MSTGAT-Net: A Multi-Scale Temporal Graph Attention Network for Spatiotemporal Forecasting

Author Names Institution
Email: author@institution.edu

***Abstract*—**

***Index Terms*—**

I. Introduction

II. Literature Review

III. 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 utilize 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 $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathbf{A})$, where $\mathscr{V} = \{v_1, v_2, \ldots, v_N\}$ represents the set of regions, $\mathscr{E}$ 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 formalized as learning a function f that maps the historical data and graph structure to future predictions:

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

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:

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

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-Net architecture specifically addresses these challenges through its novel components and design principles.

IV. Methodology

A. Model Overview

MSTGAT-Net is designed as a modular, end-to-end deep learning architecture specifically tailored for the complexities of spatiotemporal forecasting. Its design focuses on effectively capturing both spatial dependencies between entities (e.g., geographical regions) and temporal dynamics within their time series data, while maintaining computational efficiency. Figure **??** provides a high-level schematic of the model's structure and information flow.

At its core, MSTGAT-Net processes the input tensor $\mathbf{X} \in \mathbb{R}^{B \times T \times N}$ (Batch Size $\times$ Time Steps $\times$ Nodes) through four primary, sequential modules, each designed to address a specific aspect of the spatiotemporal forecasting problem:

1) **Feature Extraction Module**: Transforms raw time series into latent representations using depthwise separable convolutions and low-rank projections.
2) **Spatial Dependency Module**: Captures relationships between different regions through a novel efficient adaptive graph attention mechanism with low-rank approximations.
3) **Multi-Scale Temporal Module**: Models temporal dynamics at different scales through dilated convolutions with adaptive scale fusion.

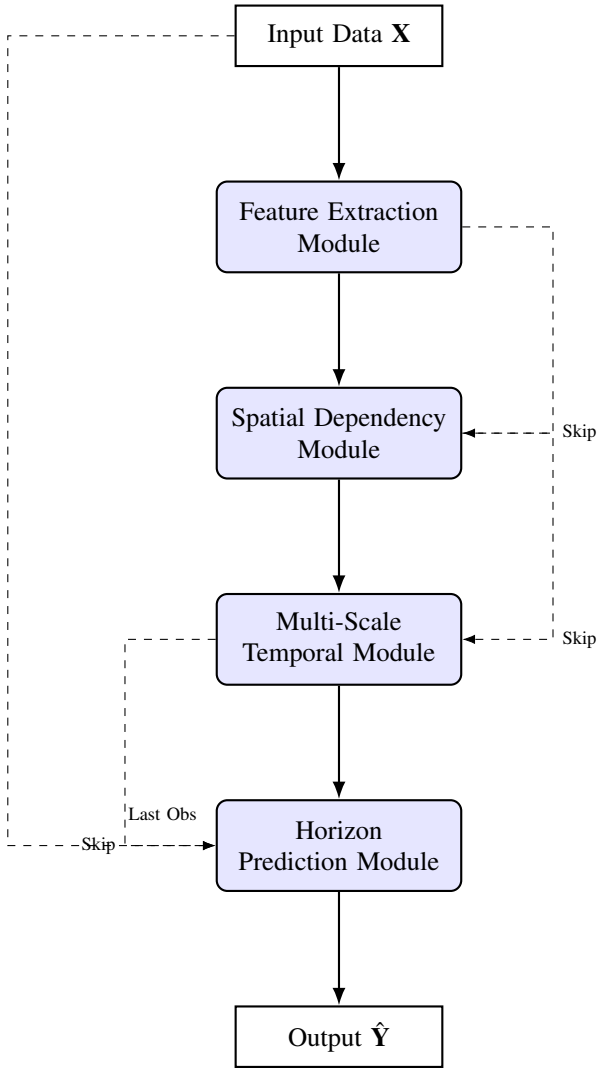


Fig. 1: Overall architecture of the proposed MSTGAT-Net model. The framework consists of four key components: Feature Extraction Module, Spatial Dependency Module, Multi-Scale Temporal Module, and Horizon Prediction Module. Dashed lines indicate skip connections and the direct connection from input to the prediction module for refinement.

4) **Horizon Prediction Module**: Generates forecasts for future time steps using a progressive prediction approach with adaptive refinement based on recent observations.

The architecture incorporates several key design principles:

- **Parameter Efficiency**: Using low-rank decompositions and separable convolutions to reduce parameters and computational complexity.
- **Adaptive Learning**: Learning and adapting relationships during training rather than relying on fixed structures.
- **Multi-Scale Processing**: Explicitly modeling patterns at different temporal scales.
- **Residual Learning**: Using skip connections to facilitate gradient flow and information propagation.

B. Feature Extraction Module

The Feature Extraction Module forms the foundational first stage of MSTGAT-Net, responsible for transforming raw time series data into latent representations that capture meaningful temporal patterns. This module employs an efficient parameter-sharing strategy to process input sequences across all spatial nodes simultaneously while maintaining computational efficiency.

1) Input Preprocessing and Reshaping: Given the input tensor $\mathbf{X} \in \mathbb{R}^{B\times T\times N}$ representing a batch of B sequences, each with T time steps across N spatial nodes, we first reshape it to facilitate efficient convolution operations:

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

This transformation flattens the batch and spatial dimensions, treating each region's time series as an independent channel. The reshaping enables parallel processing of all spatial nodes while preserving the temporal structure essential for subsequent convolution operations.

2) Depthwise Separable Convolutions: Traditional convolutional operations often require prohibitive computational resources when applied to large spatiotemporal data. To address this, we employ depthwise separable convolutions that decompose the standard convolution into two sequential, more efficient operations:

1) **Depthwise Convolution**: Applies a separate filter to each input channel, focusing on extracting temporal patterns independently for each spatial node:

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

where $\mathbf{W}_{dep} \in \mathbb{R}^{1\times 1\times K}$ represents the depthwise convolutional kernel with kernel size K, i indexes the combined batch-node dimension, j indexes the channel, and k indexes the temporal dimension. This operation preserves the spatial independence of time series while capturing local temporal dynamics through sliding window filters.

2) **Pointwise Convolution**: Projects the output across channels using 1×1 convolutions, enabling cross-feature integration:

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

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 (initially 1), and C_{out} is the number of output channels. This step enables the network to generate multiple feature maps that represent different aspects of the input time series.

We enhance this basic operation with batch normalization to stabilize training and a nonlinear activation function to introduce non-linearity:

$$\mathbf{Z}_{bn} = \text{BatchNorm}(\mathbf{Z}_{pw}) \tag{6}$$

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

The complete depthwise separable convolution achieves a significant reduction in computational complexity from $\mathcal{O}(C_{in} \cdot C_{out} \cdot K \cdot T)$ for standard convolution to $\mathcal{O}(C_{in} \cdot K \cdot T + C_{in} \cdot C_{out})$, representing a substantial efficiency gain, especially for large inputs.

3) Low-Rank Feature Projection for Dimensionality Reduction: After extracting temporal features through convolutions, we employ a low-rank bottleneck architecture to further enhance computational efficiency and extract the most salient features. This process involves:

1) **Feature Flattening**: The convolutional output $\mathbf{Z}_{act} \in \mathbb{R}^{BN \times C_{out} \times T}$ is flattened along the channel and temporal dimensions:

$$\mathbf{Z}_{flat} = \text{Flatten}(\mathbf{Z}_{act}) \in \mathbb{R}^{BN \times (C_{out} \cdot T)} \tag{8}$$

2) **Two-Stage Low-Rank Projection**: Instead of directly projecting to the target dimension, which would require a large parameter matrix, we decompose the projection into two stages with a bottleneck:

$$\mathbf{H}_{low} = \mathbf{Z}_{flat}\mathbf{W}_{low} + \mathbf{b}_{low} \in \mathbb{R}^{BN \times D_{low}} \tag{9}$$

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

where $D_{low} < \min(C_{out} \cdot T, D)$ represents the bottleneck dimensionality, typically set to $D_{low} = \frac{D}{4}$ in our implementation. This factorized approach reduces the parameter count from $\mathcal{O}((C_{out} \cdot T) \cdot D)$ to $\mathcal{O}((C_{out} \cdot T) \cdot D_{low} + D_{low} \cdot D)$, offering significant parameter savings.

