S., AND METZLER, D. Long range arena: A benchmark for efficient transformers, 2020.
- [44] TAYLOR, S. J., AND LETHAM, B. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45.
- [45] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.
- [46] WANG, S., LI, B. Z., KHABSA, M., FANG, H., AND MA, H. Linformer: Self-attention with linear complexity, 2020.
- [47] WU, S., XIAO, X., DING, Q., ZHAO, P., YING, W., AND HUANG, J. Adversarial sparse transformer for time series forecasting.
- [48] WU, Z., PAN, S., CHEN, F., LONG, G., ZHANG, C., AND YU, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (Jan 2021), 4–24.
- [49] WU, Z., PAN, S., LONG, G., JIANG, J., CHANG, X., AND ZHANG, C. Connecting the dots: Multivariate time series forecasting with graph neural networks, 2020.
- [50] WU, Z., PAN, S., LONG, G., JIANG, J., AND ZHANG, C. Graph wavenet for deep spatial-temporal graph modeling, 2019.
- [51] WU, Z., WU, L., MENG, Q., XIA, Y., XIE, S., QIN, T., DAI, X., AND LIU, T.-Y. Unidrop: A simple yet effective technique to improve transformer without extra cost, 2021.
- [52] WU, Z., ZHENG, D., PAN, S., GAN, Q., LONG, G., AND KARYPIS, G. Traversenet: Unifying space and time in message passing, 2021.
- [53] XIONG, R., YANG, Y., HE, D., ZHENG, K., ZHENG, S., XING, C., ZHANG, H., LAN, Y., WANG, L., AND LIU, T.-Y. On layer normalization in the transformer architecture, 2020.
- [54] XIONG, Y., ZENG, Z., CHAKRABORTY, R., TAN, M., FUNG, G., LI, Y., AND SINGH, V. Nyströmformer: A nyström-based algorithm for approximating self-attention, 2021.
- [55] XU, M., DAI, W., LIU, C., GAO, X., LIN, W., QI, G.-J., AND XIONG, H. Spatial-temporal transformer networks for traffic flow forecasting, 2021.
- [56] YE, Z., GUO, Q., GAN, Q., QIU, X., AND ZHANG, Z. Bp-transformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070* (2019).
- [57] YU, B., YIN, H., AND ZHU, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (Jul 2018).
- [58] ZAHEER, M., GURUGANESH, G., DUBEY, K. A., AINSLIE, J., ALBERTI, C., ONTANON, S., PHAM, P., RAVULA, A., WANG, Q., YANG, L., ET AL. Big bird: Transformers for longer sequences. In *NeurIPS* (2020).
- [59] ZERVEAS, G., JAYARAMAN, S., PATEL, D., BHAMIDIPATY, A., AND EICKHOFF, C. A transformer-based framework for multivariate time series representation learning. *arXiv preprint arXiv:2010.02803* (2020).
- [60] ZHANG, H., GONG, Y., SHEN, Y., LI, W., LV, J., DUAN, N., AND CHEN, W. Pooling-former: Long document modeling with pooling attention, 2021.
- [61] ZHENG, C., FAN, X., PAN, S., WU, Z., WANG, C., AND YU, P. S. Spatio-temporal joint graph convolutional networks for traffic forecasting, 2021.
- [62] ZHOU, H., ZHANG, S., PENG, J., ZHANG, S., LI, J., XIONG, H., AND ZHANG, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI* (2021).
- [63] ZHOU, W., GE, T., XU, K., WEI, F., AND ZHOU, M. Scheduled dropout: A regularization method for transformer models, 2020.
- [64] ZHU, C., PING, W., XIAO, C., SHOEYBI, M., GOLDSTEIN, T., ANANDKUMAR, A., AND CATANZARO, B. Long-short transformer: Efficient transformers for language and vision, 2021.

A Dataset Details

- **Toy Dataset (based on [41]).** We generate D sequences where sequence i at timestep t is defined by:

$$Y_t^i = \sin\left(\frac{2\pi it}{64}\right) + \frac{1}{D+1} \sum_{j=1, j \neq i}^D \sin\left(\frac{2\pi jt}{64}\right) \quad (3)$$

We map 2,000 timesteps to a sequence of daily calendar dates beginning on Jan 1, 2000. We set $D = 20$ and use a context length of 128 and a target length of 32.

