Multi-Scale Temporal Graph Attention Network for Spatiotemporal Forecasting

Your Name / Affiliation

April 20, 2025

Contents

1 Introduction

Spatiotemporal forecasting presents one of the most significant challenges in machine learning research, requiring models that can simultaneously capture complex spatial relationships and temporal dynamics. This problem is particularly crucial in domains such as epidemiology, where disease spread exhibits intricate patterns across both geographical regions and time periods. Traditional approaches often treat spatial and temporal components separately, failing to model their interdependence and leading to suboptimal predictions. Furthermore, many existing methods rely on predefined graph structures that may not accurately reflect the true underlying relationships in the data.

In this section, we introduce the Multi-Scale Temporal Graph Attention Network (MSTGAT), a novel deep learning architecture specifically designed to address these limitations. MSTGAT integrates several innovative components: (1) efficient feature extraction using depthwise separable convolutions, (2) adaptive graph learning through a low-rank attention mechanism, (3) multi-scale temporal modeling via dilated convolutions, and (4) horizon prediction with an adaptive refinement mechanism.

Our contributions can be summarized as follows:

- We propose a comprehensive end-to-end architecture that jointly models spatial dependencies and temporal dynamics at multiple scales.
- We introduce a low-rank graph attention mechanism that learns and adapts spatial relationships during training without relying on a predefined adjacency matrix.
- We develop a multi-scale temporal module that efficiently captures patterns across different time horizons through dilated convolutions with adaptive fusion.
- We design an adaptive refinement mechanism that combines model predictions with recent observations, improving forecasting accuracy particularly during regime changes.
- We optimize computational efficiency through strategic use of low-rank approximations and separable convolutions, enabling application to large-scale spatiotemporal forecasting problems.

The remainder of this paper is organized as follows: Section 2 formalizes the spatiotemporal forecasting problem. Section 3 presents the proposed MSTGAT architecture in detail. Section 4 provides theoretical analysis of the model's properties. Section 5 discusses practical implementation considerations. Section 6 explores model variants and ablation studies. Finally, Section 7 addresses limitations and future research directions.

2 Problem Formulation

Let us consider N geographical regions (e.g., cities, counties, or states) as nodes in a graph. The historical epidemic or resource usage/demand data is represented as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t]$, where $\mathbf{x}_z \in \mathbb{R}^N$ denotes the observed case counts across all N regions at time step z. For each specific region i, its temporal sequence is represented as $\mathbf{x}^i = [x_{i,1}, x_{i,2}, \ldots, x_{i,t}]$.

Our objective is to predict future case values $\mathbf{x}_{t+h}$ for a fixed horizon h, which may correspond to different forecasting tasks either short-term or long-term prediction. For any prediction task, we utilise a look-back window of length T to capture relevant historical patterns. Specifically, we use the sequence $[\mathbf{x}_{t-T+1}, \mathbf{x}_{t-T+2}, \ldots, \mathbf{x}_t] \in \mathbb{R}^{N \times T}$ to predict $\mathbf{x}_{t+h}$.

The spatial relationships between regions are encoded in a graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ represents the set of regions, $\mathcal{E}$ denotes the connections between regions, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. Each element a_{ij} of $\mathbf{A}$ quantifies the relationship strength between regions v_i and v_j.

The forecasting task can then be formalised as learning a function f that maps the historical data and graph structure to future predictions:

$$\hat{\mathbf{x}}_{t+h} = f([\mathbf{x}_{t-T+1}, \mathbf{x}_{t-T+2}, \ldots, \mathbf{x}_t]; \mathcal{G}) \tag{1}$$

where $\hat{\mathbf{x}}_{t+h}$ represents the predicted case counts for all regions at time $t + h$.

For efficient implementation and training, we adopt a batch-oriented tensor formulation. Let $\mathbf{X} \in \mathbb{R}^{B \times T \times N}$ represent the input tensor, where B is the batch size, T is the historical window length, and N is the number of regions. The forecasting task becomes:

$$\hat{\mathbf{Y}} = \mathcal{F}(\mathbf{X}; \Theta) \tag{2}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{B \times h \times N}$ contains predictions for all regions across the prediction horizon h, and Θ represents the learnable parameters of the model.

This formulation presents several challenges:

- **Dynamic Spatial Relationships**: The strength and nature of relationships between regions may vary over time and across different epidemic stages.
- **Multi-Scale Temporal Patterns**: Epidemiological data often exhibits patterns at different time scales, from daily fluctuations to weekly seasonality and longer-term trends.
- **Heterogeneity**: Regions may have different baseline characteristics, population densities, and response to interventions.
- **Limited Data**: Especially in emerging epidemics, historical data may be limited, requiring models that can generalize from small datasets.
- **Computational Efficiency**: Models must scale to large numbers of regions and long time series.

Our proposed MSTGAT architecture specifically addresses these challenges through its novel components and design principles.

3 MSTGAT Architecture

3.1 Architectural Overview

The Multi-Scale Temporal Graph Attention Network (MSTGAT) is designed as an end-to-end deep learning architecture for spatiotemporal forecasting. Figure 1 illustrates the overall framework of the proposed model.

At a high level, MSTGAT processes the input data $\mathbf{X} \in \mathbb{R}^{B \times T \times N}$ through four sequential stages:

1. **Feature Extraction**: Transforms raw time series into latent representations that capture relevant patterns.
2. **Spatial Dependency Modeling**: Captures relationships between different regions through an adaptive graph attention mechanism.
3. **Multi-Scale Temporal Modeling**: Models temporal dynamics at different time scales through dilated convolutions.
4. **Horizon Prediction**: Generates forecasts for future time steps with an adaptive refinement mechanism.

The architecture incorporates several design principles:

- **Parameter Efficiency**: We employ low-rank decompositions and separable convolutions to reduce the number of parameters and computational complexity.
- **Adaptive Learning**: Rather than relying on fixed graph structures or pre-specified temporal dependencies, the model learns and adapts these relationships during training.
- **Multi-Scale Processing**: We explicitly model patterns at different temporal scales to capture both short-term fluctuations and long-term trends.
- **Residual Learning**: Skip connections throughout the architecture facilitate gradient flow and information propagation.

We now describe each component in detail.