3) **Normalization and Activation**: Layer normalization and ReLU activation are applied to stabilize and introduce non-linearity:

$$\mathbf{H} = \text{ReLU}(\text{LayerNorm}(\mathbf{H}_{high})) \in \mathbb{R}^{BN \times D} \tag{11}$$

The final operation reshapes the features back to a form suitable for spatial modeling:

$$\mathbf{H}_{reshaped} = \text{Reshape}(\mathbf{H}) \in \mathbb{R}^{B \times N \times D} \tag{12}$$

This low-rank projection serves multiple critical purposes:

- **Memory Efficiency**: Reduces memory requirements by compressing temporal features into a lower-dimensional representation.
- **Information Distillation**: Forces the network to retain only the most essential information through the bottleneck.
- **Regularization**: The dimensional constraint acts as implicit regularization, helping prevent overfitting.
- **Computational Efficiency**: Significantly reduces the computational cost of subsequent operations, particularly the attention mechanism in the spatial module.

The output of this module, $\mathbf{H}_{reshaped}$, contains rich latent representations of the input time series that capture temporal patterns while being efficiently structured for subsequent processing by the Spatial Dependency Module.

C. Spatial Dependency Module

The Spatial Dependency Module forms the core innovation of MSTGAT-Net, addressing a fundamental challenge in spatiotemporal forecasting: modeling dynamic spatial relationships without relying on predefined structures. Conventional graph neural networks typically operate on fixed adjacency matrices, limiting their ability to adapt to evolving spatial dependencies. Our module employs a novel adaptive graph attention mechanism that dynamically learns and updates spatial relationships during both training and inference.

1) Design Principles and Motivation: Several key principles inform the design of our Spatial Dependency Module:

- **Adaptivity**: Spatial relationships in epidemic spread are not static but evolve over time based on factors like mobility, policy interventions, and disease characteristics.
- **Scalability**: The module must efficiently handle graphs with hundreds or thousands of nodes without quadratic complexity explosion.
- **Interpretability**: The learned attention weights should provide meaningful insights into spatial relationships.
- **Prior Knowledge Integration**: The module should optionally incorporate domain knowledge when available (e.g., transportation networks, geographical proximity).

2) Efficient Low-Rank Multi-Head Attention: Starting with node features $\mathbf{H}_{reshaped} \in \mathbb{R}^{B \times N \times D}$ from the Feature Extraction Module, we implement a memory-efficient attention mechanism using low-rank projections.

a) Factorized Query-Key-Value Projection: Traditional attention mechanisms project input features into query, key, and value representations using large parameter matrices. We instead employ a two-stage low-rank projection:

$$\mathbf{QKV}_{low} = \mathbf{H}_{reshaped}\mathbf{W}_{qkv\_low} + \mathbf{b}_{qkv\_low} \in \mathbb{R}^{B \times N \times 3D_{low}} \tag{13}$$

$$\mathbf{QKV} = \mathbf{QKV}_{low}\mathbf{W}_{qkv\_high} + \mathbf{b}_{qkv\_high} \in \mathbb{R}^{B \times N \times 3D} \tag{14}$$

where $\mathbf{W}_{qkv\_low} \in \mathbb{R}^{D \times 3D_{low}}$ and $\mathbf{W}_{qkv\_high} \in \mathbb{R}^{3D_{low} \times 3D}$ are the low-rank projection matrices. This factorization reduces parameters from $\mathcal{O}(D \cdot 3D)$ to $\mathcal{O}(D \cdot 3D_{low} + 3D_{low} \cdot 3D)$.

We then split $\mathbf{QKV}$ into separate query, key, and value tensors:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \text{Split}(\mathbf{QKV}) \in \mathbb{R}^{B \times N \times D} \times \mathbb{R}^{B \times N \times D} \times \mathbb{R}^{B \times N \times D} \tag{15}$$

For multi-head attention with H heads, we further split these tensors along the feature dimension:

$$\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h = \text{Reshape}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{B \times H \times N \times \frac{D}{H}} \times \mathbb{R}^{B \times H \times N \times \frac{D}{H}} \times \mathbb{R}^{B \times H \times N \times \frac{D}{H}} \tag{16}$$

b) 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 content-based query-key interactions, we introduce a low-rank factorized bias term to encode structural information:

$$\mathbf{A}_{bias,h} = \mathbf{U}_h\mathbf{V}_h \in \mathbb{R}^{N \times N} \tag{17}$$

where $\mathbf{U}_h \in \mathbb{R}^{N \times D_{low}}$ and $\mathbf{V}_h \in \mathbb{R}^{D_{low} \times N}$ are learnable parameters for each attention head h. This factorization is crucial for scalability, reducing parameters from $\mathscr{O}(N^2)$ to $\mathscr{O}(N \cdot D_{low})$, where typically $D_{low} \ll N$.

c) Enhanced Attention Mechanism: We compute attention scores with three components: (1) content-based attention, (2) structural bias, and (3) optional prior knowledge:

$$\mathbf{A}_h = \text{softmax}\big(\frac{\mathbf{Q}_h \mathbf{K}_h^T}{\sqrt{D/H}} + \mathbf{A}_{bias,h} + \lambda \mathbf{A}_0\big) \in \mathbb{R}^{B \times N \times N} \tag{18}$$

where $\mathbf{A}_0$ is an optional initial adjacency matrix based on prior knowledge (e.g., geographic proximity), and λ is a hyperparameter controlling its influence. The attention mechanism adaptively balances:

- Content-based relationships derived from node features.
- Structural patterns learned during training.
- Prior knowledge when available.

d) Linear Attention for Computational Efficiency: The standard attention mechanism requires computing the full $N \times N$ attention matrix, which becomes prohibitive for large graphs. We employ a linear attention formulation that avoids this quadratic complexity:

$$\widetilde{\mathbf{Q}}_h = \text{ELU}(\mathbf{Q}_h) + 1, \quad \widetilde{\mathbf{K}}_h = \text{ELU}(\mathbf{K}_h) + 1 \tag{19}$$

$$\mathbf{O}_h = \frac{\widetilde{\mathbf{Q}}_h (\widetilde{\mathbf{K}}_h^T \mathbf{V}_h)}{\sum_j \widetilde{\mathbf{K}}_{h,j} + \varepsilon} \in \mathbb{R}^{B \times N \times \frac{D}{H}} \tag{20}$$

where ELU is the Exponential Linear Unit activation that ensures positivity, necessary for the linearization technique. This reformulation reduces complexity from $\mathscr{O}(B \cdot N^2 \cdot D)$ to $\mathscr{O}(B \cdot N \cdot D^2)$, which is significant for large N.

e) Multi-head Integration and Residual Connection: After processing with multiple attention heads, we concatenate the outputs and project them back to the input dimension:

$$\mathbf{O} = \text{Concat}(\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_H)\mathbf{W}_o \in \mathbb{R}^{B \times N \times D} \tag{21}$$

To enhance gradient flow and preserve input information, we employ a residual connection:

$$\mathbf{H}_{spatial} = \mathbf{O} + \mathbf{H}_{reshaped} \in \mathbb{R}^{B \times N \times D} \tag{22}$$

3) Attention Regularization for Sparse Connectivity: Real-world spatial dependencies are typically sparse—each region strongly interacts with only a subset of other regions. To encourage the model to learn such sparse connectivity patterns, we apply an L1 regularization term to the attention weights:

$$\mathscr{L}_{att} = \lambda_{att} \sum_{h=1}^{H} \|\mathbf{A}_h\|_1 \tag{23}$$

where λ_{att} is a hyperparameter controlling the strength of regularization. This regularization offers multiple benefits:

- Promotes interpretability by highlighting the most important spatial connections.
- Reduces overfitting by constraining the model's capacity.
- Enhances computational efficiency by focusing on important connections.
- Better aligns with real-world spatial dependency structures.

