

Bios 740- Chapter 11. Spatio-temporal process

Acknowledgement: Many thanks to Mr. Shuai Huang for preparing some of these slides, and to Yuxuan Liang for sharing their slides. I also drew on material from the ECML/PKDD 2023 tutorial by Cesare Alippi, Daniele Zambon, Andrea Cini, and Ivan Marisca., as well as content generated by ChatGPT.

Content

1 Spatio-temporal Data and Tasks

2 Problem Definitions and Evaluation Metrics

3 Spatio-temporal Modeling

4 Applications

5 Challenges and future works

Content

1 Spatio-temporal Data and Tasks

2 Problem Definitions and Evaluation Metrics

3 Spatio-temporal Modeling

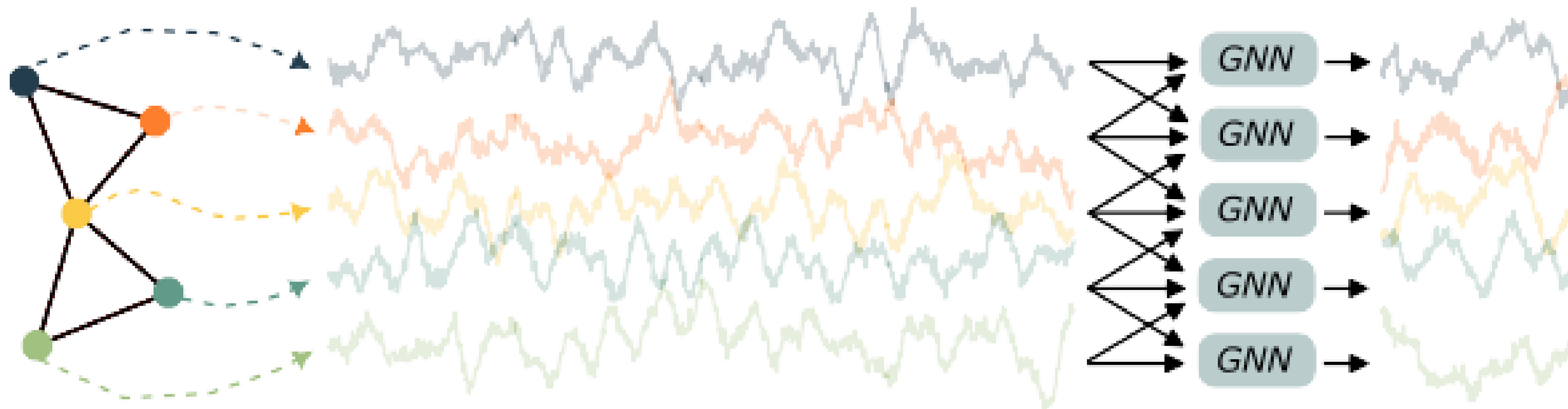
4 Applications

5 Challenges and future works

Spatio-temporal Data

Spatial data are data derived from spatial measurements that have only one parameter: the space and its characteristics are mainly location coordinates and constitute the mining data.

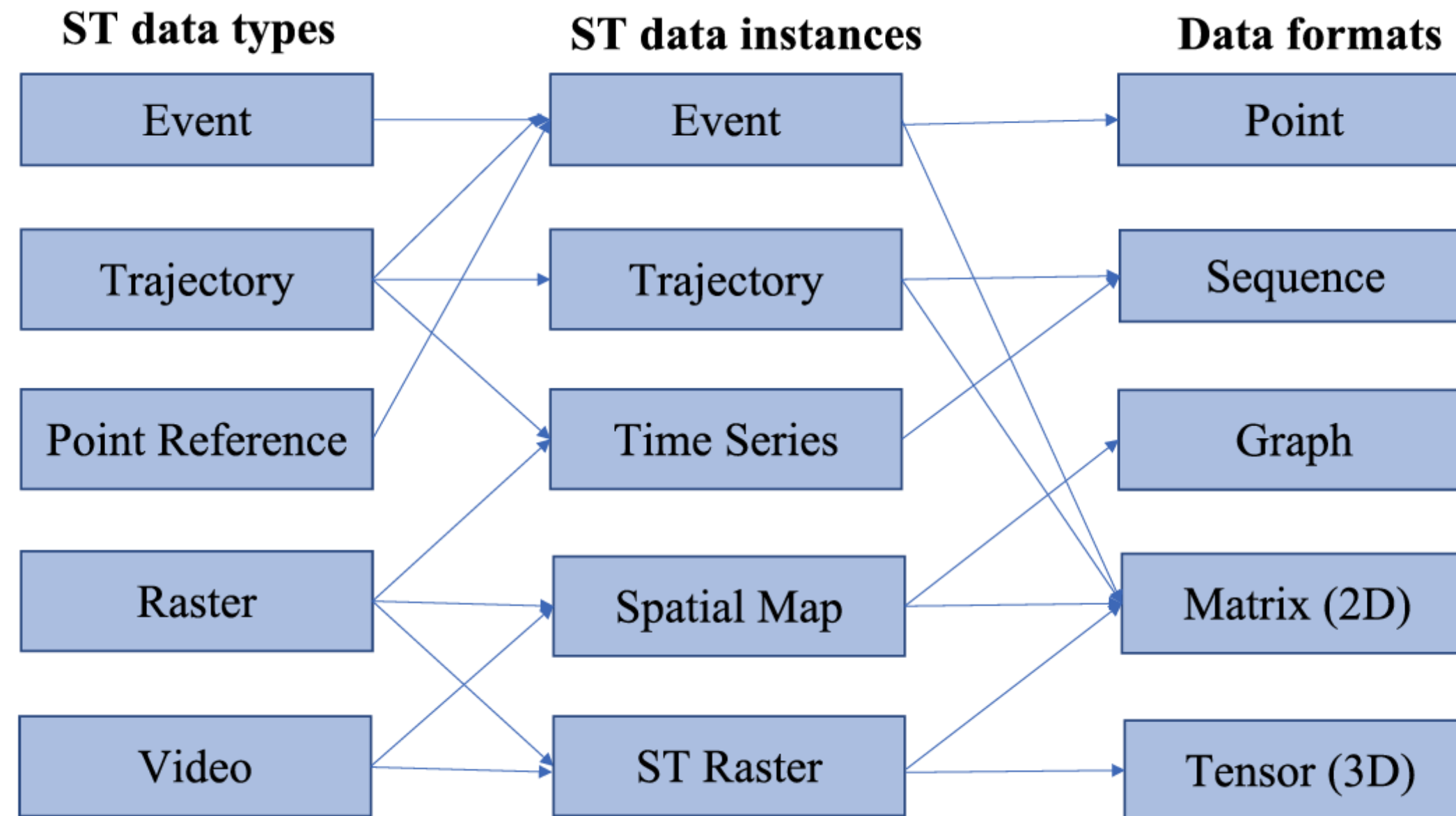
Spatiotemporal data, however, are data derived from measurements, which take into account both the parameters of space and time.



Spatio-temporal Data

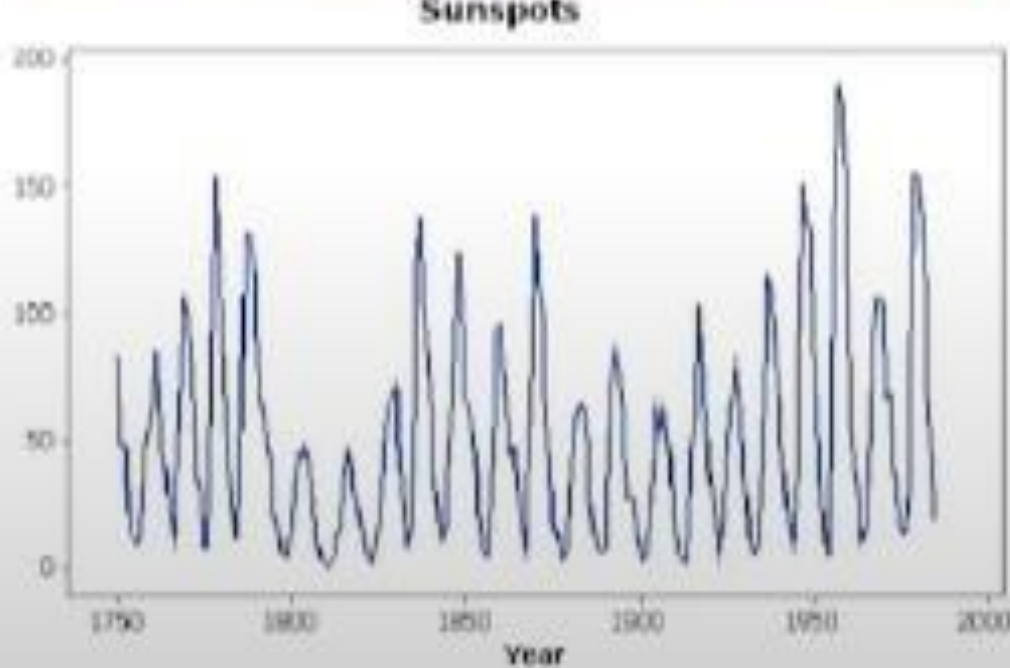
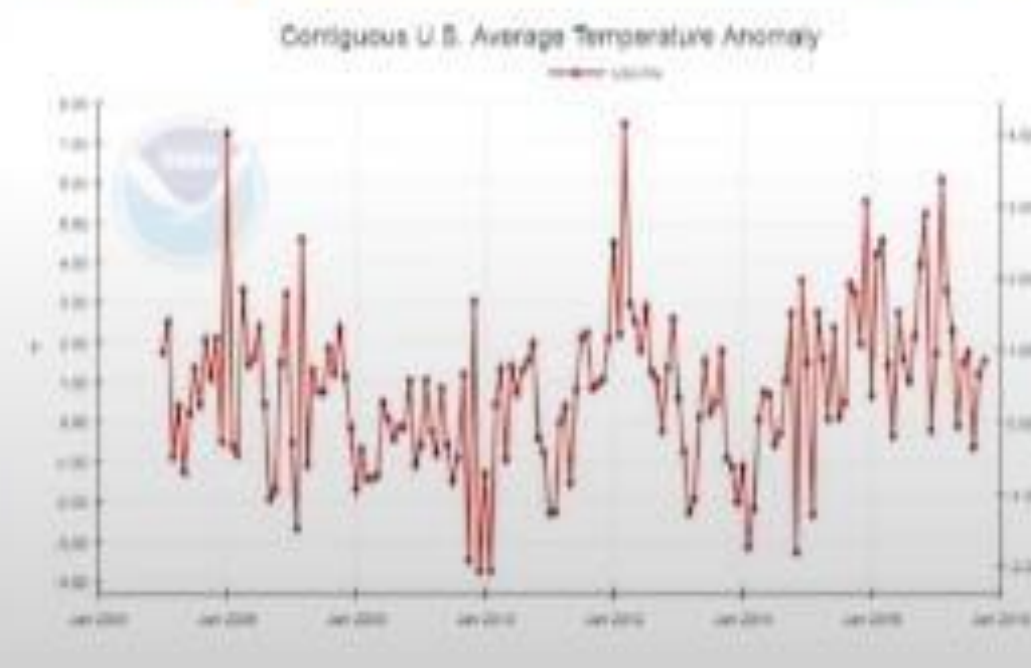
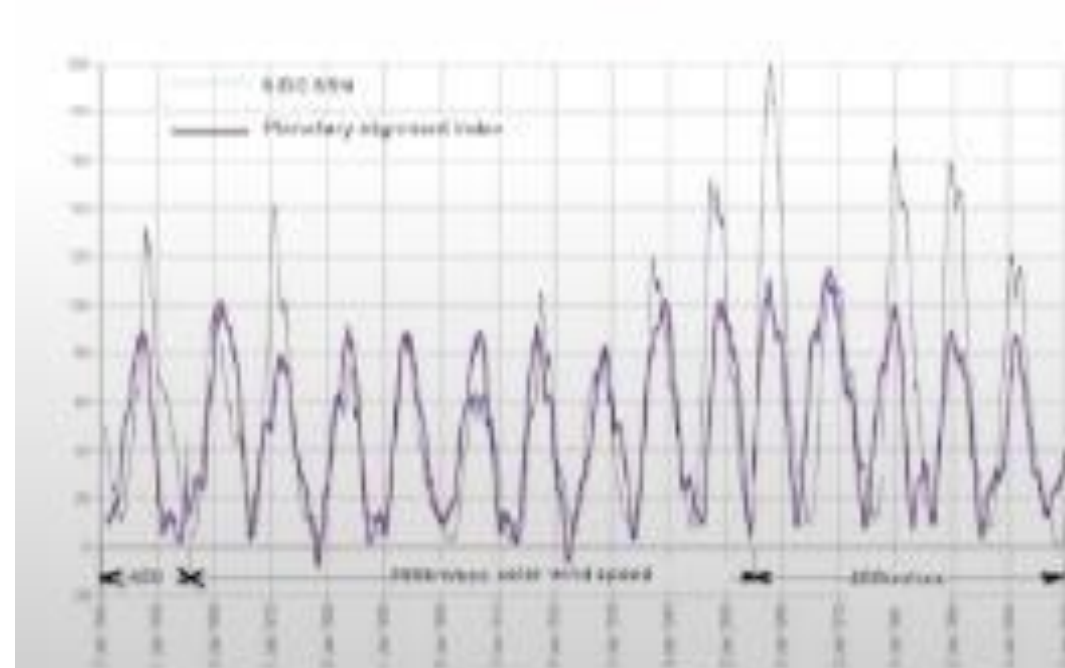
Spatial data are data derived from spatial measurements that have only one parameter: the space and its characteristics are mainly location coordinates and constitute the mining data.

Spatiotemporal data, however, are data derived from measurements, which take into account both the parameters of space and time.



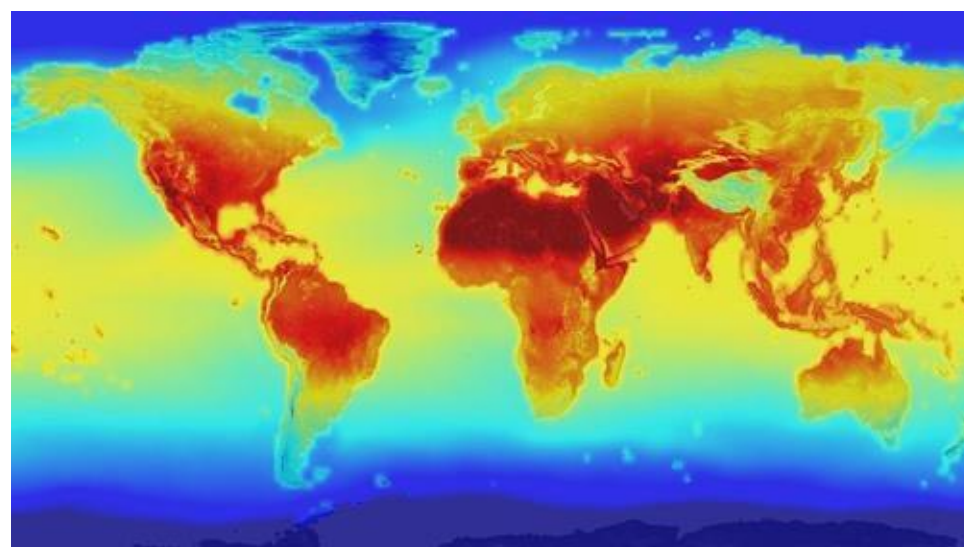
Spatio-temporal Data – Time series

- Time series is a sequence of data points collected or recorded at specific time intervals, showing how a variable changes over time
- Multivariate time series is a typical spatio-temporal data

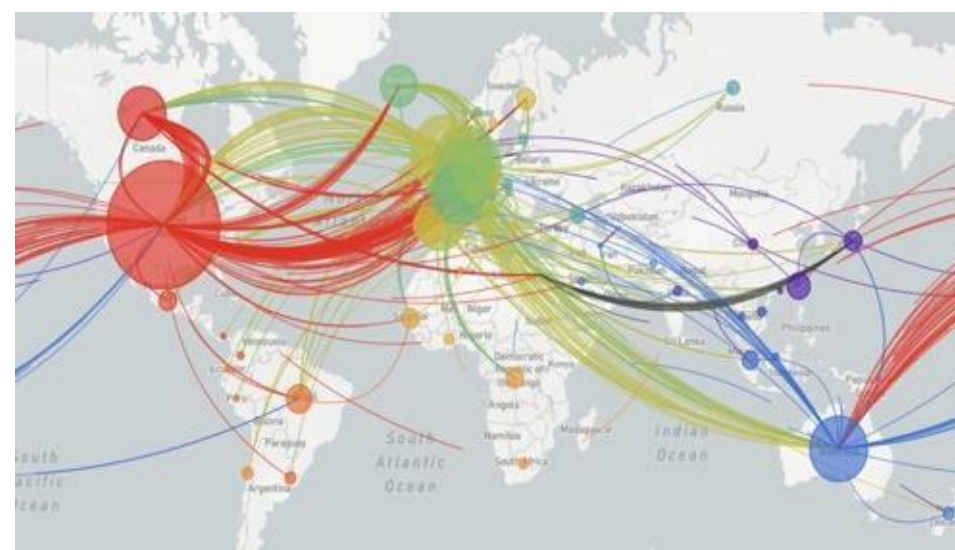


Spatio-temporal Data – Time series

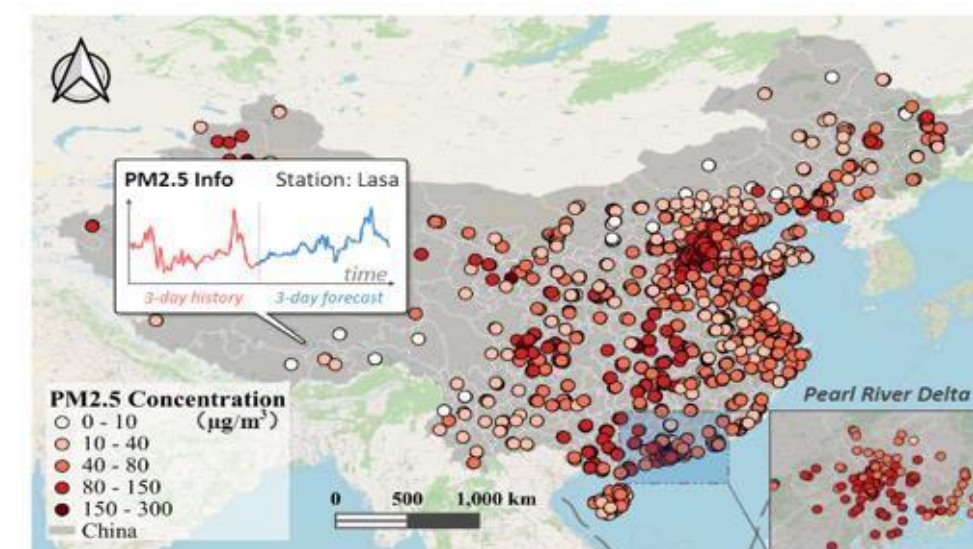
With recent advances in sensing technologies, a myriad of **Time Series (TS) Data** has been collected and contributed to various disciplines



Climate



Epidemiology



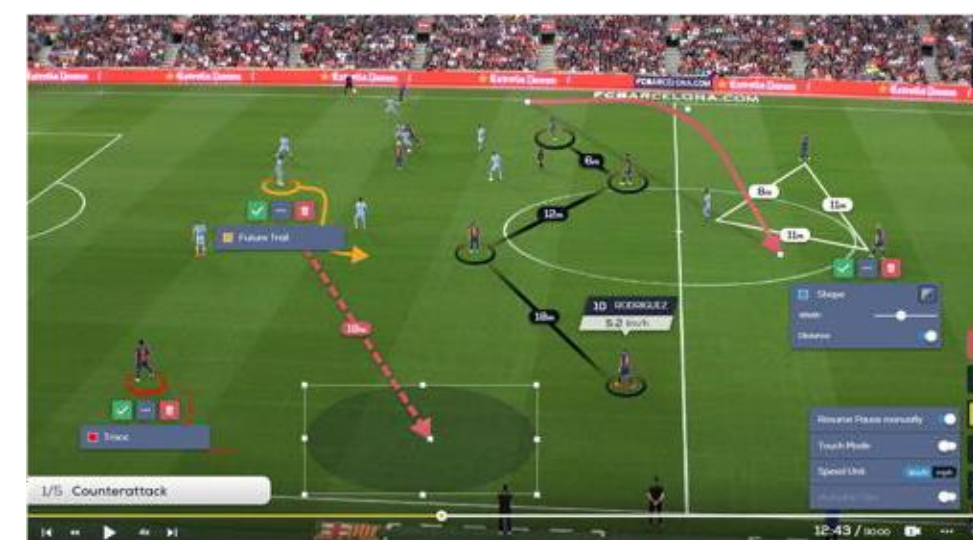
Environment



Social Science

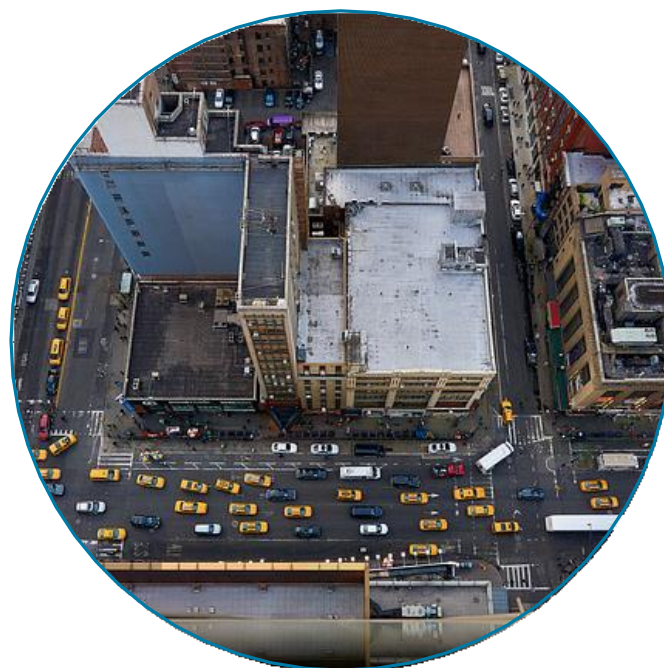


Transportation



Sports Analysis

Spatio-temporal Data – Time series



Traffic monitoring



Smart cities



Energy analytics



Physics



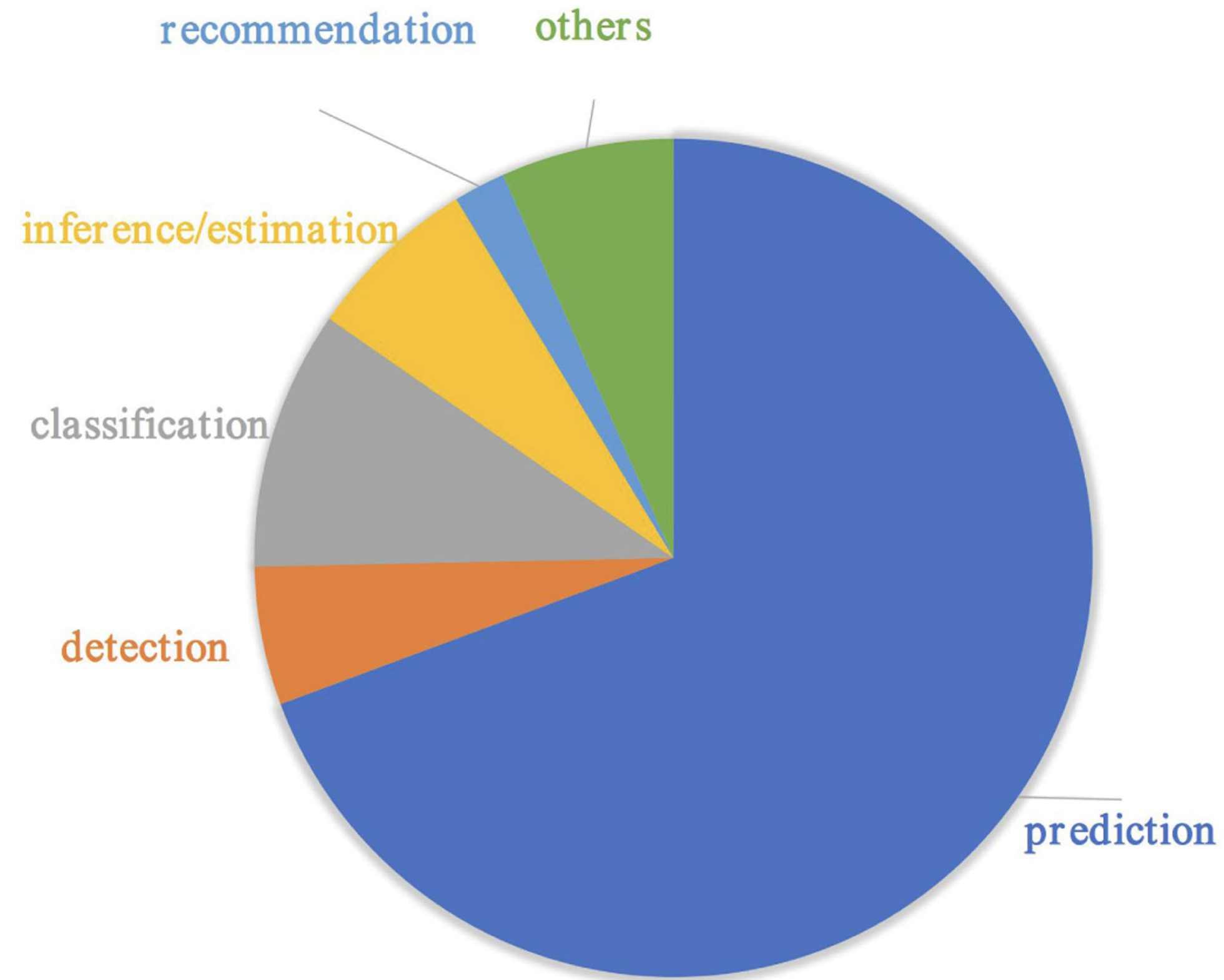
Stock markets

Spatio-temporal Data

Model Name*	Dataset Names Used in Experiments	Applications
HyDCNN [13]	Traffic, Solar-Energy, Electricity	Traffic,Energy
SCINet [14]	Traffic, Solar-Energy, Electricity, ETT, Exchange-Rate, PEMS	Traffic,Energy,Finance
DA-RNN [15]	SML 2010, NASDAQ 100 Stock	Finance
MQRNN [16]	GEFCom2014	Energy
MTGNN [17]	Traffic, Solar-Energy, Electricity, Exchange-Rate, PEMS-BAY, METR-LA	Traffic,Energy,Finance
AutoSTG [18]	PEMS-BAY, METR-LA	Traffic
DMSTGCN [19]	PeMSD4, PeMSD8	Traffic
TPGNN [20]	Traffic, Solar-Energy, Electricity, Exchange-Rate	Traffic,Energy,Finance
MAGNN [21]	Traffic, Solar-Energy, Electricity, Exchange-Rate, Nasdaq	Traffic,Energy,Finance
TFT [22]	Electricity, Traffic, Retail	Traffic,Energy
Informer [23]	ECL, Weather, ETT	Weather,Energy
Autoformer [24]	ILI, Weather, ETT, Traffic, Exchange-Rate, Electricity	Medical,Weather,Energy,Traffic,Finance
Pyraformer [25]	ETT, Electricity	Energy
FEDformer [26]	ILI, Weather, ETT, Traffic, Exchange-Rate, Electricity	Medical,Weather,Energy,Traffic,Finance
Non-stationary Transformer [27]	ILI, Weather, ETT, Traffic, Exchange-Rate, Electricity	Medical,Weather,Energy,Traffic,Finance

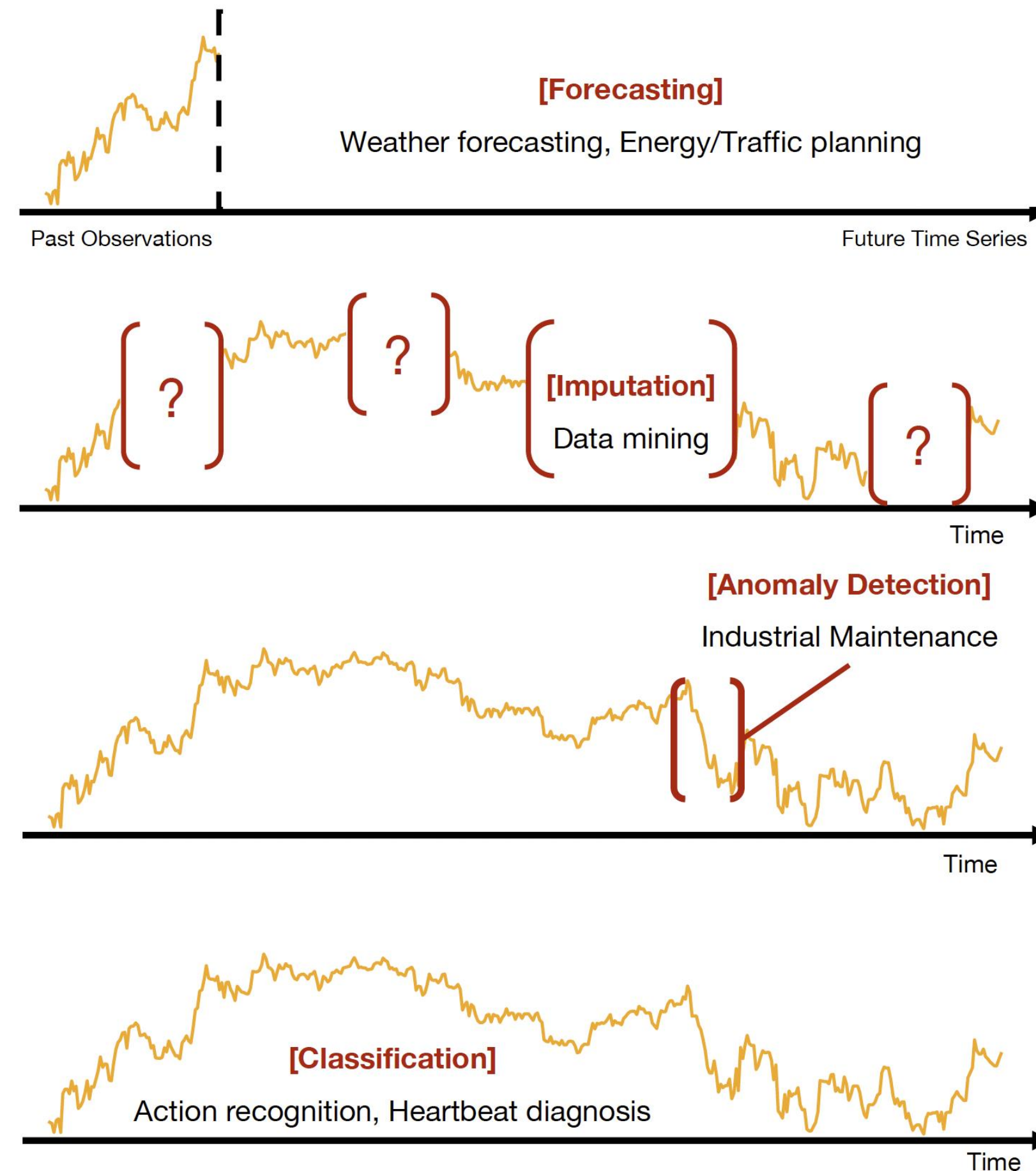
Spatio-temporal Tasks

- Predictive learning:
Prediction; Forecasting
- Classification
- Estimation and Inference
- Anomaly detection



Spatio-temporal Tasks

- Predictive learning:
Prediction; Forecasting
- Classification
- Estimation and Inference
- Anomaly detection



Wang, Y., Wu, H., Dong, J., Liu, Y., Long, M. and Wang, J., 2024. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*.

Content

1 Spatio-temporal Data and Tasks

2 Problem Definitions and Evaluation Metrics

3 Spatio-temporal Modeling

4 Applications

5 Challenges and future works

Problem formulations

Table 1. Summary of deep Forecasting Models based on Forecast and Model Type

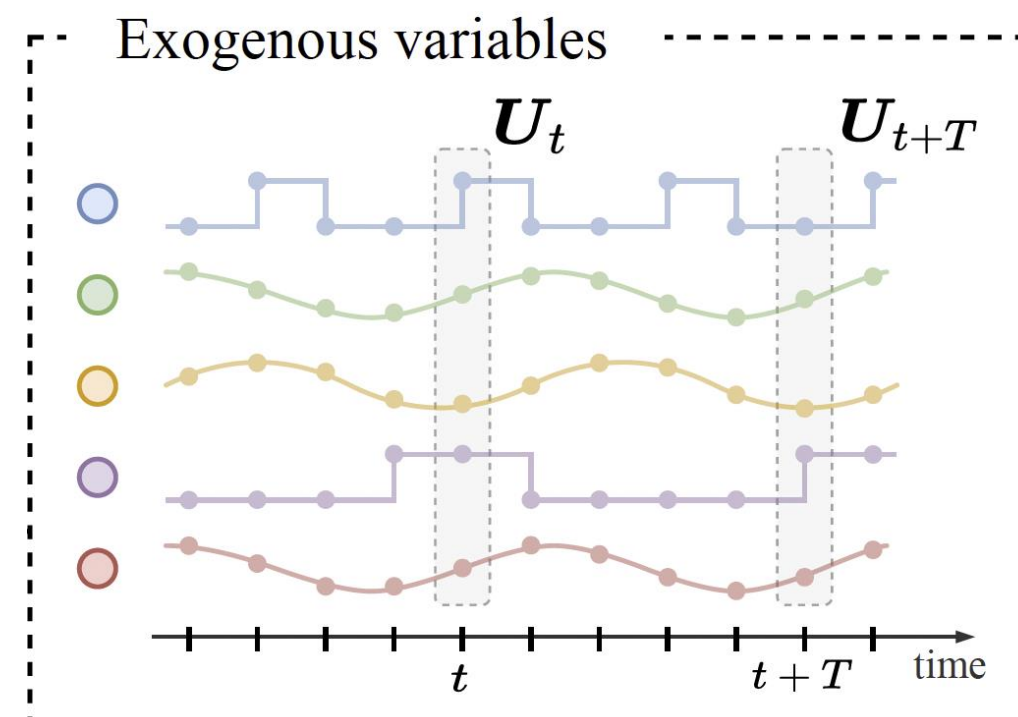
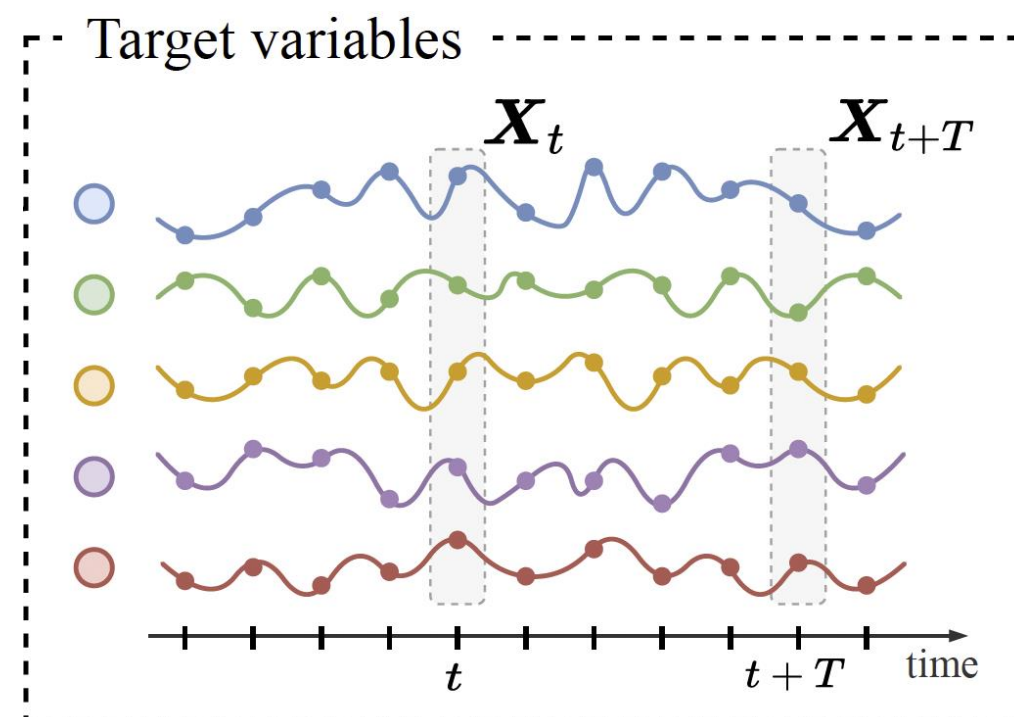
Forecast type	Model type	Formulation
Point	Local univariate	$\hat{z}_{i,t+1:t+h} = \Psi(z_{i,1:t}, X_{i,1:t+h})$
	Global univariate	$\hat{z}_{i,t+1:t+h} = \Psi(z_{i,1:t}, X_{i,1:t+h}, \Phi)$
	Multivariate	$\hat{Z}_{t+1:t+h} = \Psi(Z_{1:t}, X_{1:t+h}, \Phi)$
Probabilistic	Local univariate	$P(z_{i,t+1:t+h} z_{i,1:t}, X_{i,1:t+h}; \theta_i), \quad \theta_i = \Psi(z_{i,1:t}, X_{i,1:t+h})$
	Global univariate	$P(z_{i,t+1:t+h} Z_{1:t}, X_{1:t+h}; \theta_i), \quad \theta_i = \Psi(z_{i,1:t}, X_{i,1:t+h}, \Phi)$
	Multivariate	$P(Z_{t+1:t+h} Z_{1:t}, X_{1:t+h}; \theta), \quad \theta = \Psi(Z_{1:t}, X_{1:t+h}, \Phi)$

For one-step and multi-step forecasting models $h = 1$ and $h > 1$, respectively.