- **NY-TX Weather (new).** We obtain hourly temperature readings from the ASOS weather network. We use three stations located in central Texas and three more located hundreds of miles away in eastern New York. The data covers the years 1949–2021, making this a very large dataset by TSF standards. Many of the longest active stations are airport weather stations. We use the airport codes ACT (Waco, TX), ABI (Abilene, TX), AMA (Amarillo, TX), ALB (Albany, New York) as well as the two largest airports in the New York City metro, LGA and JFK.
- **Metr-LA Traffic [25].** A popular benchmark dataset in the GNN literature consisting of traffic measurements from Los Angeles highway sensors at 5-minute intervals over 4 months in 2012. Both the context and target sequence lengths are set to 12. We use the same train/test splits as [25].
- **AL Solar [22].** A popular benchmark dataset in the time series literature consisting of solar power production measurements across the state of Alabama in 2006. We use a context length of 168 and a target length of 24.
- **Pems-Bay Traffic [25].** Similar to Metr-LA but covering the Bay Area over 6 months in 2017. The context and target lengths are 12 and we use the standard train/test split.

Normalization and Time Representation. We normalize the values of each variable sequence independently based on training set statistics. We represent calendar dates by splitting the information into separate year, month, day, hour, minute, and second values and then re-scaling each to be $\in [0, 1]$. This works out so that only the year value is unbounded; we divide by the latest year present in the training set. The reformatted calendar values form the input to the Time2Vec embedding layer. As a brief aside, the time embedding mechanism of our model could be used in situations where the context and target sequence timestamps are different for each variable (e.g., they are sampled at different intervals). We do not take advantage of this in our experimental results because it is not relevant to common benchmark datasets and is not applicable to all the baselines we consider.

B Model Details

B.1 Spacetimeformer Details

There is significant empirical work investigating technical improvements to the Transformer architecture and training routine [18] [28] [53] [31] [51] [63] [1] [15] [40]. We incorporate some of these techniques to increase performance and hyperparameter robustness while retaining simplicity. A Pre-Norm architecture [53] is used to forego the standard learning rate warmup phase. We also find that replacing LayerNorm with BatchNorm is advantageous in the time series domain. [40] argue that BatchNorm is more popular

in Computer Vision applications because reduced training variance improves performance over the LayerNorms that are the default in NLP. Our experiments add empirical evidence that this may also be the case in time series problems. We also experiment with PowerNorm [40] and ScaleNorm [31] layers with mixed results. All four variants are included as a configurable hyperparameter in our open-source release.

As mentioned in Sec 3.3, the output of our model is a sequence of parameterized Normal distributions that folds the flattened spatiotemporal sequence back into its original concatenated format and discards predictions on start tokens. This lets us choose from several different loss functions. Most datasets use either Mean Squared Error (MSE) or Mean Absolute Error (MAE) based solely on the mean of the output distribution; in these cases the standard deviation outputs are untrained and should be ignored. We occasionally compare Root Relative Squared Error (RRSE) and Mean Absolute Percentage Error (MAPE) - though they are never used as an objective function. We also consider a Negative Log Likelihood (NLL) loss that maximizes the probability of the target sequence according to the output distribution and makes use of the model’s uncertainty estimates.

TSF datasets are generally much smaller than their NLP counterparts. As a result, we employ a number of regularization techniques and tend to prefer narrower architectures than are common in NLP. We add dropout to the embedding layer, query/key/value layers, feed-forward layers and attention outputs. Inspired by discussion on Transformer training on small NLP datasets [12], we also experiment with full token dropout in which entire input tokens are zeroed after the embedding phase - with little success.