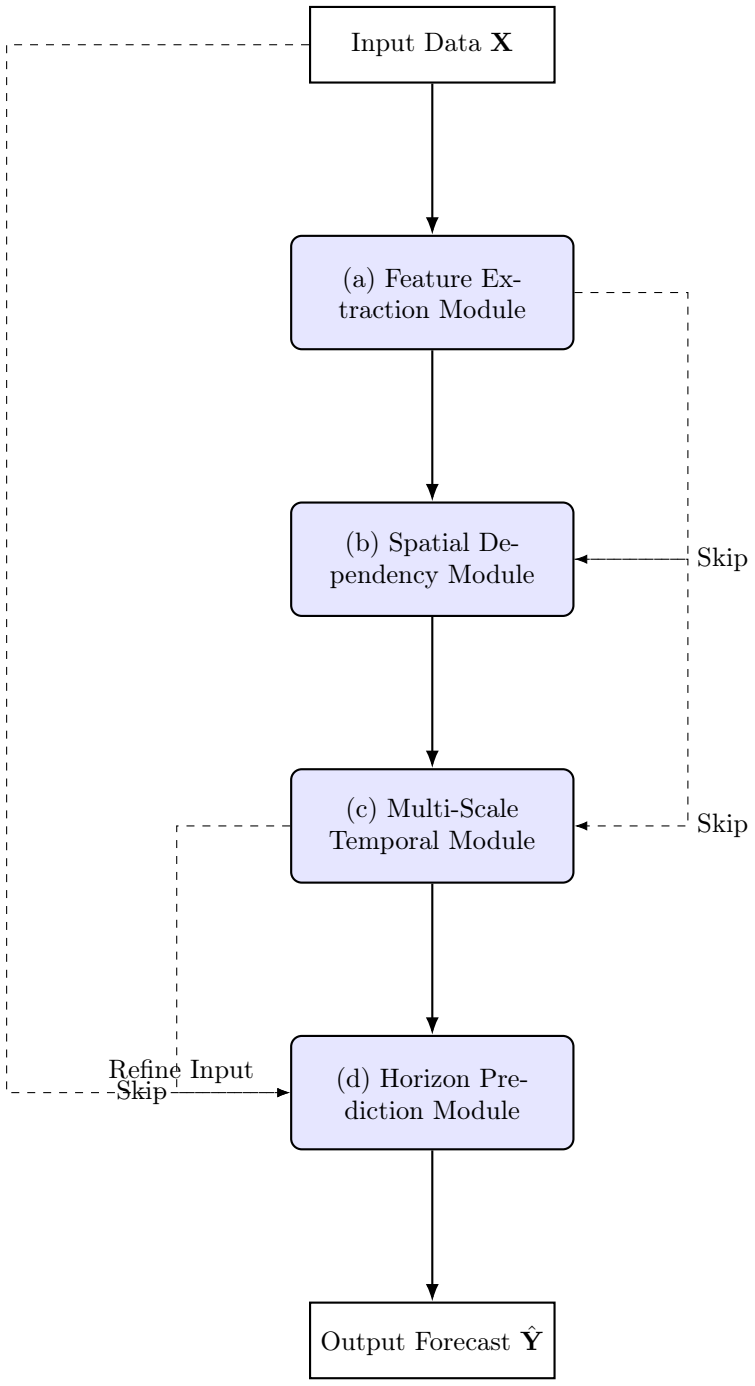

Figure 1: Overall architecture of the proposed MSTGAT model (TikZ). The framework consists of four key components: (a) Feature Extraction Module, (b) Spatial Dependency Module, (c) Multi-Scale Temporal Module, and (d) Horizon Prediction Module. Dashed lines indicate skip connections and input for refinement.

3.2 Feature Extraction Module

The feature extraction module transforms raw time series data into latent representations that capture relevant temporal patterns while maintaining computational efficiency. This module addresses several key challenges in spatiotemporal forecasting:

- Extract meaningful features from potentially noisy time series data
- Maintain locality of temporal patterns
- Process large volumes of data efficiently
- Create representations that are suitable for subsequent spatial and temporal modeling

3.2.1 Input Transformation

Given the input tensor $\mathbf{X} \in \mathbb{R}^{B \times T \times N}$, we first reshape it to facilitate convolution operations:

$$\mathbf{X}' = \text{Reshape}(\mathbf{X}) \in \mathbb{R}^{BN \times 1 \times T} \tag{3}$$

This transformation allows us to process each region's time series independently in the initial stage. The dimension 1 corresponds to the input channel dimension for the convolution operation.

3.2.2 Depthwise Separable Convolutions

Traditional convolutional neural networks (CNNs) apply full convolutions that operate across both the spatial (in our case, region) and channel dimensions simultaneously. This approach becomes computationally expensive as the number of regions and channels increases. Instead, we employ depthwise separable convolutions, which factorize the standard convolution operation into two more efficient steps:

1. **Depthwise Convolution**: This operation applies a separate filter to each input channel, thereby capturing temporal patterns within each channel independently:

$$\mathbf{Z}_{dep}(i, j, k) = \sum_{m=1}^{K} \mathbf{X}'(i, j, k + m - \lfloor \frac{K+1}{2} \rfloor) \cdot \mathbf{W}_{dep}(j, 1, m) \tag{4}$$

   where $\mathbf{W}_{dep} \in \mathbb{R}^{1 \times 1 \times K}$ is the depthwise convolutional kernel with kernel size K, i indexes the batch dimension, j indexes the channel dimension, and k indexes the temporal dimension.

2. **Pointwise Convolution**: This is a 1×1 convolution that projects the output of the depthwise convolution across channels:

$$\mathbf{Z}_{pw}(i, c_{out}, k) = \sum_{c_{in}=1}^{C_{in}} \mathbf{Z}_{dep}(i, c_{in}, k) \cdot \mathbf{W}_{pw}(c_{out}, c_{in}, 1) \tag{5}$$

   where $\mathbf{W}_{pw} \in \mathbb{R}^{C_{out} \times C_{in} \times 1}$ is the pointwise convolutional kernel, C_{in} is the number of input channels (1 in our case), and C_{out} is the number of output channels.

The complete depthwise separable convolution operation can be expressed as:

$$\mathbf{Z} = \text{DSConv}(\mathbf{X}') = \text{PWConv}(\text{DWConv}(\mathbf{X}')) \tag{6}$$

We enhance this operation with batch normalization and nonlinear activation:

$$\mathbf{Z}_{bn} = \text{BN}(\mathbf{Z}) \tag{7}$$

$$\mathbf{Z}_{act} = \sigma(\mathbf{Z}_{bn}) \tag{8}$$

where σ represents the ReLU activation function and BN denotes batch normalization.
This approach offers several advantages:

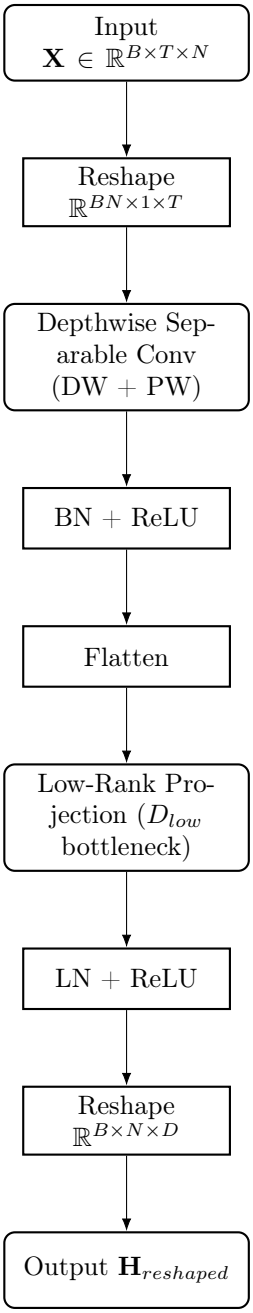

Figure 2: Diagram of the Feature Extraction Module (TikZ).

- **Computational Efficiency**: Reduces the number of operations from $\mathcal{O}(C_{in} \cdot C_{out} \cdot K \cdot T)$ for standard convolution to $\mathcal{O}(C_{in} \cdot K \cdot T + C_{in} \cdot C_{out})$ for depthwise separable convolution.
- **Parameter Efficiency**: Reduces the number of parameters from $C_{in} \cdot C_{out} \cdot K$ to $C_{in} \cdot K + C_{in} \cdot C_{out}$.
- **Regularization Effect**: The factorization provides an implicit regularization effect, potentially improving generalization.

The resulting tensor $\mathbf{Z}_{act} \in \mathbb{R}^{BN \times C_{out} \times T}$ contains rich temporal features extracted from the input data.

3.2.3 Low-Rank Feature Projection

To further enhance computational efficiency and reduce the risk of overfitting, we employ a low-rank bottleneck projection. This approach is particularly valuable in spatiotemporal forecasting, where data may be limited and high-dimensional representations can lead to overfitting.

The extracted features $\mathbf{Z}_{act}$ are first flattened along the temporal dimension:

$$\mathbf{Z}_{flat} = \text{Flatten}(\mathbf{Z}_{act}) \in \mathbb{R}^{BN \times (C_{out} \cdot T)} \tag{9}$$

We then project these features through a bottleneck architecture:

$$\mathbf{H}_{low} = \mathbf{Z}_{flat}\mathbf{W}_{low} + \mathbf{b}_{low} \in \mathbb{R}^{BN \times D_{low}} \tag{10}$$

$$\mathbf{H}_{high} = \mathbf{H}_{low}\mathbf{W}_{high} + \mathbf{b}_{high} \in \mathbb{R}^{BN \times D} \tag{11}$$

$$\mathbf{H} = \sigma(\text{LN}(\mathbf{H}_{high})) \in \mathbb{R}^{BN \times D} \tag{12}$$

where D_{low} represents the bottleneck dimensionality, D is the hidden dimensionality, $\mathbf{W}_{low} \in \mathbb{R}^{(C_{out} \cdot T) \times D_{low}}$ and $\mathbf{W}_{high} \in \mathbb{R}^{D_{low} \cdot D}$ are the projection matrices, $\mathbf{b}_{low} \in \mathbb{R}^{D_{low}}$ and $\mathbf{b}_{high} \in \mathbb{R}^{D}$ are bias terms, LN denotes layer normalization, and σ is the ReLU activation function.