Setup of multivariate time series

A set of N correlated time series, where each i -th time series is associated with:

- an observation vector $\mathbf{x}_t^i \in R^{d_x}$ at each time step t ;
- a vector of exogenous variable $\mathbf{u}_t^i \in R^{d_u}$ at each time step t ;
- a vector of static (time-independent) attributes $\mathbf{v}_t^i \in R^{d_v}$.



$$\mathbf{V} = \begin{bmatrix} \text{blue} & \text{blue} & \text{blue} & \text{blue} \\ \text{green} & \text{green} & \text{green} & \text{green} \\ \text{yellow} & \text{yellow} & \text{yellow} & \text{yellow} \\ \text{purple} & \text{purple} & \text{purple} & \text{purple} \\ \text{red} & \text{red} & \text{red} & \text{red} \end{bmatrix} \quad \mathcal{E}_t = \left\{ \begin{bmatrix} \text{green} & \text{green} & \text{blue} \\ \text{green} & \text{yellow} & \text{blue} \\ \text{blue} & \text{yellow} & \text{blue} \\ \text{yellow} & \text{purple} & \text{yellow} \\ \text{yellow} & \text{red} & \text{yellow} \\ \text{purple} & \text{red} & \text{purple} \end{bmatrix} \right\}$$

Matrices denote the stacked N observations at time t ,

e.g., $\mathbf{X}_t \in R^{N \times d_x}$, $\mathbf{U}_t \in R^{N \times d_u}$.

Setup of multivariate time series

We consider a setup where observations have been generated by a **time-invariant** spatiotemporal stochastic process such that

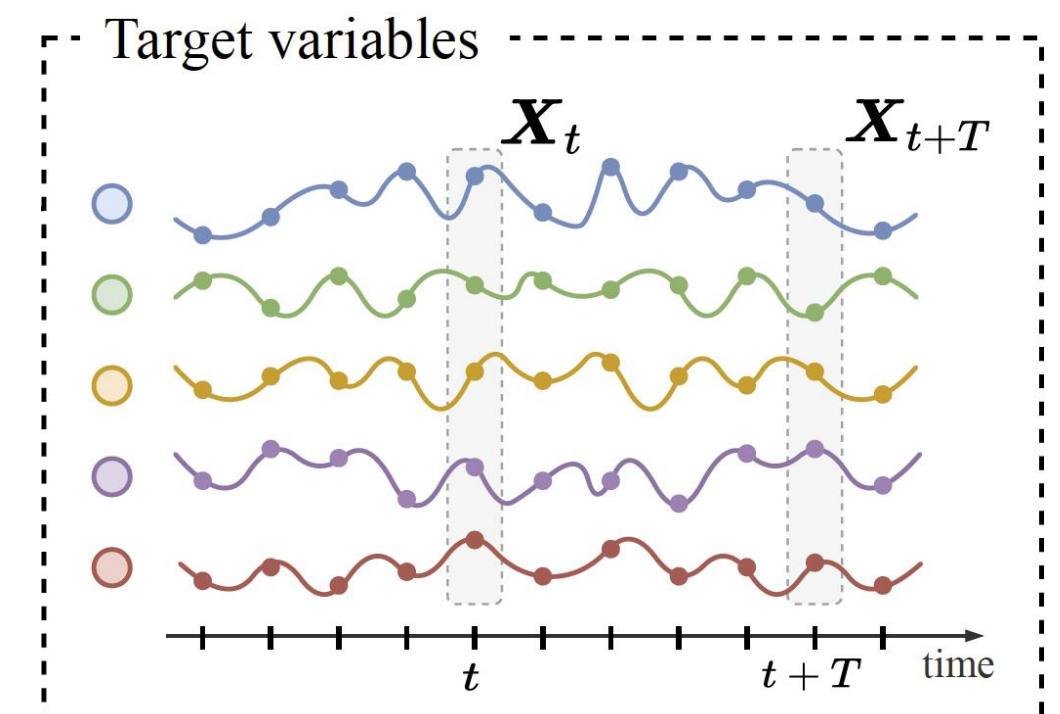
$$\mathbf{x}_t^i \sim p^i(\mathbf{x}_t^i | \mathbf{X}_{<t}, \mathbf{U}_{\leq t}, \mathbf{V}) \quad \forall i = 1, \dots, N;$$

$\mathbf{X}_{t:t+T}$: the sequence of observations within time interval $[t, t + T)$;

$\mathbf{X}_{<t}$: observations at time steps up to t (excluded)

Note that the time series:

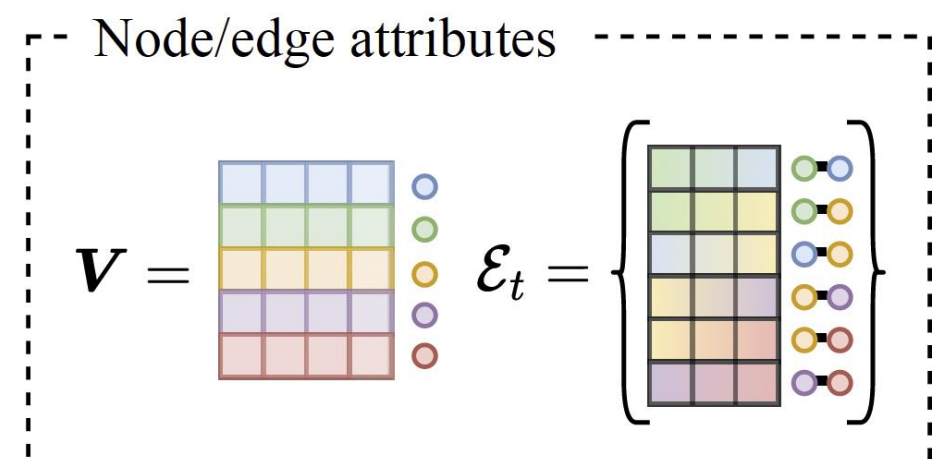
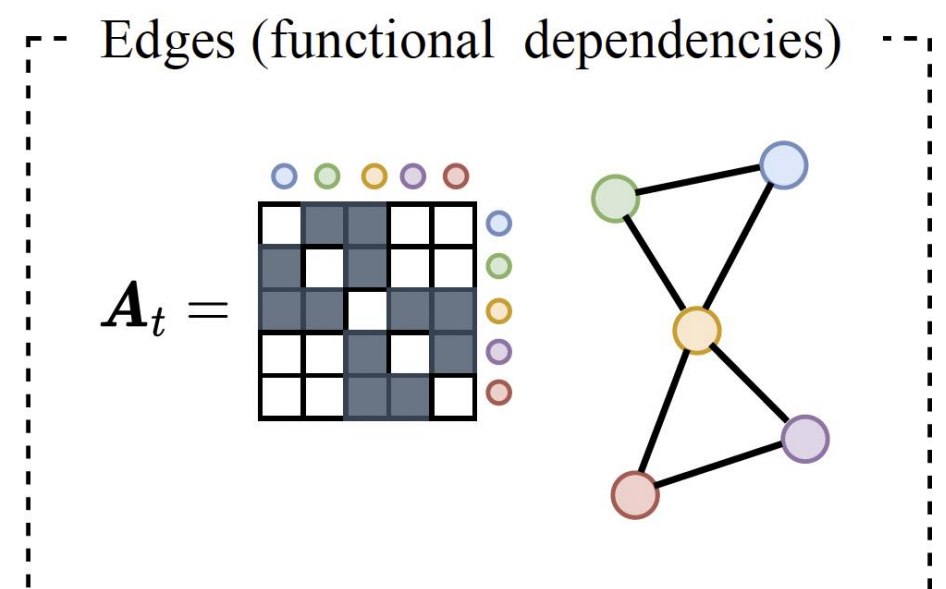
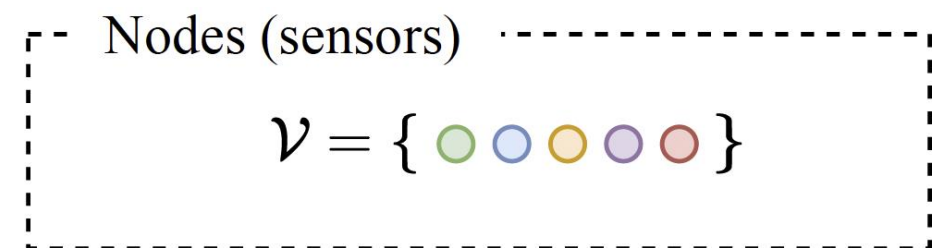
- can be generated by **different processes**,
- can **depend on each other**,
- are assumed **homogenous, synchronous, regularly sampled**.



Relational information

- Pairwise relationships existing among the time series at time step t can be encoded by a **adjacency matrix** $\mathbf{A}_t \in \{0, 1\}^{N \times N}$.
- \mathbf{A}_t can be asymmetric and dynamic (can vary with t).
- optional edge attributes $\mathbf{e}_t^{ij} \in \mathbb{R}^{d_e}$ can be associated to each non-zero entry of \mathbf{A}_t .
- $\mathcal{E}_t \doteq \{ \langle (i, j), \mathbf{e}_t^{ij} \rangle \mid \forall i, j : \mathbf{A}_t[i, j] \neq 0 \}$: the set of attributed edges encoding all the available relational information
- Tuple $\mathcal{G}_t \doteq \langle \mathbf{X}_t, \mathbf{U}_t, \mathcal{E}_t, \mathbf{V} \rangle$ indicates all the available information at time step t .

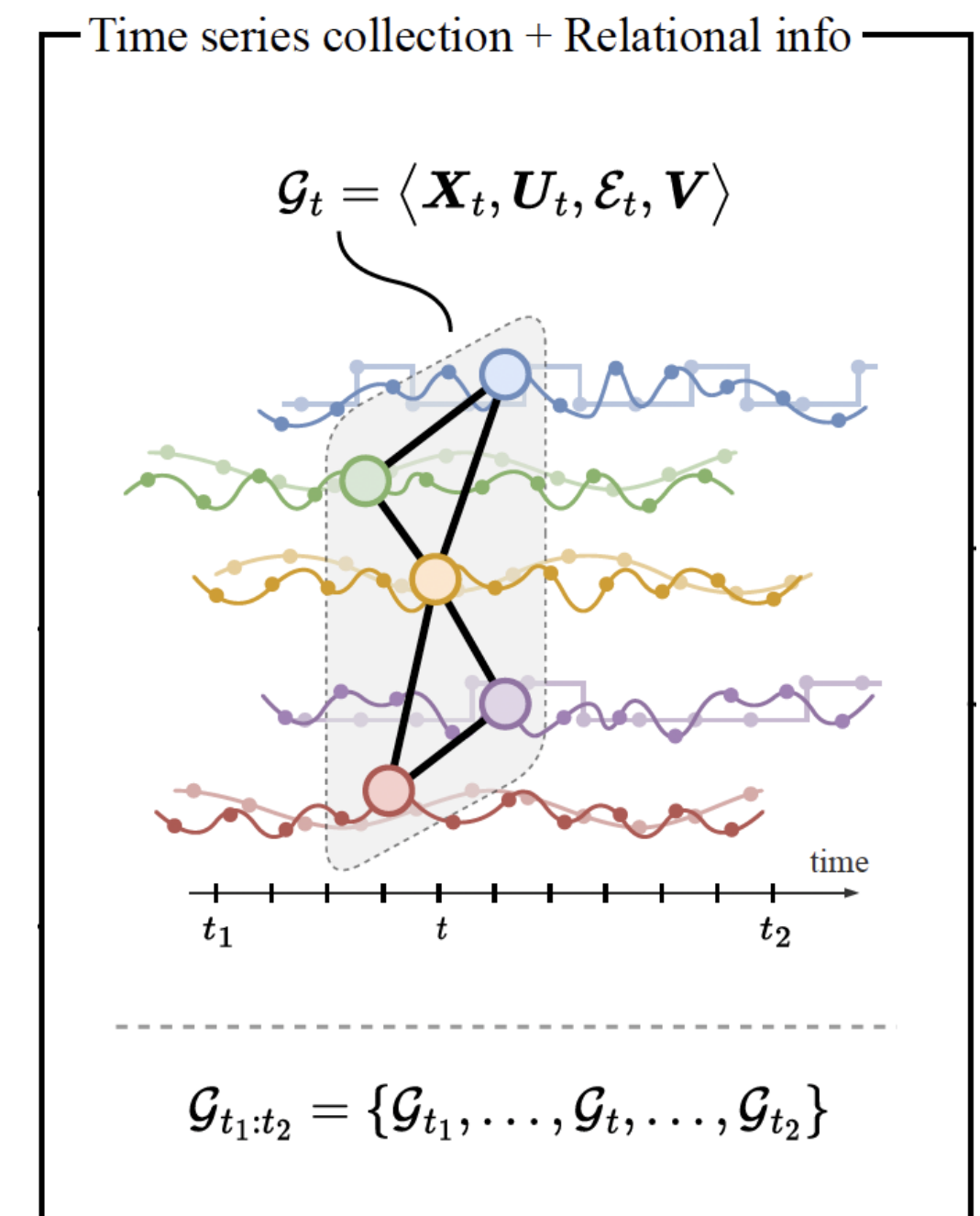
- Pairwise relationships existing among the time series at time step t can be encoded by a **adjacency matrix** $\mathbf{A}_t \in \{0, 1\}^{N \times N}$.
- \mathbf{A}_t can be asymmetric and dynamic (can vary with t).
- optional edge attributes $\mathbf{e}_t^{ij} \in \mathbb{R}^{d_e}$ can be associated to each non-zero entry of \mathbf{A}_t .
- $\mathcal{E}_t \doteq$: the set of attributed edges encoding all the available relational information
- Tuple indicates all the available information at time step t .



Cini, A., Marisca, I., Zambon, D. and Alippi, C., 2023. Graph deep learning for time series forecasting. arXiv preprint arXiv:2310.15978.

Relational information

- The term **spatial** refers to the dimension of size N , that spans the time series collection; in the case of fMRI, the term spatial reflects the fact that each time series might correspond to a different physical location.
- We use the terms **node** and **sensor** to indicate the N entities generating the time series.
- We assume the existence of **functional dependencies** between the time series.
 - e.g., forecasts for one time series can be improved by accounting for the past values of other time series.



Content

1 Spatio-temporal Data and Tasks

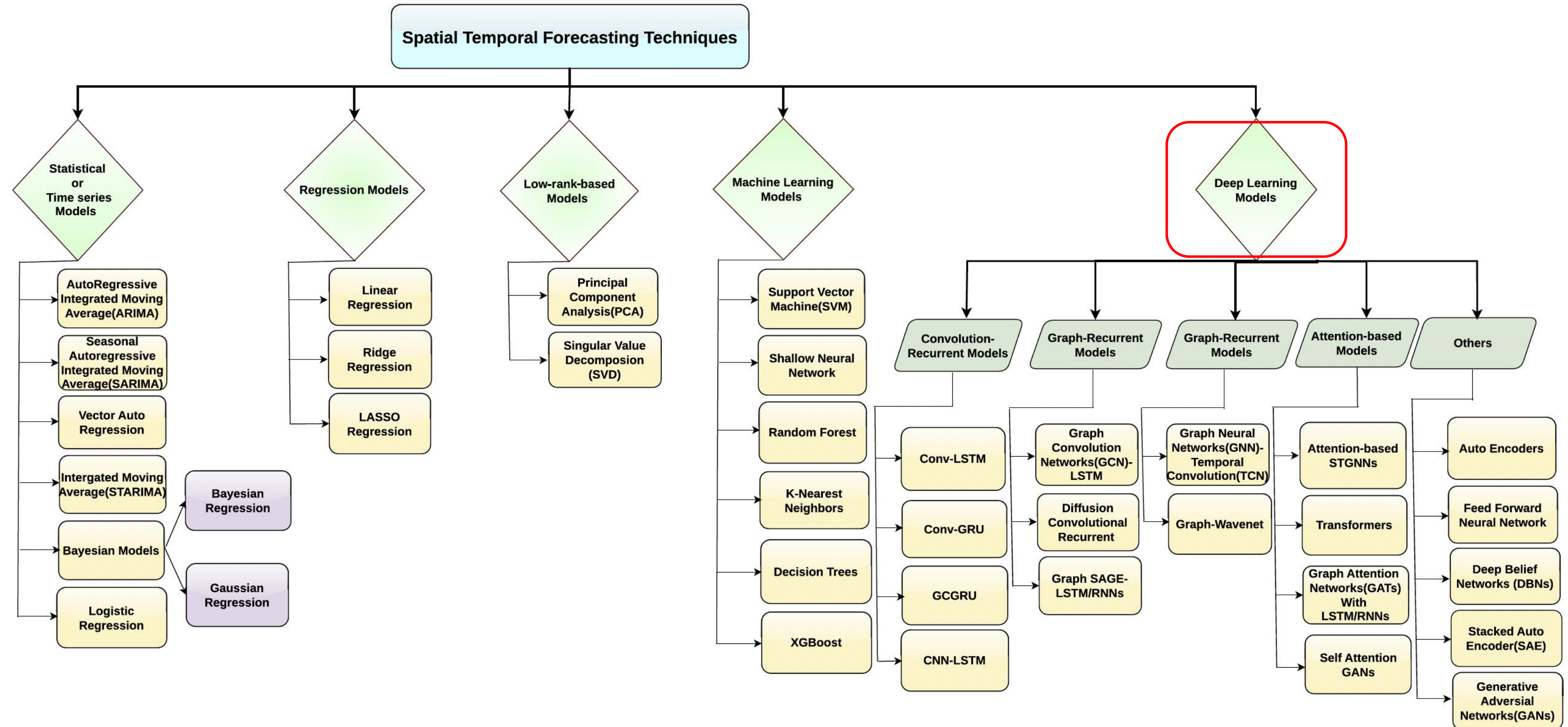
2 Problem Definitions and Evaluation Metrics

3 Spatio-temporal Modeling

4 Applications

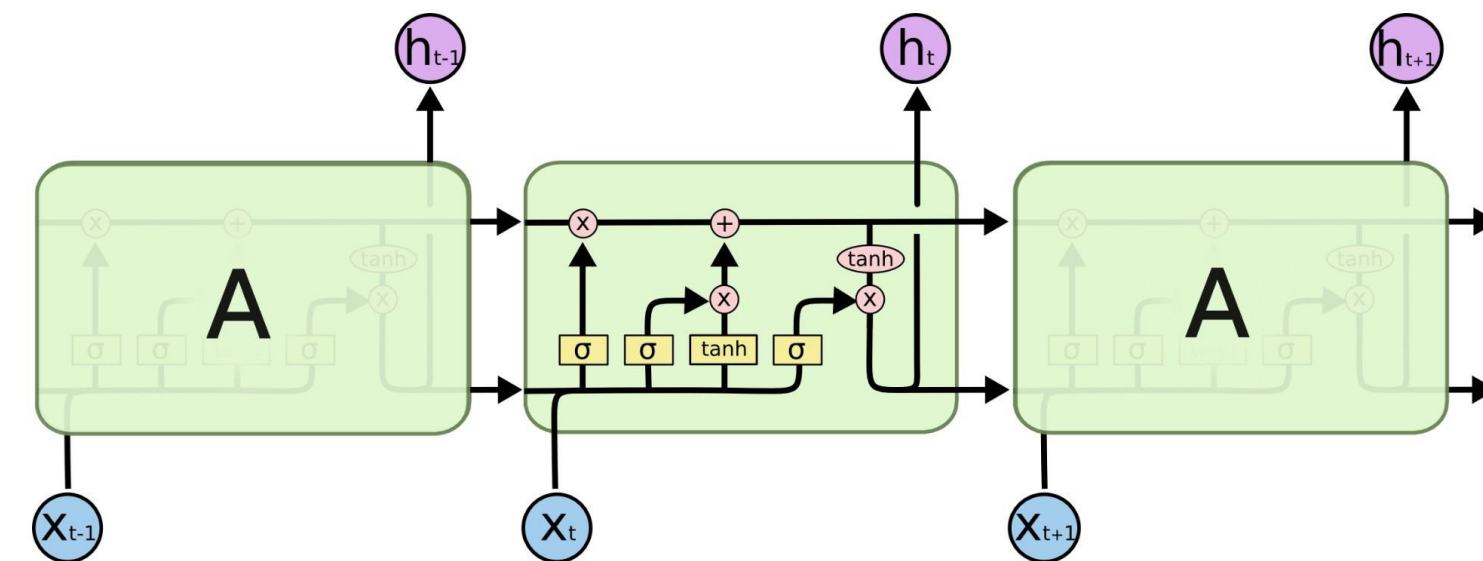
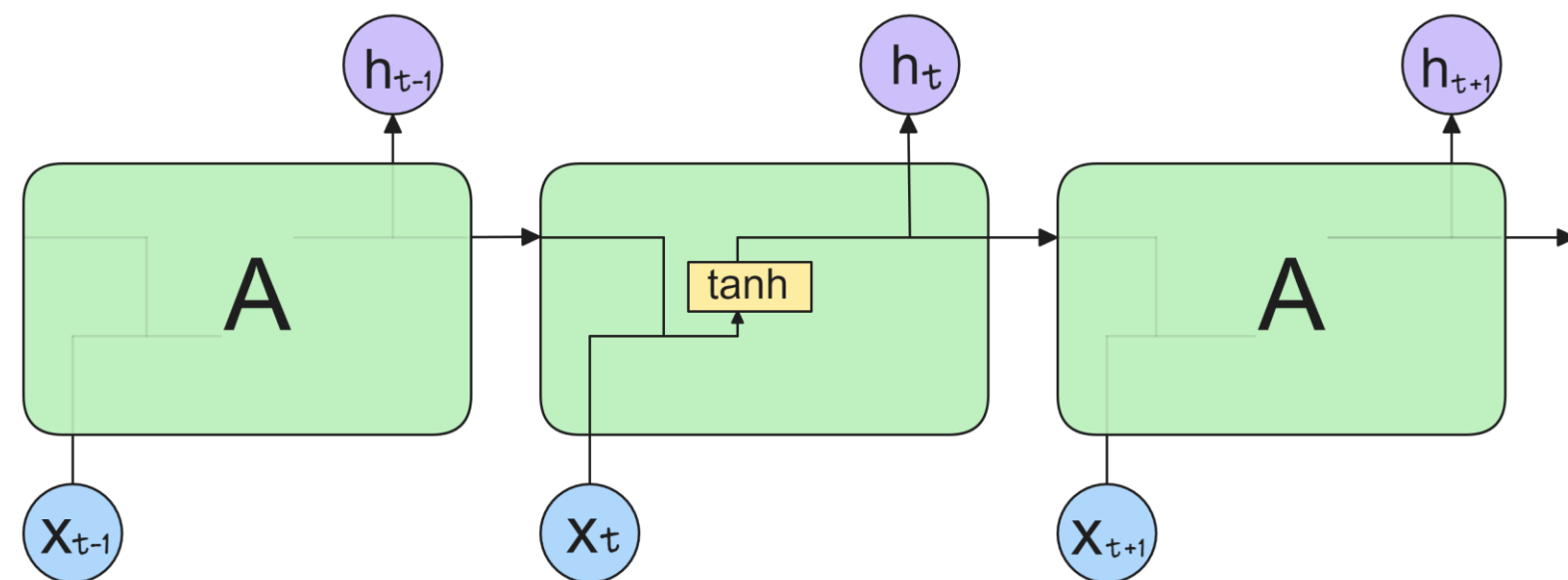
5 Challenges and future works

Spatio-temporal Modeling



Kumar, R., Bhanu, M., Mendes-Moreira, J. and Chandra, J., 2024. Spatio-Temporal Predictive Modeling Techniques for Different Domains: a Survey. *ACM Computing Surveys*, 57(2), pp.1-42.

RNN based models



RNN Structure

output distribution

$$\hat{y}^{(t)} = \text{softmax} \left(\mathbf{U} \mathbf{h}^{(t)} + \mathbf{b}_2 \right) \in \mathbb{R}^{|V|}$$

$$\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their})$$

hidden states

$$\mathbf{h}^{(t)} = \sigma \left(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_e \mathbf{e}^{(t)} + \mathbf{b}_1 \right)$$

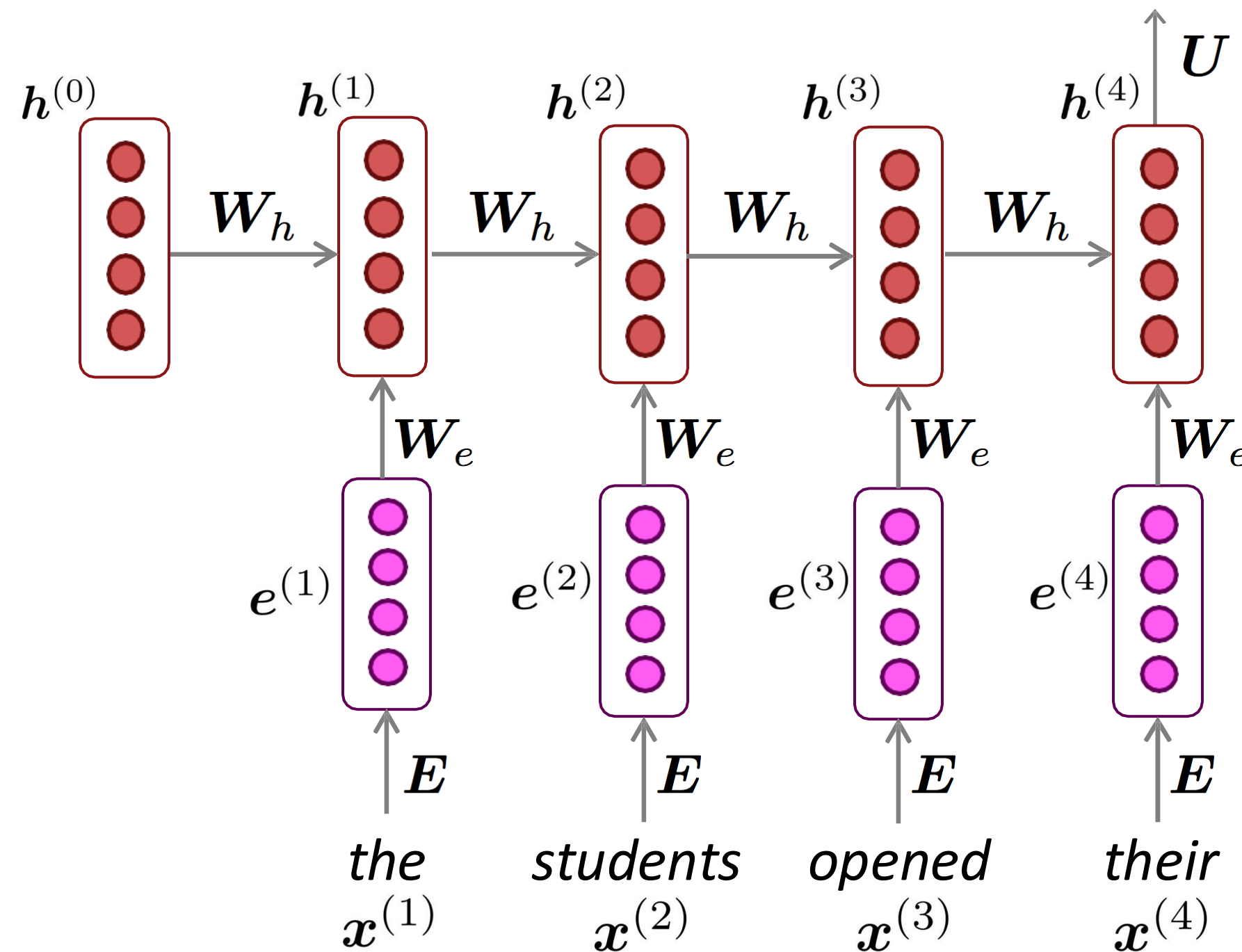
$\mathbf{h}^{(0)}$ is the initial hidden state

word embeddings

$$\mathbf{e}^{(t)} = \mathbf{E} \mathbf{x}^{(t)}$$

words / one-hot vectors

$$\mathbf{x}^{(t)} \in \mathbb{R}^{|V|}$$



Note: this input sequence could be much longer now!

LSTM Structure

We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t :

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

New cell content: this is the new content to be written to the cell

Cell state: erase (“forget”) some content from last cell state, and write (“input”) some new cell content

Hidden state: read (“output”) some content from the cell

Sigmoid function: all gate values are between 0 and 1

$$\begin{aligned} f^{(t)} &= \sigma \left(W_f h^{(t-1)} + U_f x^{(t)} + b_f \right) \\ i^{(t)} &= \sigma \left(W_i h^{(t-1)} + U_i x^{(t)} + b_i \right) \\ o^{(t)} &= \sigma \left(W_o h^{(t-1)} + U_o x^{(t)} + b_o \right) \end{aligned}$$

$$\tilde{c}^{(t)} = \tanh \left(W_c h^{(t-1)} + U_c x^{(t)} + b_c \right)$$

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$

$$h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$$

All these are vectors of same length n

Variant of RNN: DilatedRNN

- **Key Idea:**

Introduce dilated connections in RNNs to efficiently model long-range dependencies without deeper networks.

- **Architecture:**

At time t , hidden state connects to h_{t-d} instead of h_{t-1}

Use increasing dilations (e.g., 1, 2, 4, 8...) across layers

Analogous to dilated convolutions in CNNs

- **Benefits:**

Captures multi-scale temporal patterns

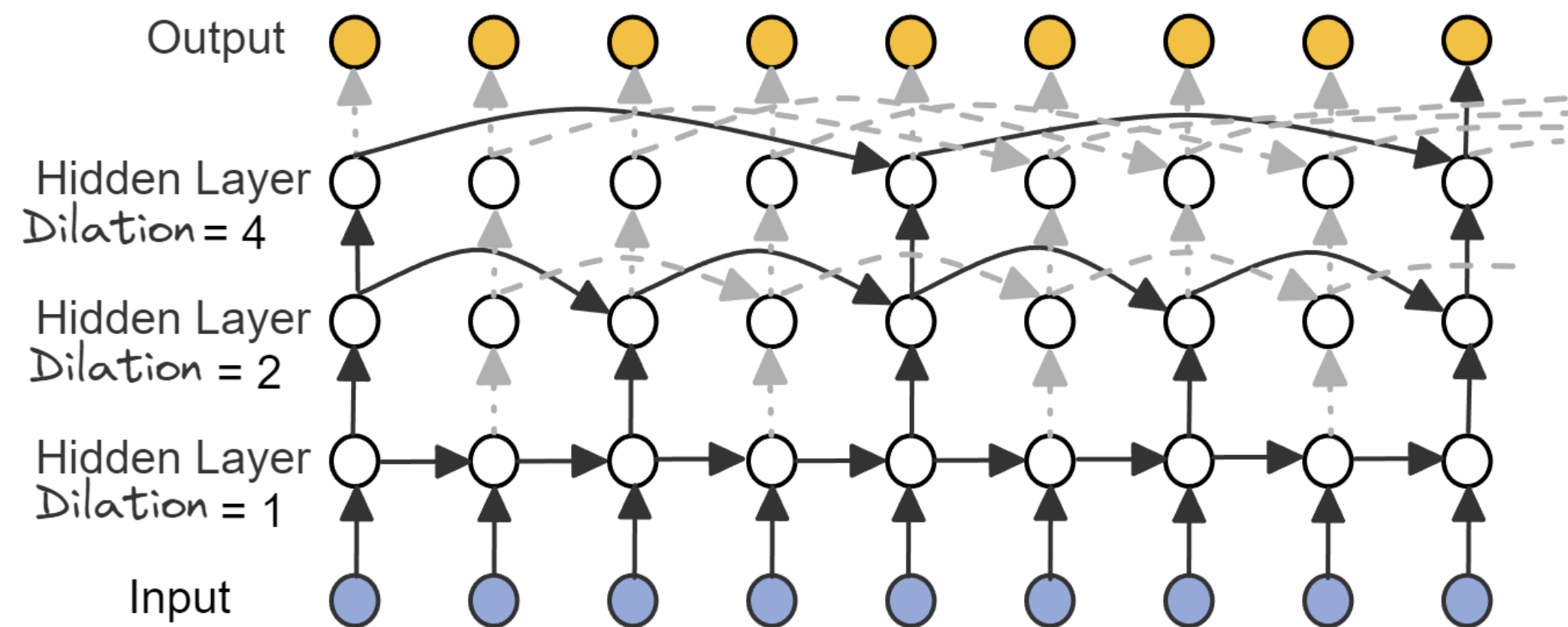
Reduces gradient vanishing

Enables faster & more stable training

- **Applications:**

Long-horizon time series forecasting

Memory-intensive tasks (e.g., permuted MNIST, copy task)



S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. A. Hasegawa-Johnson, and T. S. Huang, "Dilated recurrent neural networks," in Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS), Dec. 2017, pp. 76–86.