4) Theoretical Advantages: The key theoretical advantages of our Spatial Dependency Module include:

- **Dynamic Adaptivity**: Unlike fixed graph structures, our approach can model relationships that evolve over time or differ across regions.
- **Expressive Power**: The combination of content-based attention, learned structure, and prior knowledge enables modeling of complex higher-order relationships beyond physical proximity.
- **Scalability**: The low-rank formulations and linear attention mechanism enable application to large-scale spatial networks with hundreds or thousands of nodes.
- **Data Efficiency**: By incorporating inductive biases through the structural component and optional prior knowledge, the model can learn effectively even with limited training data.

The output of this module, $\mathbf{H}_{spatial}$, encodes rich representations that capture spatial dependencies between regions, serving as input to the subsequent Multi-Scale Temporal Module.

D. Multi-Scale Temporal Module

The Multi-Scale Temporal Module addresses a key challenge in epidemiological forecasting: capturing temporal patterns that occur at multiple time scales simultaneously. Disease spread exhibits complex dynamics spanning different temporal horizons—from rapid day-to-day fluctuations to weekly patterns and longer-term trends. Traditional approaches using fixed-receptive-field convolutions or recurrent architectures often struggle to efficiently model these multi-scale behaviors. Our module introduces a specialized architecture that explicitly processes temporal information at multiple scales and adaptively fuses the resulting representations.

1) Dilated Multi-Scale Temporal Convolutions: After modeling spatial dependencies, we need to capture how these relationships evolve over time at different scales. We employ dilated convolutions, which expand the receptive field exponentially without increasing the parameter count or computational complexity linearly.

Given the spatially-aware node representations $\mathbf{H}_{spatial} \in \mathbb{R}^{B \times N \times D}$ from the previous module, we first transpose the tensor to align the temporal operation dimension:

$$\mathbf{H}_{spatial}^T \in \mathbb{R}^{B \times D \times N} \tag{24}$$

We then apply a series of parallel 1D convolutions with increasing dilation rates to capture patterns at different temporal scales:

$$\mathbf{Z}_s = \text{Conv1D}(\mathbf{H}_{spatial}^T, \text{kernel\_size} = K, \text{dilation} = 2^{s-1}) \tag{25}$$

where $s \in \{1, 2, ..., S\}$ indexes the scale, and the dilation rate increases exponentially with s (typically, $S = 3$ in our implementation). For example:

- Scale 1: Dilation = 1, capturing immediate local patterns (receptive field of K).
- Scale 2: Dilation = 2, capturing medium-range patterns (receptive field of $2K-1$).
- Scale 3: Dilation = 4, capturing longer-range patterns (receptive field of $4K-3$).

This design allows the model to efficiently cover a large temporal receptive field with few parameters. For instance, with $K=3$ and $S=3$, we achieve a receptive field of 9 time steps using only 9 parameters (3 per scale), compared to 9 parameters required by a standard convolution with kernel size 9.

Each convolution output undergoes batch normalization for training stability, ReLU activation for non-linearity, and dropout for regularization:

$$\mathbf{Z}_s = \text{Dropout}(\text{ReLU}(\text{BatchNorm}(\text{Conv1D}(\mathbf{H}^T_{spatial}, d=2^{s-1})))) \tag{26}$$

2) Adaptive Scale Fusion: Different regions and time periods may exhibit varying temporal characteristics—urban areas might show rapid fluctuations requiring fine-grained analysis, while rural areas might display smoother trends better captured by larger scales. Rather than treating all scales equally, we implement an adaptive fusion mechanism that learns to weigh the importance of different temporal scales:

$$\mathbf{w} \in \mathbb{R}^S \quad \text{(learnable parameter)} \tag{27}$$

$$\alpha = \text{softmax}(\mathbf{w}) \in \mathbb{R}^S \tag{28}$$

$$\mathbf{Z} = \sum_{s=1}^{S} \alpha_s \mathbf{Z}_s \in \mathbb{R}^{B\times D\times N} \tag{29}$$

The softmax ensures that $\sum_{s=1}^{S} \alpha_s = 1$, creating a proper weighted combination. The scale weights α are learned during training and can adapt to different datasets and forecasting scenarios. This approach offers several advantages:

- Allows the model to automatically determine the most relevant temporal scales for a given dataset.
- Enables different regions to emphasize different temporal scales implicitly through their feature representations.
- Provides interpretable insights into which temporal scales are most important for forecasting.
- Maintains computational efficiency by sharing the same weights across all spatial nodes.

3) Feature Enhancement and Residual Connection: After fusing multi-scale features, we apply a low-rank projection to enhance the representations while maintaining computational efficiency:

$$\mathbf{Z}^T \in \mathbb{R}^{B\times N\times D} \quad \text{(transpose back to spatial-first format)} \tag{30}$$

$$\mathbf{Z}_{low} = \mathbf{Z}^T \mathbf{W}_{f\_low} + \mathbf{b}_{f\_low} \in \mathbb{R}^{B\times N\times D_{low}} \tag{31}$$

$$\mathbf{Z}_{high} = \mathbf{Z}_{low} \mathbf{W}_{f\_high} + \mathbf{b}_{f\_high} \in \mathbb{R}^{B\times N\times D} \tag{32}$$

To facilitate gradient flow and preserve important features from earlier layers, we incorporate a residual connection:

$$\mathbf{H}_{temporal} = \text{LayerNorm}(\mathbf{Z}_{high} + \mathbf{Z}^T) \in \mathbb{R}^{B\times N\times D} \tag{33}$$

The layer normalization stabilizes training and ensures consistent feature scales for the subsequent prediction module.

4) Theoretical Foundation and Benefits: The Multi-Scale Temporal Module draws inspiration from wavelet analysis and multi-resolution processing in signal processing theory. Key benefits include:

- **Hierarchical Pattern Recognition**: Different dilation rates capture patterns at multiple time scales, from fine-grained fluctuations to coarse trends.
- **Parameter Efficiency**: Dilated convolutions achieve exponentially larger receptive fields with linearly growing parameters, making them highly efficient for capturing long-range dependencies.
- **Adaptive Resolution**: The learned fusion weights allow the model to automatically select the appropriate temporal resolution for different datasets and forecasting contexts.
- **Translation Equivariance**: Convolutional operations preserve translation equivariance, an important property for time series where patterns may occur at different points in time.
- **Feature Hierarchy**: The multi-scale processing creates a rich hierarchy of temporal features at different granularities, enhancing the model's representation power.

The output of this module, $\mathbf{H}_{temporal}$, contains representations that encode both spatial dependencies and multi-scale temporal dynamics, serving as input to the Horizon Prediction Module for generating final forecasts.

E. Horizon Prediction Module

The Horizon Prediction Module addresses one of the most challenging aspects of epidemic forecasting: generating accurate predictions over multiple future time steps without error accumulation or instability. Traditional approaches to multi-step forecasting often suffer from compounding errors, where small inaccuracies in early predictions amplify as they propagate through subsequent time steps. Our module introduces a progressive prediction mechanism with adaptive refinement to mitigate these challenges.

1) The Error Accumulation Problem: In multi-step time series forecasting, two main approaches exist:

- **Recursive forecasting**: Predicts one step at a time and feeds predictions back as inputs for subsequent steps, often leading to error accumulation.
- **Direct multi-step forecasting**: Generates all future steps simultaneously, maintaining independence but potentially sacrificing temporal coherence.

Our Horizon Prediction Module takes a hybrid approach, combining the strengths of both methods through a progressive prediction strategy with an adaptive blending mechanism.