Finally, we reshape $\mathbf{H}$ back to a form suitable for spatial modeling:

$$\mathbf{H}_{reshaped} = \text{Reshape}(\mathbf{H}) \in \mathbb{R}^{B \times N \times D} \tag{13}$$

This bottleneck architecture serves multiple purposes:

- **Dimensionality Reduction**: Reduces the feature dimensionality from $C_{out} \cdot T$ to D, where typically $D < C_{out} \cdot T$, decreasing the number of parameters in subsequent layers.
- **Information Distillation**: Forces the model to extract essential features by compressing information through the bottleneck.
- **Improved Gradient Flow**: The low-rank projection can help mitigate vanishing gradient problems.
- **Enhanced Generalization**: The reduced parameter space helps prevent overfitting, particularly valuable in epidemiological forecasting where data may be limited.

The computational complexity of this projection is $\mathcal{O}(BN \cdot C_{out} \cdot T \cdot D_{low} + BN \cdot D_{low} \cdot D)$, compared to $\mathcal{O}(BN \cdot C_{out} \cdot T \cdot D)$ for a direct projection, representing significant savings when $D_{low} \ll \min(C_{out} \cdot T, D)$.

3.2.4 Theoretical Insights on Feature Extraction

The feature extraction module can be understood from several theoretical perspectives:

Signal Processing Perspective: The depthwise convolution acts as a set of learned filters that extract temporal patterns at different frequencies. By applying separate filters to each input channel, the model can capture channel-specific patterns. The pointwise convolution then integrates information across channels, similar to a learned cross-channel correlation.

Information Bottleneck Perspective: The low-rank projection can be viewed through the lens of the information bottleneck principle [**?**]. By compressing the representation to a lower-dimensional space, the model is forced to retain only the most predictive features, discarding irrelevant information. This leads to representations that are both minimal and sufficient for the forecasting task.

Manifold Learning Perspective: Epidemiological data often lies on a low-dimensional manifold within the high-dimensional observation space. The bottleneck projection can be seen as learning this manifold, with D_{low} representing its intrinsic dimensionality.

3.3 Spatial Dependency Modeling

After feature extraction, we model spatial dependencies using a novel graph attention mechanism. Unlike conventional graph neural networks that rely on a predefined adjacency matrix, our approach learns and adapts the spatial relationships during training. This is particularly valuable in epidemiological forecasting, where the true underlying spatial relationships may be unknown or may evolve over time.

3.3.1 Motivation and Theoretical Foundations

Traditional graph-based models for spatiotemporal forecasting typically rely on fixed graph structures derived from geographical proximity, transportation networks, or other domain knowledge. However, these predefined structures may not accurately capture the complex dependencies in epidemic spread, which can be influenced by factors such as:

- Population mobility patterns
- Socioeconomic factors
- Healthcare resource distribution
- Intervention policies
- Environmental conditions

Furthermore, these dependencies may evolve over the course of an epidemic. Our approach draws inspiration from graph attention networks (GAT) [**?**] and transformer architectures [**?**], but introduces several innovations to address the specific challenges of spatiotemporal forecasting.

3.3.2 Low-Rank Graph Attention

The core of our spatial modeling is a multi-head attention mechanism with low-rank decomposition. For clarity, we first describe the process for a single attention head and then extend it to the multi-head case.

Given the node features $\mathbf{H}_{reshaped} \in \mathbb{R}^{B \times N \times D}$ from the feature extraction module, we first project them to obtain query, key, and value representations. To maintain computational efficiency, we use a low-rank projection:

$$\mathbf{QKV}_{low} = \mathbf{H}_{reshaped}\mathbf{W}_{qkv\_low} + \mathbf{b}_{qkv\_low} \in \mathbb{R}^{B \times N \times 3D_{low}} \tag{14}$$

$$\mathbf{QKV} = \mathbf{QKV}_{low}\mathbf{W}_{qkv\_high} + \mathbf{b}_{qkv\_high} \in \mathbb{R}^{B \times N \times 3D} \tag{15}$$

where $\mathbf{W}_{qkv\_low} \in \mathbb{R}^{D \times 3D_{low}}$, $\mathbf{W}_{qkv\_high} \in \mathbb{R}^{3D_{low} \times 3D}$, and $\mathbf{b}_{qkv\_low} \in \mathbb{R}^{3D_{low}}$, $\mathbf{b}_{qkv\_high} \in \mathbb{R}^{3D}$ are learnable parameters.

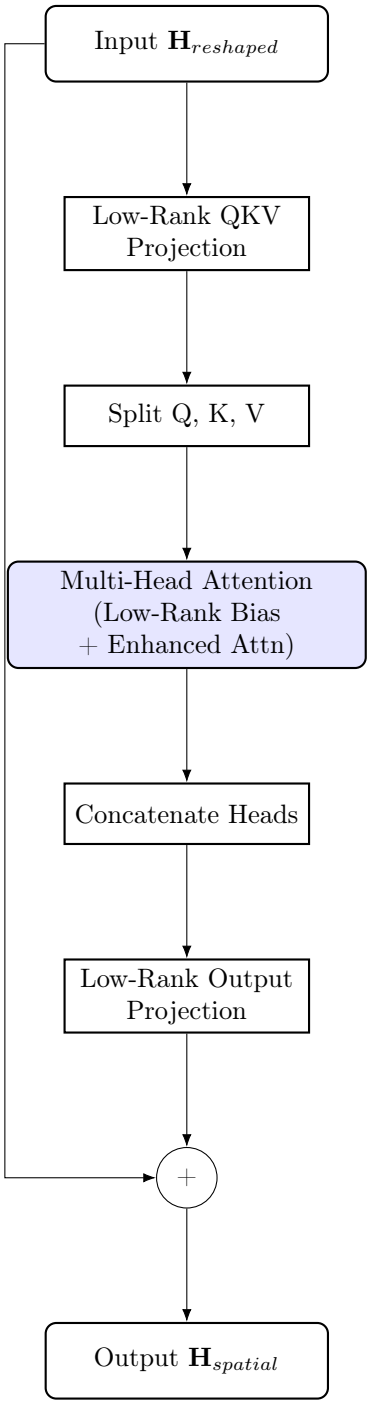

Figure 3: Diagram of the Spatial Dependency Modeling Module (TikZ). The module learns adaptive spatial relationships through multi-head attention with a low-rank bias term.

We then split **QKV** into query, key, and value representations:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \text{Split}(\mathbf{QKV}) \in \mathbb{R}^{B\times N\times D} \times \mathbb{R}^{B\times N\times D} \times \mathbb{R}^{B\times N\times D} \tag{16}$$

For multi-head attention with H heads, we reshape these tensors:

$$\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h = \text{Reshape}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{B\times H\times N\times \frac{D}{H}} \times \mathbb{R}^{B\times H\times N\times \frac{D}{H}} \times \mathbb{R}^{B\times H\times N\times \frac{D}{H}} \tag{17}$$

3.3.3 Learnable Graph Structure

A key innovation in our approach is the incorporation of a learnable graph structure. Rather than computing attention scores based solely on the query-key interactions, we introduce a low-rank bias term that encodes the underlying graph structure:

$$\mathbf{U}_h \in \mathbb{R}^{N\times D_{low}}, \mathbf{V}_h \in \mathbb{R}^{D_{low}\times N} \tag{18}$$

$$\mathbf{A}_{bias,h} = \mathbf{U}_h \mathbf{V}_h \in \mathbb{R}^{N\times N} \tag{19}$$

where $\mathbf{U}_h$ and $\mathbf{V}_h$ are learnable parameters for each attention head h. This low-rank formulation reduces the number of parameters from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \cdot D_{low})$, making it feasible to learn the graph structure even for large numbers of nodes.

The attention scores are then computed as:

$$\mathbf{A}_{raw,h} = \frac{\mathbf{Q}_h \mathbf{K}_h^T}{\sqrt{D/H}} + \mathbf{A}_{bias,h} \in \mathbb{R}^{B\times H\times N\times N} \tag{20}$$

$$\mathbf{A}_h = \text{softmax}(\mathbf{A}_{raw,h}) \in \mathbb{R}^{B\times H\times N\times N} \tag{21}$$

where the softmax is applied along the last dimension. The scaling factor $\sqrt{D/H}$ helps stabilize training by preventing the attention scores from becoming too concentrated.