RNN and Attention Mechanism

- **Key Idea:**

Learn to dynamically attend to both: Important input features (variables); Relevant past time steps

- **Architecture:**

Encoder with Input Attention

→ Selects key features at each time step

Decoder with Temporal Attention

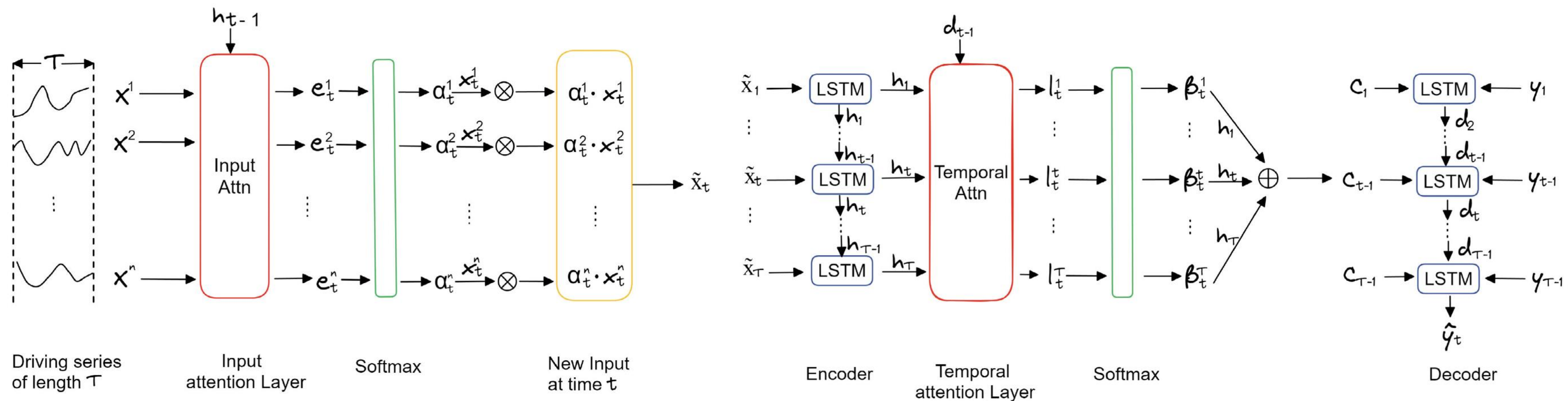
→ Focuses on key time steps for prediction

- **Benefits:**

Handles multivariate time series

Improves interpretability & accuracy

Flexible for long- and short-term forecasting



RNN and Attention Mechanism

- **Key Idea:**

Jointly model when and which features matter in multivariate time series via spatiotemporal attention.

- **Architecture:**

Temporal Attention: Learns importance of time steps for each variable

Spatial Attention: Learns importance of variables at each time step

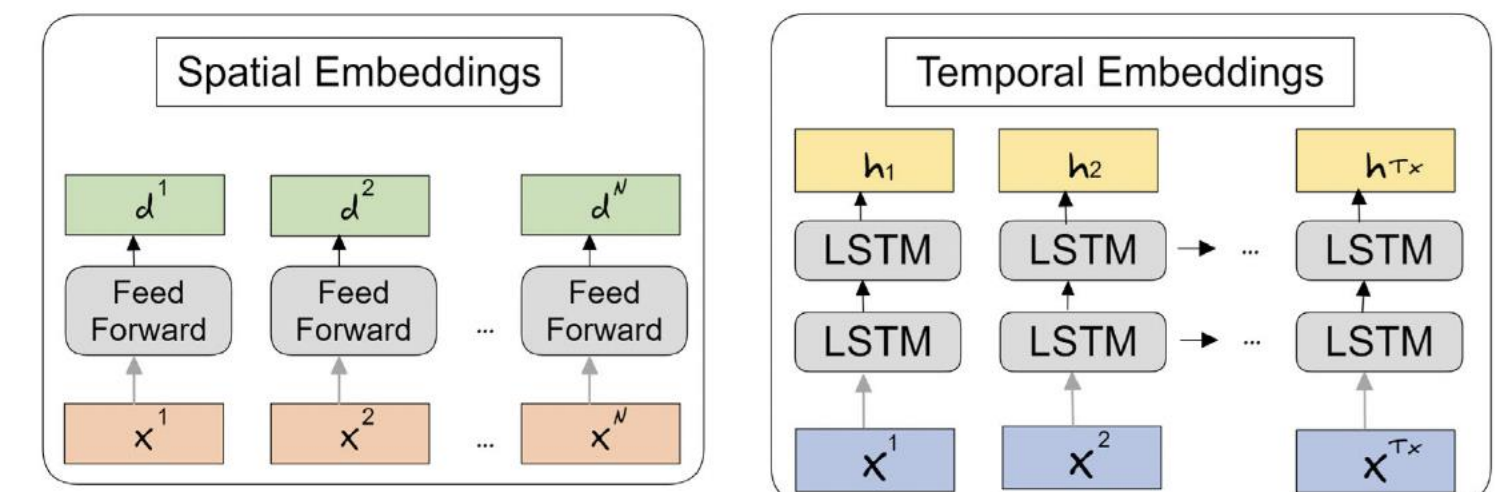
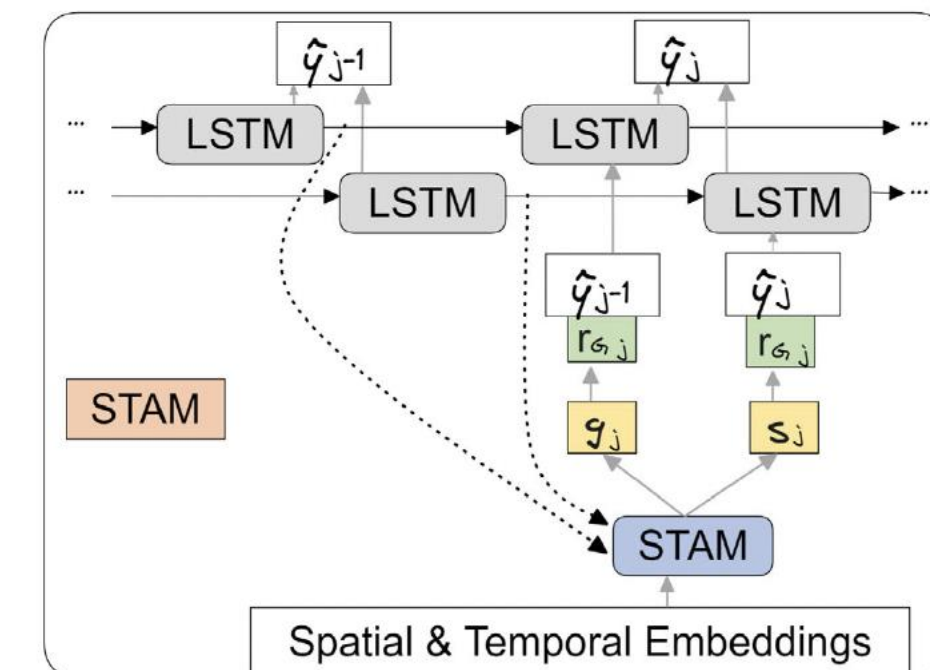
Generates a context tensor for final prediction

- **Benefits:**

Captures complex feature-time interactions

Produces interpretable attention maps

Outperforms separate temporal/spatial attention models



RNN and Seq2Seq

MQRNN

- **Key Idea:**

Forecast multiple future time steps (multi-horizon) with quantile regression using an RNN backbone.

- **Architecture:**

Input: Historical time series + optional covariates

Uses RNN encoder (e.g., LSTM/GRU) to extract context

At each future step, predicts multiple quantiles (e.g., 0.1, 0.5, 0.9)

Forking-sequences strategy: Makes all horizon predictions in one forward pass

- **Benefits:**

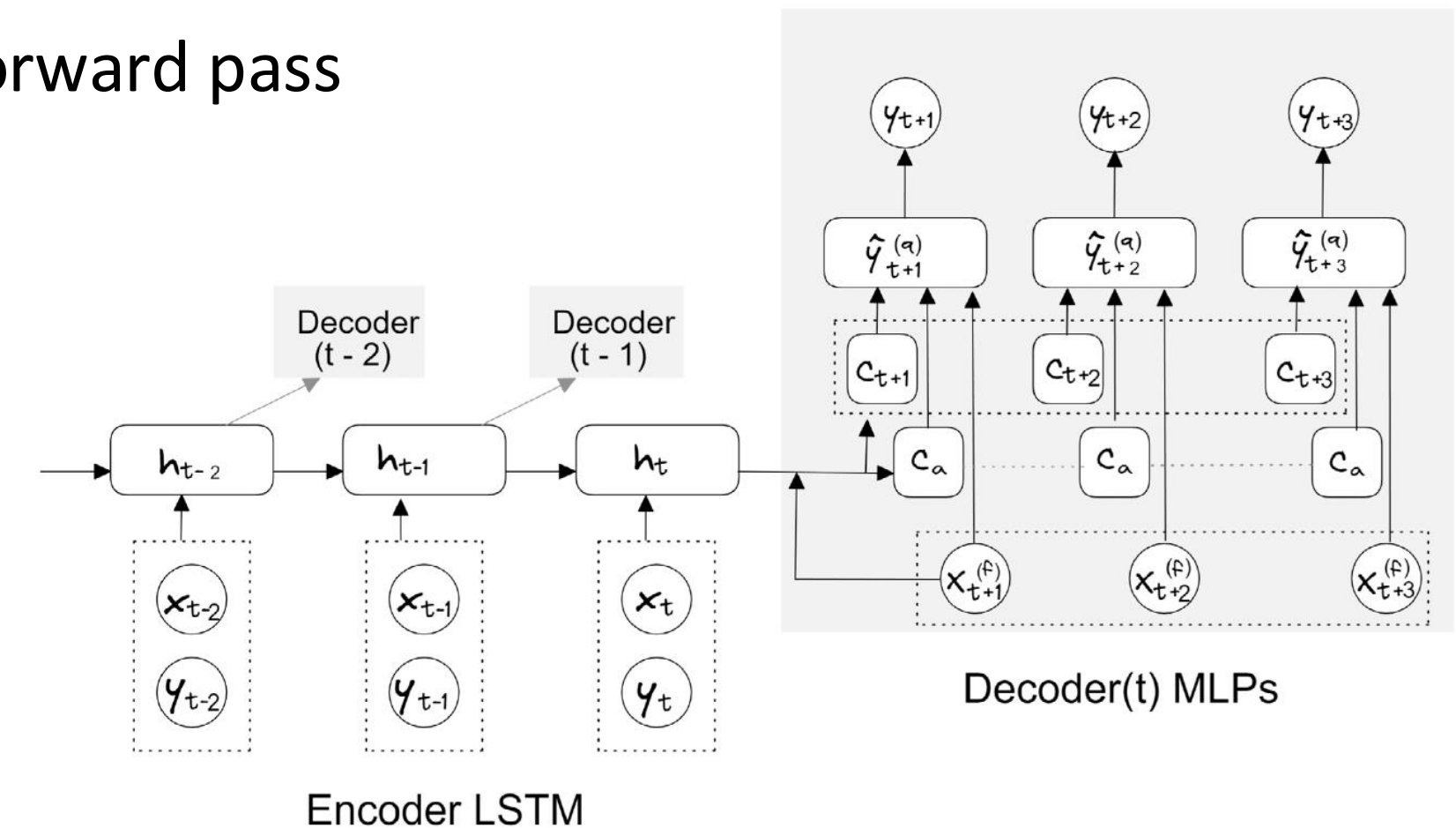
Handles uncertainty via quantile output

Efficient multi-step forecasting (vs. recursive methods)

Well-suited for demand forecasting, energy, finance

- **Output:**

Predicts a distribution range for each future step (e.g., 10th, 50th, 90th percentile)



RNN and Seq2Seq

- **Key Idea:**

Use an attention-based encoder–decoder to forecast multivariate time series by capturing temporal dependencies and feature interactions.

- **Architecture:**

Encoder: GRU/LSTM to encode historical multivariate time series

Decoder: GRU/LSTM with temporal attention

Attention helps the decoder focus on relevant past time steps

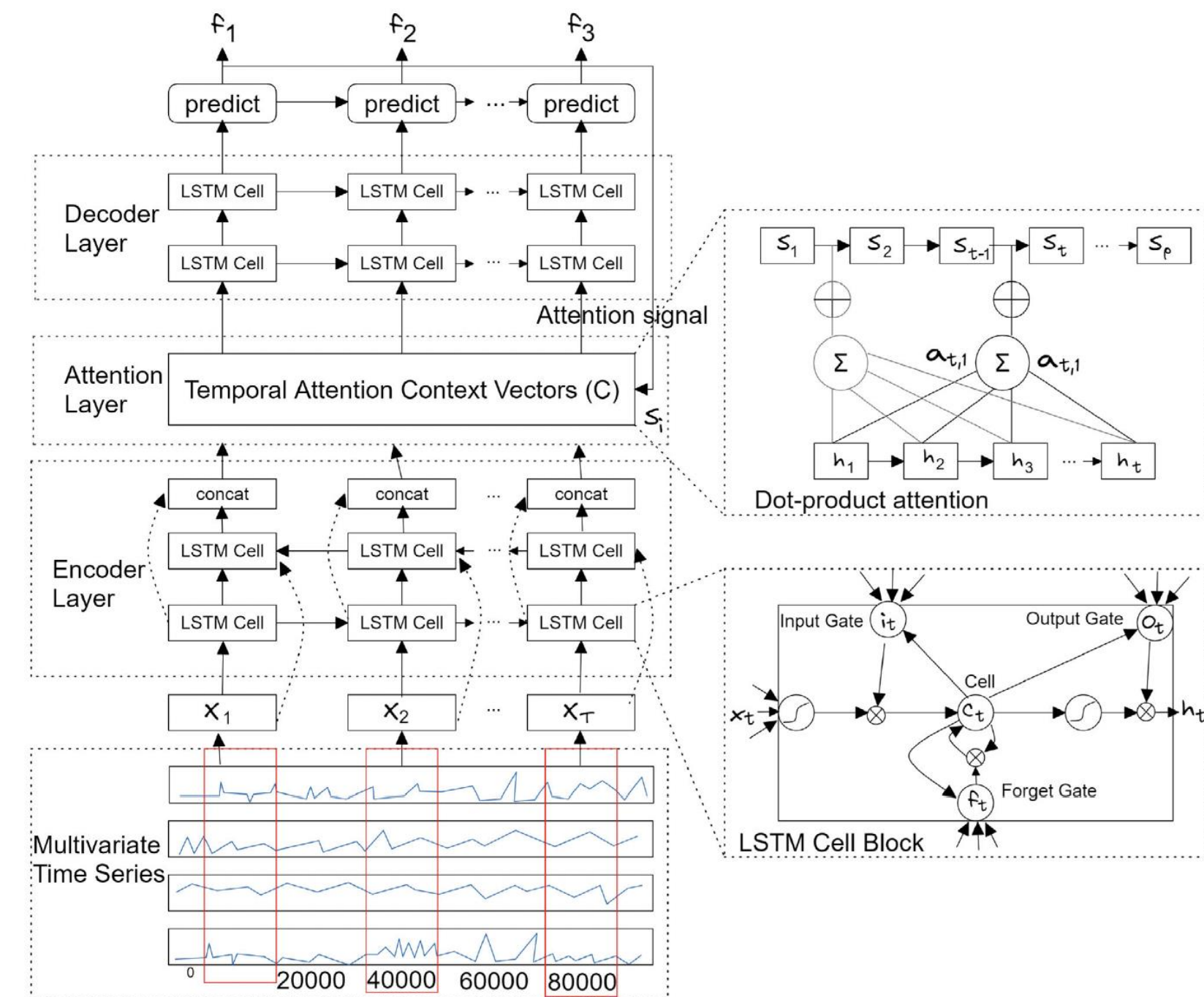
Outputs forecast for multiple time steps (multi-step prediction)

- **Benefits:**

Captures both short- and long-term dependencies

Enhances forecasting accuracy with temporal attention

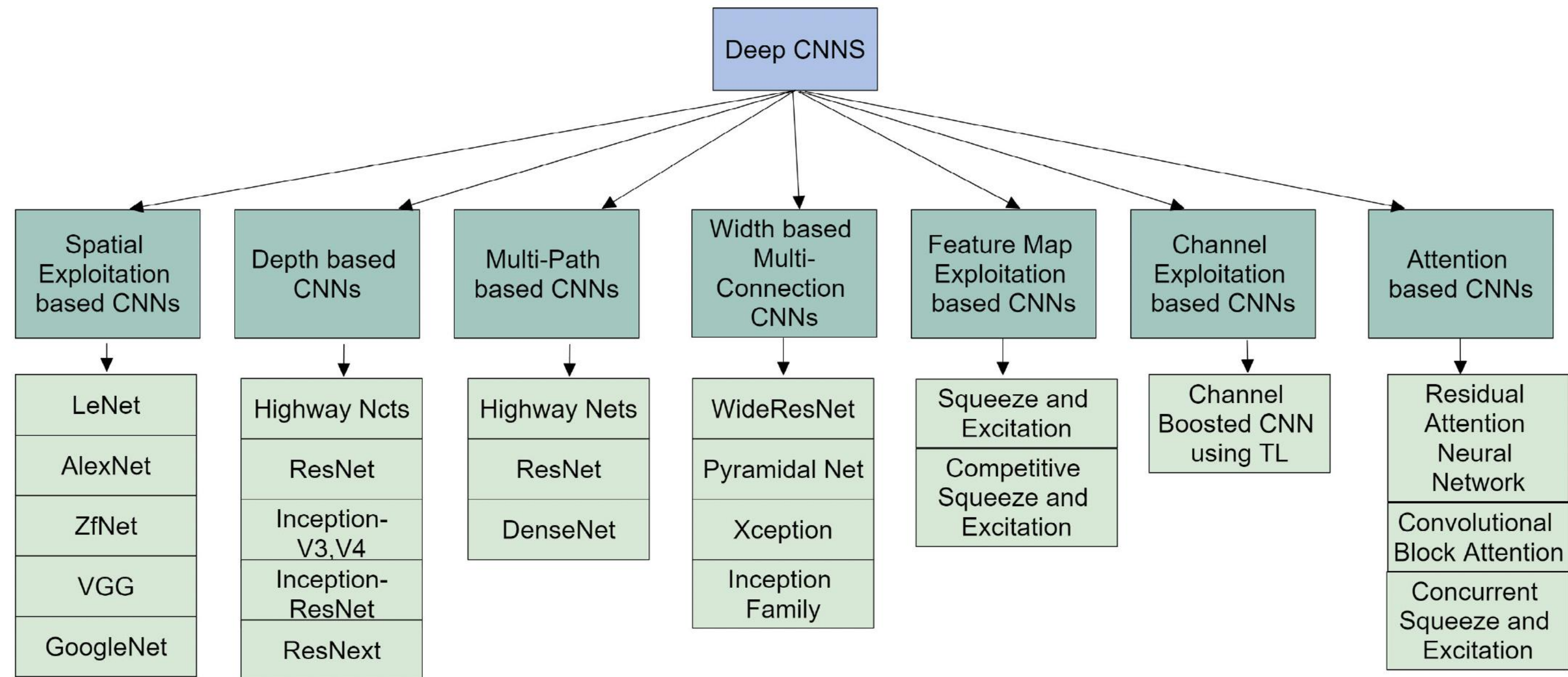
Generalizable to various MTS domains



RNN based models

Model	Year	Models Applied	Notes	Contributions
DA-RNN [15]	2017	Attention + LSTM	Select the most relevant feature variables and hidden states in the LSTM for both stages of the Attention network.	<ol style="list-style-type: none"> 1. DA-RNN model was proposed based on a dual-stage attention mechanism. 2. Captured long-term dependencies in sequences and select relevant input feature sequences for prediction.
MQRNN [16]	2017	LSTM + Encoder-Decoder + MLP	A multi-horizon LSTM with an encoder-decoder architecture can be used for cold-starting Single Target Forecasting (STF) tasks.	<ol style="list-style-type: none"> 1. An efficient training approach combined sequence neural networks with multi-view prediction. 2. A network substructure was designed to address a previously overlooked problem: how to interpret known future information.
Dilated RNN [64]	2017	Dilated connections + RNN	Dilations skip connections by: <ol style="list-style-type: none"> 1. capturing long-term and complicated dependencies; 2. addressing the gradient disappearance and explosion issue. 	<ol style="list-style-type: none"> 1. Introduced a new dilated recurrent skip connection as the core component of the architecture. 2. By incorporating multiple dilated recurrent layers with hierarchical dilations, the DilatedRNN effectively captured temporal dependencies across various dimensions and layers.
MTSMFF [69]	2020	LSTM + Encoder-Decoder + Attention CNN	The Attention network selected the hidden states of BiLSTM.	<ol style="list-style-type: none"> 1. A novel time attention-based encoder-decoder model was applied to multi-step prediction tasks for multivariate time series. 2. A time attention mechanism was introduced between the encoder and decoder networks.
STAM [67]	2021	LSTM + Attention	STAM captured relevant variables at each time step.	<ol style="list-style-type: none"> 1. A new STAM architecture introduced a unique method for multi-step prediction in the realm of interpretability for multivariate time series issues. 2. By combining spatial and temporal attention mechanisms in a unified structure, it enabled an understanding of the impacts of both time and space. 3. Analysis of STAM complexity offered additional insights to enrich understanding.
PSO-Bi-LSTM [65]	2023	PSO+BI-LSTM	Bi-LSTM prediction model was fine-tuned with PSO (known for its quick convergence, resilience, and wide search capacity).	<ol style="list-style-type: none"> 1. Employed nonlinear change weights to enhance the convergence speed of the particle swarm algorithm. 2. Integrated the particle swarm algorithm with the Bi-LSTM algorithm, addressing the limitations of manual parameter selection in Bi-LSTM.

CNN based models



CNN based models

TCN: Temporal Convolutional Network

- **Key Idea:**

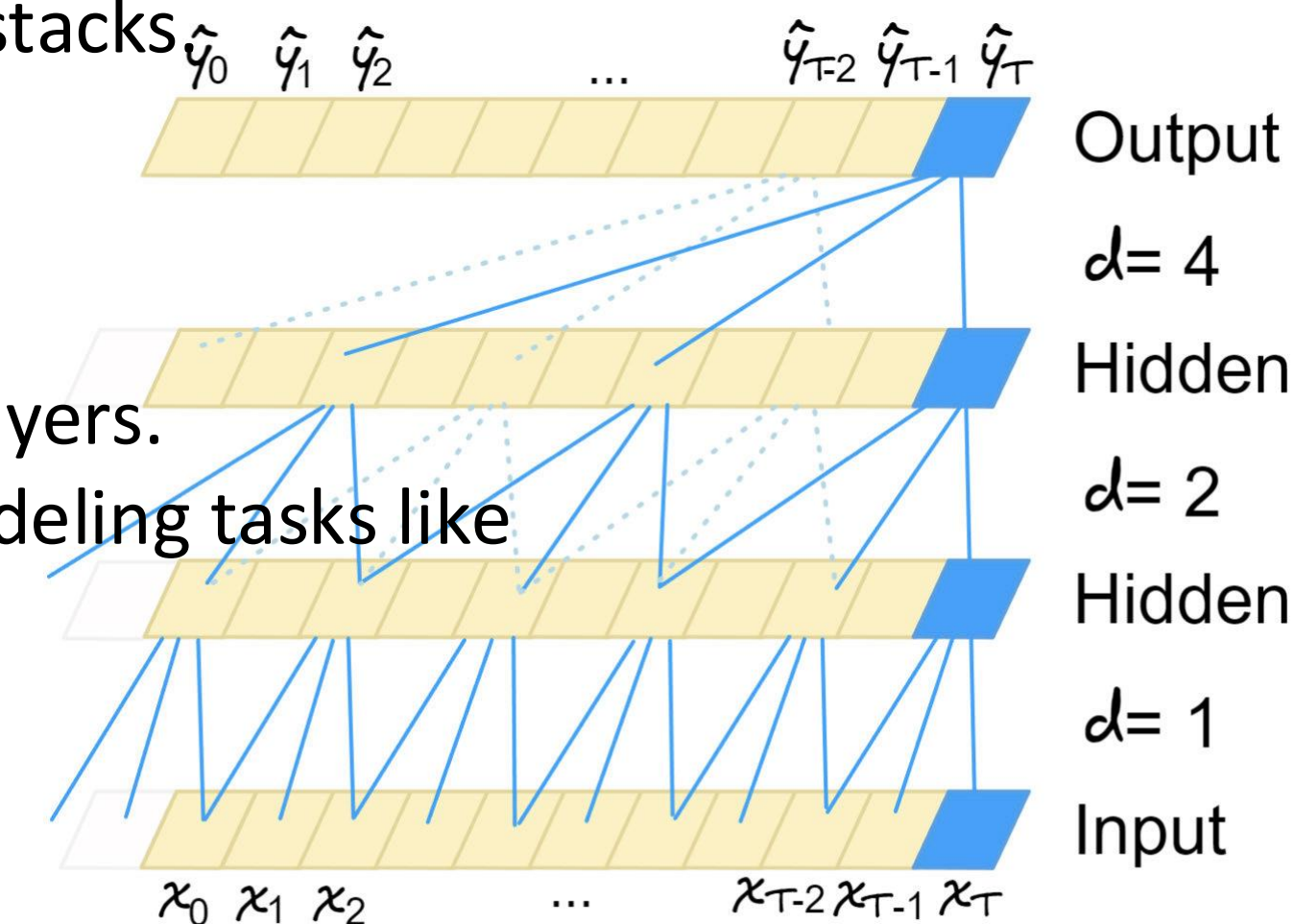
Use 1D dilated causal convolutions instead of RNNs to model sequential data efficiently and effectively, with better parallelism and longer effective memory.

Key Features:

- **Causal Convolutions:** Ensure predictions at time t depend only on inputs at or before t .
- **Dilated Convolutions:** Expand receptive field exponentially without deepening the network (see figure: dilation factors $d=1, 2, 4$).
- **Residual Connections:** Improve stability and learning in deep convolutional stacks

Benefits:

- **Parallelizable:** Unlike RNNs, TCNs can be fully parallelized during training.
- **Flexible receptive field:** Easily covers long-range dependencies with fewer layers.
- **Superior performance:** Outperforms RNNs/LSTMs on multiple sequence modeling tasks like audio synthesis, language modeling, and time series forecasting.



CNN based models

HyDCNN (Hybrid Dilated CNN)

Key Idea:

Combine dilated convolutions and VAR models in a hybrid residual framework to effectively capture both nonlinear local patterns and global temporal dependencies.

Model Highlights:

•Dilated CNN Blocks:

- Capture multi-scale **local temporal patterns** via expanding receptive fields.

•Hop VAR Module:

- Exploits long-range **linear dependencies** with vector autoregression.

•Hybrid Residual Learning:

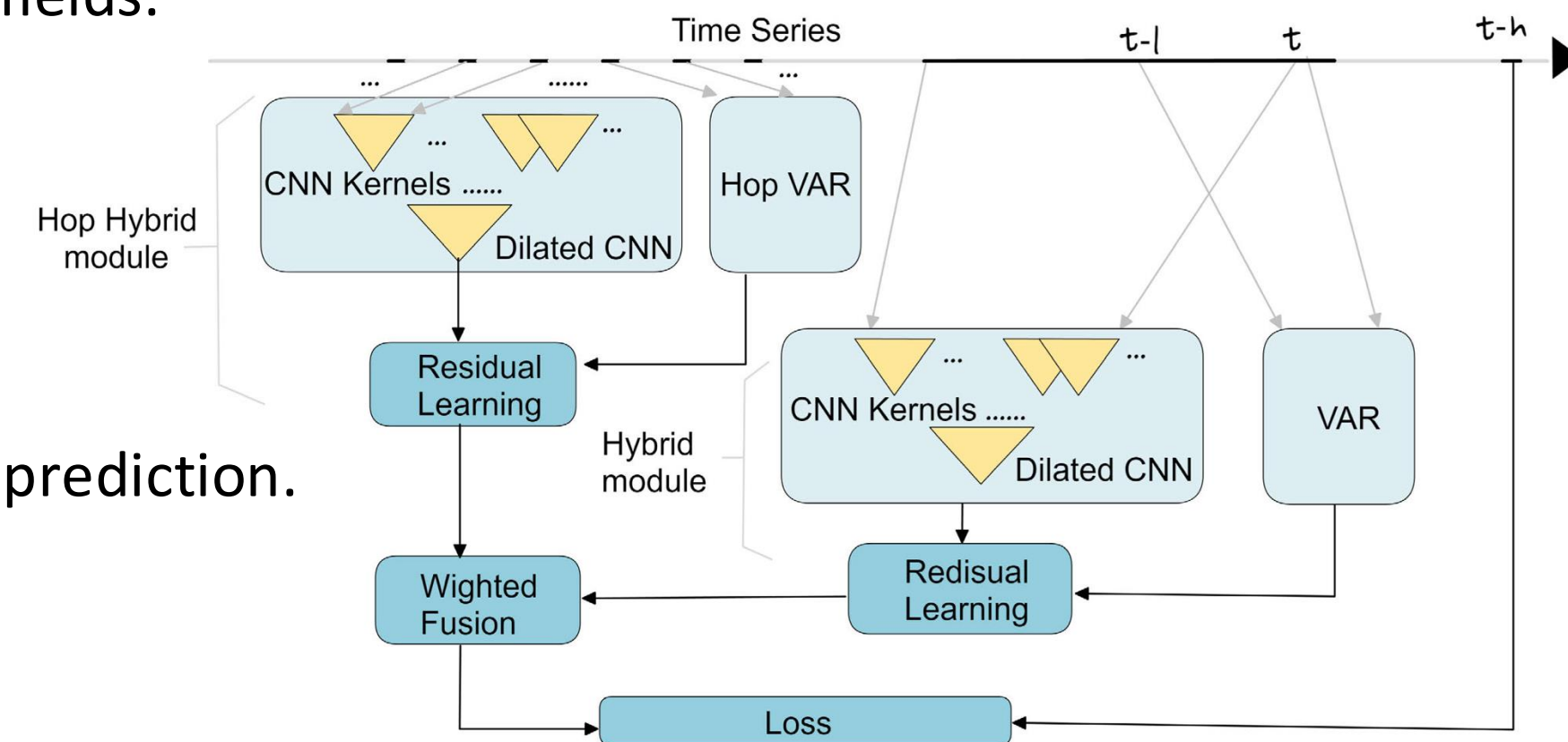
- Learns residuals from both CNN and VAR branches.

•Weighted Fusion Layer:

- Adaptively merges outputs from both CNN and VAR to form the final prediction.

Benefits:

- Effectively combines **nonlinear** and **linear** temporal modeling.
- Robust to time series with mixed patterns (e.g., sharp trends + smooth components).
- Outperforms traditional CNN, VAR, and RNN baselines on several forecasting benchmarks.



CNN based models – CNN and Attention

DSANet: Dual Self-Attention for MTS Forecasting

Key Idea:

DSANet introduces a dual self-attention mechanism to model both global and local temporal dependencies in multivariate time series data, combining convolutional encoders and attention modules for high interpretability and accuracy.

Model Highlights:

Local Temporal Convolution:

- Captures short-term dependencies via convolution + pooling.

Global Temporal Convolution:

- Captures long-range patterns with broader receptive field.

Self-Attention Module:

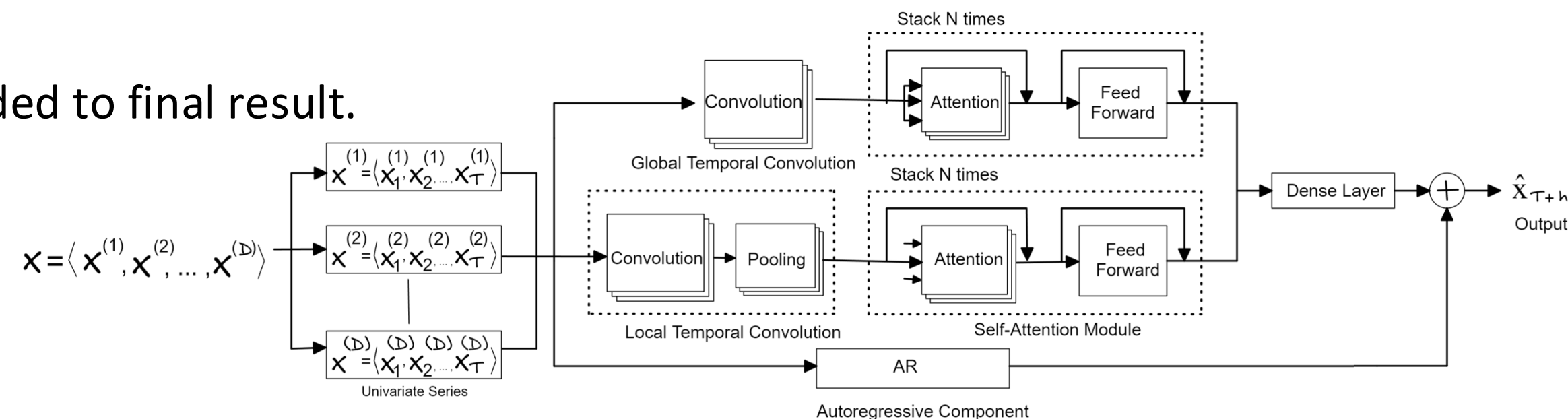
- Applied in both paths to capture inter-time-step and inter-feature relationships.

Autoregressive Component (AR):

- Models linear dependencies for stability.

Fusion:

- Dense layer integrates all components; AR output is added to final result.



CNN based models

DSANet: Dual Self-Attention for MTS Forecasting

Key Idea:

SCINet introduces a **hierarchical sample-wise interaction mechanism** for time series forecasting, decomposing the input sequence into **even and odd components** and modeling their **sample-level interactions** through convolutional structures.