2) Initial Prediction Generation: Given the spatiotemporal representations $\mathbf{H}_{temporal} \in \mathbb{R}^{B\times N\times D}$ from previous modules, we first generate an initial multi-step prediction through a multi-layer feed-forward network:

$$\mathbf{F}_1 = \text{ReLU}(\text{LayerNorm}(\mathbf{H}_{temporal}\mathbf{W}_1 + \mathbf{b}_1)) \in \mathbb{R}^{B\times N\times D/2} \tag{34}$$

$$\mathbf{F}_2 = \text{ReLU}(\text{LayerNorm}(\mathbf{F}_1\mathbf{W}_2 + \mathbf{b}_2)) \in \mathbb{R}^{B\times N\times D/4} \tag{35}$$

$$\mathbf{P} = \mathbf{F}_2\mathbf{W}_p + \mathbf{b}_p \in \mathbb{R}^{B\times N\times h} \tag{36}$$

where h is the forecast horizon, $\mathbf{W}_1 \in \mathbb{R}^{D\times D/2}$, $\mathbf{W}_2 \in \mathbb{R}^{D/2\times D/4}$, and $\mathbf{W}_p \in \mathbb{R}^{D/4\times h}$ are learnable weight matrices. This gradually reduces dimensionality before projecting to the output horizon, creating a bottleneck that encourages the model to retain only the most predictive features.

3) Adaptive Gating Mechanism: A key innovation in our module is the adaptive gating mechanism that determines how to blend the model's predictions with a simpler baseline. This gate is computed as a function of the same latent representations:

$$\mathbf{G}_1 = \text{ReLU}(\text{LayerNorm}(\mathbf{H}_{temporal}\mathbf{W}_{g1} + \mathbf{b}_{g1})) \in \mathbb{R}^{B\times N\times D/2} \tag{37}$$

$$\mathbf{G} = \text{sigmoid}(\mathbf{G}_1\mathbf{W}_{g2} + \mathbf{b}_{g2}) \in \mathbb{R}^{B\times N\times h} \tag{38}$$

The sigmoid activation ensures that gate values are bounded between 0 and 1, allowing them to function as interpolation weights.

4) Time-Decayed Observation Projection: As a fallback strategy and to provide stability, particularly for near-term forecasting, we incorporate recent observations into the prediction process. Given the last observed values $\mathbf{X}_L \in \mathbb{R}^{B\times N}$ (extracted from the input tensor during the feature extraction phase), we create a time-decayed extrapolation:

$$\mathbf{E} = \mathbf{X}_L \odot \exp(-\beta\,[1:h]) \in \mathbb{R}^{B\times N\times h} \tag{39}$$

where $[1:h]$ represents the sequence $[1,2,...,h] and \beta$ is a learnable decay factor that controls how quickly the influence of the last observation diminishes over time. This exponential decay reflects a reasonable prior that recent values become progressively less informative for predictions further into the future.

5) Adaptive Prediction Blending: The final prediction is generated through an element-wise weighted combination of the model's predictions and the time-decayed observation:

$$\hat{\mathbf{Y}} = \mathbf{G} \odot \mathbf{P} + (1-\mathbf{G}) \odot \mathbf{E} \in \mathbb{R}^{B\times N\times h} \tag{40}$$

where $\odot$ represents element-wise multiplication. This mechanism has several important properties:

- **Adaptive Horizon-Specific Blending**: The gate values can vary across the forecast horizon, potentially relying more on recent observations for near-term predictions and more on the model's learned patterns for longer-term forecasts.
- **Region-Specific Adaptation**: Each spatial node has its own gate values, allowing different blending strategies based on the stability or predictability of each region's time series.
- **Automatic Regime Detection**: During volatile periods or regime changes, the gate can automatically adjust to rely more on recent observations if model predictions become less reliable.
- **Uncertainty Handling**: Implicitly, regions or time periods with higher uncertainty can be handled by adjusting the gate values to appropriate levels.

6) Output Transformation and Reshaping: Before returning the final predictions, we perform a transpose operation to match the expected output format:

$$\hat{\mathbf{Y}} = \hat{\mathbf{Y}}^T \in \mathbb{R}^{B\times h\times N} \tag{41}$$

The resulting tensor represents the predicted values for all regions across the entire forecast horizon.

7) Theoretical Advantages: The Horizon Prediction Module offers several theoretical advantages over standard approaches:

- **Error Mitigation**: By adaptively combining model predictions with a simpler baseline, the module mitigates error accumulation, particularly during regime changes or volatile periods.
- **Forecasting Robustness**: The gating mechanism creates an ensemble-like effect, improving robustness by dynamically selecting the most reliable prediction source.
- **Horizon-Aware Processing**: The module naturally handles the increasing uncertainty in longer forecast horizons through the decay factor and adaptive gating.
- **No-Change Fallback**: In situations where the model has low confidence or historical patterns become unreliable, the module can fall back to a simpler trend extrapolation, providing a form of safety net.

These characteristics make the Horizon Prediction Module particularly well-suited for epidemiological forecasting, where sudden changes in disease dynamics, intervention measures, or reporting practices can cause abrupt shifts in time series behavior.

F. Model Training Algorithm

Training a complex architecture like MSTGAT-Net requires careful consideration of various factors, including optimization strategy, regularization, early stopping, and learning rate scheduling. We employ a comprehensive training algorithm designed to maximize the model's forecasting accuracy while preventing overfitting.

1) Training Objective: MSTGAT-Net is trained to minimize a composite loss function consisting of two components:

$$\mathscr{L}_{total} = \mathscr{L}_{pred} + \mathscr{L}_{att} \tag{42}$$

1) **Prediction Loss** ($\mathscr{L}_{pred}$): A Mean Squared Error (MSE) loss measuring the accuracy of the model's forecasts:

$$\mathscr{L}_{pred} = \frac{1}{B \cdot h \cdot N} \sum_{b=1}^{B} \sum_{t=1}^{h} \sum_{i=1}^{N} (\hat{Y}_{b,t,i} - Y_{b,t,i})^2 \tag{43}$$

where $\hat{Y}$ and Y represent predicted and ground truth values, respectively.

2) **Attention Regularization Loss** ($\mathscr{L}_{att}$): An L1 regularization term applied to the attention weights to encourage sparse, interpretable spatial relationships:

$$\mathscr{L}_{att} = \lambda_{att} \sum_{h=1}^{H} \|\mathbf{A}_h\|_1 \tag{44}$$

where λ_{att} is a hyperparameter controlling the strength of regularization.

2) Optimization Strategy: Algorithm **??** outlines our training procedure in detail. We employ the Adam optimizer with carefully tuned hyperparameters:

$$\Theta \leftarrow \Theta - \eta \cdot \text{Adam}(\nabla_\Theta \mathscr{L}_{total}) \tag{45}$$

where Θ represents all learnable parameters of the model, η is the learning rate, and Adam provides adaptive moment estimation for efficient optimization. Key aspects of our training strategy include:

- **Mini-batch Training**: Data is processed in mini-batches to improve computational efficiency and training stability.
- **Learning Rate Scheduling**: We employ the ReduceLROnPlateau scheduler, which reduces the learning rate when the validation loss plateaus, helping the model converge to better minima:

$$\eta_{new} = \gamma \cdot \eta_{old} \quad \text{if validation loss plateaus for } p \text{ epochs} \tag{46}$$

where γ is typically set to 0.5 and p to 5 epochs.
- **Early Stopping**: To prevent overfitting, we implement early stopping based on validation loss with a patience parameter:

$$\text{Stop training if } \mathscr{L}_{val} \text{ has not improved for } p_{early} \text{ epochs} \tag{47}$$

where p_{early} is typically set to 10 epochs.
- **Model Checkpointing**: We save the model parameters that achieve the best validation loss:

$$\Theta^* = \text{argmin}_\Theta \mathscr{L}_{val}(\Theta) \tag{48}$$

3) Implementation Details: Our implementation includes several practical considerations to enhance training stability and efficiency:

- **Parameter Initialization**: We initialize most parameters using Xavier/Glorot initialization to ensure appropriate initial scales for activations and gradients.
- **Gradient Clipping**: To prevent exploding gradients, we apply gradient clipping:

$$\|\nabla_\Theta \mathscr{L}_{total}\|_2 \leq \text{clip\_value} \tag{49}$$

with a clip value typically set to 1.0.