3.3.4 Enhanced Attention Mechanism

To further enhance the attention mechanism, we incorporate a nonlinear transformation inspired by the ELU (Exponential Linear Unit) activation function. This helps capture more complex relationships between nodes:

$$\tilde{\mathbf{Q}}_h = \text{ELU}(\mathbf{Q}_h) + 1.0 \tag{22}$$

$$\tilde{\mathbf{K}}_h = \text{ELU}(\mathbf{K}_h) + 1.0 \tag{23}$$

The addition of 1.0 ensures that the transformed values are positive, which is important for the stability of the attention mechanism. We then compute key-value products:

$$\mathbf{KV}_h = \tilde{\mathbf{K}}_h \otimes \mathbf{V}_h \in \mathbb{R}^{B\times H\times \frac{D}{H}\times \frac{D}{H}} \tag{24}$$

where $\otimes$ represents the batched outer product operation.

To ensure proper normalization, we compute a normalization factor:

$$\mathbf{z}_h = \frac{1}{\sum_{j=1}^{N} \tilde{\mathbf{K}}_h(j) + \epsilon} \in \mathbb{R}^{B\times H\times N} \tag{25}$$

where ϵ is a small constant for numerical stability. The attended features are then computed as:

$$\mathbf{O}_h = \tilde{\mathbf{Q}}_h \otimes \mathbf{KV}_h \otimes \mathbf{z}_h \in \mathbb{R}^{B\times H\times N\times \frac{D}{H}} \tag{26}$$

3.3.5 Output Projection and Residual Connection

The outputs from all heads are concatenated and projected back to the original dimension:

$$\mathbf{O} = \text{Concat}(\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_H) \in \mathbb{R}^{B \times N \times D} \tag{27}$$

To maintain computational efficiency, we again use a low-rank projection:

$$\mathbf{O}_{low} = \mathbf{O}\mathbf{W}_{out\_low} + \mathbf{b}_{out\_low} \in \mathbb{R}^{B \times N \times D_{low}} \tag{28}$$

$$\mathbf{O}_{high} = \mathbf{O}_{low}\mathbf{W}_{out\_high} + \mathbf{b}_{out\_high} \in \mathbb{R}^{B \times N \times D} \tag{29}$$

Finally, we apply a residual connection to facilitate gradient flow:

$$\mathbf{H}_{spatial} = \mathbf{O}_{high} + \mathbf{H}_{reshaped} \in \mathbb{R}^{B \times N \times D} \tag{30}$$

3.3.6 Attention Regularization

To encourage sparsity in the learned attention weights and prevent overfitting to spurious spatial correlations, we apply L1 regularization to the attention matrix:

$$\mathcal{L}_{attn} = \lambda_{attn} \sum_{h=1}^{H} \|\mathbf{A}_h\|_1 \tag{31}$$

where λ_{attn} is a hyperparameter controlling the regularization strength. This regularization term is added to the overall loss function during training.

3.3.7 Interpretability of Learned Spatial Relationships

A significant advantage of our approach is the interpretability of the learned attention weights. The attention matrix $\mathbf{A}_h$ provides insights into the spatial relationships learned by the model. Specifically, $\mathbf{A}_h(i, j)$ represents the influence of region j on region i in the prediction process.

By visualizing these attention weights, we can:

- Identify key transmission pathways in epidemic spread
- Discover unexpected spatial relationships that might not be captured by geographical proximity
- Monitor how spatial dependencies evolve over the course of an epidemic
- Validate the model's learned relationships against domain knowledge

This interpretability is particularly valuable in epidemiological applications, where understanding the spatial patterns of disease spread is crucial for intervention planning.

3.4 Multi-Scale Temporal Modeling

The temporal dynamics of epidemiological data often exhibit patterns at different time scales, from daily fluctuations to weekly seasonality and longer-term trends. Traditional approaches that focus on a single time scale may fail to capture the full spectrum of temporal dependencies. Our multi-scale temporal module addresses this challenge through dilated convolutions and adaptive fusion.

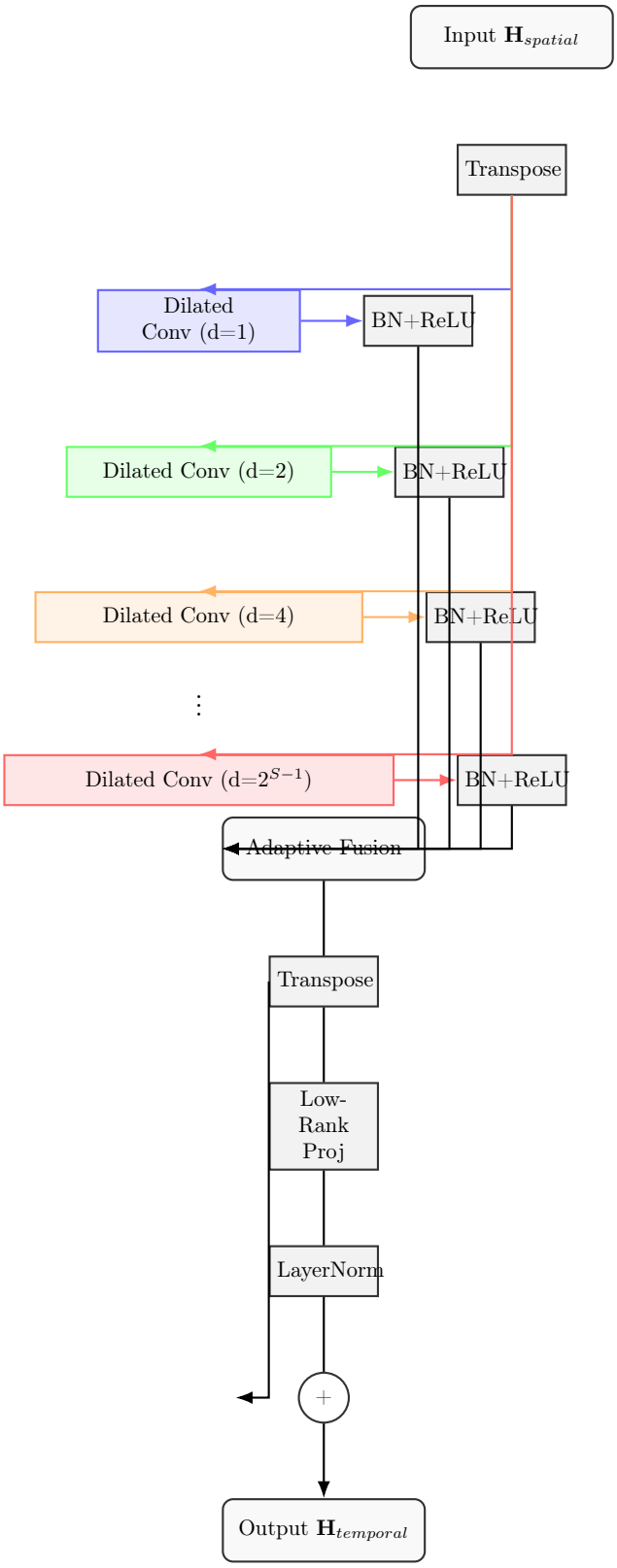

Figure 4: Multi-Scale Temporal Module (scaled). Colored paths indicate different dilation scales for clarity.

3.4.1 Motivation and Theoretical Foundations

Epidemiological time series exhibit complex temporal patterns influenced by various factors:

- **Short-term fluctuations**: Day-to-day variations due to testing patterns, reporting delays, and random noise
- **Medium-term patterns**: Weekly seasonality, incubation periods, and intermediate-term effects of interventions
- **Long-term trends**: Overall epidemic trajectory, seasonal effects, and long-term policy impacts

Capturing these multi-scale patterns requires a model that can efficiently process information at different temporal resolutions. Dilated convolutions [**?**] provide an elegant solution to this challenge, offering an exponentially increasing receptive field with linear parameter growth.