Model Highlights:

SCI-Block (Left):

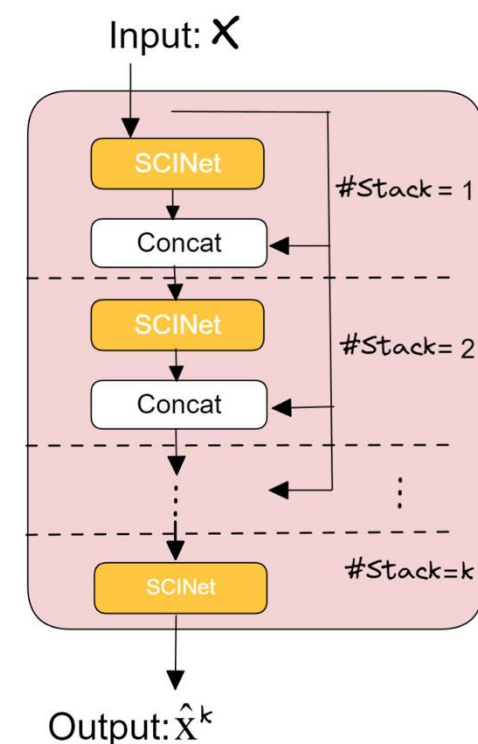
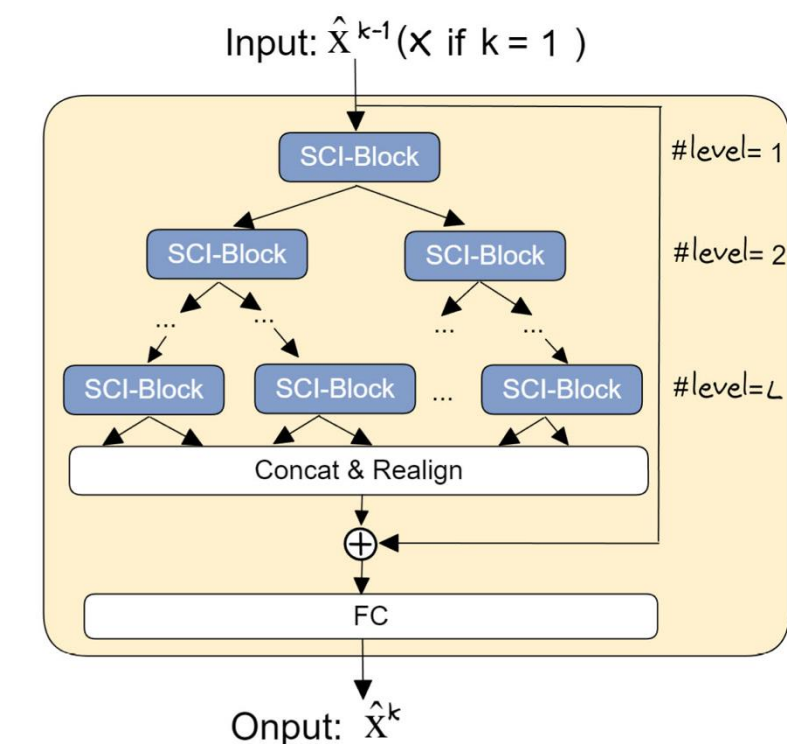
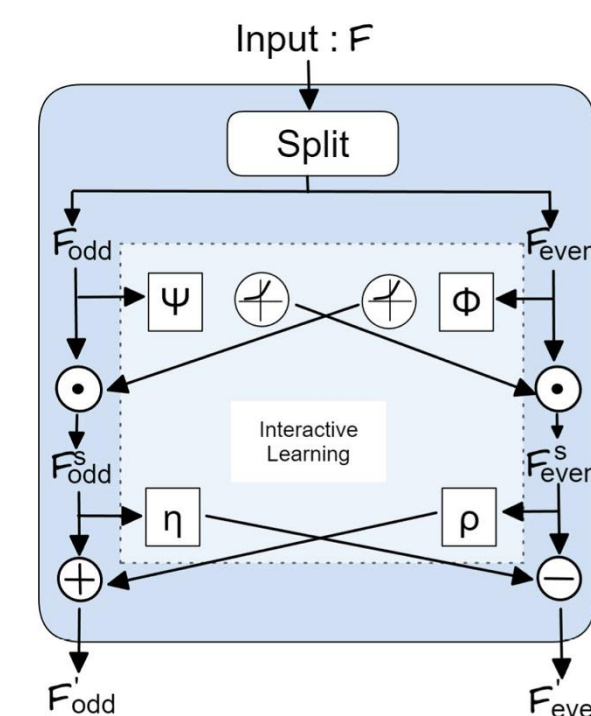
- Splits the input into **odd and even** samples.
- Learns **inter-sample dependencies** via a dual convolutional mechanism.
- Performs **interactive learning** between split branches using convolution operators
- Merges outputs via addition/subtraction to form transformed even/odd components.

Hierarchical SCI Module (Middle):

- Stacks SCI-Blocks in a **tree-like structure** to progressively model complex patterns.
- Outputs are concatenated and realigned for further processing.

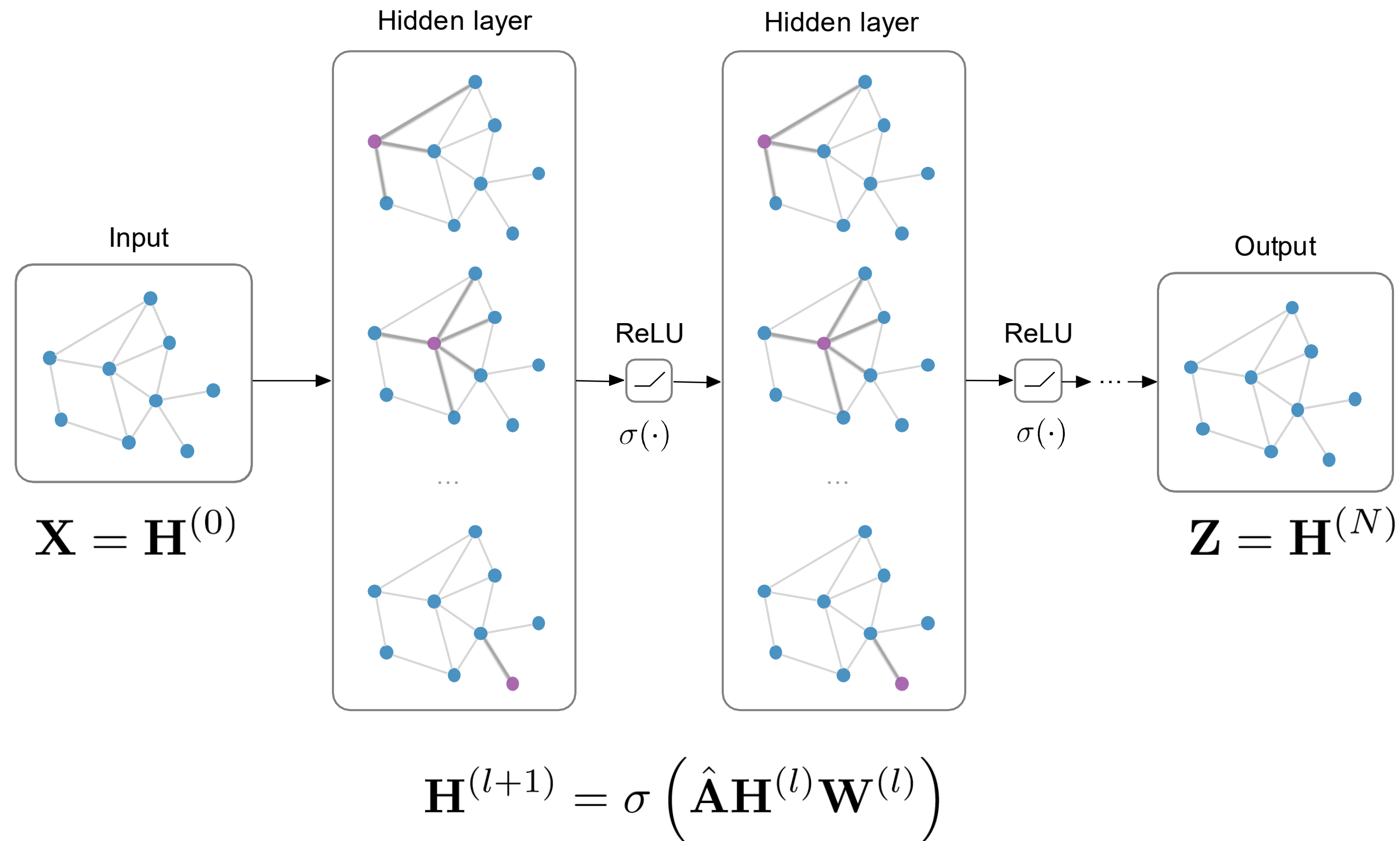
SCINet Stacking (Right):

- Multiple SCINet modules are **stacked** to enhance representation depth.
- Uses intermediate concatenation and residual pathways to preserve information flow.



GNN based models

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



GNN based models

DMSTGCN

- **Key Idea:**

Learn both dynamic spatial dependencies and multi-granular temporal patterns for accurate traffic speed forecasting.

- **Architecture:**

Dynamic Spatial Graph: Learns time-varying spatial connectivity using attention-based graph learning

Temporal Encoder: Combines:

Short-term (recent) trends

Periodic (e.g., daily/weekly) patterns

Long-term dependencies

Uses Gated Temporal Convolution + Graph Convolutional Networks

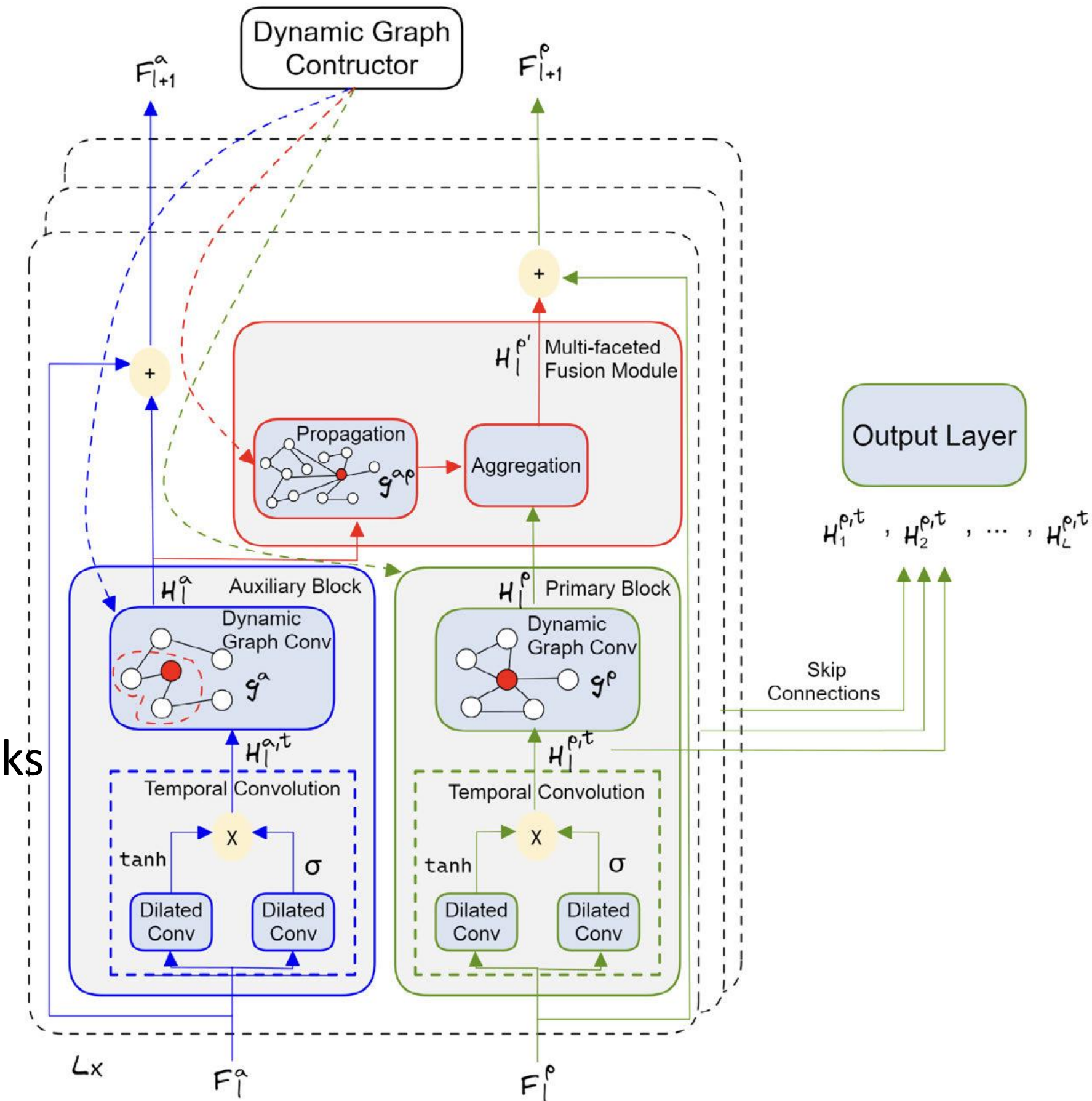
Output: Multi-step traffic speed predictions at multiple locations

- **Benefits:**

Captures non-stationary spatial dynamics

Models multi-scale temporal behavior

Outperforms static GCN-based methods in traffic forecasting



GNN based models

AutoSTG

Key Idea:

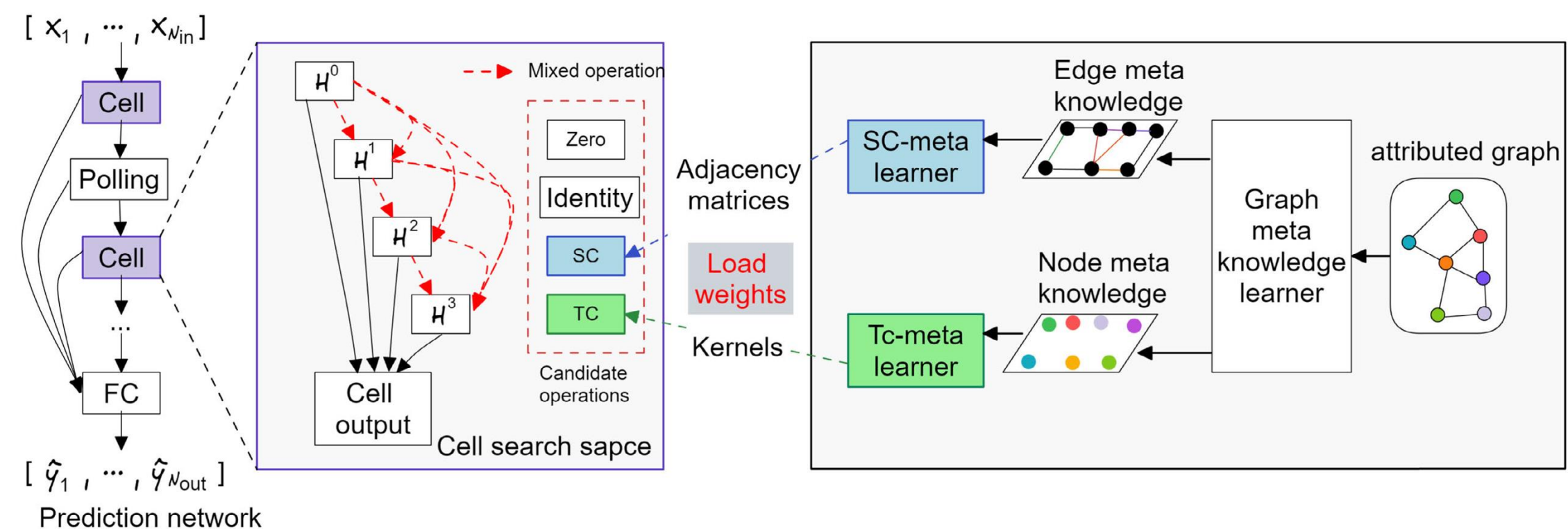
Automatically discover optimal spatiotemporal graph neural network architectures for time series forecasting using Neural Architecture Search (NAS).

Architecture:

- Search Space: Encodes design choices for:
 - Spatial modules (e.g., GCN, attention-based GNN)
 - Temporal modules (e.g., GRU, TCN, self-attention)
 - Fusion strategies for combining spatiotemporal features
- Uses Differentiable NAS to efficiently search
- Produces a customized model architecture

Benefits:

Data-specific architecture without manual design
Outperforms handcrafted STGNNs
(e.g., STGCN, DCRNN)
Scalable to large graphs and long horizons



GNN based models

MTGNN

- **Key Idea:**

Learn both temporal dynamics and implicit variable dependencies using a graph neural network (GNN) framework for multivariate time series forecasting.

- **Architecture:**

1. Graph Learning Module:

Learns a dynamic graph structure between variables (no predefined adjacency matrix needed)

2. Graph Convolution:

Captures spatial (cross-variable) dependencies

3. Gated Temporal Convolution:

Captures temporal dependencies in each variable

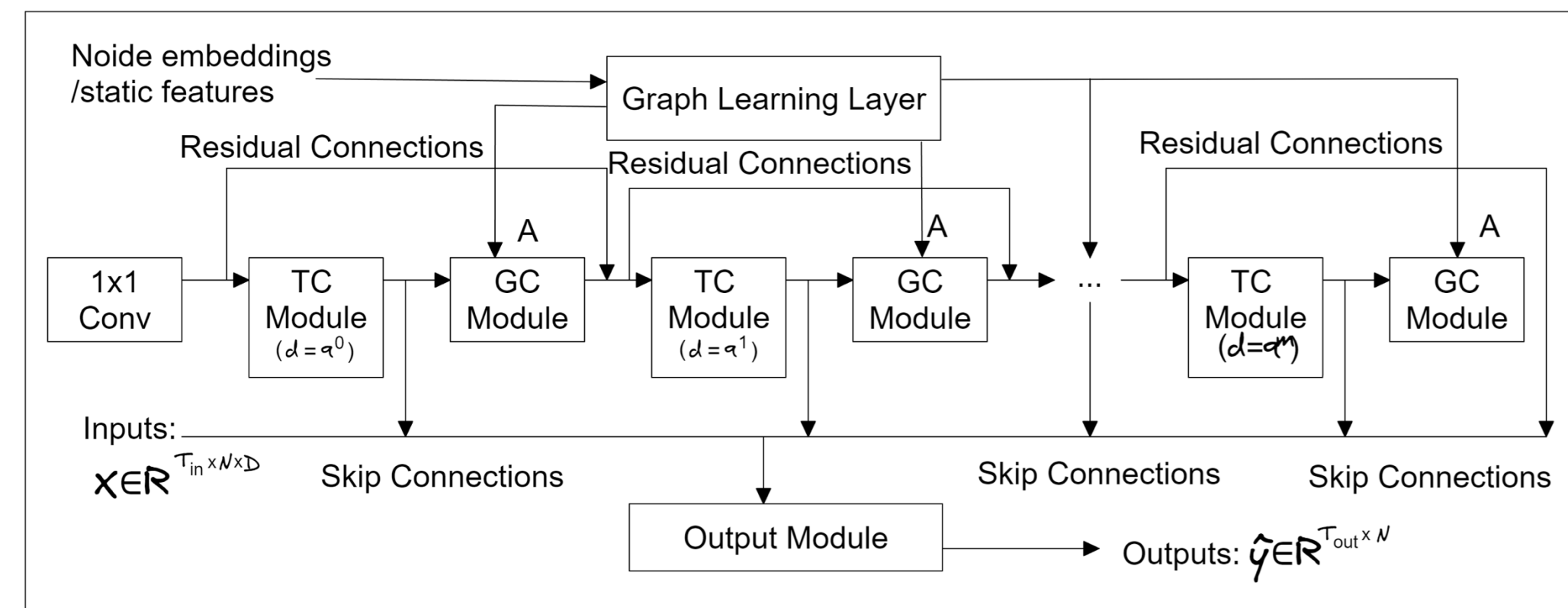
Jointly trained for end-to-end forecasting

- **Benefits:**

Models hidden relationships among variables

Scalable to high-dimensional time series

State-of-the-art performance on benchmarks (traffic, energy, finance)



GNN based models

REST

- **Key Idea:**

Design a reciprocal framework that couples spatial and temporal dependencies bidirectionally to improve spatiotemporal forecasting.

- **Architecture:**

1. Spatial Module:

Learns spatial relationships conditioned on temporal information

2. Temporal Module:

Learns temporal dynamics conditioned on spatial information

3. Reciprocal Learning:

Both modules refine each other iteratively

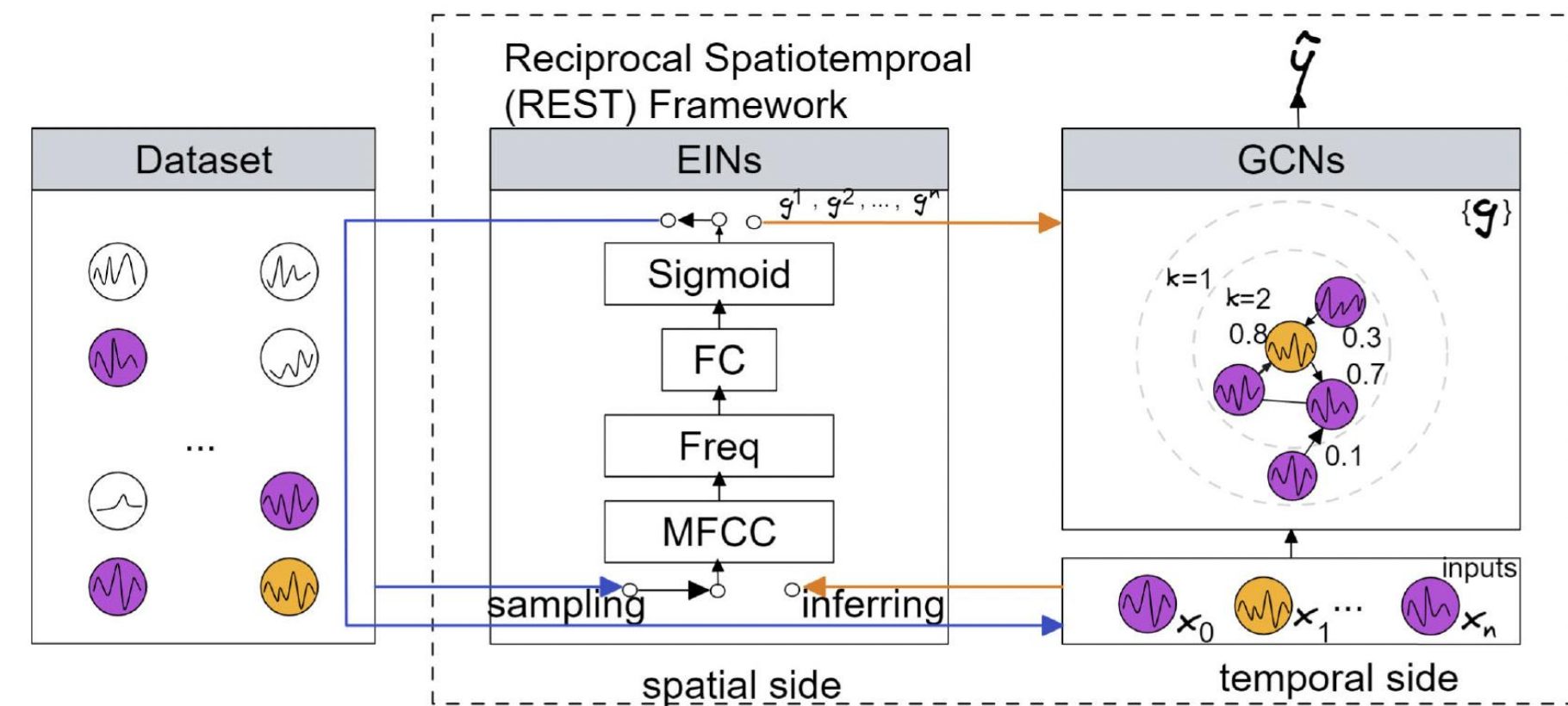
Achieves spatiotemporal mutual enhancement during training

- **Benefits:**

Stronger coupling between space and time features

Captures complex mutual dependencies (vs. simple stacking)

Boosts accuracy on dynamic forecasting tasks



GNN based models

TPGNN

- **Key Idea:**

Model complex temporal dependencies in multivariate time series by introducing polynomial-based temporal graphs into GNNs.

- **Architecture:**

Temporal Polynomial Graph (TPG):

Each time series variable forms a graph where edges are weighted polynomials of time lag

Graph Convolution on TPG:

Captures both short- and long-term temporal dependencies explicitly

Joint Learning:

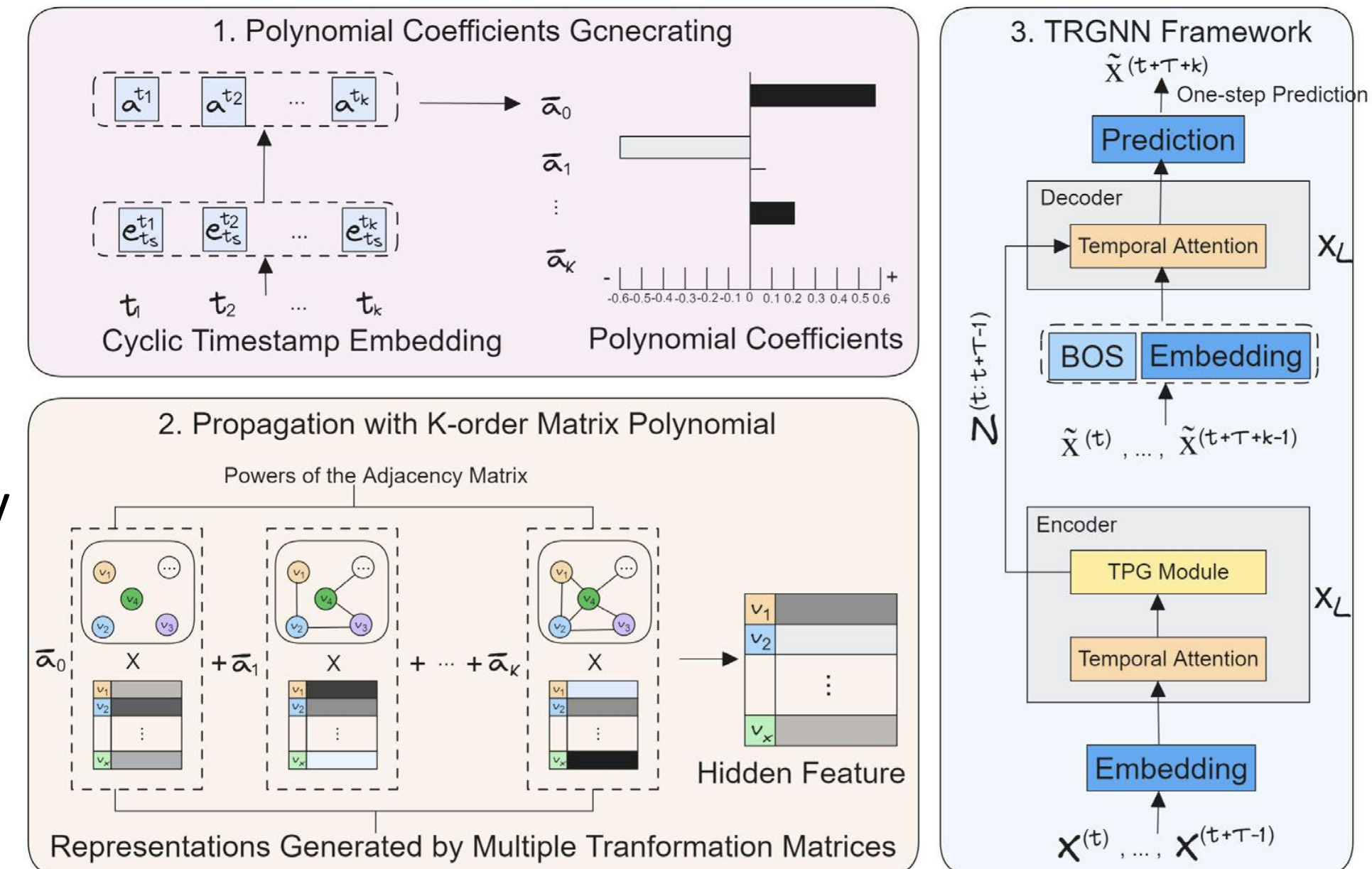
Learns graph structure and forecasting model simultaneously

- **Benefits:**

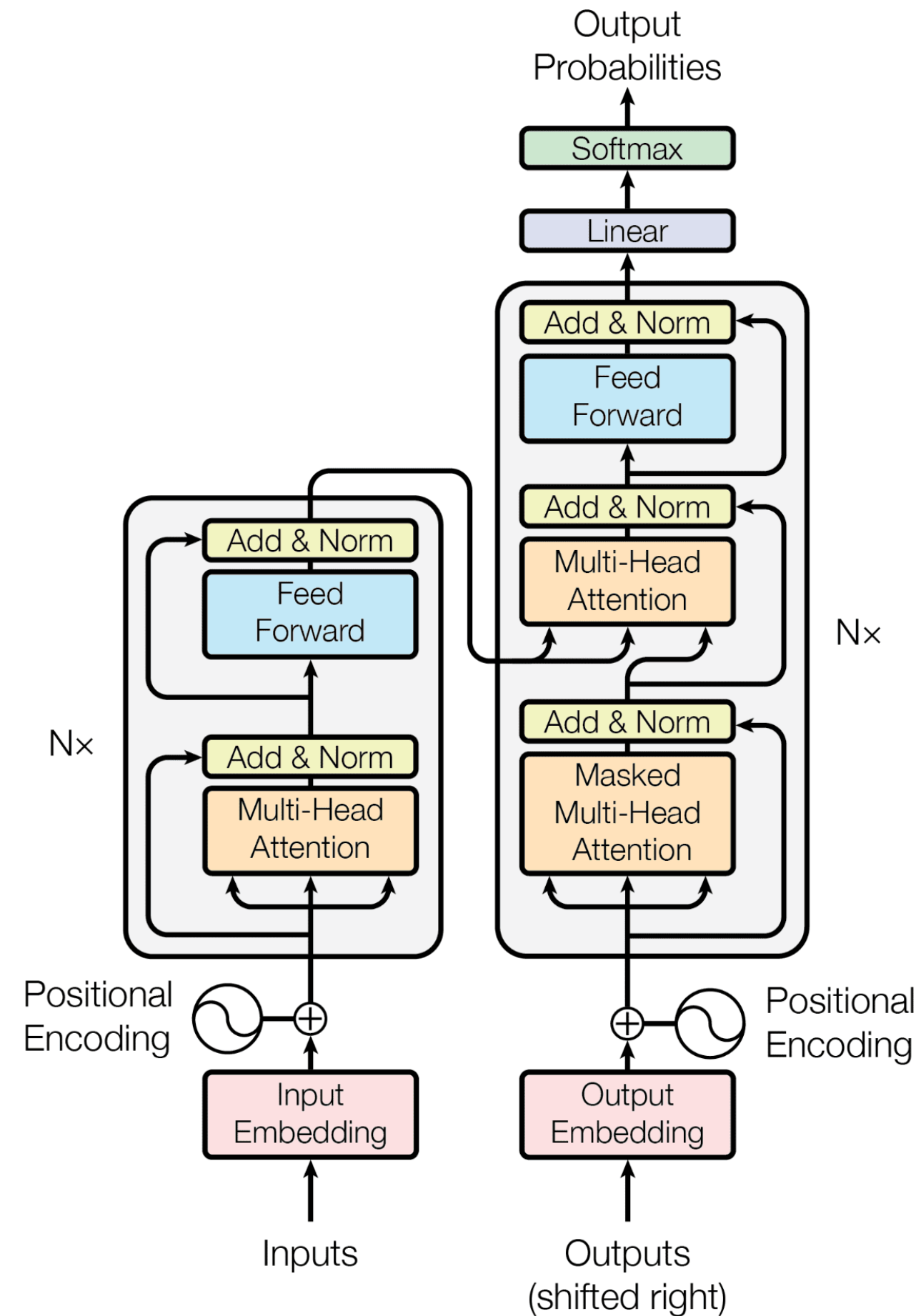
Models non-linear time-lag relationships naturally

Better captures long-range temporal patterns

Outperforms existing spatiotemporal GNNs on multiple MTS benchmarks



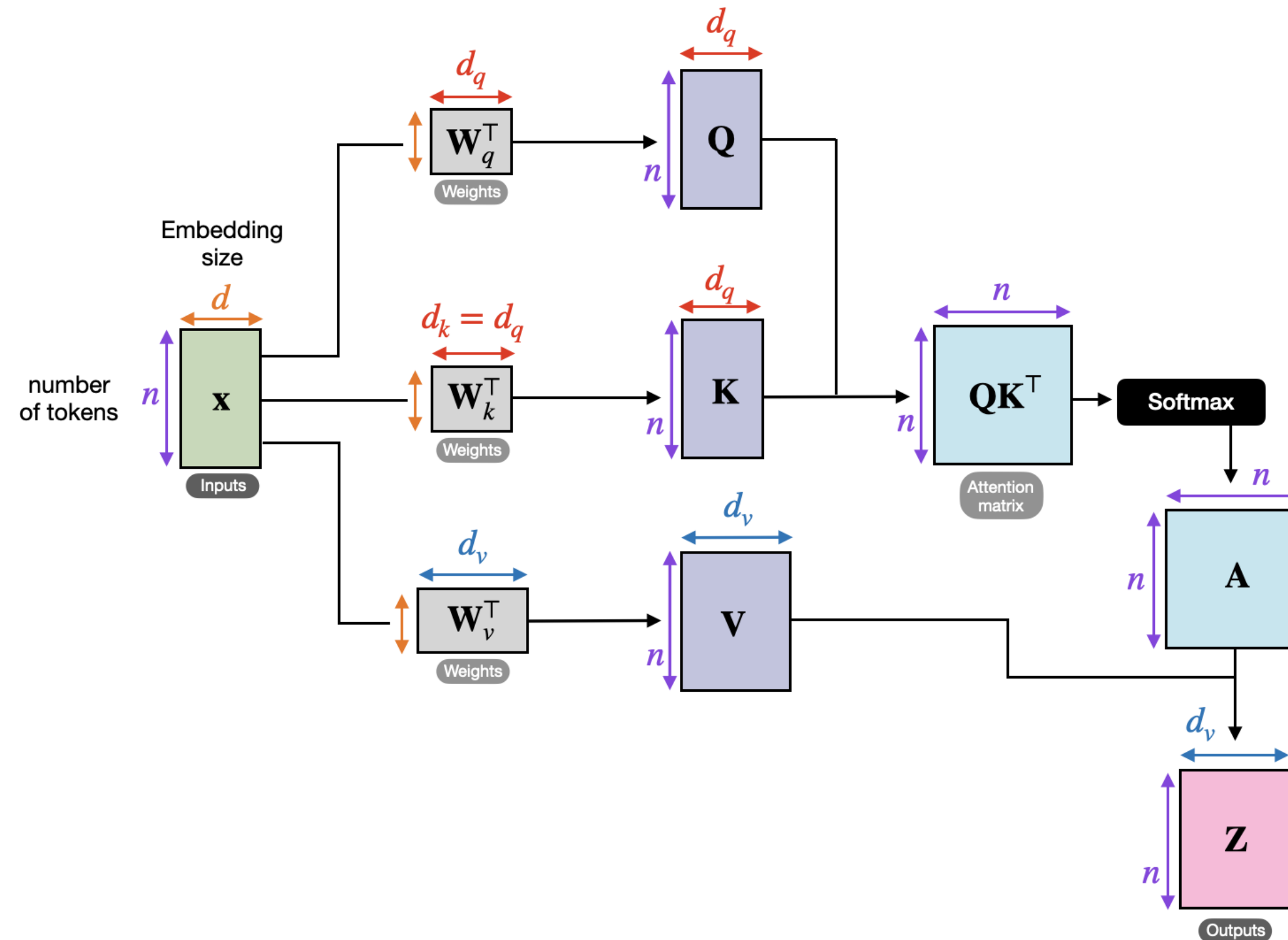
Transformer based models



Dynamic Graph Transformer for Brain Disorder Diagnosis

Transformer based models

Scaled Dot-Product Attention mechanism used in Transformers:



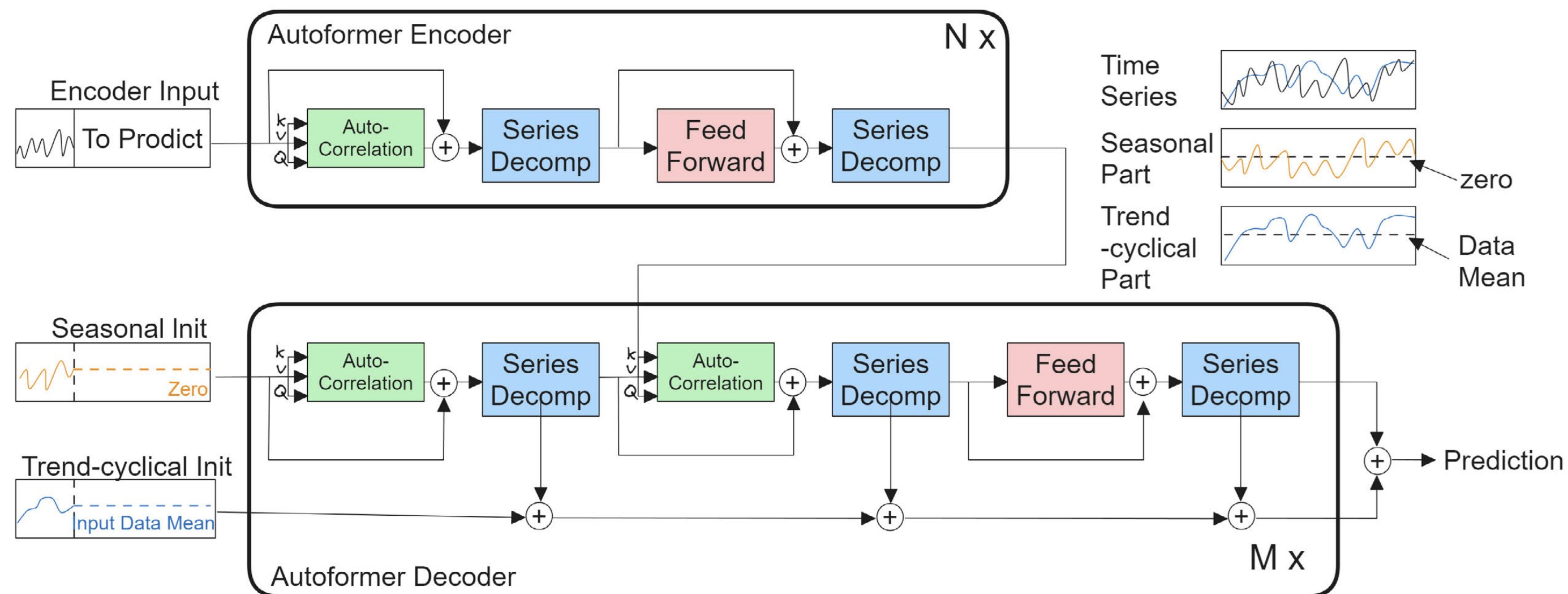
Transformer based models - Autoformer

Design a Transformer specialized for long-term time series forecasting by combining series decomposition and auto-correlation-based attention.