Our architecture consists of four types of attention (Fig 3). “Global Self Attention” and “Global Cross Attention” attend to the entire spatiotemporal sequence, while “Local Self Attention” and “Local Cross Attention” attend to the timesteps of a single variable in isolation. Many Long-Range Transformer architectures include a concept of “local” vs “global” attention. For example, BigBird [58], Star-Transformer [17] and BP-Transformer [56] connect each token to local tokens in addition to global tokens. In these methods, “local” tokens are *adjacent* in terms of sequence order. This aligns well with the assumption that NLP sequences have meaningful order. Our flattened spatiotemporal sequence complicates that assumption because tokens can be adjacent while originating from different variables at vastly different timesteps. Regardless, we use the term “local” in a spatial sense where it refers to all timesteps within a single variable. Our local attention is straightforwardly implemented on top of any open-source attention variant by splitting the spatiotemporal sequence into its separate variables, folding this extra dimension into the batch dimension, performing attention as normal and then reversing the transformation. The attention mechanism used in all four types is configurable, and our open-source release includes ProbSparse Self Attention [62], NystromFormer Self Attention [54], Performer Self/Cross Attention in addition to the standard Self/Cross Full Attention. In practice, the sequence lengths considered in this paper require Performer Global Attention. The Local Attention modules are much more flexible; we default to Performer for simplicity.

Table 8 provides an overview of the hyperparameters used in our experiments. The values were determined by a brief sweep based

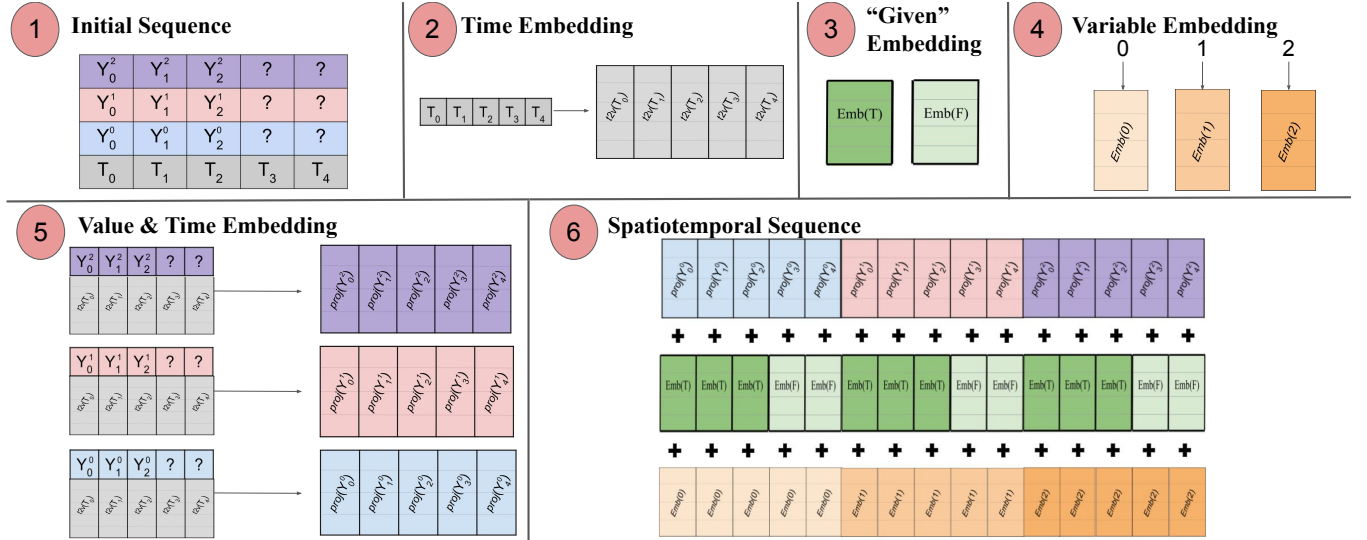


Figure 5: Input Encoding Pipeline: (1) The standard multivariate input format with time information included. We depict a three variable timeseries input with variables shown in purple, pink and blue. The input consists of three context steps and two target steps with missing values that need to be predicted (represented by “?”). (2) The time sequence is passed through a Time2Vec layer to generate a frequency embedding that represents periodic input patterns. (3) A binary embedding indicates whether this value is given as context or needs to be predicted. (4) The integer index of each time series is mapped to a “spatial” representation with a lookup-table embedding. (5) The Time2Vec embedding and variable values of each time series are projected with a feed-forward layer. (6) Value&Time, Variable, and Given embeddings are summed and laid out so that attention learns relationships across both time and variable space at the cost of a longer input sequence.

on validation loss. It is worth reiterating that these models are quite small relative to Informer and tiny in comparison to standard NLP architectures [45]. The architectures listed in Table 8 consist of 1 – 3M parameters. The memory requirements of our model are a central engineering challenge in this work, but Transformers do scale well with sequence length in terms of total parameter count. In contrast, methods like MTGNN grow larger with L and can become significantly larger than our model in datasets like NY-TX Weather or AL Solar. Temporal-only ablations add an extra encoder and decoder layer to help even out the difference in parameter count that comes with skipping local attention modules.