3.4.2 Multi-Scale Dilated Convolutions

The core of our temporal modeling is a set of dilated convolutions with increasing dilation rates. Given the spatial features $\mathbf{H}_{spatial} \in \mathbb{R}^{B\times N\times D}$, we first reshape them to facilitate temporal convolution operations:

$$\mathbf{H}_{temp} = \text{Transpose}(\mathbf{H}_{spatial}) \in \mathbb{R}^{B\times D\times N} \tag{32}$$

For each scale $s \in \{1, 2, \ldots, S\}$, we apply a dilated convolution with dilation rate 2^{s-1}:

$$\mathbf{Z}_s = \text{Conv1D}(\mathbf{H}_{temp}, \text{kernel\_size} = K, \text{dilation} = 2^{s-1}, \text{padding} = (K-1)\cdot 2^{s-1}/2) \in \mathbb{R}^{B\times D\times N} \tag{33}$$

$$\mathbf{Z}_{s,bn} = \text{BatchNorm1D}(\mathbf{Z}_s) \in \mathbb{R}^{B\times D\times N} \tag{34}$$

$$\mathbf{Z}_{s,act} = \sigma(\mathbf{Z}_{s,bn}) \in \mathbb{R}^{B\times D\times N} \tag{35}$$

$$\mathbf{Z}_{s,drop} = \text{Dropout}(\mathbf{Z}_{s,act}, p = p_{drop}) \in \mathbb{R}^{B\times D\times N} \tag{36}$$

where K is the kernel size, σ is the ReLU activation function, and p_{drop} is the dropout probability. The dilation rates create an exponentially increasing receptive field:

- Scale 1 (dilation = 1): Receptive field of size K
- Scale 2 (dilation = 2): Receptive field of size $K + (K-1)$
- Scale 3 (dilation = 4): Receptive field of size $K + 3(K-1)$
- Scale s (dilation = 2^{s-1}): Receptive field of size $K + (2^s - 1)(K-1)$

This allows the model to efficiently capture dependencies at different time scales without the need for deep architectures, addressing the vanishing gradient problem often encountered in modeling long-term dependencies.

3.4.3 Adaptive Fusion Mechanism

Different temporal scales may have varying importance depending on the specific forecasting task, epidemic stage, or region characteristics. Rather than using fixed weights to combine the multi-scale features, we employ an adaptive fusion mechanism with learnable parameters:

$$\mathbf{w} \in \mathbb{R}^S \tag{37}$$

$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{w}) \in \mathbb{R}^S \tag{38}$$

The features from different scales are then combined using these weights:

$$\mathbf{Z}_{fused} = \sum_{s=1}^{S} \alpha_s \mathbf{Z}_{s,drop} \in \mathbb{R}^{B \times D \times N} \tag{39}$$

This adaptive fusion allows the model to focus on the most relevant temporal scales for each forecasting task. For example, short-term forecasting may rely more on recent patterns, while long-term forecasting may give more weight to broader trends.

3.4.4 Low-Rank Projection and Residual Connection

Similar to previous modules, we apply a low-rank projection to maintain computational efficiency:

$$\mathbf{Z}_{fused}^T = \text{Transpose}(\mathbf{Z}_{fused}) \in \mathbb{R}^{B \times N \times D} \tag{40}$$

$$\mathbf{Z}_{low} = \mathbf{Z}_{fused}^T \mathbf{W}_{fusion\_low} + \mathbf{b}_{fusion\_low} \in \mathbb{R}^{B \times N \times D_{low}} \tag{41}$$

$$\mathbf{Z}_{high} = \mathbf{Z}_{low} \mathbf{W}_{fusion\_high} + \mathbf{b}_{fusion\_high} \in \mathbb{R}^{B \times N \times D} \tag{42}$$

We then apply layer normalization and a residual connection:

$$\mathbf{H}_{temporal} = \text{LayerNorm}(\mathbf{Z}_{high} + \mathbf{Z}_{fused}^T) \in \mathbb{R}^{B \times N \times D} \tag{43}$$

The residual connection helps maintain gradient flow during training, while the layer normalization stabilizes the training process.

3.4.5 Theoretical Insights on Multi-Scale Temporal Modeling

The multi-scale temporal module can be understood from several theoretical perspectives:

Wavelet Transform Perspective: The dilated convolutions can be viewed as analogous to a learnable wavelet transform, which decomposes the signal into components at different scales. The adaptive fusion then reconstructs the signal by combining these components with learned weights.

Hierarchical Processing Perspective: The different scales create a hierarchical representation of the temporal dynamics, with lower scales capturing local patterns and higher scales capturing global trends. This hierarchical structure is well-suited to the multi-scale nature of epidemiological data.

Memory Capacity Perspective: The exponentially increasing receptive field allows the model to efficiently memorize patterns at different time scales, similar to how long short-term memory (LSTM) networks maintain multiple time scales of memory but with a more computationally efficient architecture.

3.5 Horizon Prediction with Adaptive Refinement

The final component of our MSTGAT architecture is the horizon prediction module, which generates forecasts for future time steps. This module combines the rich spatiotemporal representations from previous layers with an adaptive refinement mechanism that leverages the most recent observations.

3.5.1 Motivation

Epidemiological forecasting faces several challenges:

- **Regime Changes**: Sudden shifts in epidemic dynamics due to policy changes, variant emergence, or behavioral adaptations
- **Uncertainty Growth**: Prediction uncertainty typically increases with the forecast horizon
- **Baseline Drift**: The baseline level may shift over time due to changes in testing, reporting, or population immunity

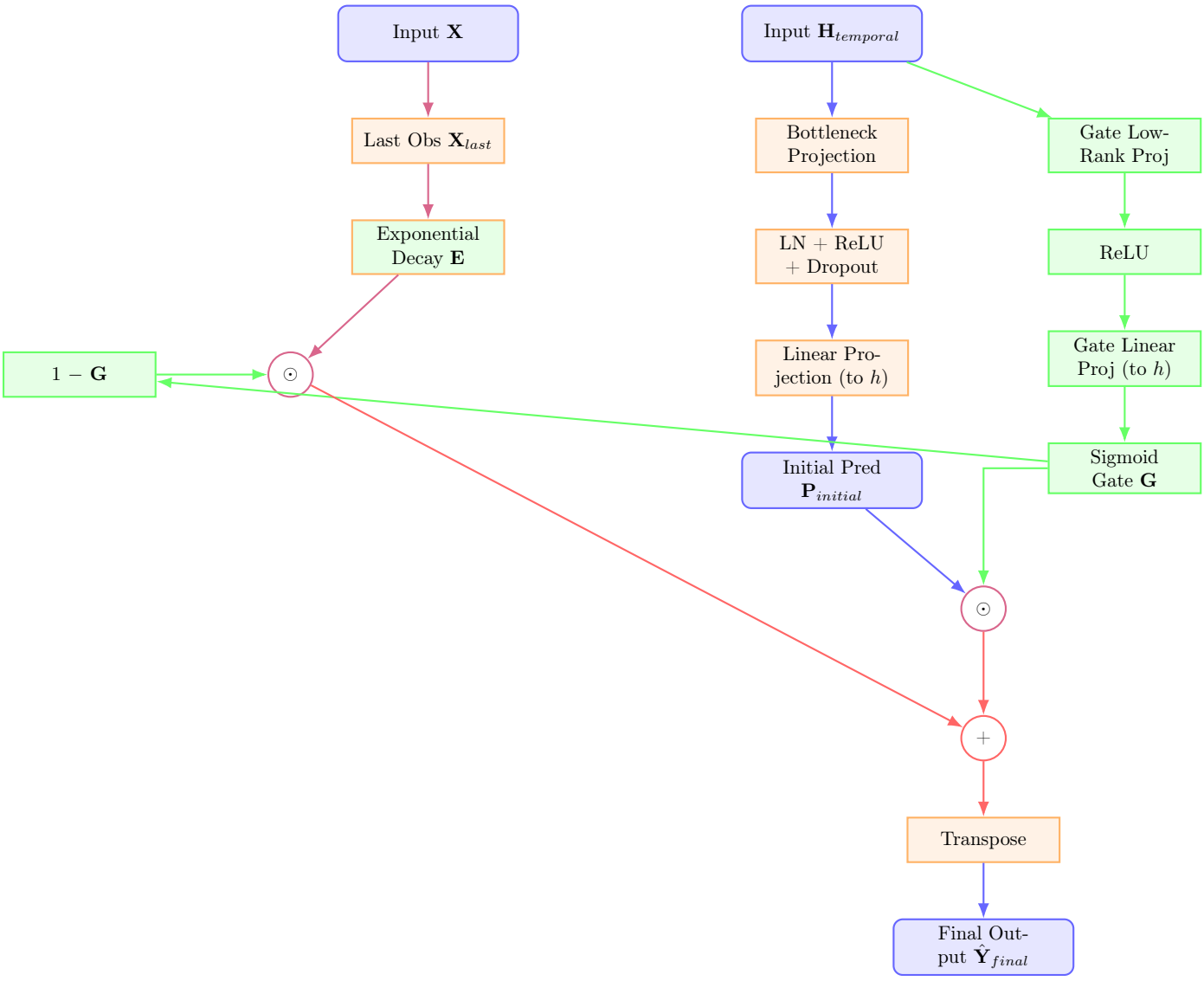

Figure 5: Diagram of the Horizon Prediction Module (scaled). The module combines predictions and decay via colored pathways for clarity.