- Series Decomposition:
Decomposes input into trend and seasonality components
- Auto-Correlation Attention:
Focuses on periodic dependencies by aligning similar subseries across time
- Decoder:
Predicts trend directly and reconstructs seasonal part from autocorrelated patterns

Benefits:

- Reduces forecasting error accumulation over long horizons
- More efficient and interpretable than standard self-attention
- State-of-the-art on long-term forecasting benchmarks (ETT, Traffic, Electricity)



Transformer based models - Pyraformer

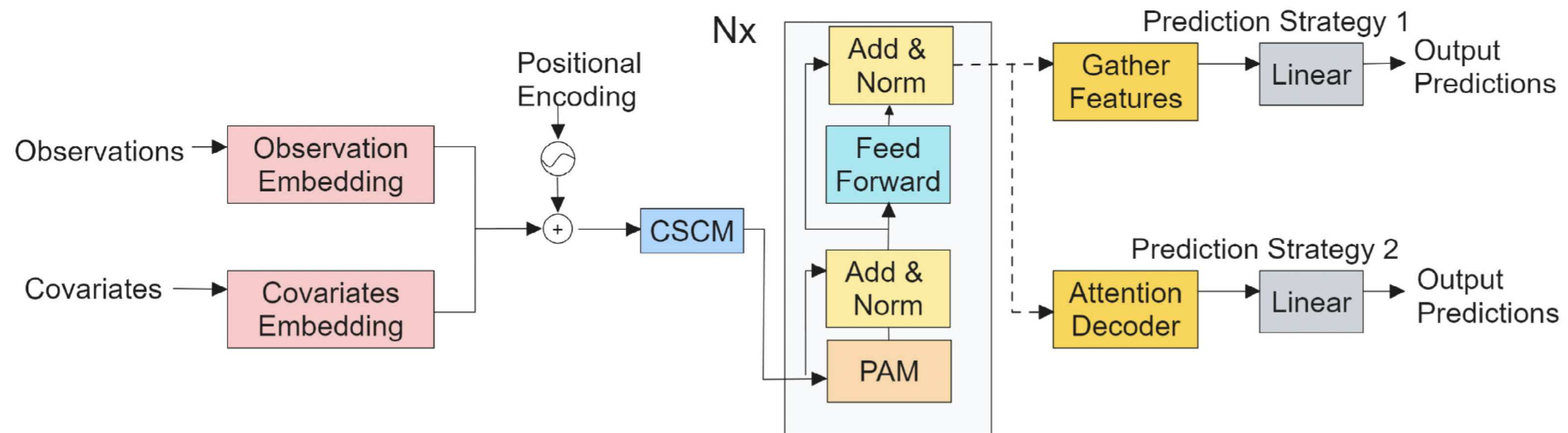
low-complexity Transformer for long-range time series by introducing **pyramidal attention** to capture hierarchical temporal dependencies efficiently

➤ Architecture:

- Pyramidal Attention:
 - Groups time steps into a hierarchical pyramid structure
 - Computes attention within local groups and across pyramid levels
- Sparse Attention Mechanism:
 - Reduces complexity from $O(n^2)$ to $O(n \log n)$
 - Models both short-term details and long-term trends effectively

➤ Benefits:

- Scalable to very long time series
- Efficient memory and computation usage
- Outperforms Informer and standard Transformer on long-range forecasting tasks



Transformer based models - non-stationary Transformer

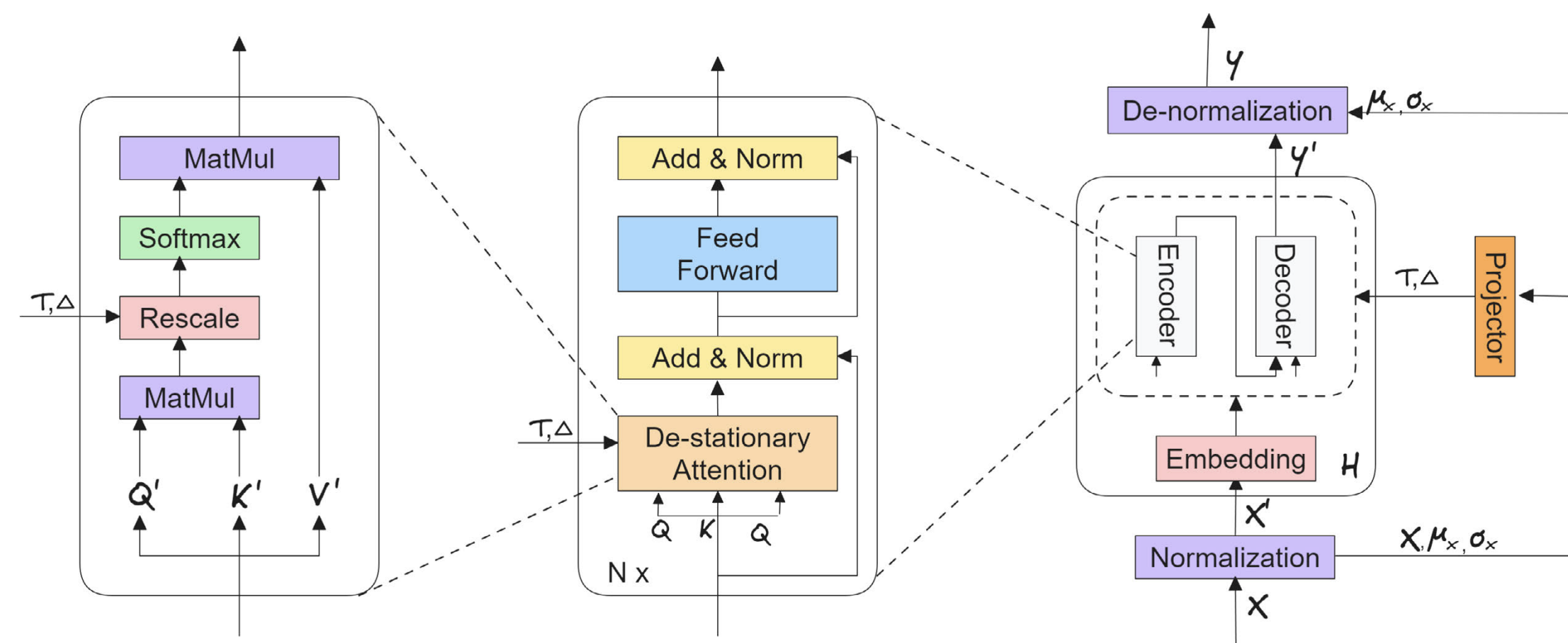
Address the challenge that real-world time series are **non-stationary** (i.e., changing distribution over time) by explicitly modeling stationary and non-stationary components in Transformers.

➤ **Architecture:**

- Decompose input sequences into:
 - Stationary part: stable trends/patterns
 - Non-stationary part: dynamic changes
- Two attention modules:
 - Stationary Attention: captures stable dependencies
 - Non-stationary Attention: models shifting patterns
- Combine outputs for final prediction

➤ **Benefits:**

- Better handles distribution shifts over time
- Improved long-term forecasting compared to Autoformer and Informer
- Adaptable to highly dynamic and irregular time series



Transformer based models - Informer

➤ Key Idea:

Improve Transformer efficiency for **long sequence time series forecasting** using **ProbSparse Attention** to focus only on the most relevant query-key pairs.

➤ Architecture:

• ProbSparse Attention:

Instead of full self-attention $O(n^2)$, it selectively attends to top queries with largest impacts

Reduces complexity to $O(n \log n)$

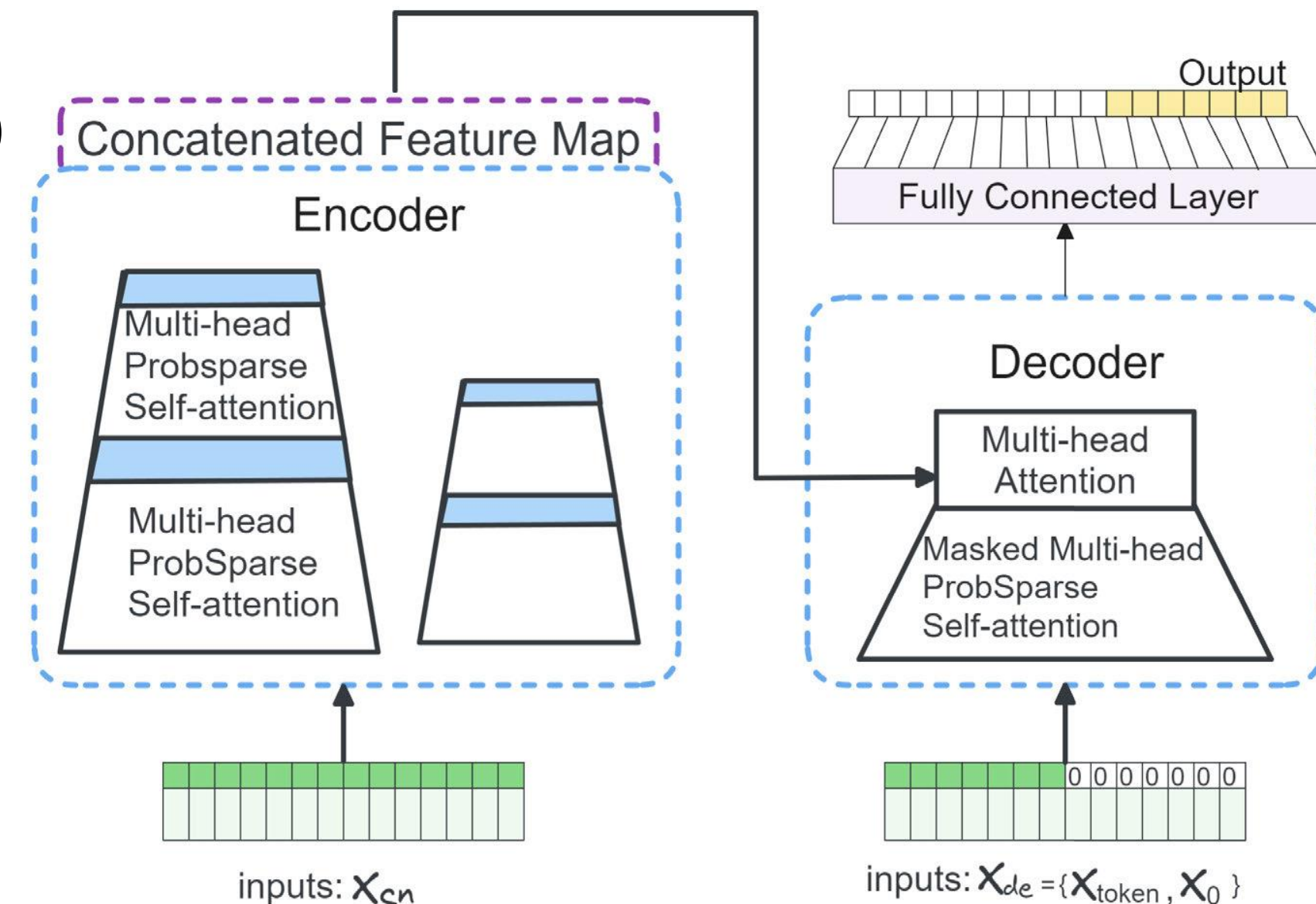
• Encoder-Decoder Structure:

Encoder compresses long input sequences

Decoder uses generative prediction (one forward pass for multiple steps)

➤ Benefits:

- Handles very long sequences efficiently
- Fast training and inference compared to vanilla Transformer
- Outperforms RNNs and vanilla Transformer on multivariate long-horizon forecasting



Transformer combined models - TFT

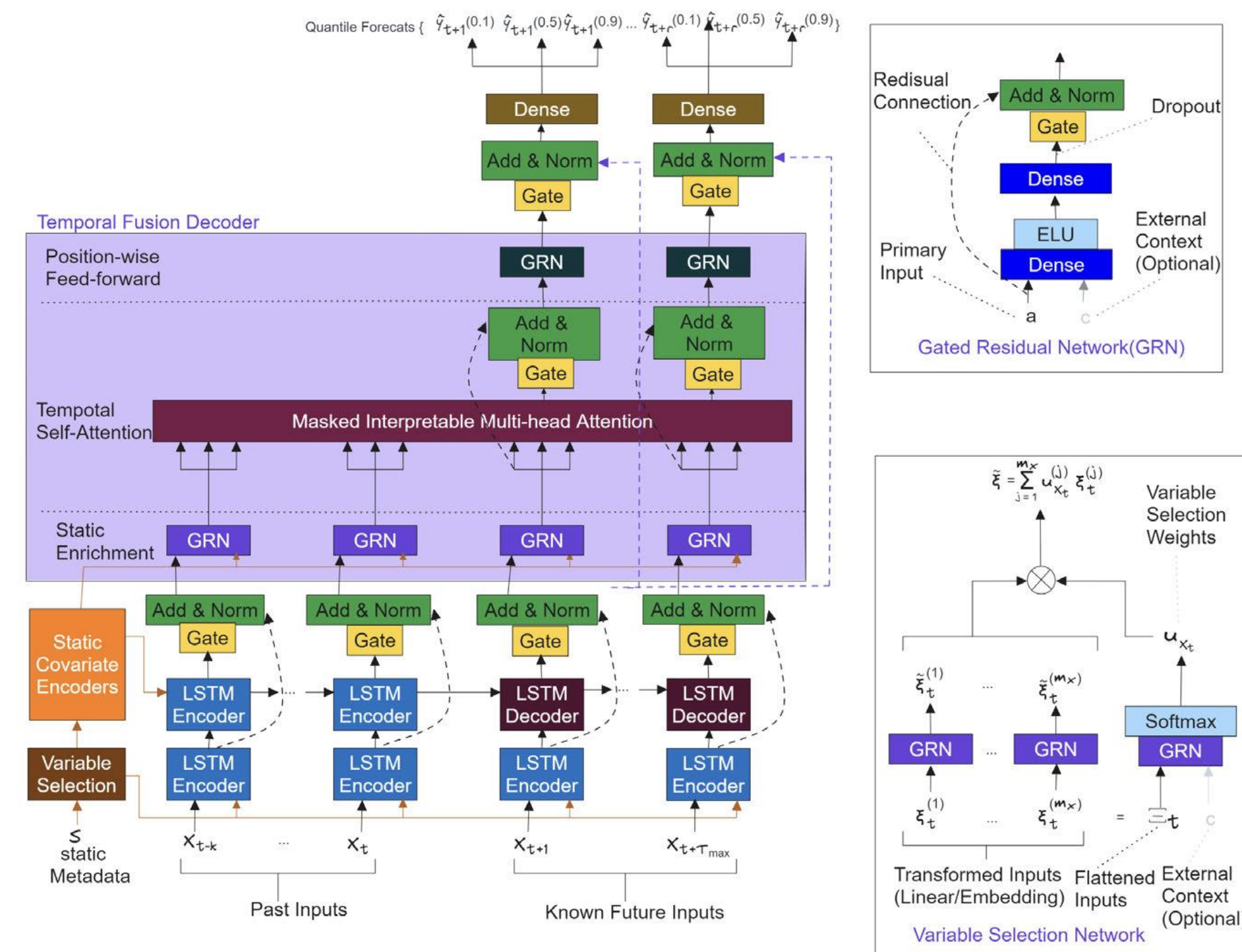
Temporal Fusion Transformer

➤ Key Idea:

Combine high forecasting accuracy with interpretability for multi-horizon time series prediction using a Transformer-based architecture with feature selection.

➤ Architecture:

- **Static Covariate Encoder:**
Learns embeddings for time-invariant features (e.g., location, ID)
- **Variable Selection Networks:**
Dynamically select important features at each time step
- **Gated Residual Network (GRN):**
Enhances representation learning and prevents overfitting
- **Multi-Head Attention:**
Focuses on temporal relationships for long-term prediction
- **Benefits:**
 - Interpretable: Identifies key time-varying and static features
 - Flexible: Handles static/dynamic inputs, missing data, and known future covariates
 - Strong performance on a wide range of real-world forecasting tasks



Transformer based models - FEDformer

➤ Key Idea:

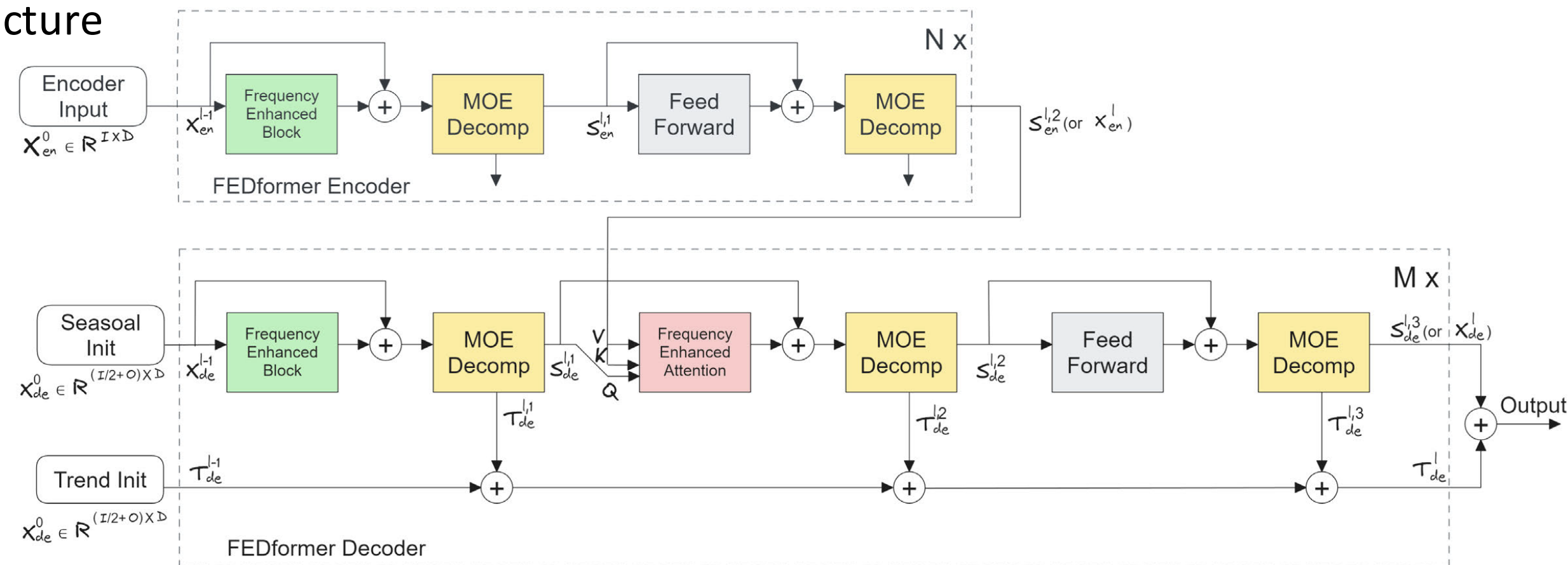
Forecast long-term time series by combining **series decomposition** and **frequency domain attention** to improve accuracy and efficiency.

➤ Architecture:

- Series Decomposition:
Separates inputs into trend and seasonality components using MOE Decomp (Mixture of Experts Decomposition)
- Frequency Enhanced Attention:
Models seasonal patterns directly in the Fourier frequency domain
- Trend Learning:
Trend component is updated separately at each layer

➤ Benefits:

- Reduces complexity by frequency-domain processing
- Better long-range forecasting than Autoformer and Informer
- Improved interpretability through decomposition structure



Complexity analysis of transformer-based models

Model	Training		Testing (steps needed)
	Time	Memory	
Transformer [40]	$O(L^2)$	$O(L^2)$	L
Informer [23]	$O(L \log L)$	$O(L \log L)$	1
Autoformer [24]	$O(L \log L)$	$O(L \log L)$	1
Pyraformer [25]	$O(L)$	$O(L)$	1
Fedformer [26]	$O(L)$	$O(L)$	1

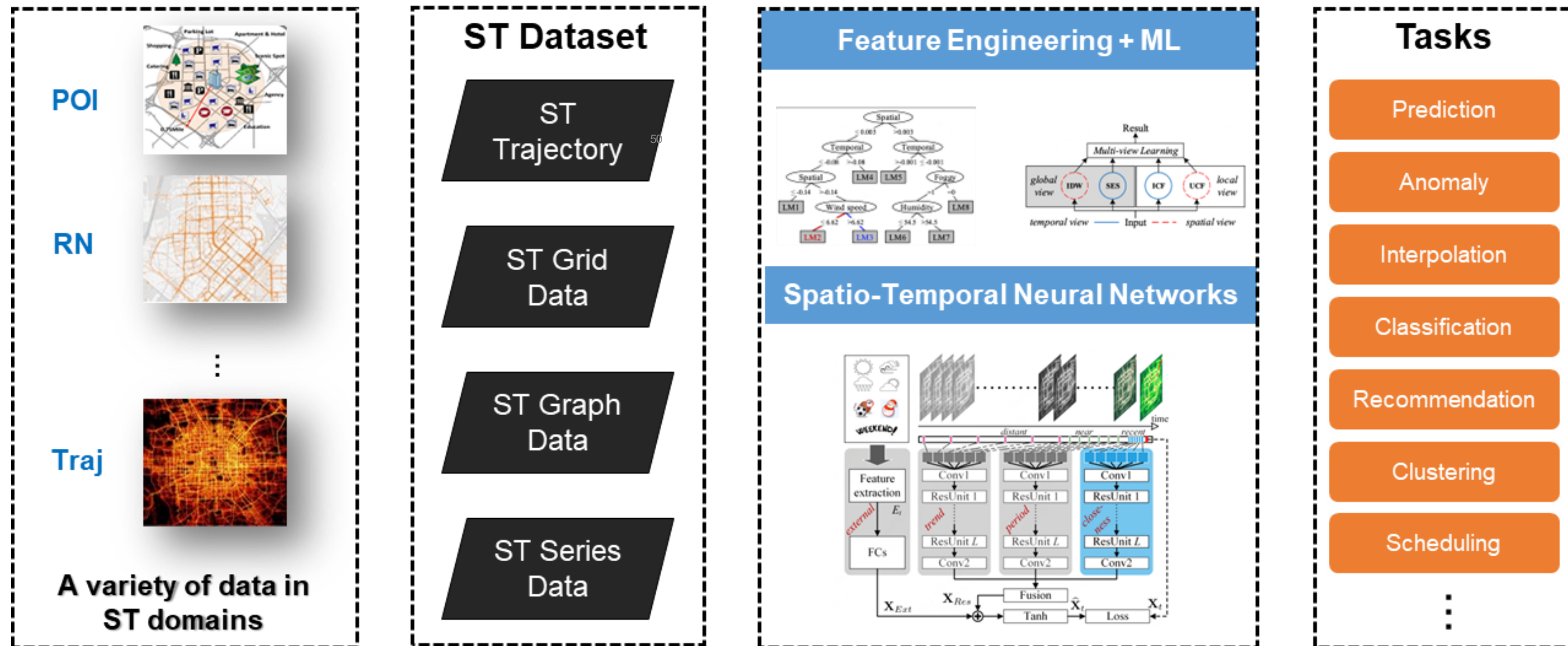
Effectiveness of Transformers in TS Forecasting

➤ Key Points:

- Challenge to Transformers:
A simple linear model was shown to outperform complex Transformers on forecasting tasks.
 - MTS-Mixers:
Inspired by vision mixers, replaces Transformer attention with full connections across time and channel dimensions.
Achieves better performance by learning on sub-sequences.
 - TiDE:
A fully MLP-based model (no attention, no RNNs, no CNNs).
Reached state-of-the-art results across multiple datasets, outperforming Transformers again.
- ## ➤ Insight:
- Recent work shows simple architectures (e.g., MLPs, mixers) can match or even surpass Transformers for time series forecasting.

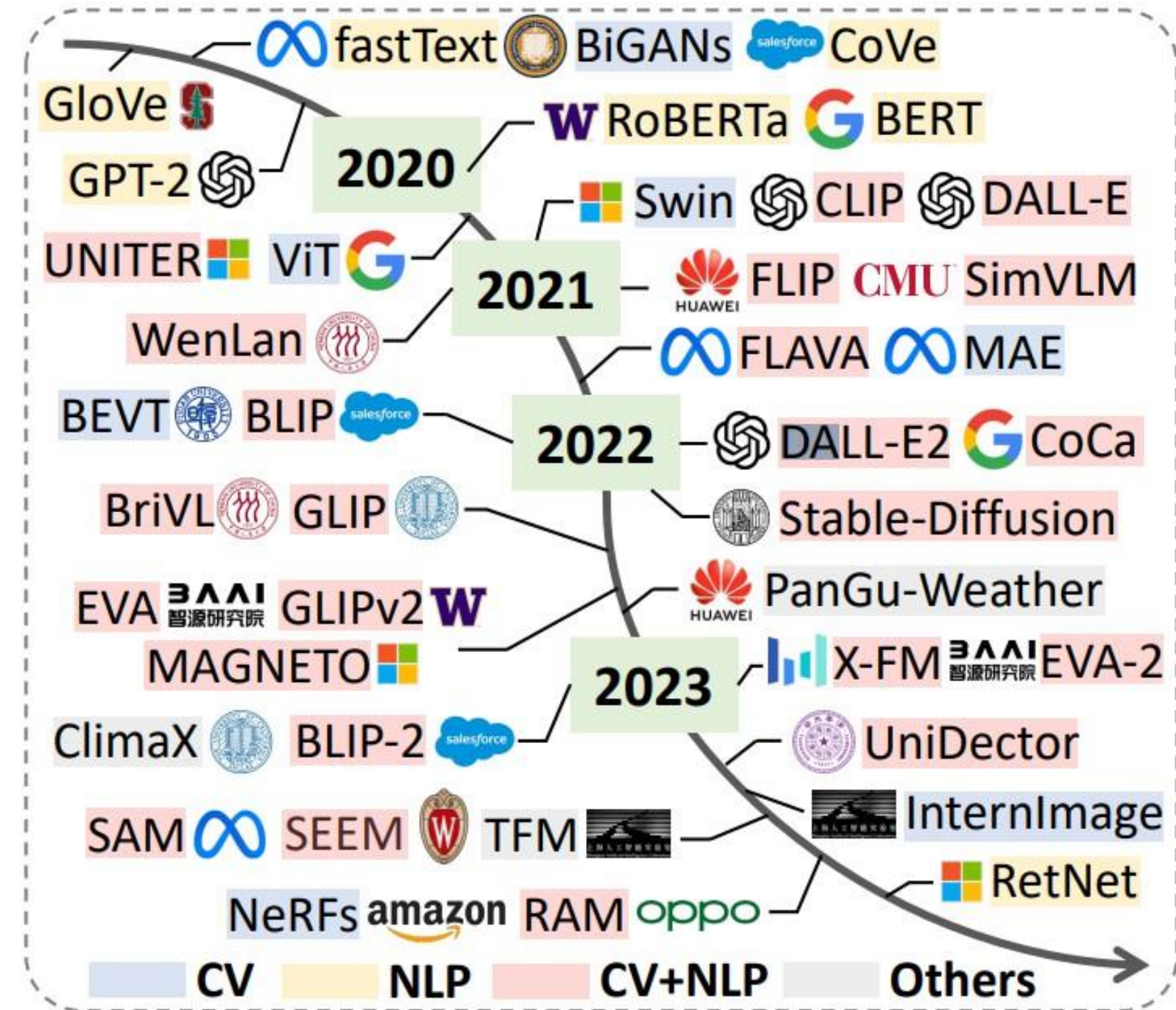
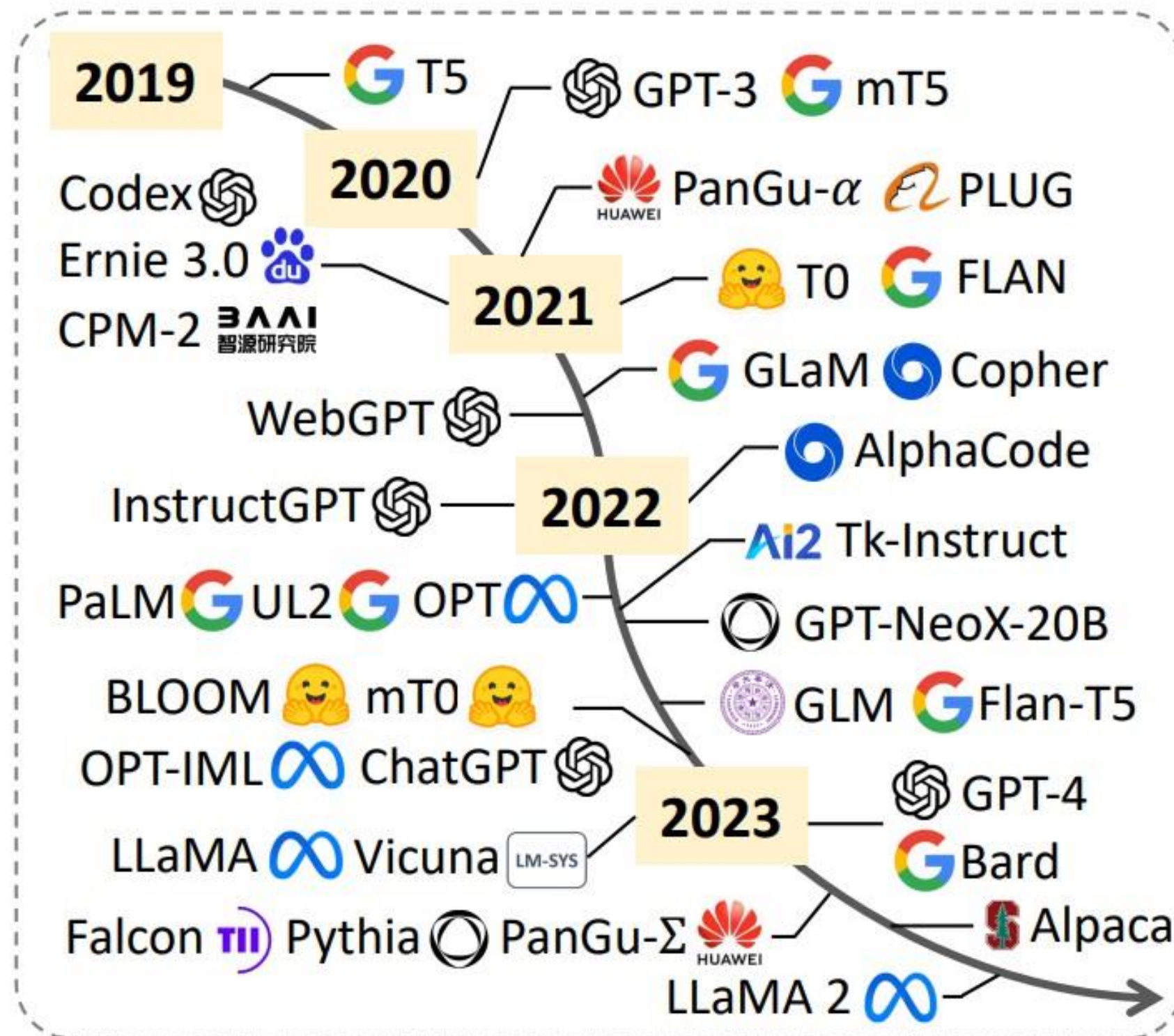
Limitations

- There is no free lunch
- Prior literature mostly concentrated on solving specific tasks