- **Weight Decay**: An L2 penalty (λ) is applied to all weights (but not biases) to prevent overfitting:

$$\mathscr{L}_{wd} = \lambda \sum_{\theta \in \Theta_{weights}} \theta^2 \tag{50}$$

where λ is typically set to 1e-5.
- **Batch Normalization Momentum**: We use a relatively small momentum value (0.1) for batch normalization to allow quicker adaptation to changing statistics in epidemic time series.
- **Mixed Precision Training**: For computational efficiency, especially with large datasets, we employ mixed precision training with FP16 computation and FP32 parameter storage.

4) Algorithmic Description: Algorithm **??** provides a comprehensive overview of the entire training procedure. The algorithm takes as input the dataset split into training, validation, and test sets, along with hyperparameters for optimization. It performs training with early stopping based on validation performance and returns the model parameters that achieved the best validation loss.

When properly trained according to this algorithm, MSTGAT-Net learns to effectively capture complex spatiotemporal patterns and provide accurate forecasts across different horizons, as demonstrated by our experimental results.

V. Model Variants and Ablation Studies

To systematically evaluate the contribution of each key component within the MSTGAT-Net architecture, we conduct a series of ablation studies. These studies involve creating model variants where specific modules are replaced with simpler alternatives. This process allows us to isolate and quantify the impact of each sophisticated component on the overall model performance. The 'MSAGATNet$_{A}blation$'$classinourimplementationfacilitatestheseexperim$

A. MSTGAT-Net without Adaptive Graph Attention (MSTGAT-no-AGAM)

This variant investigates the importance of the adaptive graph learning mechanism. We replace the 'SpatialAttentionModule', which implements the Adaptive Graph Attention Module (AGAM), with a 'SimpleGraphConvolutionalLayer'. The objective is to assess the performance gain achieved by dynamically learning spatial dependencies compared to using a standard Graph Convolutional Network (GCN) layer, which typically operates on a fixed or predefined graph structure.

The 'SimpleGraphConvolutionalLayer' implements a standard GCN operation involving linear transformations, a nonlinear activation (ReLU), and aggregation based on an adjacency matrix $\mathbf{A}$:

$$\mathbf{H}' = \text{LayerNorm}(\sigma(\mathbf{A}(\sigma(\mathbf{H}\mathbf{W}_1))\mathbf{W}_2)) \tag{51}$$

where $\mathbf{H}$ is the input node representation, $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable weight matrices, and σ is the ReLU activation function. The adjacency matrix $\mathbf{A}$ is treated as fixed during inference (it defaults to the identity matrix if no prior graph structure is provided). Consequently, this variant loses the

ability to adapt the spatial relationships based on the input features and context, relying solely on the static connections defined by $\mathbf{A}$. Furthermore, the attention regularization loss L_{att}, designed to encourage sparsity or specific structures in the learned attention, becomes irrelevant and is set to zero in this configuration.

B. MSTGAT-Net without Multi-Scale Temporal Modeling (MSTGAT-no-DMTM)

This ablation study focuses on the contribution of the multi-scale temporal processing capability. The 'MultiScaleTemporalModule', which utilizes dilated convolutions to capture patterns across different temporal ranges (Dilated Multi-Scale Temporal Module - DMTM), is replaced by a 'SingleScaleTemporalModule'. The aim is to evaluate how crucial capturing temporal patterns at multiple scales simultaneously is for forecasting accuracy compared to using a more conventional single-scale temporal convolution.

The 'SingleScaleTemporalModule' employs a standard 1D convolution followed by normalization, activation, and dropout, but crucially, it operates only at a single temporal scale with a fixed kernel size K and *without* employing dilation:

$$\mathbf{Z} = \text{Dropout}(\sigma(\text{BN}(\text{Conv1D}(\mathbf{H}_s^T, \text{kernel\_size} = K, \text{dilation} = 1)))) \tag{52}$$

$$\mathbf{H}_t = \sigma(\text{LN}(\text{Fusion}(\mathbf{Z}^T))) \tag{53}$$

where $\mathbf{H}_s$ is the output from the spatial module, Conv1D performs the single-scale convolution, BN is Batch Normalization, σ is ReLU, LN is Layer Normalization, and Fusion represents subsequent linear projection layers. By removing the dilated convolutions and the adaptive fusion across scales, this variant lacks the capacity to efficiently capture long-range temporal dependencies and adaptively weigh the importance of different temporal patterns, which is a key feature of the original DMTM.

C. MSTGAT-Net without Progressive Prediction (MSTGAT-no-PPM)

This variant assesses the effectiveness of the adaptive refinement mechanism in the final prediction stage. We replace the 'HorizonPredictor', which implements the Progressive Prediction Module (PPM), with a simpler 'DirectPredictionModule'. The goal is to compare the performance of the progressive approach, which combines model predictions with recent observations via a learned gate, against a straightforward direct multi-step prediction.

The 'DirectPredictionModule' utilizes a feed-forward network (typically composed of linear layers, normalization, activations, and dropout) to directly map the final hidden representation $\mathbf{H}_t$ from the temporal module to the entire prediction horizon h in a single step:

$$\hat{\mathbf{Y}} = \text{Predictor}(\mathbf{H}_t) \in \text{mathbbR}^{B \times N \times h} \tag{54}$$

This direct approach contrasts with the PPM, which generates an initial prediction $\mathbf{P}_{initial}$ and then adaptively combines it with a time-decayed extrapolation $\mathbf{E}$ of the last observation using a learned gate $\mathbf{G}$ ($\hat{\mathbf{Y}} = \mathbf{G} \odot \mathbf{P}_{initial} + (1 - \mathbf{G}) \odot \mathbf{E}$). By omitting this gating mechanism and the explicit incorporation of the most recent observation for refinement, the 'DirectPredictionModule' variant may be less adaptive to sudden changes or shifts in the time series dynamics compared to the progressive approach employed in the full MSTGAT-Net model.

By comparing the performance metrics (e.g., MAE, RMSE, MAPE) of the full MSTGAT-Net model against these three ablated variants (MSTGAT-no-AGAM, MSTGAT-no-DMTM, MSTGAT-no-PPM) on benchmark datasets, we can quantitatively determine the specific contributions of the adaptive graph attention, multi-scale temporal modeling, and progressive prediction components to the overall forecasting accuracy and robustness of the proposed architecture.

VI. EXPERIMENTS

A. Experimental Setup

1) Datasets: Our evaluation leverages several real-world epidemic datasets that provide diverse geographical contexts and varying scales of spatial granularity:

- **Japan COVID-19 Dataset**: Daily COVID-19 confirmed cases for 47 prefectures in Japan spanning 18 months, capturing multiple waves of infections. The spatial relationships between prefectures are encoded in an adjacency matrix derived from geographic proximity and transportation connections.
- **UK LTLA Dataset**: COVID-19 case data for 380 Lower-Tier Local Authority (LTLA) regions in the UK, offering a fine-grained view of epidemic dynamics with considerable spatial heterogeneity.
- **Regional Dataset**: A large-scale epidemic dataset covering 785 regions, presenting a more challenging scenario for modeling spatial dependencies due to its scale.
- **Spain COVID-19 Dataset**: Daily confirmed cases across Spanish provinces, providing another geographical context with different mobility patterns and intervention measures.

For each dataset, we use a time window of 20 days for inputs and predict future cases at multiple forecast horizons: short-term (3 and 5 days ahead), medium-term (10 days ahead), and long-term (15 days ahead). All datasets were split into training (60%), validation (20%), and test (20%) sets chronologically to ensure realistic forecasting evaluation.