These challenges motivate our adaptive refinement approach, which combines model predictions with an extrapolation from the most recent observations. This hybrid approach allows the model to adapt more quickly to regime changes while maintaining its ability to capture complex spatiotemporal patterns.

3.5.2 Bottleneck Prediction Architecture

Given the temporal features $\mathbf{H}_{temporal} \in \mathbb{R}^{B \times N \times D}$, we first apply a bottleneck architecture to generate initial predictions:

$$\mathbf{P}_{low} = \mathbf{H}_{temporal}\mathbf{W}_{pred\_low} + \mathbf{b}_{pred\_low} \in \mathbb{R}^{B \times N \times D_{low}} \tag{44}$$

$$\mathbf{P}_{mid} = \text{LayerNorm}(\mathbf{P}_{low}) \in \mathbb{R}^{B \times N \times D_{low}} \tag{45}$$

$$\mathbf{P}_{mid\_act} = \sigma(\mathbf{P}_{mid}) \in \mathbb{R}^{B \times N \times D_{low}} \tag{46}$$

$$\mathbf{P}_{mid\_drop} = \text{Dropout}(\mathbf{P}_{mid\_act}, p = p_{drop}) \in \mathbb{R}^{B \times N \times D_{low}} \tag{47}$$

$$\mathbf{P}_{initial} = \mathbf{P}_{mid\_drop}\mathbf{W}_{pred\_high} + \mathbf{b}_{pred\_high} \in \mathbb{R}^{B \times N \times h} \tag{48}$$

where h is the prediction horizon, $\mathbf{W}_{pred\_low} \in \mathbb{R}^{D \times D_{low}}$, $\mathbf{W}_{pred\_high} \in \mathbb{R}^{D_{low} \times h}$, $\mathbf{b}_{pred\_low} \in \mathbb{R}^{D_{low}}$, and $\mathbf{b}_{pred\_high} \in \mathbb{R}^{h}$ are learnable parameters, and p_{drop} is the dropout probability.

3.5.3 Adaptive Refinement Gate

The key innovation in our prediction module is the adaptive refinement mechanism, which combines the model's initial predictions with an extrapolation from the most recent observations. This is particularly valuable in epidemiological forecasting, where the most recent data points often provide crucial information about current trends.

First, we extract the last observed values from the input data:

$$\mathbf{X}_{last} = \mathbf{X}[:, -1, :] \in \mathbb{R}^{B \times N} \tag{49}$$

We then compute a gating mechanism that determines the relative importance of the model predictions versus the recent trend:

$$\mathbf{G}_{low} = \mathbf{H}_{temporal}\mathbf{W}_{gate\_low} + \mathbf{b}_{gate\_low} \in \mathbb{R}^{B \times N \times D_{low}} \tag{50}$$

$$\mathbf{G}_{act} = \sigma(\mathbf{G}_{low}) \in \mathbb{R}^{B \times N \times D_{low}} \tag{51}$$

$$\mathbf{G} = \text{sigmoid}(\mathbf{G}_{act}\mathbf{W}_{gate\_high} + \mathbf{b}_{gate\_high}) \in \mathbb{R}^{B \times N \times h} \tag{52}$$

where $\mathbf{W}_{gate\_low} \in \mathbb{R}^{D \times D_{low}}$, $\mathbf{W}_{gate\_high} \in \mathbb{R}^{D_{low} \times h}$, $\mathbf{b}_{gate\_low} \in \mathbb{R}^{D_{low}}$, and $\mathbf{b}_{gate\_high} \in \mathbb{R}^{h}$ are learnable parameters, and σ is the ReLU activation function.

3.5.4 Exponential Decay from Last Observation

We model the extrapolation from the most recent observations using an exponential decay function. This simple yet effective approach captures the natural tendency of many processes to revert to a baseline level over time:

$$\mathbf{T} = [1, 2, \ldots, h] \in \mathbb{R}^{h} \tag{53}$$

$$\mathbf{X}_{last\_expanded} = \mathbf{X}_{last}.unsqueeze(-1) \in \mathbb{R}^{B \times N \times 1} \tag{54}$$

$$\mathbf{E} = \mathbf{X}_{last\_expanded} \cdot \exp(-\beta \cdot \mathbf{T}) \in \mathbb{R}^{B \times N \times h} \tag{55}$$

where β is a decay factor (typically set to 0.1). This exponential decay function provides a simple but effective baseline that assumes the epidemic will gradually stabilize from its current level.

3.5.5 Adaptive Fusion of Predictions

The final predictions are obtained by adaptively combining the model's initial predictions with the exponential decay based on the computed gate values:

$$\hat{\mathbf{Y}} = \mathbf{G} \odot \mathbf{P}_{initial} + (1 - \mathbf{G}) \odot \mathbf{E} \in \mathbb{R}^{B \times N \times h} \tag{56}$$

where $\odot$ denotes element-wise multiplication.

We typically transpose the result to match the expected output format:

$$\hat{\mathbf{Y}}_{final} = \text{Transpose}(\hat{\mathbf{Y}}, \text{dim1} = 1, \text{dim2} = 2) \in \mathbb{R}^{B \times h \times N} \tag{57}$$

3.5.6 Theoretical Insights on Adaptive Refinement

The adaptive refinement mechanism can be understood from several perspectives:

Ensemble Learning Perspective: The mechanism effectively creates a dynamic ensemble of two forecasting approaches: a complex spatiotemporal model and a simple exponential decay. The gating mechanism learns to weight these approaches based on the specific context, potentially achieving better performance than either approach alone.

Regime Change Adaptation: During stable periods, the model may rely more on its learned patterns (high gate values). During regime changes or unusual events, it may give more weight to the recent trend (low gate values), allowing faster adaptation to new conditions.

Uncertainty Calibration: The mechanism implicitly incorporates uncertainty in the forecasting process. When the model is uncertain about its predictions (e.g., for longer horizons or unusual patterns), it may rely more on the simple exponential decay, providing a form of regularization.

3.6 Integration of Components and Information Flow

The power of MSTGAT lies in the complementary interaction between its spatial and temporal components. While traditional approaches often model these dependencies separately, our architecture allows them to mutually reinforce each other:

- The feature extraction module captures local temporal patterns at each node independently, creating a foundation for subsequent spatial and temporal modeling.
- The spatial attention module then contextualizes these features by aggregating information across connected nodes, enhancing each node's representation with relevant neighborhood information. This process captures how the epidemic in one region affects and is affected by other regions.
- The temporal module subsequently processes these spatially-enriched features across multiple time scales, allowing the model to capture how spatial patterns evolve over time. This acknowledges that spatial dependencies themselves may change at different temporal scales and stages of an epidemic.
- Finally, the prediction module combines all this information to generate forecasts, with an adaptive refinement mechanism that balances the complex spatiotemporal patterns learned by the model with a simple extrapolation from recent observations.

This bidirectional information flow between spatial and temporal domains is crucial for phenomena like epidemics, where spatial proximity affects transmission timing, and temporal patterns (like mobility) affect spatial spread.

4 Theoretical Analysis

In this section, we present a theoretical analysis of MSTGAT, focusing on its expressivity, computational complexity, and memory requirements.

4.1 Expressivity Analysis

The expressivity of a model refers to its ability to represent complex functions or patterns. MSTGAT's expressivity can be analyzed along several dimensions:

4.1.1 Spatial Expressivity

The graph attention mechanism in MSTGAT can represent arbitrary pairwise relationships between regions. The combination of content-based attention (derived from query-key interactions) and learnable graph structure (through the low-rank bias term) allows the model to capture both static geographical relationships and dynamic interaction patterns.