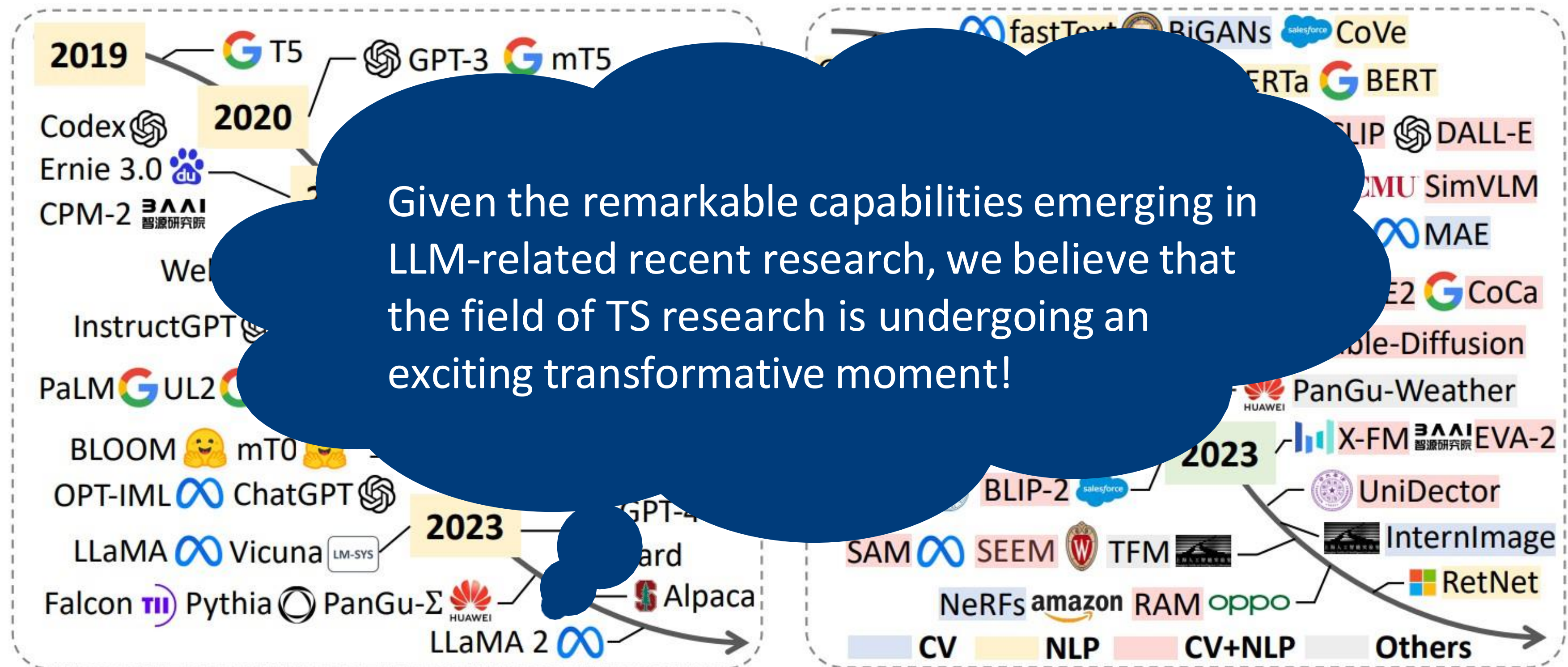
Roadmap of LLMs

- LLMs and Foundation Models



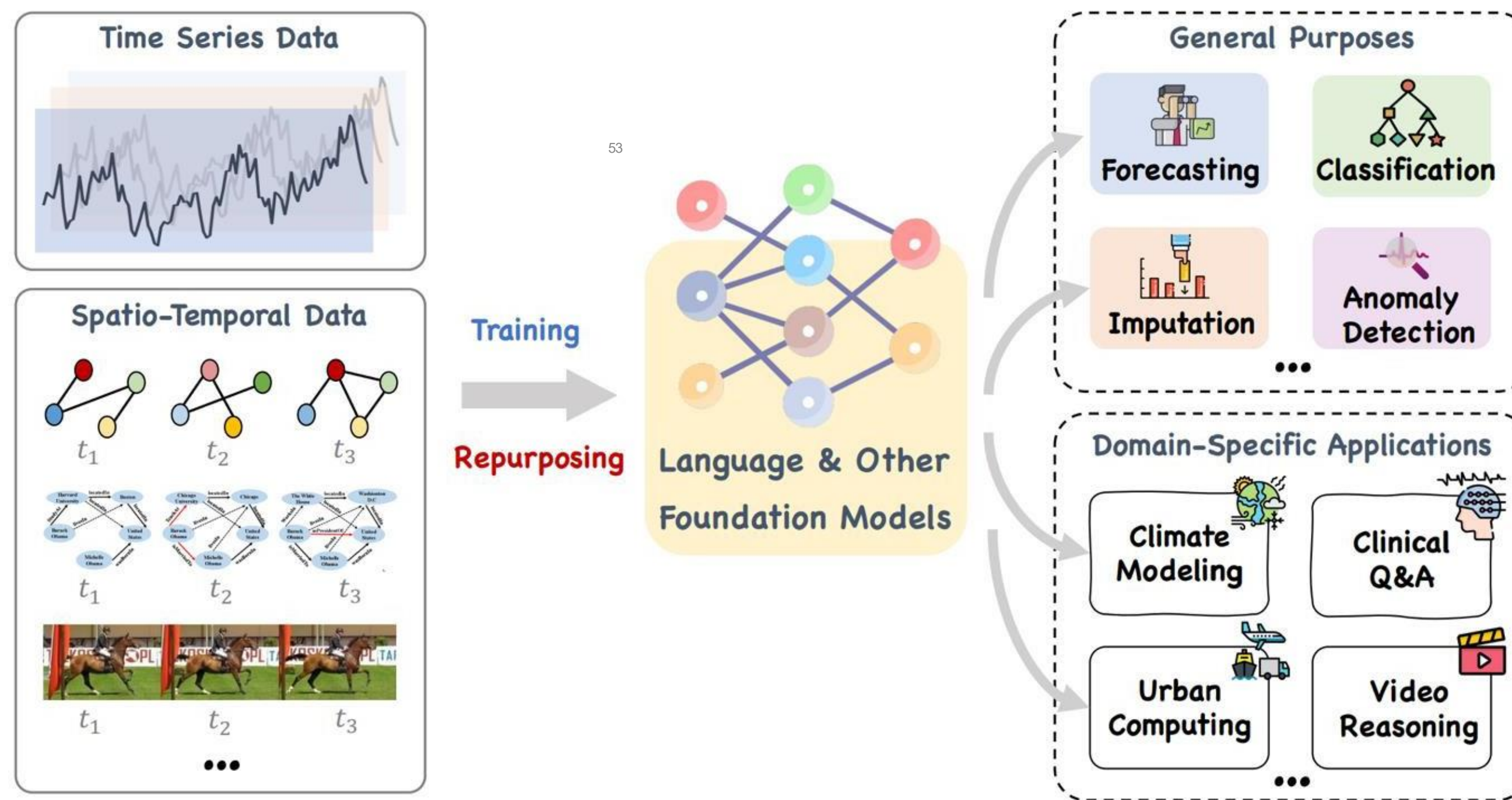
Roadmap of LLMs

- LLMs and Foundation Models

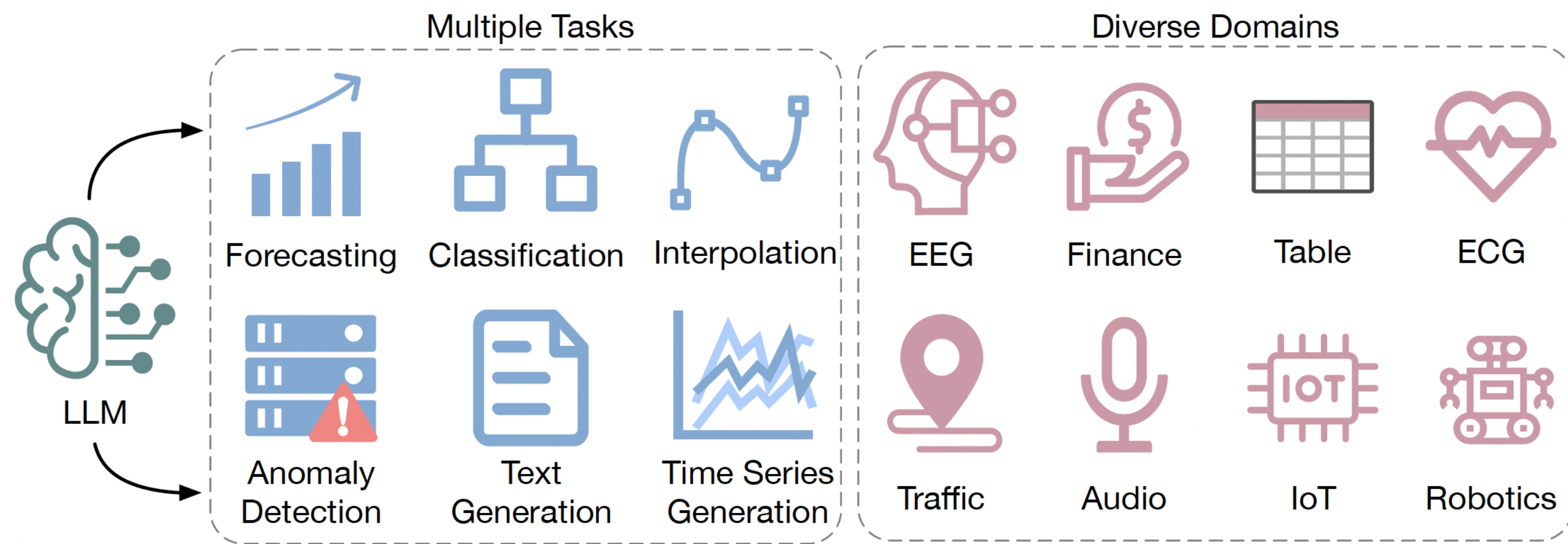


Towards General Intelligence for TS

- LLMs can be either trained or adeptly repurposed to handle TS data for a range of general-purpose tasks and specialized domain applications.

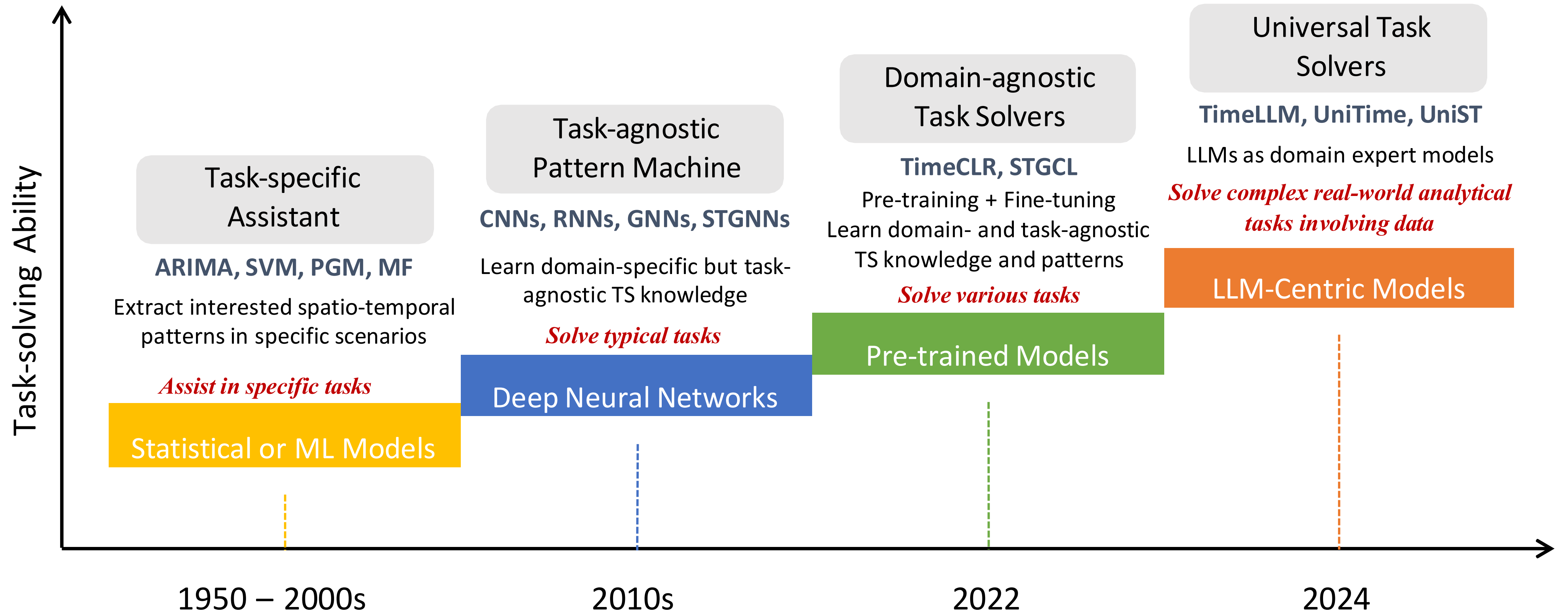


LLM for Time Series

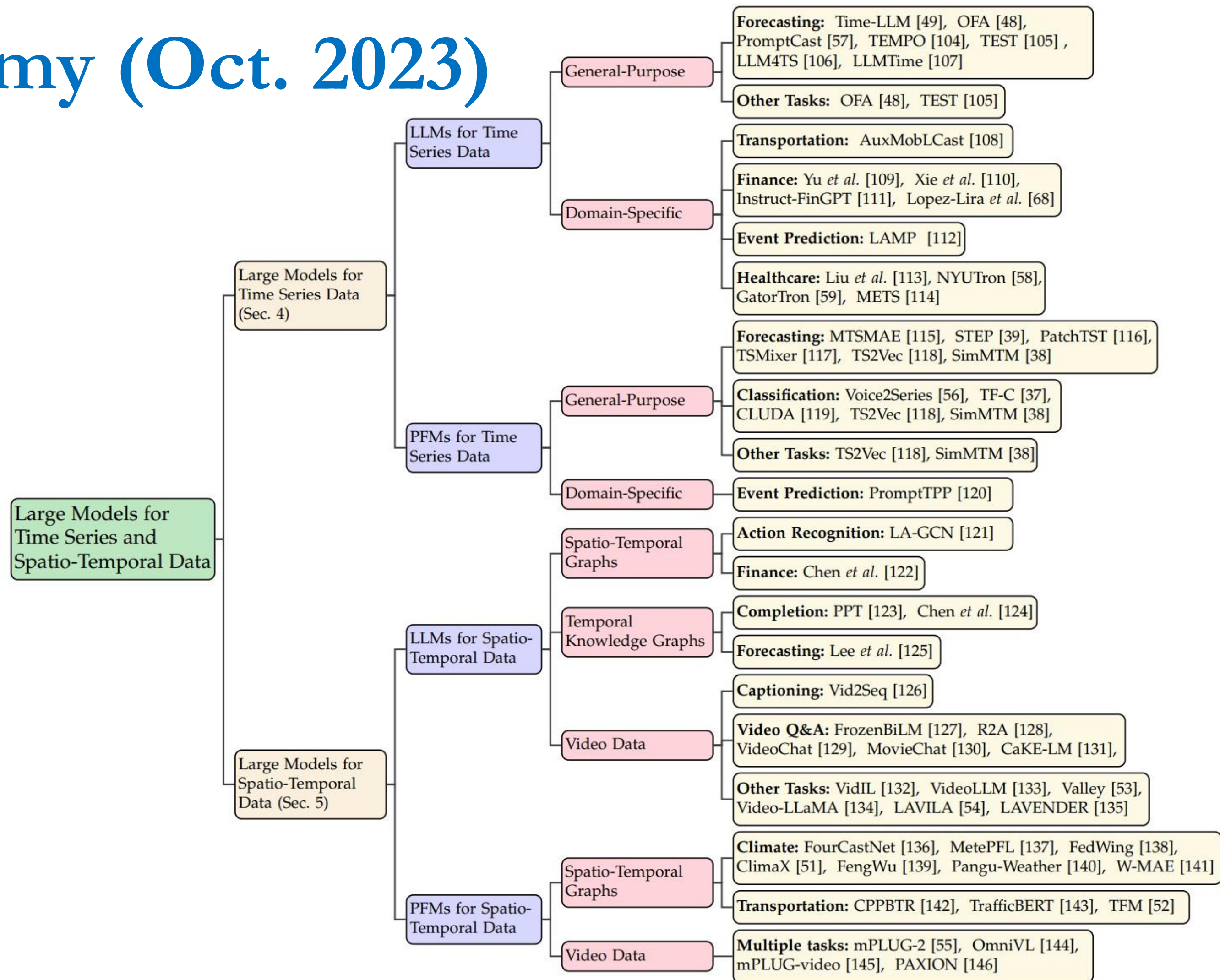


Large language models for various time series tasks in diverse application domains.

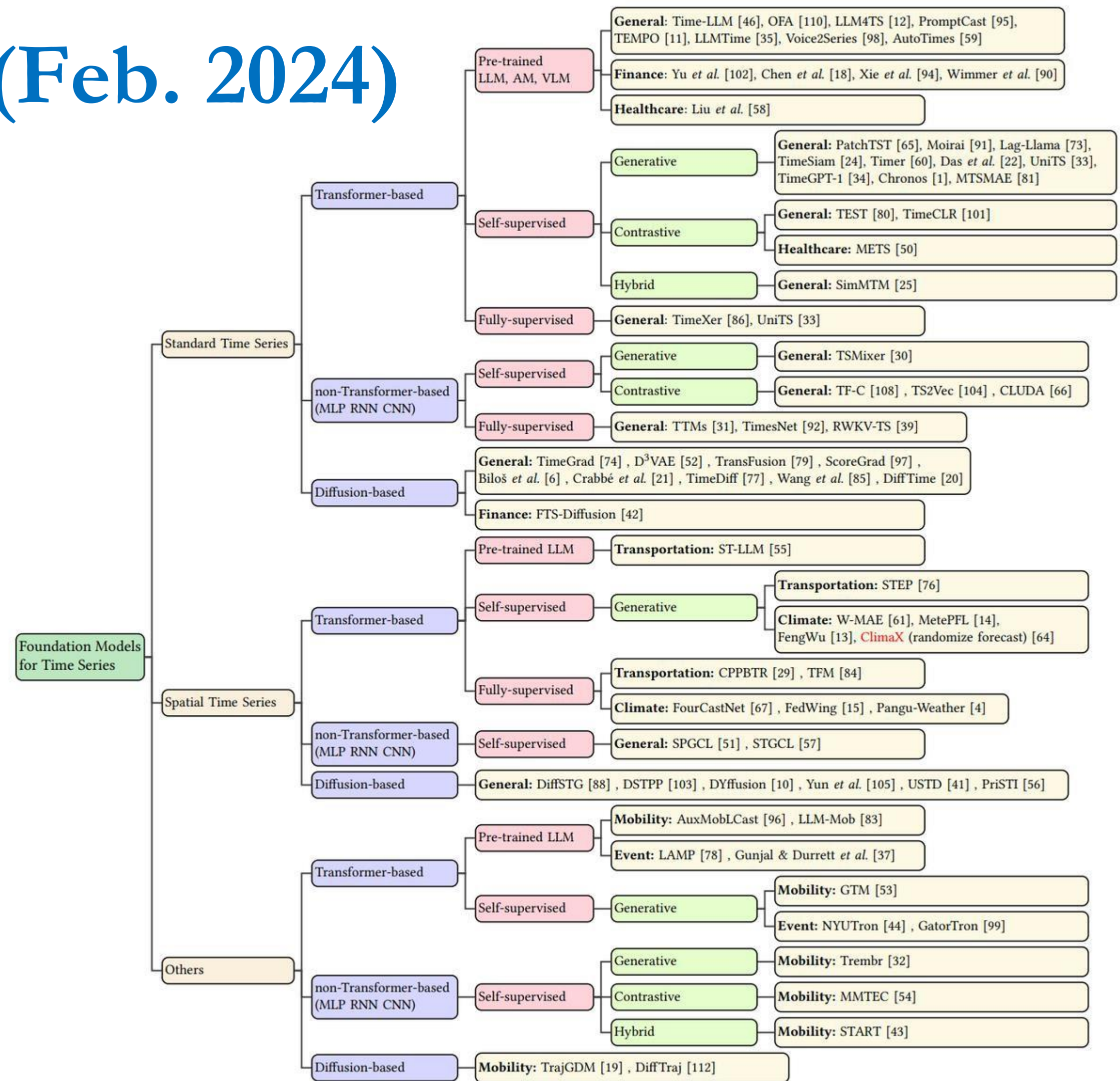
Roadmap of Time Series Analysis



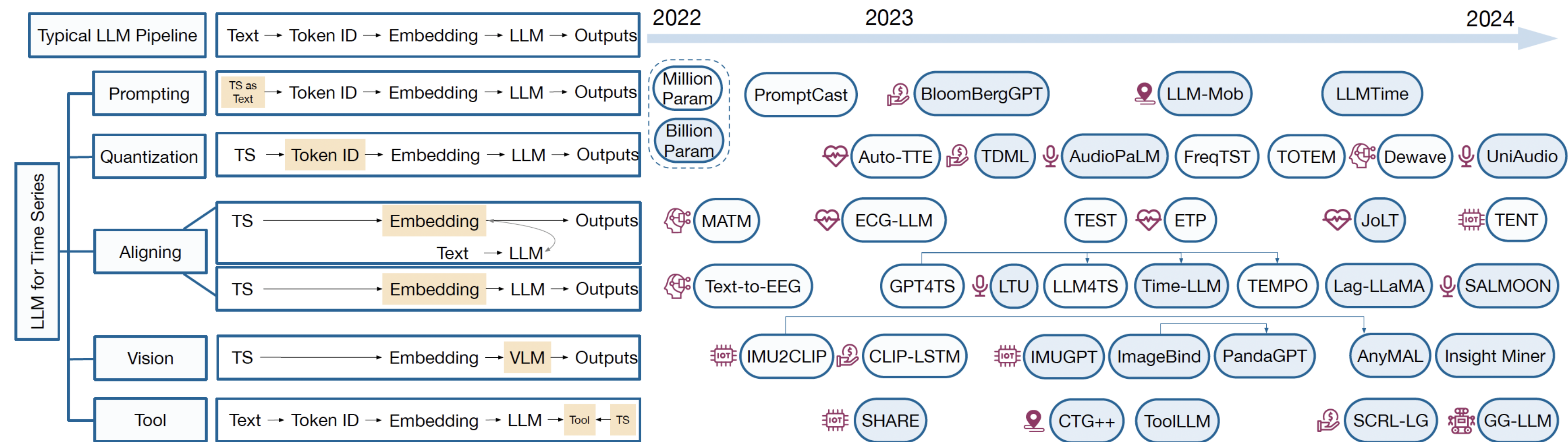
Taxonomy (Oct. 2023)



Taxonomy (Feb. 2024)



LLM for Time Series



- Left: Taxonomy of LLMs for time series analysis (prompting, quantization, aligning which is further categorized into two groups as detailed in Figure 4, vision as bridge, tool integration). For each category, key distinctions are drawn in comparison to the standard LLM pipeline shown at the top of the figure.
- Right: We present representative works for each category, sorted by their publication dates. The use of arrows indicates that later works build upon earlier studies. Dark(light)-colored boxes represent billion(million)-parameter models. Icons to the left of the text boxes represent the application domains of domain-specific models, with icons' meanings illustrated in Figure 1.

LLM for Time Series - Prompting

- **Number-Agnostic Tokenization:** The method treats numerical time series as raw textual data and directly prompts existing LLMs.

Table 1: Examples of representative direct prompting methods.

Method	Example
PromptCast [Xue and Salim, 2022]	“From $\{t_1\}$ to $\{t_{\text{obs}}\}$, the average temperature of region $\{U_m\}$ was $\{x_t^m\}$ degree on each day. What is the temperature going to be on $\{t_{\text{obs}}\}$?”
Liu <i>et al.</i> [2023d]	“Classify the following accelerometer data in meters per second squared as either walking or running: 0.052,0.052,0.052,0.051,0.052,0.055,0.051,0.056,0.06,0.064”
TabLLM [Hegselmann <i>et al.</i> , 2023]	“The person is 42 years old and has a Master’s degree. She gained \$594. Does this person earn more than 50000 dollars? Yes or no? Answer:”
LLMTime [Gruver <i>et al.</i> , 2023]	“0.123, 1.23, 12.3, 123.0” \rightarrow “1 2 , 1 2 3 , 1 2 3 0 , 1 2 3 0 0”

LLM for Time Series - Prompting

- **Number-Specific Tokenization:**

Key Points:

- Problem with BPE Tokenization:

Standard Byte Pair Encoding (BPE) breaks numbers inconsistently, complicating numerical reasoning [Gruver et al., 2023; Spathis and Kawsar, 2023].

- Solutions:

Insert spaces between digits to tokenize each digit distinctly (following LLaMA, PaLM).

Use commas to separate time steps.

Scale and fix precision (e.g., 2 decimal places) to optimize context length.

- Applications:

BloombergGPT: Digit-level chunking improves financial text and number modeling.

Mirchandani et al. (2023): Space-prefixed tokenization allows LLMs to perform sequence transformation, completion, and improvement tasks effectively.

Insight:

Proper number-specific tokenization significantly improves LLMs' ability to handle time series and numerical data.

LLM for Time Series - Quantization

Two Types of Index-Based Quantization Methods for Time Series

(a) VQ-VAE Based Quantization:

Time Series Encoder generates embeddings.

Embeddings are mapped to the closest codebook vectors via vector quantization.

Indices representing the codebook entries are used for decoding.

Enables efficient reconstruction through the Time Series Decoder.

(b) K-Means Based Quantization:

Feature Extraction generates embeddings.

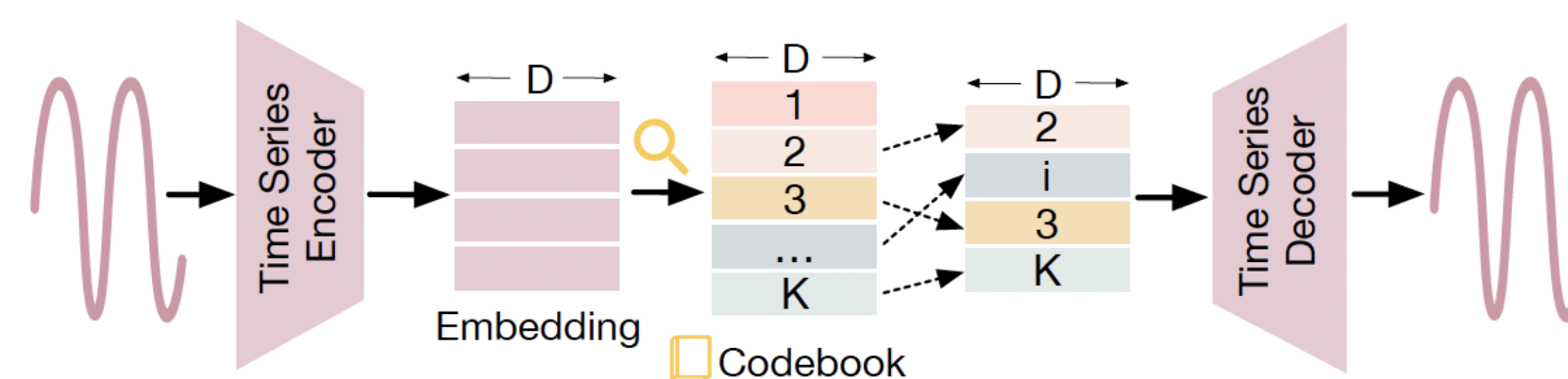
Embeddings are clustered using K-Means to assign cluster indices.

Indices are used for further masking encoding and downstream tasks.

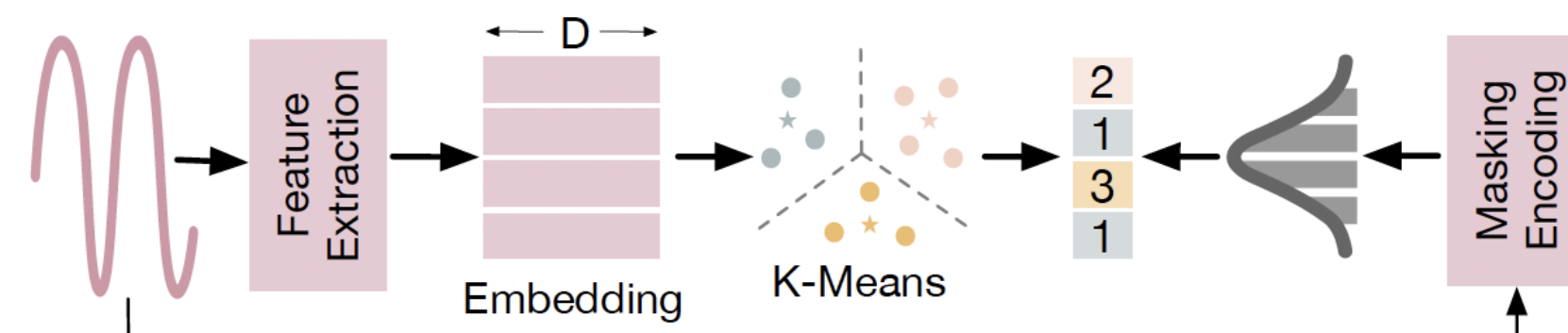
Simplifies representation without full reconstruction.

Insight:

Both methods compress time series into discrete indices for efficient modeling, but VQ-VAE focuses on reconstruction, while K-Means targets feature encoding and masking for downstream learning.



(a) VQ-VAE based quantization method.



(b) K-Means based quantization method.

LLM for Time Series - Aligning

Two Types of Aligning-Based Methods for Time Series and LLMs

(a) Aligning by Similarity Matching (Type 1):

Time series and text are separately encoded into embeddings.

Similarity matching is used to align time series embeddings with LLM (text) embeddings.

No joint decoding; mainly used for retrieval or classification tasks.

(b) Aligning with LLMs as Backbones (Type 2):

Time series and text embeddings are both fed into a shared LLM encoder-decoder.

The model can generate either time series outputs (e.g., forecasting) or text outputs (e.g., EEG-to-text) depending on the task.

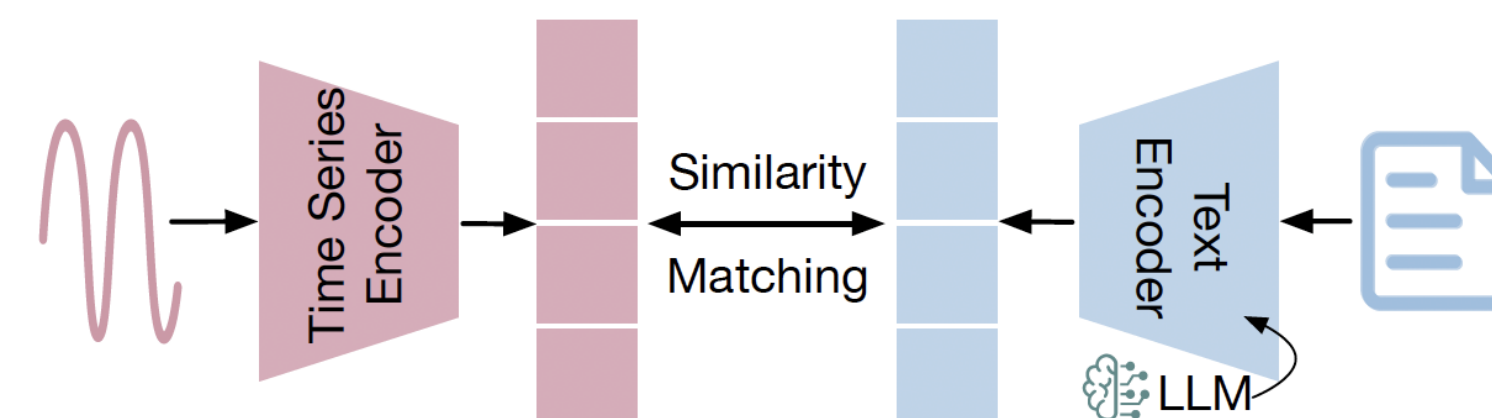
Enables flexible sequence-to-sequence generation.



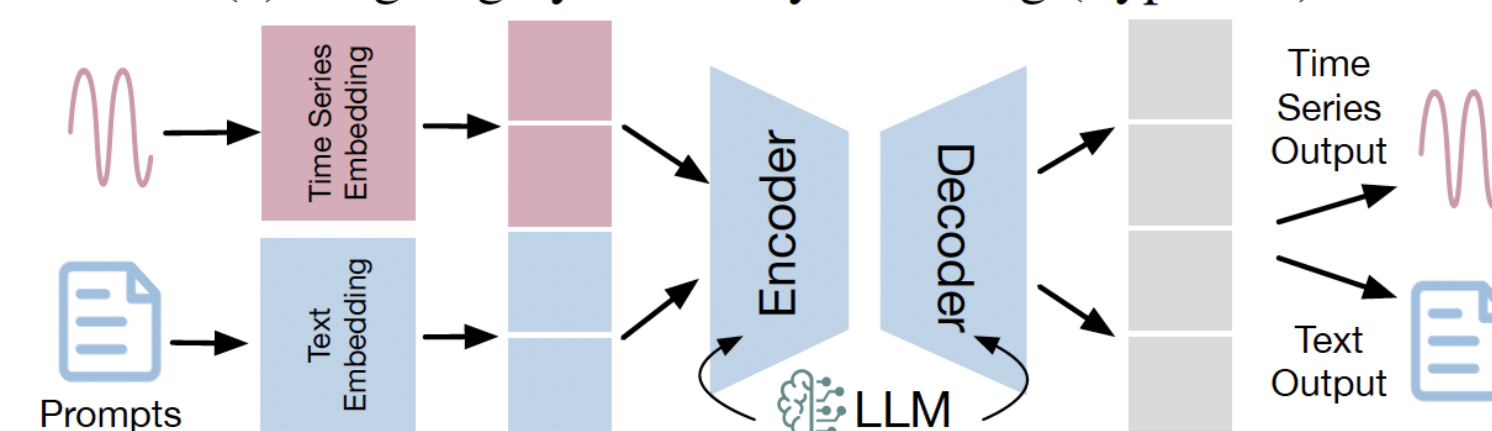
Insight:

Type 1: Independent encoding + similarity alignment.

Type 2: Joint encoding + versatile output generation using LLMs.



(a) Aligning by similarity matching (Type one).



(b) Aligning with large language models as backbones (Type two), where the output could be time series (e.g., forecasting) or text (e.g., EEG-to-text) depending on the downstream tasks.

LLM for Time Series - Aligning

Aligning Time Series with Language Models

Key Idea:

- Train a separate encoder for time series, then align it with the semantic space of language models using various strategies.