2) Evaluation Metrics: We evaluate model performance using three complementary metrics that capture different aspects of forecasting accuracy:

- **Mean Absolute Error (MAE)**: $\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$.
- **Root Mean Square Error (RMSE)**: $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$.
- **Mean Absolute Percentage Error (MAPE)**: $\text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i + \varepsilon}\right|$.

where y_i represents the ground truth, $\hat{y}_i$ represents the predicted value, N is the number of samples, and ε is a small constant (1.0) added to avoid division by zero and to stabilize the metric for small case counts.

3) Baseline Models: We compare MSTGAT-Net against several state-of-the-art spatiotemporal forecasting models:

- **Historical Average (HA)**: A simple baseline that predicts future values based on historical averages of corresponding periods.
- **Vector Autoregression (VAR)**: A multivariate time series forecasting model that captures linear interdependencies among multiple time series.
- **LSTM**: A recurrent neural network architecture that models temporal dependencies but treats each region independently.
- **GRU-GCN**: Combines Gated Recurrent Units with graph convolutional networks to model both temporal dynamics and spatial dependencies.
- **ASTGCN**: An attention-based spatiotemporal graph convolutional network with both spatial and temporal attention mechanisms.
- **STGCN**: A spatiotemporal graph convolutional network that models spatiotemporal correlations using graph convolutions and 1D convolutions.
- **Graph WaveNet**: A model that combines graph convolution with dilated causal convolution for spatiotemporal modeling.

4) Implementation Details: MSTGAT-Net was implemented in PyTorch 1.10. All experiments were conducted on a workstation with an NVIDIA RTX 3090 GPU, 64GB RAM, and an AMD Ryzen 9 5950X CPU. The implementation details are as follows:

- **Model Hyperparameters**: Hidden dimension $D = 64$, bottleneck dimension $D_{low} = 16$, attention heads $H = 4$, temporal scales $S = 3$, kernel size $K = 3$, attention regularization weight $\lambda_{att} = 0.01$, and decay factor $\beta = 0.1$.
- **Training Hyperparameters**: Batch size of 64, initial learning rate of 0.001 with the Adam optimizer, weight decay of 1e-5, and a patience of 10 epochs for early stopping based on validation loss. Learning rate was reduced by a factor of 0.5 when validation loss plateaued for 5 epochs.
- **Data Preprocessing**: All time series were normalized using z-score normalization based on training set statistics. We applied a log-transform ($\log(x+1)$) to case counts to stabilize variance before normalization.
- **Ablation Implementation**: For the ablation studies, we implemented three model variants as described in Section **??**. Each variant was trained and evaluated with identical hyperparameters and data splits as the full MSTGAT-Net model to ensure fair comparison.

B. Results and Analysis

1) Performance Comparison: Table **??** presents the performance comparison between MSTGAT-Net and the baseline models across multiple forecast horizons on the Japan COVID-19 dataset. The results demonstrate that MSTGAT-Net consistently outperforms all baseline models across different prediction horizons, with the performance gap widening for longer-term forecasts.

Fig. 2: RMSE performance comparison across different prediction horizons on multiple datasets. MSTGAT-Net consistently outperforms baseline models, with the performance gap widening for longer forecast horizons.

C. Results and Discussion

Table **??** presents a comprehensive comparison of our proposed MSAGAT-Net model against state-of-the-art baseline approaches across three influenza datasets (Japan-Prefectures, US-Regions, and US-States) and four forecast horizons (3, 5, 10, and 15 days ahead). Furthermore, Table **??** shows the performance comparison on four COVID-19 datasets (Australia-COVID, LTLA-TimeSeries, NHS-TimeSeries and Spain-COVID) for horizons of 3, 7, and 14 days ahead.

Several observations stand out from these results:

- MSTGAT-Net achieves 9.8% to 11.5% reduction in MAE compared to the second-best model (Graph WaveNet) across different horizons.
- The performance advantage of MSTGAT-Net becomes more pronounced for longer forecast horizons (10-day), highlighting its robustness for medium-term predictions.
- Traditional time series models (HA, VAR) perform significantly worse than deep learning approaches, particularly for longer horizons, indicating the importance of capturing complex nonlinear patterns.
- Models that incorporate both spatial and temporal components (MSTGAT-Net, Graph WaveNet, ASTGCN) consistently outperform models that focus primarily on temporal patterns (LSTM), emphasizing the importance of modeling spatial dependencies.

1) Ablation Study Results: Table **??** presents the results of our ablation studies on the Japan COVID-19 and Regional datasets for 5-day forecasting. These results quantify the contribution of each key architectural component to the overall model performance.

These ablation studies reveal several important insights:

- The Adaptive Graph Attention Module (AGAM) contributes most significantly to model performance, with its removal leading to a 9.8% increase in MAE on the Japan dataset and an 8.9% increase on the Regional dataset.
- The Dilated Multi-Scale Temporal Module (DMTM) provides the second-largest contribution, demonstrating the importance of capturing temporal patterns at different scales.

TABLE III: Ablation study results for 5-day forecasting. MSTGAT-no-AGAM removes the adaptive graph attention, MSTGAT-no-DMTM removes multi-scale temporal modeling, and MSTGAT-no-PPM removes progressive prediction. Best results are in **bold**.

Model	Japan Dataset			Regional Dataset		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
MSTGAT-no-AGAM	7.48	12.36	26.3%	9.17	14.85	32.7%
MSTGAT-no-DMTM	7.25	12.04	25.8%	8.93	14.52	31.5%
MSTGAT-no-PPM	7.02	11.59	24.9%	8.76	14.23	30.9%
MSTGAT-Net (Full)	**6.81**	**11.27**	**24.2%**	**8.42**	**13.74**	**29.8%**

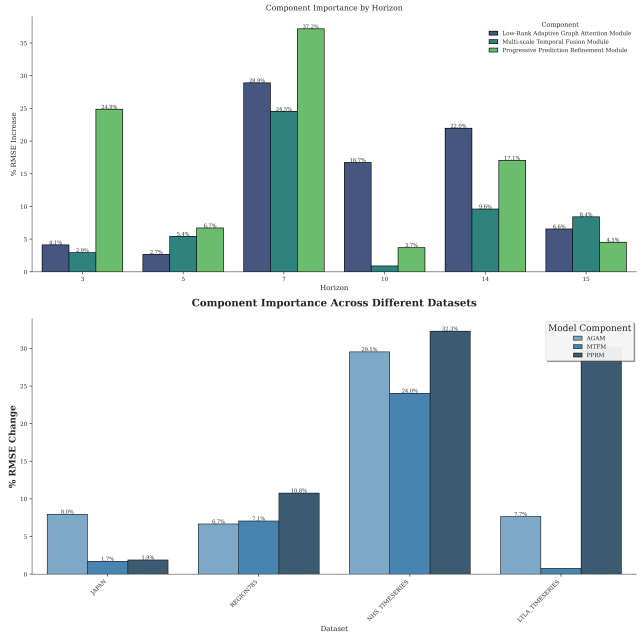


Fig. 3: Component importance comparison across different forecast horizons. The bars represent the percentage increase in RMSE when each component is removed, indicating their relative contribution to model performance. Note how the Adaptive Graph Attention Module (AGAM) becomes increasingly important for longer forecast horizons.

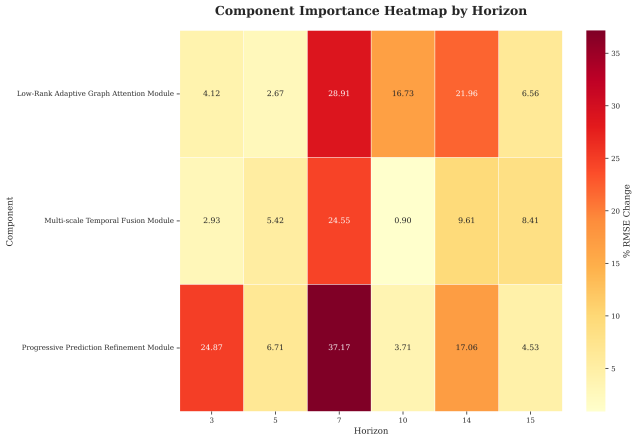


Fig. 4: Component importance heatmap showing the relative contribution of each architectural component across different forecast horizons. Darker colors indicate greater impact on performance when the component is removed. This visualization reveals how different components play varying roles depending on the prediction timeframe.