B.2 Baseline Details

In addition to variants/ablations of our Transformer framework, we consider several other Deep Learning architectures for TSF:

- **Linear AR:** A simple autoregressive linear model that acts independently on each input variable.
- **LSTNet [22]:** A mix of RNN and Conv1D layers with skip connections that predicts the difference between the final output and the above linear model. The fully autoregressive approach does not scale well with some of the sequence lengths considered in this paper. For this reason, we do not use LSTNet in all experiments.
- **LSTM:** A standard encoder-decoder RNN without attention. We use scheduled sampling to anneal the teacher forcing training mechanism.
- **MTGNN [49]:** A GNN model that learns the adjacency matrix between its input variables. MTGNN one-shots the entire

	NY-TX Weather	AL Solar	Metr-LA	Pems-Bay	Toy
<i>Model Dim</i>	200	128	160	128	100
<i>FF Dim</i>	800	512	600	512	400
<i>Attn Heads</i>	8	8	4	8	4
<i>Enc Layers</i>	2	3	4	4	4
<i>Dec Layers</i>	2	3	4	4	4
<i>Initial Convs</i>	0	1	0	0	0
<i>Inter. Convs</i>	0	0	0	0	0
<i>Attn Type</i>	performer	performer	performer	performer	performer
<i>Normalization</i>	batch	batch	batch	batch	batch
<i>Start Token Len</i>	(8, 16, 32)	8	3	3	4
<i>Time Emb Dim</i>	12	12	12	12	12
<i>Dropout FF</i>	0.2	0.3	0.3	0.4	.3
<i>Dropout Q/K/V</i>	0	0	0	0	0
<i>Dropout Token</i>	0	0	0	0	0
<i>Dropout Emb</i>	0.1	0.2	0.2	0.3	0
<i>L2 Weight</i>	0.01	0.01	0.01	0.01	0

Table 8: Hyperparameters for Spacetimeformer main experiments.

prediction sequence like our Transformer models. We use the annealed time mask that acts as a curriculum to emphasize shorter predictions early in training.

We use published hyperparameters on datasets that appear in the original work, and perform a sweep when they do not. Tricks like the linear delta in LSTNet and time mask curriculum in MTGNN are implemented for all models in our public code release; for simplicity’s sake we do not use them in the Transformer results. The evaluation statistics listed in all main tables are the average of three random seeds. We do not include variance measurements

for readability. However, all the methods are quite consistent and the variance of the main Transformer models studied is essentially negligible.

Time2Vec is a key component of our embedding pipeline; the Time2Vec paper [21] finds that learned time representations improve many types of sequence models. We account for this potential advantage by applying Time2Vec to LSTM and MTGNN models with the same embedding dimension used in the Spacetimeformer embedding scheme. Linear AR and LSTNet do not have a clear way to include them but are less competitive in general.

B.3 Training and Evaluation Process

All models use the same basic training loop for a fair comparison. We use Early Stopping with a patience of 5 epochs and reduce learning rates on validation loss plateaus. Most datasets are evaluated at the end of each full epoch; NY-TX Weather is so large that we adjust this to every quarter epoch. As a final training detail, some datasets contain missing values that should be ignored

during training. Of the datasets officially listed in this paper this only applies to Metr-LA and Pems-Bay. We adopt the “masked” loss functions used in DCRNN [25]: targets and predictions corresponding to missing timesteps are zeroed so that they do not impact gradient computation.

B.4 Code Implementation and Acknowledgements

The code for our method as well as the baselines and datasets can be found on GitHub at [QData/spacetimeformer](https://github.com/QData/spacetimeformer). All models are implemented in PyTorch [36] and the training process is conducted with PyTorch Lightning [14]. Our LSTNet and MTGNN implementations are based on public code [38] and verified by replicating experiments from the original papers. Generic models like LSTM and Linear AR are implemented from scratch and we made an effort to ensure the results are competitive. The code for the Spacetimeformer model was originally based on the Informer open-source release.