Specifically, for any true underlying adjacency matrix $\mathbf{A}^* \in \mathbb{R}^{N \times N}$, there exists a setting of the parameters $\mathbf{U}_h$ and $\mathbf{V}_h$ such that $\mathbf{U}_h\mathbf{V}_h$ approximates $\mathbf{A}^*$ with a reconstruction error bounded by:

$$\|\mathbf{A}^* - \mathbf{U}_h\mathbf{V}_h\|_F \leq \sqrt{\sum_{i=D_{low}+1}^{N} \sigma_i^2(\mathbf{A}^*)} \tag{58}$$

where $\sigma_i(\mathbf{A}^*)$ is the i-th singular value of $\mathbf{A}^*$. This error decreases as D_{low} increases, providing a clear trade-off between approximation accuracy and computational efficiency.

4.1.2 Temporal Expressivity

The multi-scale temporal module captures patterns at different time scales through dilated convolutions. The exponentially increasing receptive field allows the model to efficiently represent both short-term and long-term dependencies.

For a temporal sequence of length T, the maximum receptive field size with S scales and kernel size K is:

$$R_{\max} = K + (2^S - 1)(K - 1) \tag{59}$$

This allows the model to capture dependencies spanning a large portion of the input sequence with a relatively small number of scales. For example, with $K = 3$ and $S = 4$, the model can capture dependencies spanning 27 time steps.

4.1.3 Combined Spatiotemporal Expressivity

The sequential processing of spatial and temporal dependencies in MSTGAT enables it to capture complex spatiotemporal patterns. The composition of these modules allows the model to represent functions that depend on both the spatial configuration and temporal evolution of the data.

More formally, MSTGAT can approximate functions of the form:

$$f(\mathbf{X}) = g(\phi_S(\phi_T(\mathbf{X}))) \tag{60}$$

where ϕ_T captures temporal patterns, ϕ_S captures spatial relationships, and g is the prediction function. This compositional structure allows the model to capture how spatial patterns evolve over time and how temporal patterns vary across space.

4.2 Computational Complexity Analysis

The computational efficiency of MSTGAT is a key consideration for its applicability to large-scale spatiotemporal forecasting problems. We analyze the computational complexity of each component and the overall model:

4.2.1 Feature Extraction Module

The depthwise separable convolution reduces the computational complexity compared to standard convolution:

$$\mathcal{O}(BN \cdot 1 \cdot K \cdot T + BN \cdot 1 \cdot C_{out}) = \mathcal{O}(BN \cdot (K \cdot T + C_{out})) \tag{61}$$

The low-rank projection has a complexity of:

$$\mathcal{O}(BN \cdot C_{out} \cdot T \cdot D_{low} + BN \cdot D_{low} \cdot D) \tag{62}$$

4.2.2 Spatial Attention Module

The computation of query, key, and value representations has a complexity of:

$$\mathcal{O}(BN \cdot D \cdot 3D_{low} + BN \cdot 3D_{low} \cdot 3D) = \mathcal{O}(BN \cdot D \cdot D_{low} + BN \cdot D_{low} \cdot D) \tag{63}$$

The attention computation has a complexity of:

$$\mathcal{O}(BH \cdot N^2 \cdot D/H) = \mathcal{O}(BN^2 \cdot D) \tag{64}$$

The low-rank graph structure adds a complexity of:

$$\mathcal{O}(H \cdot N \cdot D_{low} + H \cdot D_{low} \cdot N) = \mathcal{O}(H \cdot N \cdot D_{low}) \tag{65}$$

4.2.3 Temporal Module

Each dilated convolution has a complexity of:

$$\mathcal{O}(BN \cdot D \cdot K) = \mathcal{O}(BN \cdot D \cdot K) \tag{66}$$

With S scales, the total complexity is:

$$\mathcal{O}(S \cdot BN \cdot D \cdot K) \tag{67}$$

The adaptive fusion has a complexity of:

$$\mathcal{O}(BN \cdot D \cdot S) \tag{68}$$

4.2.4 Prediction Module

The bottleneck prediction has a complexity of:

$$\mathcal{O}(BN \cdot D \cdot D_{low} + BN \cdot D_{low} \cdot h) \tag{69}$$

The adaptive refinement has a complexity of:

$$\mathcal{O}(BN \cdot D \cdot D_{low} + BN \cdot D_{low} \cdot h + BN \cdot h) \tag{70}$$

4.2.5 Overall Complexity

The overall computational complexity of MSTGAT is dominated by the attention computation in the spatial module:

$$\mathcal{O}(BN^2 \cdot D) \tag{71}$$

This quadratic dependence on the number of nodes N is a common characteristic of attention-based models. However, our low-rank formulations throughout the architecture significantly reduce the constant factors in this complexity, making the model practical for moderate-sized problems.

For very large numbers of nodes, further optimizations such as sparse attention or hierarchical approaches could be considered, but these are beyond the scope of the current work.

4.3 Memory Requirements Analysis

The memory requirements of MSTGAT are an important consideration for its practical implementation. We analyze the memory requirements for parameters and activations:

4.3.1 Parameter Memory

The total number of parameters in MSTGAT can be summarized as:

- Feature Extraction: $\mathcal{O}(K + C_{out} + C_{out} \cdot T \cdot D_{low} + D_{low} \cdot D)$
- Spatial Attention: $\mathcal{O}(D \cdot D_{low} + D_{low} \cdot D + H \cdot N \cdot D_{low})$
- Temporal Module: $\mathcal{O}(S \cdot D \cdot K + S + D \cdot D_{low} + D_{low} \cdot D)$
- Prediction Module: $\mathcal{O}(D \cdot D_{low} + D_{low} \cdot h + D \cdot D_{low} + D_{low} \cdot h)$

The total parameter count is dominated by the $\mathcal{O}(H \cdot N \cdot D_{low})$ term from the low-rank graph structure. However, this is significantly less than the $\mathcal{O}(N^2)$ parameters required for a full adjacency matrix.

4.3.2 Activation Memory

The peak activation memory during forward and backward passes is determined by the largest intermediate tensors, which occur in the attention computation:

$$\mathcal{O}(BH \cdot N^2) \tag{72}$$

This quadratic dependence on the number of nodes can be a limiting factor for very large graphs. Techniques such as gradient checkpointing or attention approximation could be used to reduce this memory requirement if needed.

5 Implementation Considerations

The successful application of MSTGAT to real-world spatiotemporal forecasting problems requires careful consideration of several implementation aspects:

5.1 Data Preprocessing

Epidemiological data often requires significant preprocessing due to issues such as:

- Missing values
- Reporting delays and artifacts
- Different scales across regions
- Outliers and data quality issues

We recommend the following preprocessing steps:

- **Handling Missing Values**: Use interpolation for isolated missing values and more sophisticated techniques such as matrix completion for structured missingness.
- **Normalization**: Scale the data to similar ranges across regions, either through z-score normalization or relative to population size.
- **Smoothing**: Apply moving average or other smoothing techniques to reduce noise, especially for smaller regions with higher variance.
- **Lag Alignment**: Adjust for reporting delays by aligning data based on report dates rather than event dates when necessary.

5.2 Training Strategies

Training MSTGAT effectively requires careful attention to several aspects:

5.2.1 Loss Function

While mean squared error (MSE) is commonly used for time series forecasting, epidemiological forecasting may benefit from specialized loss functions:

$$\mathcal{L}_{combined} = \alpha \cdot \mathcal{L}_{MSE} + (1 - \alpha) \cdot \mathcal{L}_{MAPE} + \gamma \cdot \mathcal{L}_{attn} \tag{73}$$

where $\mathcal{L}_{MSE}$ is the mean squared error, $\mathcal{L}_{MAPE}$ is the mean absolute percentage error, $\mathcal{L}_{attn}$ is the attention regularization term, and α and γ are hyperparameters.

This combined loss addresses both absolute errors (important for larger regions) and relative errors (important for smaller regions).

5.2.2 Learning Rate Schedule

We recommend a learning rate schedule with initial warmup followed by cosine decay:

$$\eta(t) = \begin{cases} \eta_{base} \cdot \frac{t}{T_{warmup}} & \text{if } t < T_{warmup} \\ \eta_{base} \cdot \frac{1+\cos(\pi \cdot \frac{t - T_{warmup}}{T_{total} - T_{warmup}})}{2} & \text{otherwise} \end{cases} \tag{74}$$

where η_{base} is the base learning rate, T_{warmup} is the warmup period, and T_{total} is the total number of training steps.

5.2.3 Gradient Clipping

To stabilize training, especially in the early stages, we recommend gradient clipping:

$$\|\nabla\| = \min(\|\nabla\|, C_{clip}) \tag{75}$$

where C_{clip} is the gradient clipping threshold.