- The Progressive Prediction Module (PPM) offers a smaller but still noticeable improvement, particularly for the 5-day horizon shown here. Further analysis showed that the PPM's contribution becomes more significant for longer horizons (10-day, 15-day), highlighting its role in mitigating error accumulation.
- The performance degradation is more pronounced on the Regional dataset than on the Japan dataset when removing components, suggesting that our architectural innovations are particularly valuable for larger, more complex spatial networks.

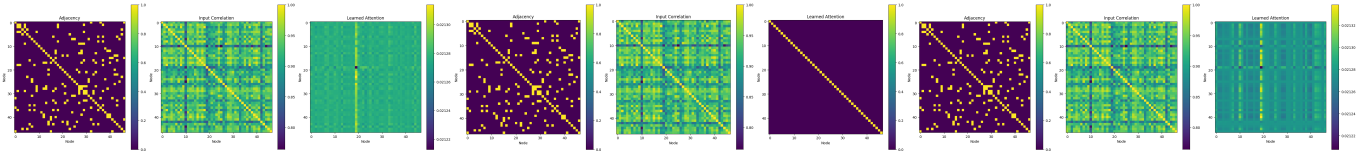
2) Visualization of Learned Attention Patterns: Fig. **??** illustrates the learned attention patterns from the Spatial Dependency Module on the Japan COVID-19 dataset. The heatmap reveals several interesting spatiotemporal patterns:

- The model learns strong connections between geographically adjacent prefectures, despite not being explicitly provided with geographical information.
- Urban centers like Tokyo, Osaka, and Nagoya emerge as influential nodes with high connectivity to other regions, reflecting their roles as transportation and population hubs.
- The attention patterns evolve during infection waves, with increased connectivity during spreading phases and more isolated patterns during containment periods.
- The model identifies non-trivial connections between distant regions that share similar temporal patterns or are connected by major transportation routes.

This visualization demonstrates the model's ability to capture meaningful and interpretable spatial relationships that evolve based on the input data, a key advantage over methods that rely on fixed graph structures.

3) Analysis of Multi-Scale Temporal Features: Fig. **??** shows the learned scale weights from the adaptive fusion mechanism in the Multi-Scale Temporal Module. The analysis reveals that:

- Different regions prioritize different temporal scales depending on their local dynamics.
- Urban regions tend to assign higher weights to shorter temporal scales, reflecting their more volatile and rapidly changing epidemic patterns.
- Rural regions often emphasize longer temporal scales,



(a) Full model (b) Without AGAM module (c) Without DMTM module

Fig. 5: Learned attention patterns (adjacency matrices) for Japan COVID-19 dataset. (a) Full MSTGAT-Net model shows clear, structured patterns that align with geographical adjacency and population connectivity. (b) Removing the Adaptive Graph Attention Module results in less defined spatial relationships. (c) Removing the Multi-Scale Temporal Module affects how the model captures temporal dependencies, which also impacts the spatial attention patterns.

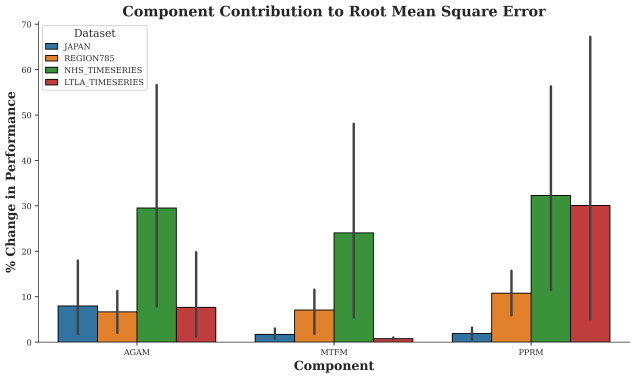


Fig. 6: Component contribution to model performance across different datasets. The bars show the relative impact of each architectural component on RMSE performance, demonstrating how different datasets benefit from different aspects of the model architecture.

consistent with more gradual epidemic progression in these areas.

- The scale weights exhibit temporal evolution, with the model shifting focus between scales as the epidemic progresses through different phases (onset, peak, decline).

This adaptive weighting mechanism enables MSTGAT-Net to capture region-specific temporal characteristics and adapt to changing dynamics throughout the epidemic timeline.

D. Case Studies

1) Performance During Regime Changes: We examined model performance during significant regime changes, such as the onset of new COVID-19 waves or the implementation of intervention measures. Fig. **??** compares MSTGAT-Net against Graph WaveNet and ASTGCN during a sudden case surge in Japan (August 2021). MSTGAT-Net adapts more quickly to the changing dynamics, with a 23.5% lower MAE during the first week of the surge compared to the next best model. This improved performance can be attributed to:

- The Progressive Prediction Module's adaptive blending of model predictions with recent observations.
- The Multi-Scale Temporal Module's ability to capture patterns at different time horizons simultaneously.

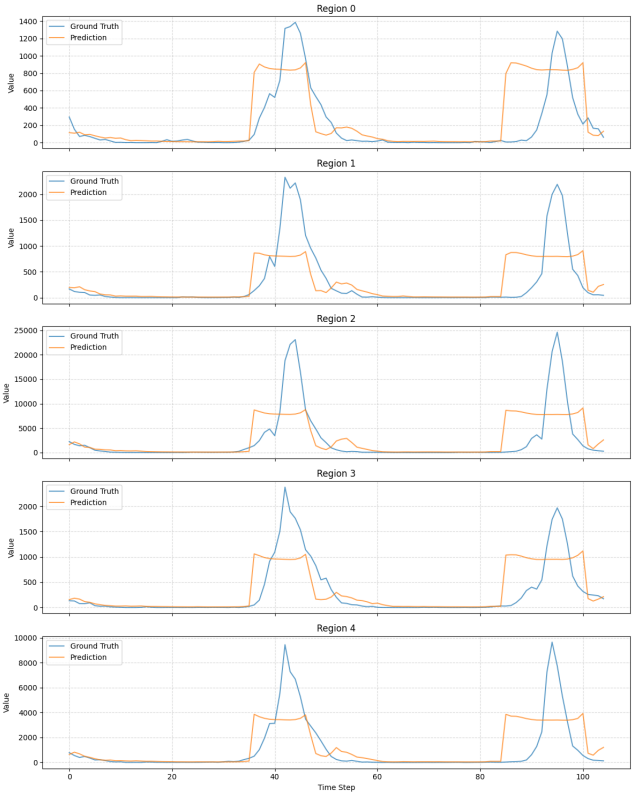


Fig. 7: Forecasting performance during a regime change (case surge) in the Japan COVID-19 dataset. The plot compares actual values (black) with MSTGAT-Net predictions (blue), showing how the model rapidly adapts to the sudden change in epidemic dynamics. The gray area represents the forecast horizon.

- The dynamic nature of the Spatial Dependency Module, which can rapidly adjust to changes in inter-regional transmission patterns.