Two Main Approaches:

1. Similarity Matching

- Contrastive Loss: Align time series and text embeddings via similarity (e.g., ECG \leftrightarrow reports, sensor \leftrightarrow clinical notes).
Examples: ETP, TEST, TENT, JoLT, Q-Former
- Other Losses: Use Optimal Transport, Canonical Correlation, Wasserstein distance for alignment.
Examples: ECG-LLM, MTAM

2. LLMs as Backbones

- Feed time series embeddings into pre-trained LLMs (e.g., GPT, BART, LLaMA).

Examples:

EEG-to-Text

GPT4TS, LLM4TS, GATGPT, Time-LLM

Lag-LLaMA for probabilistic forecasting

WavPrompt, Speech LLaMA, MU-LLAMA, SALMONN for audio/music

LLM for Time Series - Vision as a Bridge

Vision as a Bridge: Linking Time Series with LLMs

Key Idea:

- Visual representations of time series (e.g., plots, IMU motion, paired images) act as an intermediate modality to align time series with large language models (LLMs).

Three Main Approaches:

1. Paired Data & Joint Embedding
 - ImageBind, PandaGPT: Align multiple modalities (image, text, IMU, etc.) via paired data
 - IMU2CLIP, AnyMAL: Project IMU time series into CLIP or LLaMA embedding space
2. Physics-Based Alignment
 - IMUGPT: Generates human motion from text and derives IMU signals based on motion physics
3. Time Series Plots as Images
 - CLIP-LSTM: Converts price charts into images and uses CLIP for forecasting
 - Insight Miner: Feeds line plots of time series into LLaVA for generating textual trend descriptions

LLM for Time Series - Tools

Tool: Using LLMs Indirectly for Time Series Tasks

Key Idea:

- Instead of processing time series **directly**, LLMs are used to generate **indirect tools**—like code, APIs, or feature selectors—to **assist downstream time series applications**.

Three Main Tool Types:

1. Code Generation

- CTG++: Uses GPT-4 to turn text instructions into differentiable loss functions for traffic diffusion modeling.

2. API Frameworks

- ToolLLM: Provides a tool-based pipeline (data → model → evaluation) using LLM-generated API calls for tasks like weather and stock prediction.

3. Text Domain Knowledge

- SHARE: Augments human activity labels via GPT-4 to maintain semantic structure.
- GG-LLM: Encodes commonsense patterns (e.g., human actions) using LLaMA-2.
- SCRL-LG: Uses LLaMA-7B to extract features from news headlines for reinforcement learning in financial forecasting.

LLM for Time Series - Summary

Table: Summary of five major categories of applying LLMs for time series analysis

Method	Subcategory	Representative Works	Equations	Advantages	Limitations
Prompting	Number-Agnostic	PromptCast [Xue and Salim, 2022]	$\mathbf{y} = f_{\theta}(\mathbf{x}_s, \mathbf{x}_t)$	easy to implement; zero-shot capability	lose semantics; not efficient
	Number-Specific	LLMTime [Gruver <i>et al.</i> , 2023]			
Quantization	VQ-VAE	DeWave [Duan <i>et al.</i> , 2023]	$k_i = \arg \min_j \ g_{\phi}(\mathbf{x}_s)_i - \mathbf{c}_j\ _2$	flexibility of index and time series conversion	may require two-stage training
	K-Means	AudioLM [Borsos <i>et al.</i> , 2023]	$\mathbf{k} = [k_i]_{i=1}^{\frac{T}{S}}, \mathbf{y} = f_{\theta}(\mathbf{k}, \mathbf{x}_t)$		
	Text Categories	TDML [Yu <i>et al.</i> , 2023]	$\mathbf{y} = f_{\theta}(q(\mathbf{x}_s), \mathbf{x}_t)$		
Aligning	Similarity Match	ETP [Liu <i>et al.</i> , 2023a] MATM [Han <i>et al.</i> , 2022]	$\mathbf{y} = g_{\phi}(\mathbf{x}_s)$ $\mathcal{L} = \text{sim}(g_{\phi}(\mathbf{x}_s), f_{\theta}(\mathbf{x}_t))$	align semantics of different modalities; end-to-end training	complicated design and fine-tuning
	LLM Backbone	GPT4TS [Zhou <i>et al.</i> , 2023a]	$\mathbf{y} = f_{\theta}(g_{\phi}(\mathbf{x}_s), \mathbf{x}_t)$		
Vision as Bridge	Paired Data	ImageBind [Girdhar <i>et al.</i> , 2023]	$\mathcal{L} = \text{sim}(g_{\phi}(\mathbf{x}_s), h_{\psi}(\mathbf{x}_v))$	additional visual knowledge	not hold for all data
	TS Plots as Images	Wimmer and Rekabsaz [2023]	$\mathbf{y} = h_{\psi}(\mathbf{x}_s)$		
Tool	Code	CTG++ [Zhong <i>et al.</i> , 2023]	$z = f_{\theta}(\mathbf{x}_t)$	empower LLM with more abilities	optimization not end-to-end
	API	ToolLLM [Qin <i>et al.</i> , 2023]	$\mathbf{y} = z(\mathbf{x}_s)$		

LLM for Time Series - Multimodal Datasets

Internet of Things (IoT):

- Ego4D, Ego-Exo4D, DeepSQA
- Paired IMU time series + textual activity descriptions
- Used for human activity recognition and instruction following

Finance:

- PIXIU, MoAT
- Align time series (e.g., stock, fuel) with financial news
- Used for forecasting and financial NLP tasks

Healthcare:

- Zuco, PTB-XL, ECG-QA
- Eye-tracking, EEG, and ECG paired with reports, labels, or questions
- Supports clinical NLP and diagnostics

Audio / Music / Speech:

- AudioSet, OpenAQA-5M, MusicCaps, Libri-Light, Voice
- Rich collections of speech/music time series with text or labels
- Enables captioning, translation, QA

LLM for Time Series - Multimodal Datasets

Domain	Dataset	Size	Major Modalities	Task
Internet of Things	Ego4D ² [Grauman <i>et al.</i> , 2022]	3, 670h data, 3.85M narrations	text, IMU, video, audio, 3D	classification, forecasting
	DeepSQA ³ [Xing <i>et al.</i> , 2021]	25h data, 91K questions	text, imu	classification, question answering
Finance	PIXIU ⁴ [Xie <i>et al.</i> , 2023b]	136K instruction data	text, tables	5 NLP tasks, forecasting
	MoAT ⁵ [Lee <i>et al.</i> , 2023]	6 datasets, 2K timesteps in total	text, time series	forecasting
Healthcare	Zuco 2.0 ⁶ [Hollenstein <i>et al.</i> , 2019]	739 sentences	text, eye-tracking, EEG	classification, text generation
	PTB-XL ⁷ [Wagner <i>et al.</i> , 2020]	60h data, 71 unique statements	text, ECG	classification
	ECG-QA ⁸ [Oh <i>et al.</i> , 2023]	70 question templates	text, ECG	classification, question answering
Audio	OpenAQA-5M ⁹ [Gong <i>et al.</i> , 2023]	5.6M (audio, question, answer) tuples	text, audio	tagging, classification
Music	MusicCaps ¹⁰ [Agostinelli <i>et al.</i> , 2023]	5.5K music clips	text, music	captioning, generation
Speech	CommonVoice ¹¹ [Ardila <i>et al.</i> , 2019]	7, 335 speech hours in 60 languages	text, speech	ASR, translation

LLM for Time Series - Challenges

1. Theoretical Understanding

- Current research is empirically successful, but lacks theoretical grounding.
- Works like LIFT and GPT4TS suggest links between LLM representations and PCA, and hint at universal function approximation by Transformers.
- **Future need:** clarify how LLMs interpret numerical time series.

2. Multimodal & Multitask Analysis

- Most current LLM models focus on single task + modality.
- Inspired by UnifiedIO and UniAudio, more work is needed to build multitask, multimodal time series foundation models.

3. Efficient Algorithms

- Long and multivariate time series → high complexity for LLMs.
- While patching helps, it may hurt interpretability.
- Future work: efficient designs for scalable and user-friendly LLM-based time series systems.

4. Combining Domain Knowledge

- Infusing domain-specific methods (e.g., decomposition, wavelet, autocorrelation) into LLM pipelines (e.g., TEMPO, FreqTST) improves performance.
- Future potential in hybrid modeling.

5. Customization & Privacy

- Global models ≠ optimal for personalized or private clinical/home data.
- FedAlign uses federated learning + natural language anchors for privacy-preserving alignment.
- Future direction: custom LLMs for individuals and secure learning frameworks.

LLM for TS Typical Applications

What Can Large Language Models Tell Us about Time Series Analysis?

- Our standpoint is that LLMs can serve as the central hub for understanding and advancing time series analysis in three principal ways
 - **LLM-as-Enhancers**: augmenting TS data and existing approaches with enhanced external knowledge and analytical prowess
 - **LLM-as-Predictors**: utilizing their extensive internal knowledge and emerging reasoning abilities to benefit a range of downstream tasks, e.g., forecasting
 - **LLM-as-Agents**: transcending conventional roles to actively engage in and transform time series analysis

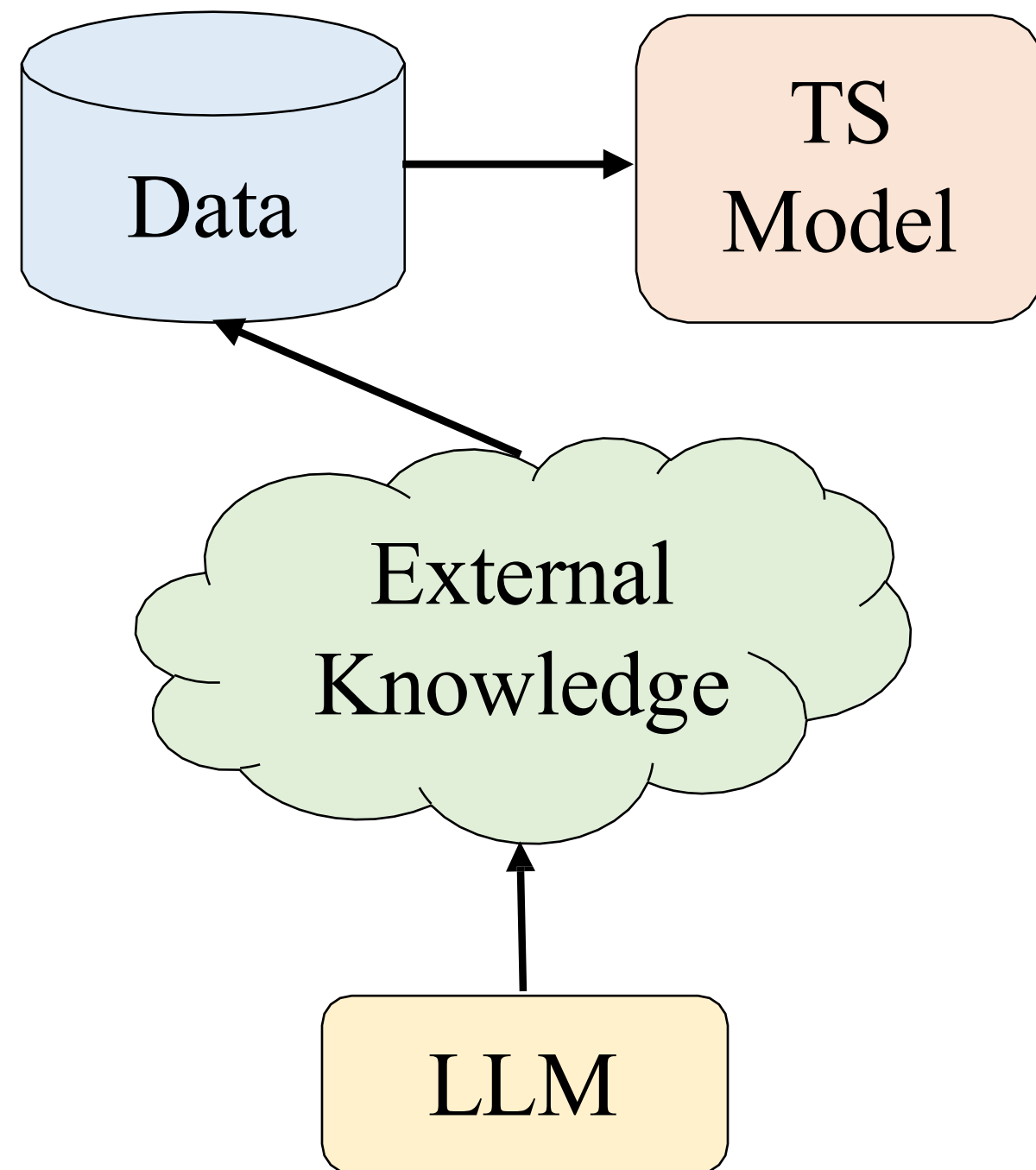
LLM-as-Enhancer

- Our standpoint is that LLMs can serve as the central hub for understanding and advancing time series analysis in three principal ways
 - **LLM-as-Enhancers**: augmenting TS data and existing approaches with enhanced external knowledge and analytical prowess
 - **LLM-as-Predictors**: utilizing their extensive internal knowledge and emerging reasoning abilities to benefit a range of downstream tasks, e.g., forecasting
 - **LLM-as-Agents**: transcending conventional roles to actively engage in and transform time series analysis

71

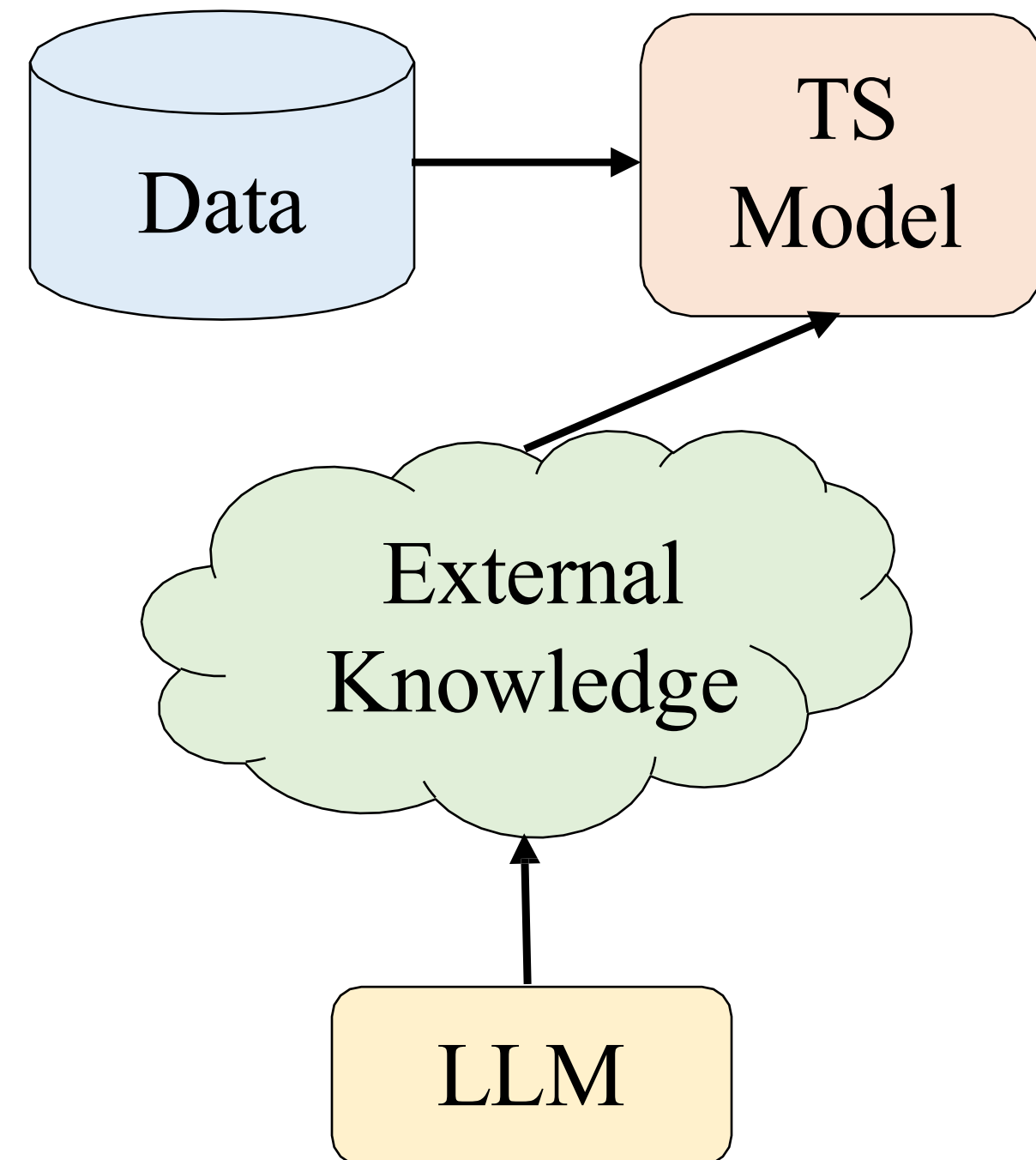
LLM-as-Enhancer

- Data-centric enhancer



72

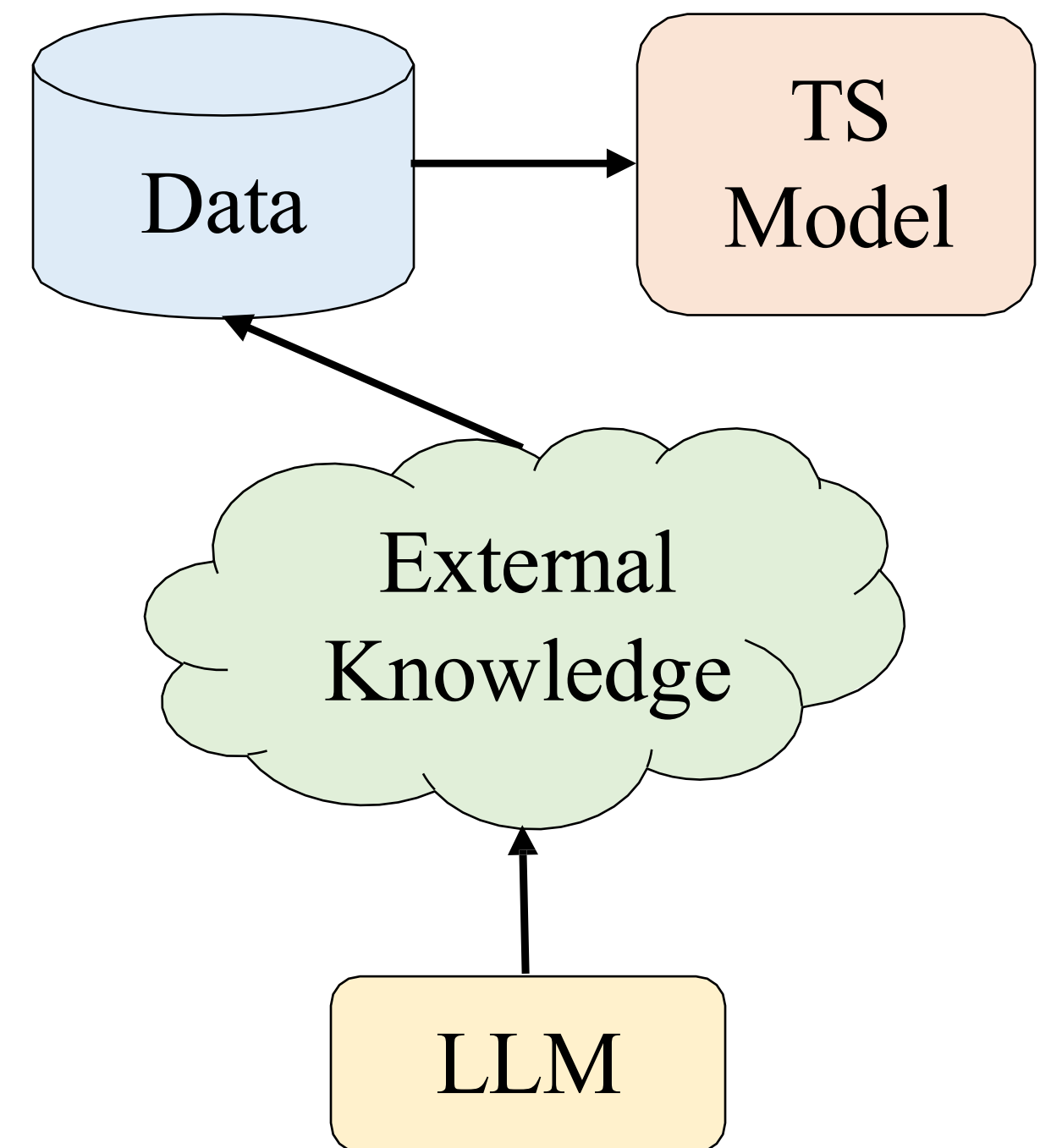
- Model-centric enhancer



Data-Centric Enhancer

- It enhances data **interpretability** or provide supplementary improvements, facilitating a more thorough understanding and effective use of TS data
- Examples
 - LLM for finance
 - LLM-MPE for human mobility

73



Example: LLM for Finance

- This study focuses on NASDAQ-100 stock price prediction, supplemented by metadata about the stock company and relevant financial news data

Prompt: Generate a short description for stock AAPL' s company. Also list general positive and negative factors that might impact the stock price; be brief and use keywords. Consider diverse general factors, such as macro economic situation (e.g. inflation, CPI growth), business factors (e.g. sales, investment, products), technology factors (e.g. innovation), and others. Use format Description: ..., Positive Factors: ..., Negative factors: ...

GPT-4 Response:

Description:

AAPL (Apple Inc.) is a world-renowned technology company based in Cupertino, California. Known for its sleek designs and innovative technologies, the company produces a wide range of products including the iPhone, iPad, Mac, Apple Watch, and services like iCloud, App Store, and more.

Positive Factors:

Macroeconomic: Economic growth, increased consumer spending

Business: Strong sales, expansion into new markets, positive earnings reports, strategic acquisitions

Technology: Innovation, new product launches, advancements in artificial intelligence and augmented reality

Brand: Strong global brand, loyal customer base

Legal/Regulatory: Favorable policy changes, successful patent applications

Negative Factors:

Macroeconomic: Inflation, economic downturn, trade restrictions

Business: Decreased sales, poor earnings reports, lack of new investments

Technology: Slow pace of innovation, delays in product launches

Brand: Damage to brand reputation, loss of customer loyalty

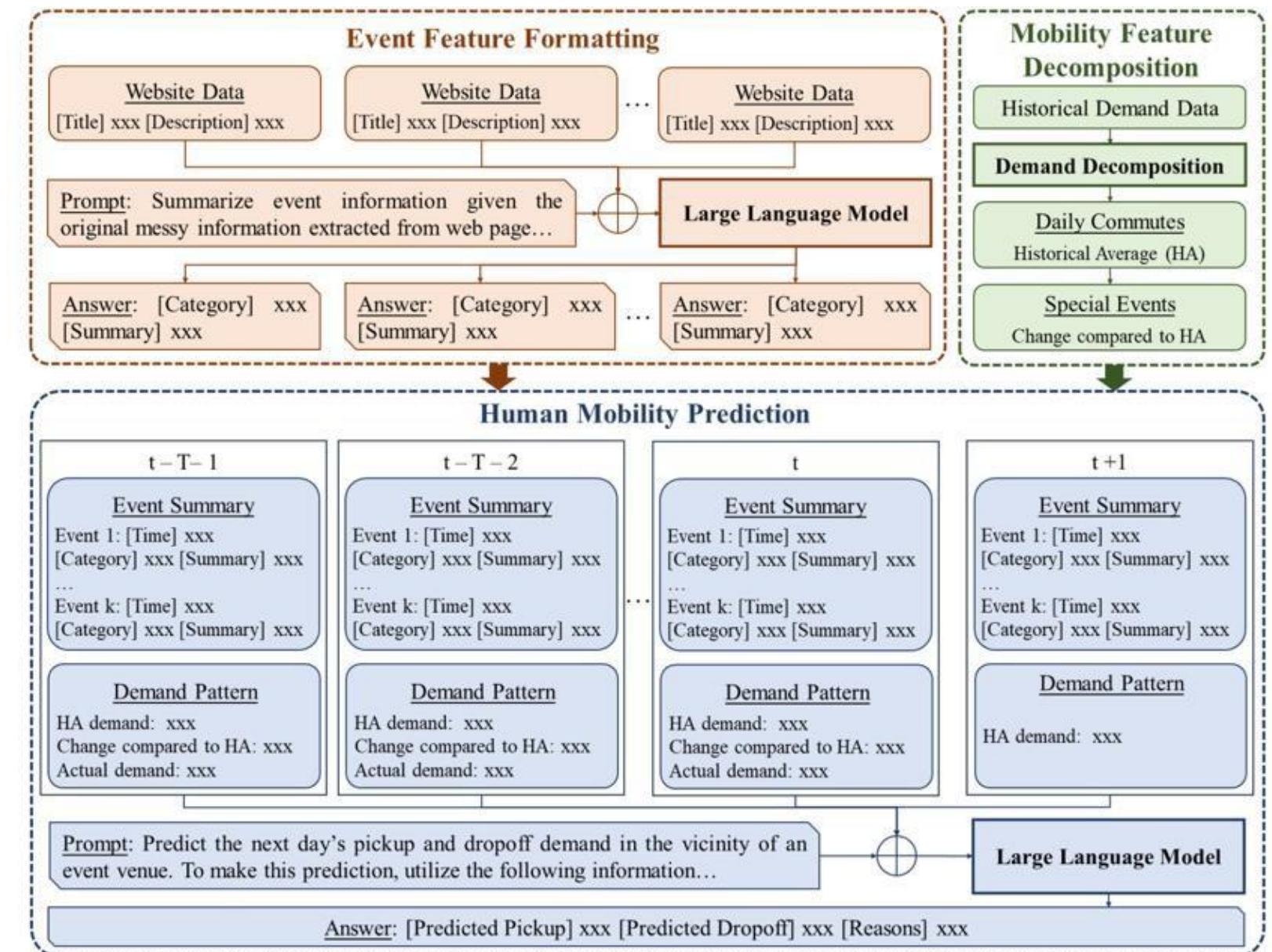
Legal/Regulatory: Antitrust investigations, unfavorable policy changes, patent disputes

Environmental: Failure to meet sustainability goals, negative environmental impact

An example of a stock's company profile consisting of the company description

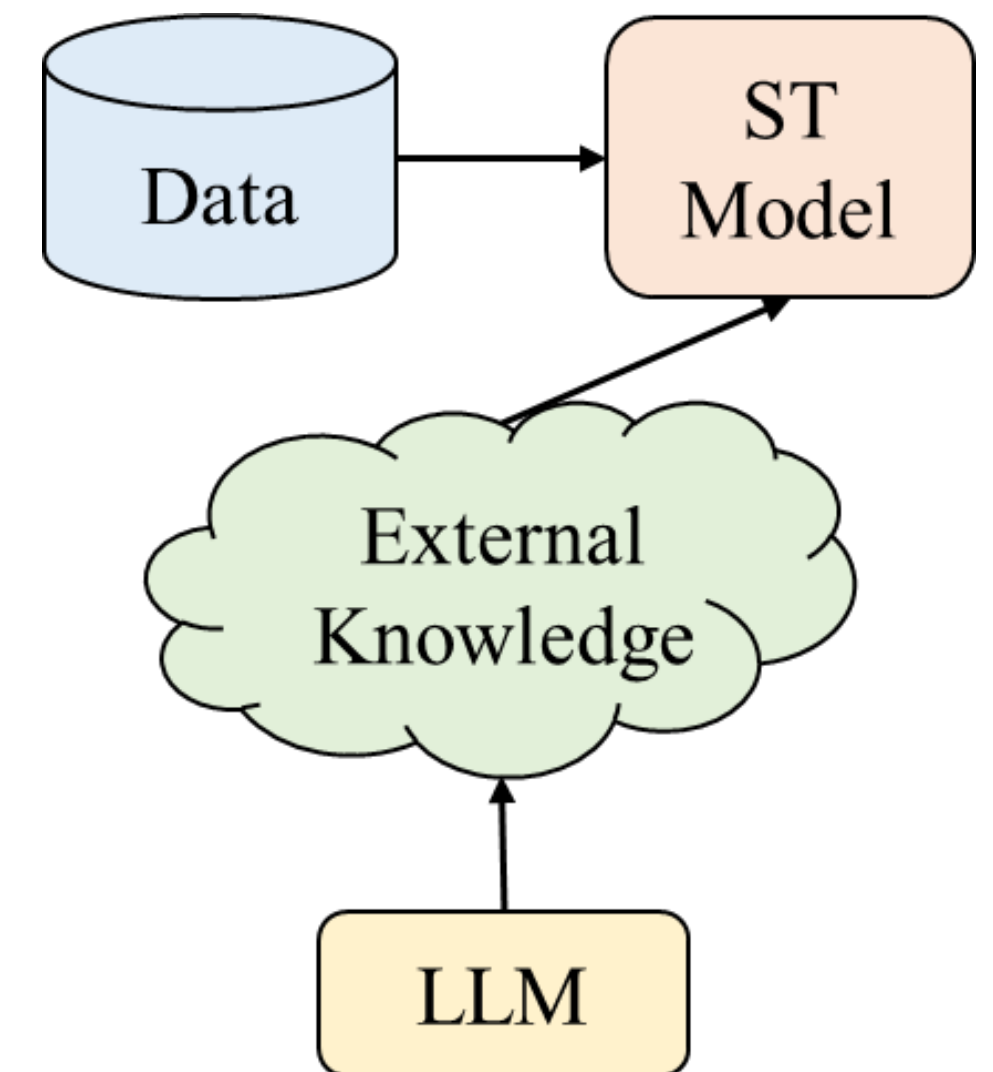
Example: LLM-MPE for Human Mobility

- While rich textual descriptions about public events are commonly available from online sources, it is challenging to encode such information in ML models
- They introduce LLM-MPE for human mobility prediction under public events based on LLMs
 - leveraging their unprecedented ability to process textual data, learn from minimal examples, and generate human-readable explanations



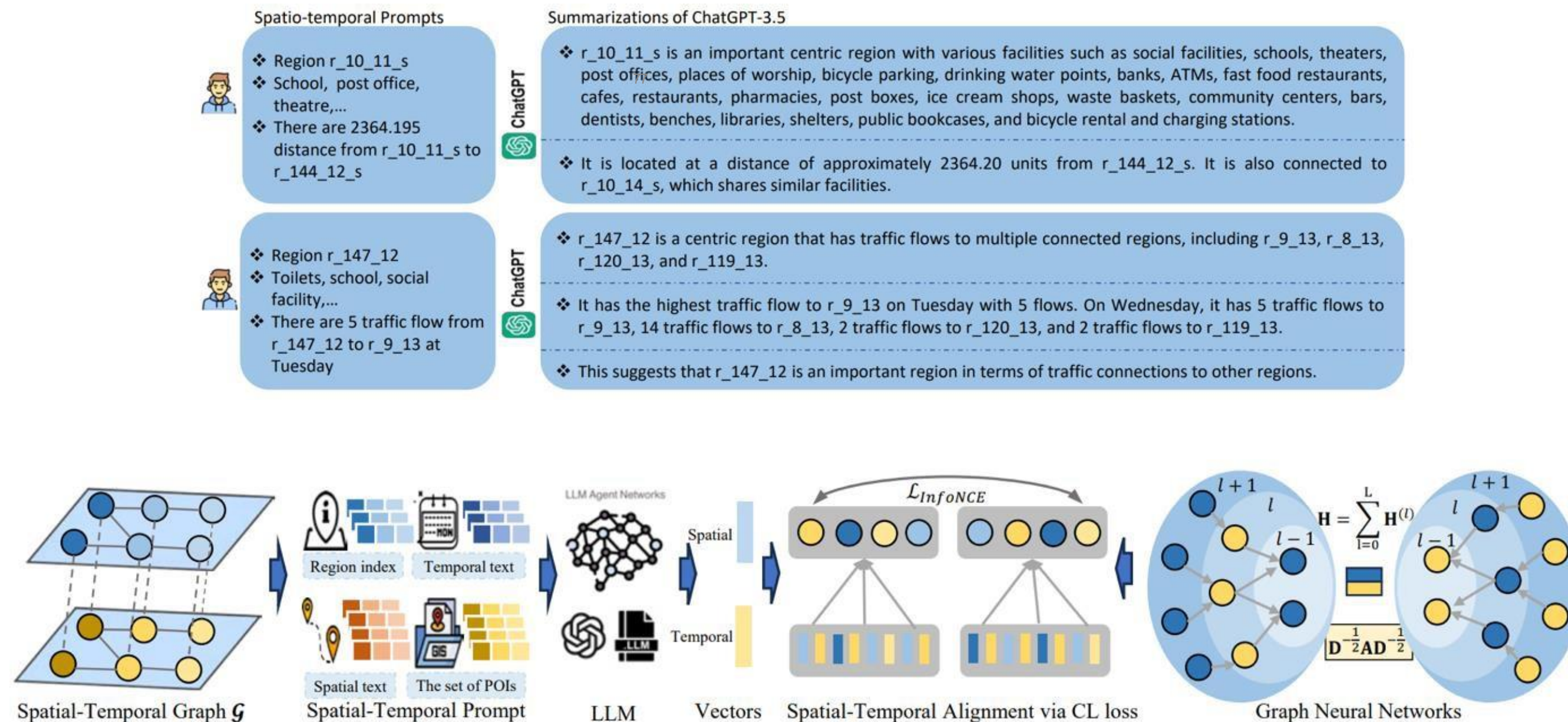
Model-Centric Enhancer

- Model-based enhancers aim to augment TS models by addressing their limitations in external knowledge and domain-specific contexts.
- Transferring knowledge from LLMs boosts the performance of domain models in handling complex tasks
- Examples
 - STLLM: LLM for spatio-temporal forecasting
 - UrbanCLIP: LLM for urban region profiling