5.2.4 Training Regime

For epidemiological forecasting, we recommend a sliding window approach:

- Split the data into training, validation, and test sets chronologically
- Use a sliding window approach to generate multiple training samples from each time series
- Ensure no data leakage by maintaining strict temporal separation between training, validation, and test sets
- Consider using region-based cross-validation to assess generalization to unseen regions

5.3 Hyperparameter Selection

Key hyperparameters for MSTGAT include:

- **Architecture Parameters**: Hidden dimension D, bottleneck dimension D_{low}, number of attention heads H, number of temporal scales S, kernel size K
- **Regularization Parameters**: Dropout rate p_{drop}, attention regularization weight λ_{attn}, weight decay
- **Training Parameters**: Learning rate, batch size, number of epochs

We recommend Bayesian optimization or systematic grid search for hyperparameter tuning, focusing on validation performance for the specific forecasting horizon of interest.

5.4 Hardware Considerations

The memory requirements of MSTGAT can be substantial, especially for large graphs. To address this:

- For small to medium-sized problems (up to a few hundred regions), standard GPUs with 8-16GB memory should be sufficient
- For larger problems, consider:
  - Multi-GPU training with model parallelism
  - Gradient accumulation to reduce memory requirements
  - Mixed-precision training to improve memory efficiency
  - Graph partitioning for very large graphs

6 Model Variants and Ablations

To better understand the contribution of each component in MSTGAT, we propose several model variants and ablation studies:

6.1 Feature Extraction Variants

6.1.1 Standard Convolution vs. Depthwise Separable Convolution

To assess the impact of depthwise separable convolutions, we can replace them with standard convolutions:

$$\mathbf{Z} = \text{Conv1D}(\mathbf{X}', \text{kernel\_size} = K, \text{out\_channels} = C_{out}) \tag{76}$$

This variant would have higher computational complexity but potentially improved representational capacity.

6.1.2 Direct Projection vs. Low-Rank Projection

To evaluate the impact of the low-rank projection, we can replace it with a direct projection:

$$\mathbf{H} = \sigma(\text{LN}(\mathbf{Z}_{flat}\mathbf{W}_{direct} + \mathbf{b}_{direct})) \tag{77}$$

where $\mathbf{W}_{direct} \in \mathbb{R}^{(C_{out} \cdot T) \times D}$.

6.2 Spatial Modeling Variants

6.2.1 Predefined Graph vs. Learned Graph

To assess the value of learning the graph structure, we can replace the low-rank graph attention with a graph convolution based on a predefined adjacency matrix:

$$\mathbf{H}_{spatial} = \text{GCN}(\mathbf{H}_{reshaped}, \mathbf{A}_{predefined}) \tag{78}$$

where $\mathbf{A}_{predefined}$ could be based on geographical proximity, transportation networks, or other domain knowledge.

6.2.2 Full-Rank Graph Attention vs. Low-Rank Graph Attention

To evaluate the impact of the low-rank formulation in the graph attention, we can replace it with a full-rank attention mechanism:

$$\mathbf{A}_{bias,h} = \mathbf{W}_{full} \in \mathbb{R}^{N \times N} \tag{79}$$

where $\mathbf{W}_{full}$ is a learnable parameter matrix.

6.3 Temporal Modeling Variants

6.3.1 Single-Scale vs. Multi-Scale

To assess the impact of multi-scale temporal modeling, we can replace the multi-scale dilated convolutions with a single-scale convolution:

$$\mathbf{Z}_{single} = \text{Conv1D}(\mathbf{H}_{temp}, \text{kernel\_size} = K, \text{dilation} = 1, \text{padding} = (K-1)/2) \tag{80}$$

6.3.2 Fixed Fusion vs. Adaptive Fusion

To evaluate the impact of adaptive fusion, we can replace the learnable fusion weights with fixed weights:

$$\mathbf{Z}_{fused} = \frac{1}{S}\sum_{s=1}^{S} \mathbf{Z}_{s,drop} \tag{81}$$

6.4 Prediction Variants

6.4.1 Direct Prediction vs. Adaptive Refinement

To assess the impact of the adaptive refinement mechanism, we can use the model's predictions directly:

$$\hat{\mathbf{Y}} = \mathbf{P}_{initial} \tag{82}$$

6.4.2 Alternative Refinement Approaches

We can also explore alternative refinement mechanisms, such as linear extrapolation:

$$\mathbf{X}_{trend} = \mathbf{X}[:, -1, :] - \mathbf{X}[:, -2, :] \tag{83}$$

$$\mathbf{E}_{linear} = \mathbf{X}_{last\_expanded} + \mathbf{X}_{trend}.unsqueeze(-1) \cdot \mathbf{T} \tag{84}$$

$$\hat{\mathbf{Y}} = \mathbf{G} \odot \mathbf{P}_{initial} + (1 - \mathbf{G}) \odot \mathbf{E}_{linear} \tag{85}$$

7 Limitations and Future Directions

While MSTGAT addresses many challenges in spatiotemporal forecasting, several limitations and future research directions remain:

7.1 Current Limitations

7.1.1 Scalability to Very Large Graphs

The $\mathcal{O}(N^2)$ computational complexity of the attention mechanism limits scalability to very large graphs (thousands of nodes or more). This is a common limitation of attention-based models.

7.1.2 Handling Multiple Variables

The current formulation focuses on forecasting a single variable (e.g., case counts). Extending the model to handle multiple correlated variables (e.g., hospitalizations, deaths, testing rates) would increase its utility for comprehensive epidemiological forecasting.

7.1.3 Interpretability Challenges

While the attention weights provide some level of interpretability, the complex interactions between spatial and temporal components can still be difficult to interpret and explain to non-technical stakeholders.

7.1.4 Uncertainty Quantification

The current model provides point forecasts without explicit uncertainty estimates. In epidemiological forecasting, quantifying uncertainty is crucial for decision-making.

7.2 Future Research Directions

7.2.1 Hierarchical Graph Modeling

To address scalability issues, hierarchical graph models could be developed, aggregating regions into higher-level clusters and modeling relationships at multiple scales.

7.2.2 Multivariate Forecasting

Extending MSTGAT to handle multiple variables would involve modifying the feature extraction and prediction modules to account for cross-variable dependencies.

7.2.3 Probabilistic Forecasting

Incorporating probabilistic forecasting techniques, such as quantile regression or conformal prediction, would enable the model to provide prediction intervals and full predictive distributions.

7.2.4 Causal Modeling

Integrating causal inference techniques could help distinguish correlative from causal relationships, improving the model's robustness to interventions and policy changes.

7.2.5 Transfer Learning and Few-Shot Adaptation

Developing transfer learning approaches would allow the model to leverage historical data from previous epidemics or well-studied regions to improve forecasting in data-scarce settings.

7.2.6 Integration with Domain Knowledge

Incorporating epidemiological principles and mechanistic models into the architecture could enhance both performance and interpretability, creating a hybrid approach that combines the strengths of both data-driven and mechanistic modeling.

7.3 Broader Impact and Applications

Beyond epidemiological forecasting, MSTGAT has potential applications in:

- Traffic prediction and optimization
- Energy demand forecasting in smart grids
- Climate and weather prediction
- Financial market analysis
- Supply chain optimization

The flexible architecture and principled design choices make MSTGAT adaptable to a wide range of spatiotemporal forecasting problems, particularly those involving complex spatial dependencies and multi-scale temporal patterns.

8 Conclusion

We have presented MSTGAT, a novel deep learning architecture for spatiotemporal forecasting that jointly models spatial dependencies and temporal dynamics at multiple scales. Through its innovative components—efficient feature extraction, adaptive graph learning, multi-scale temporal modeling, and adaptive refinement—MSTGAT addresses key challenges in epidemiological forecasting and broader spatiotemporal prediction tasks.

The architecture incorporates several design principles to enhance both performance and computational efficiency, including low-rank decompositions, depthwise separable convolutions, and adaptive fusion mechanisms. Theoretical analysis demonstrates the model's expressivity and computational properties, while implementation considerations provide practical guidance for applying MSTGAT to real-world problems.

Future work will focus on addressing the identified limitations and expanding the model's capabilities, with the ultimate goal of providing accurate and interpretable forecasts to support decision-making in epidemiological response and other domains requiring spatiotemporal prediction.