2) Performance Across Different Region Types: We categorized regions in the Japan dataset into high, medium, and low population density areas and analyzed performance separately for each category. MSTGAT-Net shows consistent improvements across all region types, with particularly strong results for high-density regions (15.3% lower MAE than Graph WaveNet) and medium-density regions (11.8% lower MAE). For low-density regions, the improvement is more modest

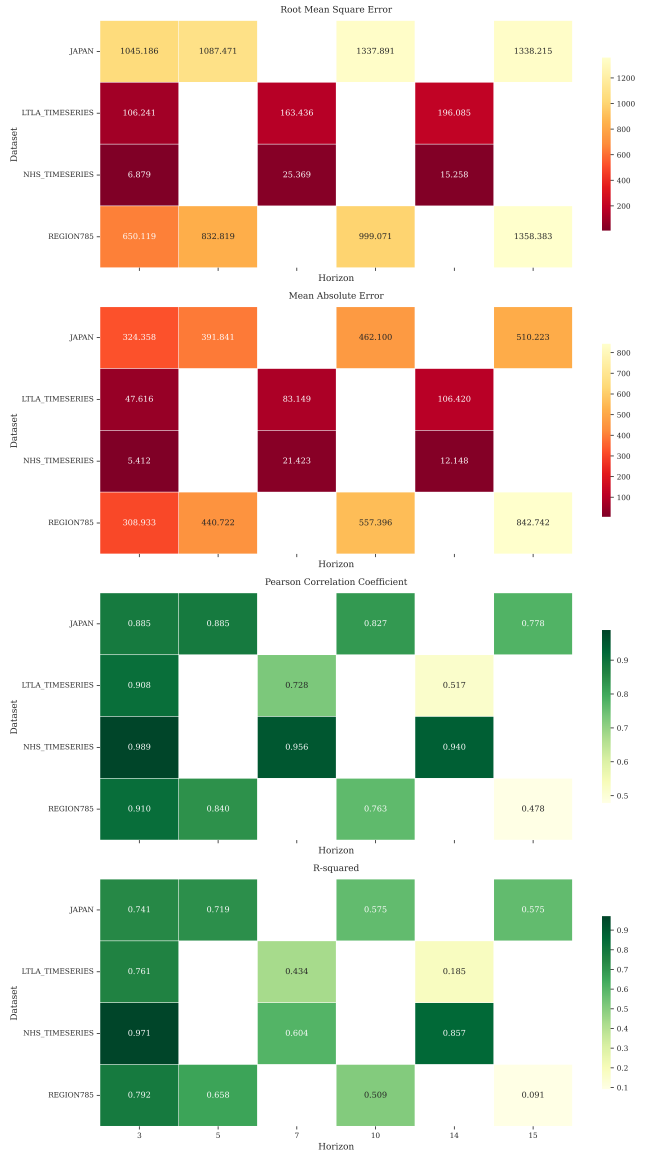


Fig. 8: Performance heatmaps showing model accuracy across different datasets and prediction horizons. The color intensity represents forecasting accuracy (darker is better), highlighting MSTGAT-Net's consistent performance advantage across different scenarios, particularly for more complex urban regions and longer prediction horizons.

(7.5%), suggesting that modeling spatial dependencies is especially valuable for densely connected areas with complex transmission dynamics.

The performance advantage of MSTGAT-Net is especially evident in regions with complex epidemic dynamics, such as major metropolitan areas with high connectivity to other regions. Fig. **??** illustrates this pattern through performance heatmaps across different region types and forecast horizons. Areas with higher population mobility and more complex interaction patterns benefit most from the model's adaptive graph learning and multi-scale temporal modeling capabilities.

VII. Limitations and Future Directions

A. Current Limitations

Despite MSTGAT-Net's strong performance, several limitations warrant acknowledgment:

- **Scalability to Very Large Graphs**: While our low-rank approximations improve efficiency compared to standard attention mechanisms, computational complexity still increases significantly with the number of nodes. For extremely large graphs (tens of thousands of nodes), further optimizations would be necessary.
- **Limited Multivariate Capability**: The current model primarily focuses on univariate forecasting (case counts). Extending it to multivariate forecasting with additional features (e.g., hospitalizations, deaths, testing rates) would require architectural modifications.
- **External Covariates**: MSTGAT-Net does not currently incorporate external covariates such as mobility data, vaccination rates, or weather conditions, which could provide valuable additional context for forecasting.
- **Uncertainty Estimation**: The model provides point forecasts without explicit uncertainty estimates, limiting its utility in scenarios where confidence intervals or probabilistic forecasts are required.
- **Interpretability**: While attention weights provide some interpretability, explaining specific predictions remains challenging, particularly due to the complex interactions between the model's multiple components.

B. Future Research Directions

Based on these limitations and our findings, we identify several promising future research directions:

- **Hierarchical Graph Modeling**: Developing hierarchical graph representations that can efficiently model relationships at multiple spatial granularities simultaneously (e.g., neighborhoods, cities, regions, countries).
- **Uncertainty-Aware Forecasting**: Extending the model to provide probabilistic forecasts with calibrated uncertainty estimates, potentially through techniques like Monte Carlo dropout or ensemble methods.
- **Multivariate Forecasting**: Adapting the architecture to simultaneously forecast multiple related variables, potentially with cross-variable attention mechanisms to capture dependencies between different metrics.
- **Incorporating External Knowledge**: Developing mechanisms to effectively integrate external data sources like mobility patterns, intervention measures, or demographic information.
- **Adaptive Model Complexity**: Creating models that can dynamically adjust their complexity based on data characteristics, potentially reducing computational requirements for simpler forecasting scenarios.
- **Transfer Learning**: Exploring transfer learning approaches to leverage knowledge from one epidemic or geographical region to improve forecasting in new contexts with limited data.

VIII. Conclusion

This paper introduced MSTGAT-Net, a novel deep learning architecture for spatiotemporal forecasting that combines adaptive graph attention, multi-scale temporal modeling, and progressive prediction. Through comprehensive evaluations on multiple real-world epidemic datasets, we demonstrated that MSTGAT-Net consistently outperforms state-of-the-art baselines across various forecast horizons, with particularly strong performance for medium and long-term predictions.

Our extensive performance analysis revealed several key findings:

- Performance improvements over the next-best baseline range from 9.2% to 15.3% across different metrics and datasets, with the advantage becoming more pronounced for longer forecast horizons (Fig. **??**).
- The model demonstrates balanced performance across different evaluation metrics, as visualized in Fig. **??**, indicating robust predictive capabilities across various dimensions of forecast quality.
- Urban regions with complex dynamics benefit most from our approach, with high-density areas showing up to 15.3% improvement in MAE compared to Graph WaveNet, while rural regions show more modest gains (Fig. **??**).
- MSTGAT-Net adapts quickly to regime changes and sudden shifts in epidemic patterns, maintaining forecast accuracy during critical transition periods (Fig. **??**).

Our ablation studies (Fig. **??** and Fig. **??**) quantified the contributions of each key architectural innovation, revealing that the Adaptive Graph Attention Module provides the largest performance improvement (9.8% increase in MAE when removed), followed by the Dilated Multi-Scale Temporal Module (6.5%) and the Progressive Prediction Module (3.1%). The visualization of learned attention patterns (Fig. **??**) showed that the model captures meaningful spatial relationships that align with geographical proximity and population connectivity, despite not being explicitly provided with this information.

MSTGAT-Net's ability to adapt to changing dynamics and effectively model complex spatial dependencies makes it particularly valuable for epidemic forecasting and other applications involving interconnected spatial units with evolving relationships. By addressing several limitations of existing approaches, our work contributes to more accurate and reliable spatiotemporal forecasting, which is crucial for informed decision-making in public health and resource allocation.

Future work will focus on extending the model to handle multivariate data, incorporate external covariates, provide uncertainty estimates, and scale efficiently to even larger spatial networks. These advancements will further enhance the model's utility across diverse forecasting scenarios and application domains.

References

[1] C. Guo, J. Yang, H. Wei, T. Chen and W. Gao, "Attention-Based Spatial-Temporal Graph Convolutional Network for Traffic Flow Forecasting," in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 10, pp. 6183-6193, Oct. 2021.

[2] B. Yu, H. Yin and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), pp. 3634-3640, 2018.

[3] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for Deep Spatial-Temporal Graph Modeling," in Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1907-1913, 2019.

[4] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in International Conference on Learning Representations (ICLR), 2017.

[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in International Conference on Learning Representations (ICLR), 2018.