Example: STLLM

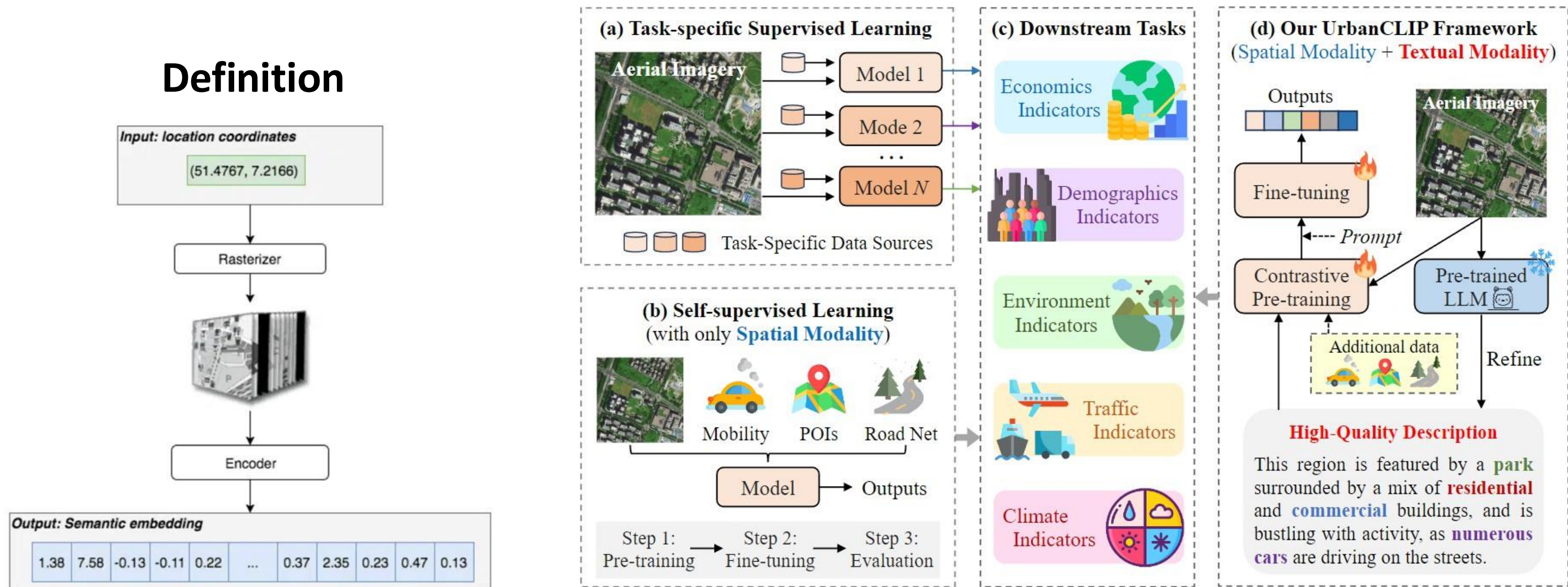
- Using a **spatio-temporal knowledge alignment** paradigm to maximize mutual information between LLM-based knowledge representations and GNN-based structural embeddings, ensuring effective alignment and information preservation across different views



Example: UrbanCLIP

- Urban Contrastive Language-Image Pre-training (UrbanCLIP) is the first framework that integrates the knowledge of text modality into **urban region profiling**

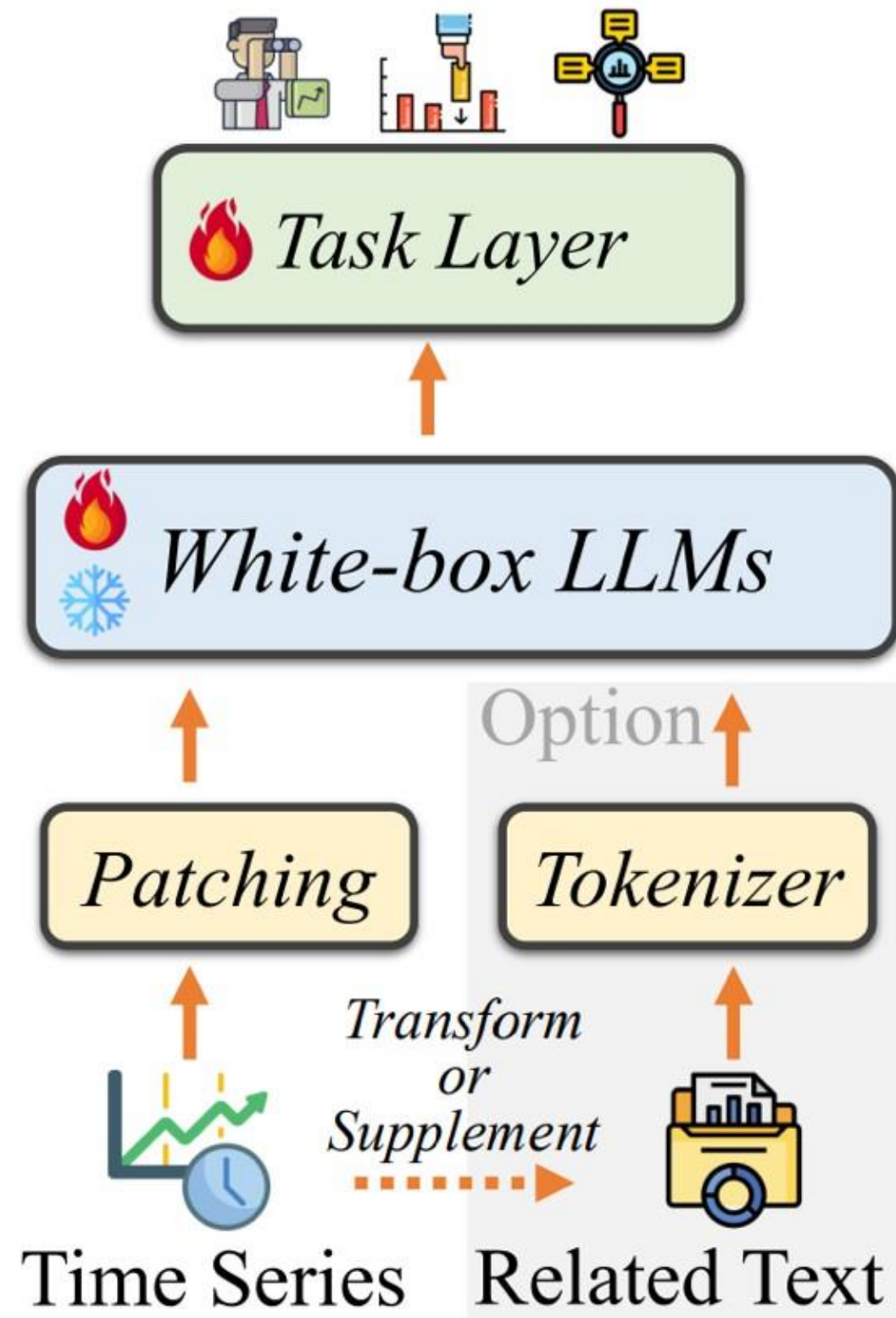
Definition



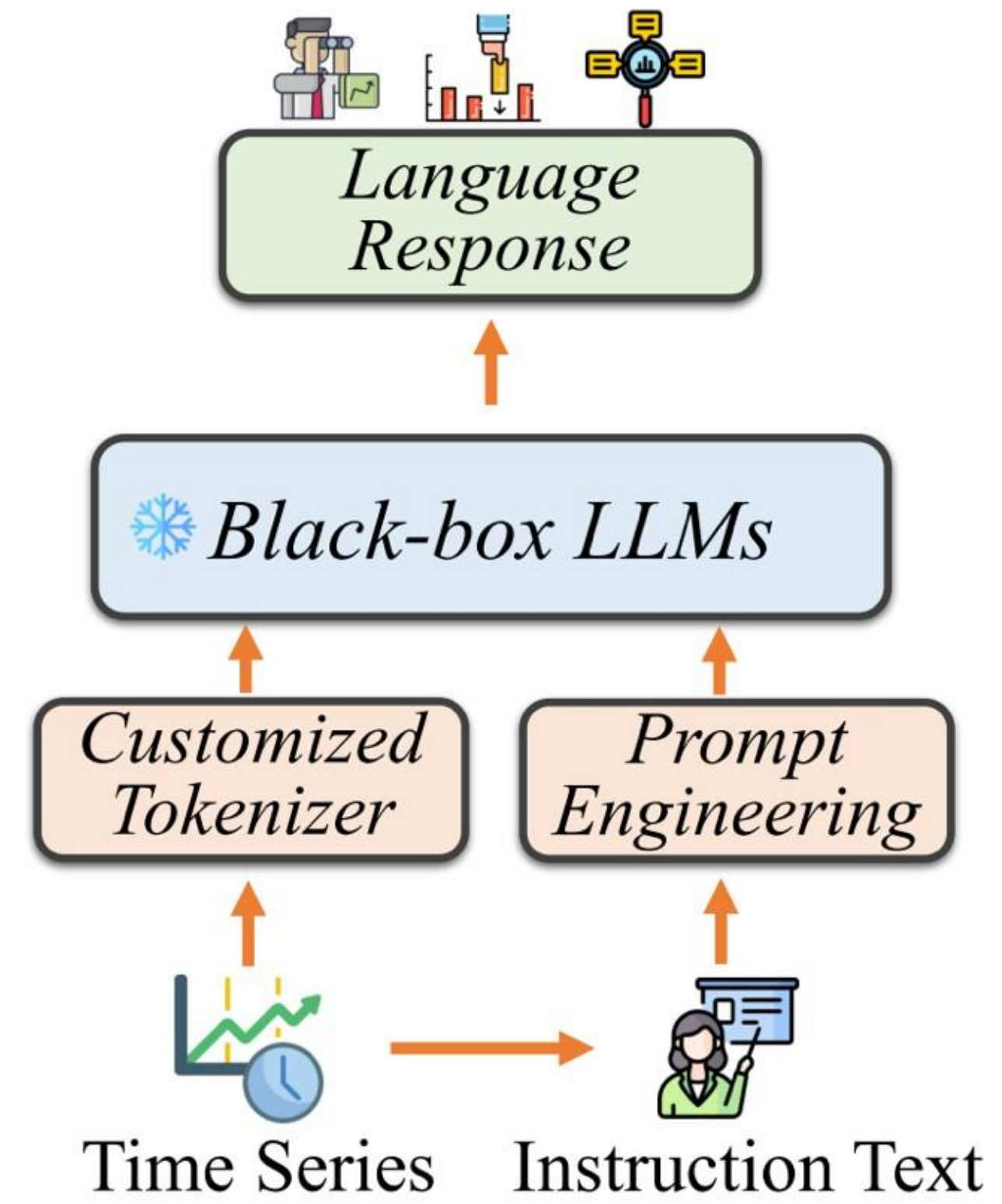
LLM-as-Predictor

- Our standpoint is that LLMs can serve as the central hub for understanding and advancing time series analysis in three principal ways
 - **LLM-as-Enhancers**: augmenting TS data and existing approaches with enhanced external knowledge and analytical prowess
 - **LLM-as-Predictors**: utilizing their extensive internal knowledge and emerging reasoning abilities to benefit a range of downstream tasks, e.g., forecasting
 - **LLM-as-Agents**: transcending conventional roles to actively engage in and transform time series analysis

LLM-as-Predictor



(a) Tuning-Based



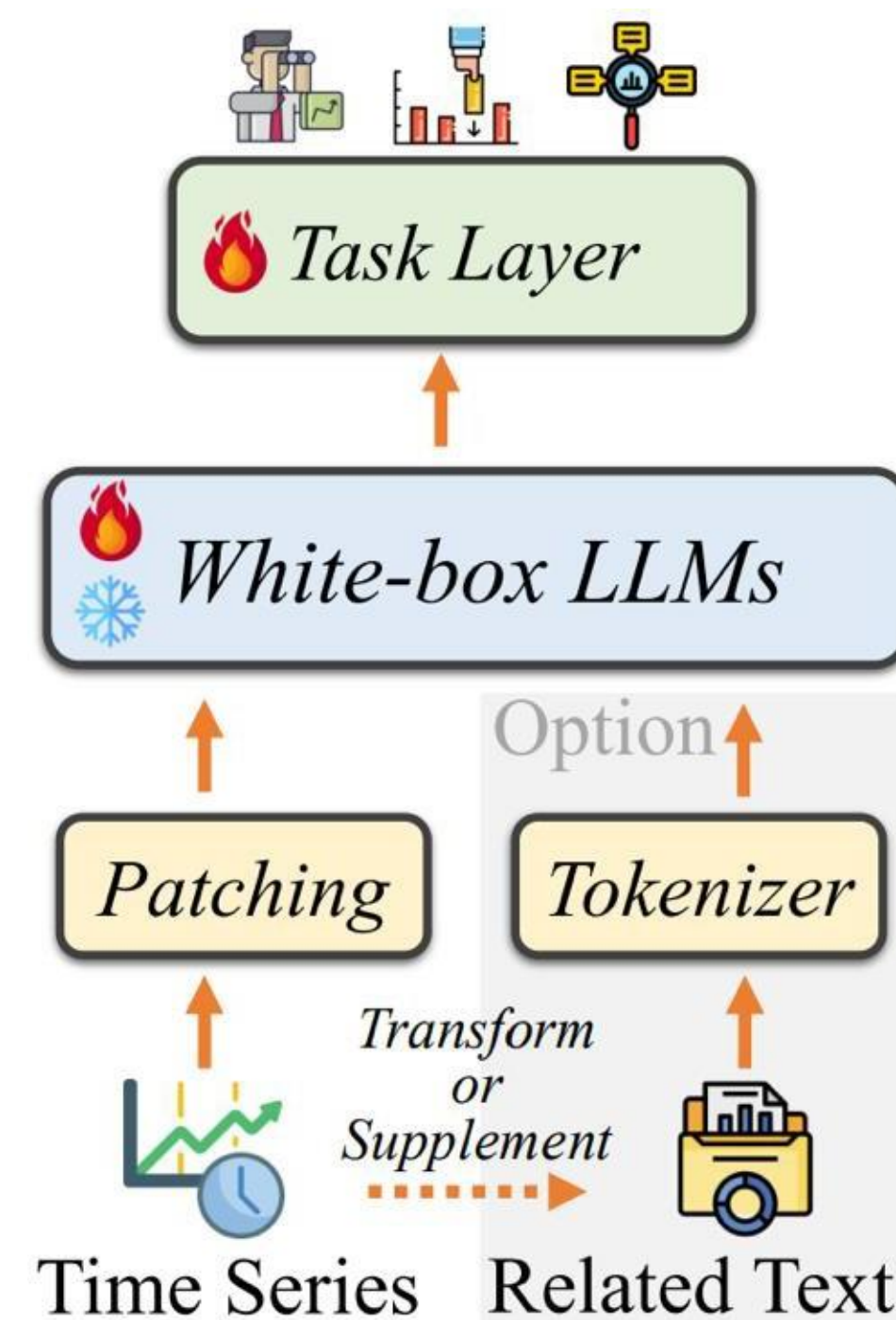
(b) Non-Tuning-Based

80

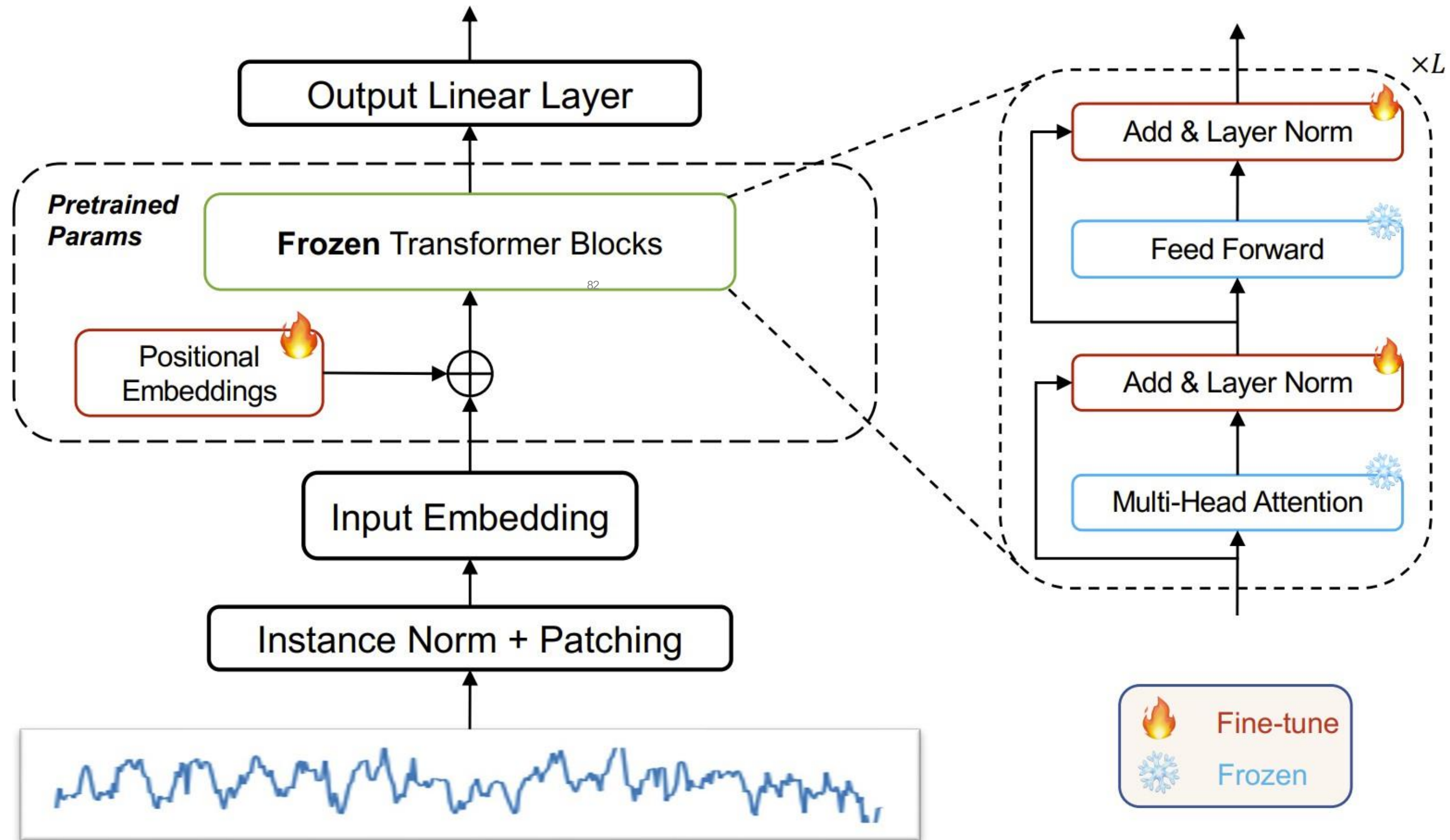
Tuning-based LLM Predictors

- Tuning-based predictors use accessible LLM parameters, typically involving **patching** and **tokenizing** numerical signals and related text data, followed by fine-tuning for TS data
- Examples
 - GPT4TS for time series analysis
 - Time-LLM for time series analysis

81

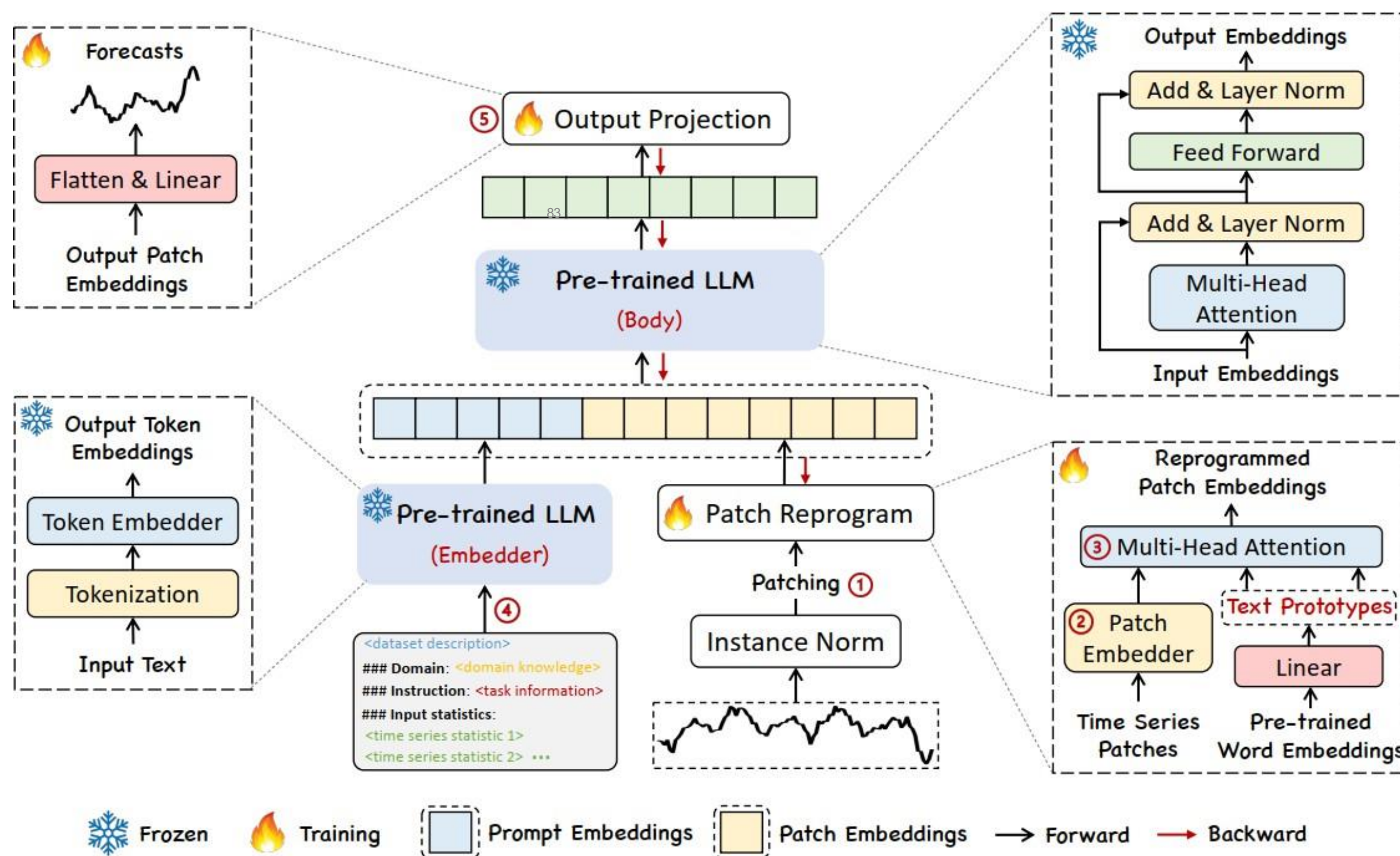


GPT4TS for Time Series Analysis



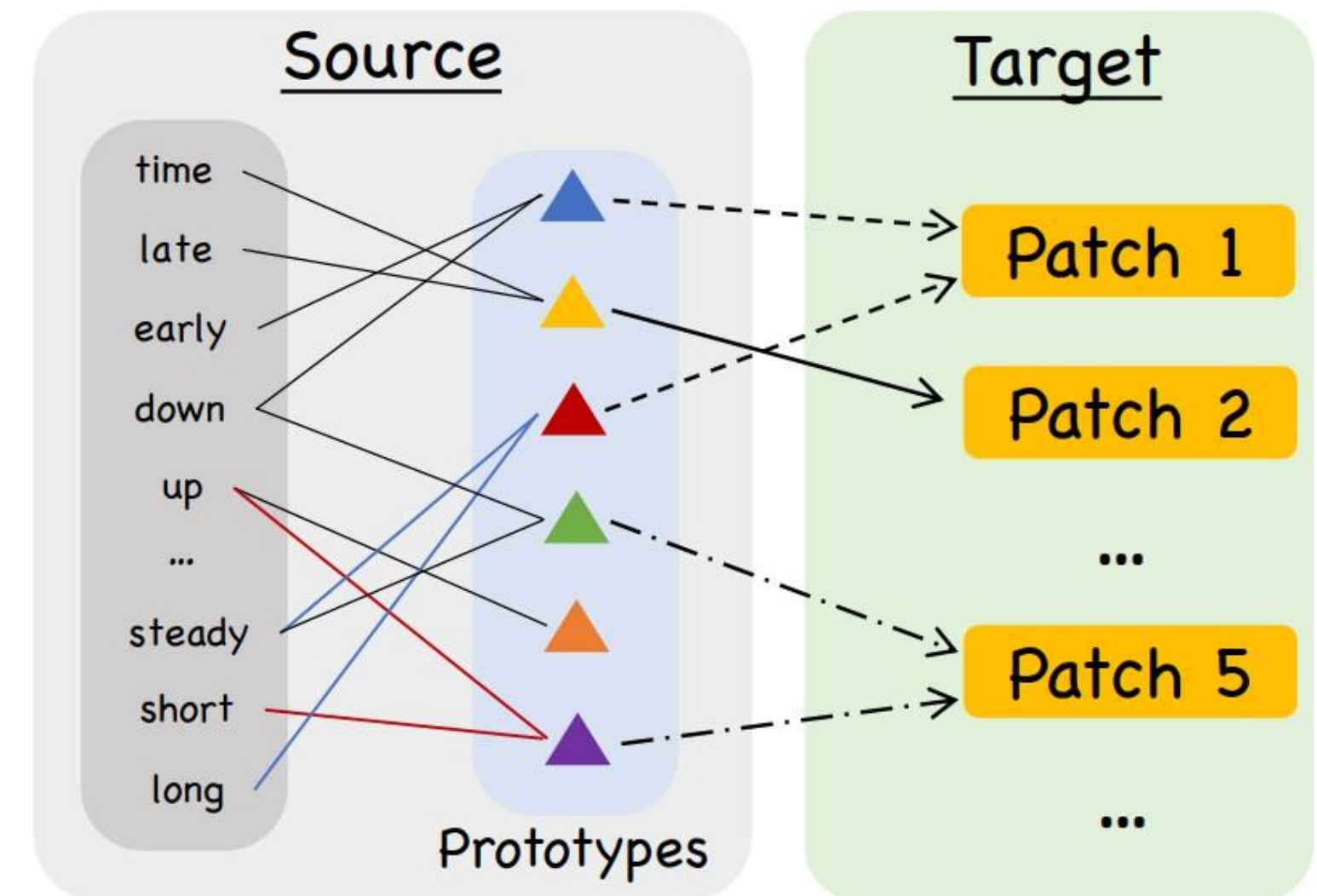
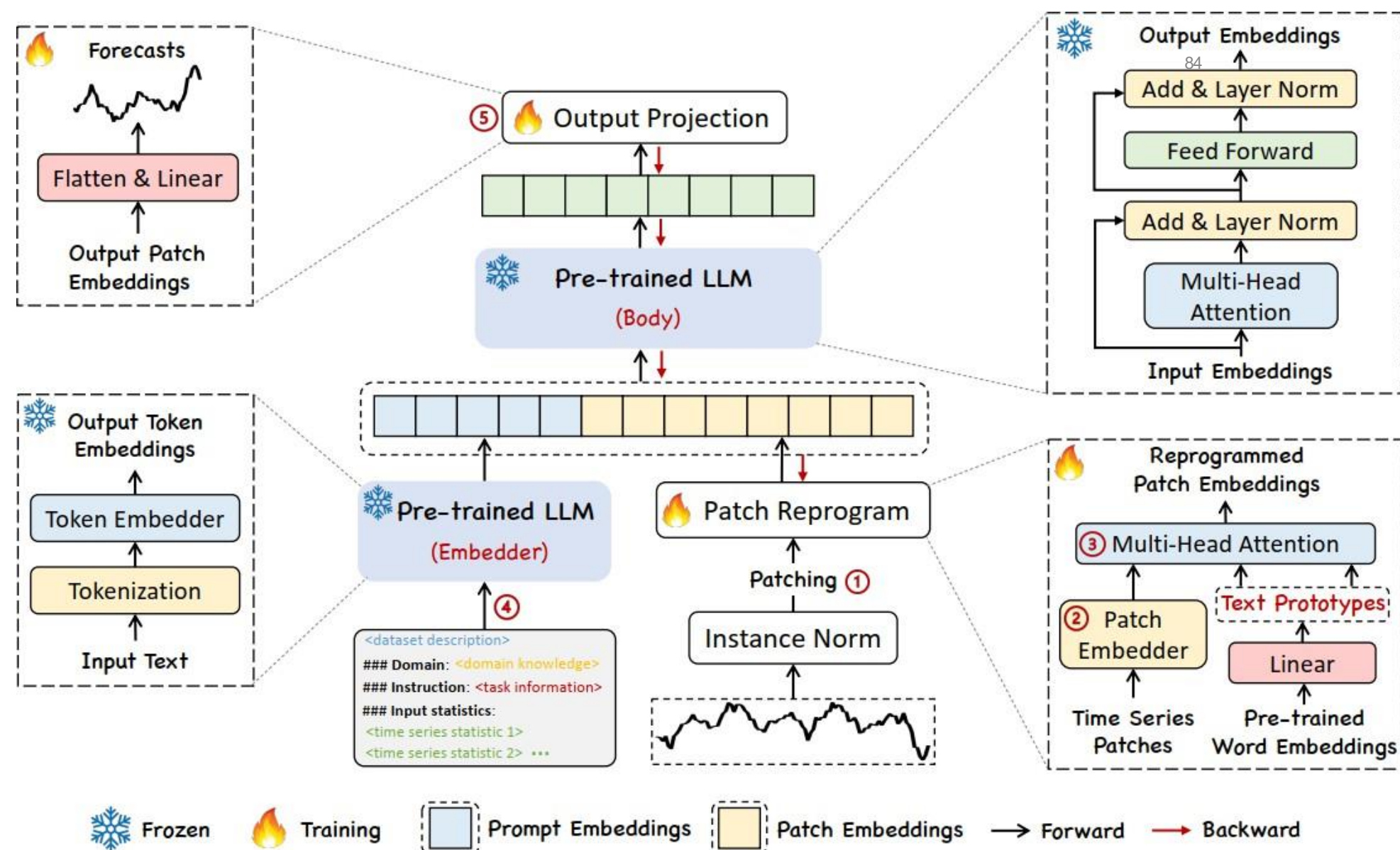
Time-LLM for Time Series Analysis

- Key question: **How to enable LLMs to understand time series?**



Contribution 1: Patch Reprogramming

- This module aims to **reprogram** time series features using pre-trained word embeddings in the backbone



Contribution 2: Prompt-as-Prefix

- **Dataset context** furnishes LLM with essential background information concerning the input time series
- **Task instruction** serves as a crucial guide in the transformation of patch embeddings for specific tasks
- We also enrich with additional crucial **statistics**, such as trends and lags, to facilitate pattern recognition and reasoning

The Electricity Transformer Temperature (ETT) indicates the electric power long-term deployment. Each data point consists of the target oil temperature and 6 power load features ... Below is the information about the input time series:

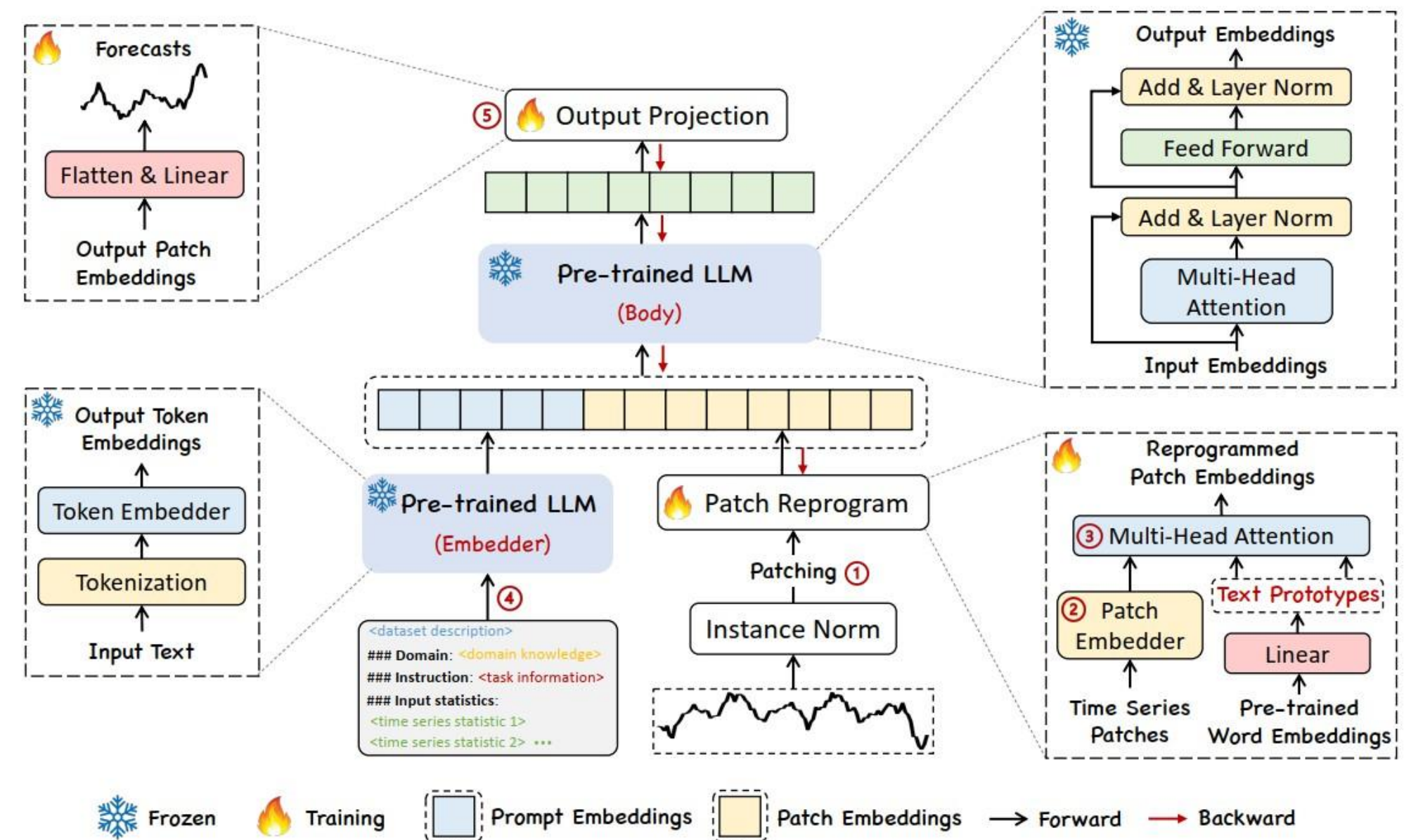
[BEGIN DATA]

[Domain]: We usually observe that electricity consumption peaks at noon, with a significant increase in transformer load

[Instruction]: Predict the next $\langle H \rangle$ steps given the previous $\langle T \rangle$ steps information attached

[Statistics]: The input has a minimum of $\langle \text{min_val} \rangle$, a maximum of $\langle \text{max_val} \rangle$, and a median of $\langle \text{median_val} \rangle$. The overall trend is $\langle \text{upward or downward} \rangle$. The top five lags are $\langle \text{lag_val} \rangle$.

[END DATA]



Results on Long-Term Forecasting

Table 1: Long-term forecasting results. We use forecasting horizons $H \in \{96, 192, 336, 720\}$. A lower value indicates better performance. **Red**: the best, **Blue**: the second best.

Methods		TIME-LLM		GPT4TS		DLinear		PatchTST		TimesNet		FEDformer		Autoformer		Stationary		ETSformer		LightTS		Informer		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT_{h1}	96	0.362	0.392	0.376	<u>0.397</u>	0.375	0.399	<u>0.370</u>	0.399	0.384	0.402	0.376	0.419	0.449	0.459	0.513	0.491	0.494	0.479	0.424	0.432	0.865	0.713	0.837	0.728
	192	0.398	<u>0.418</u>	0.416	0.418	<u>0.405</u>	0.416	0.413	0.421	0.436	0.429	0.420	0.448	0.500	0.482	0.534	0.504	0.538	0.504	0.475	0.462	1.008	0.792	0.923	0.766
	336	<u>0.430</u>	0.427	0.442	<u>0.433</u>	0.439	0.443	0.422	0.436	0.491	0.469	0.459	0.465	0.521	0.496	0.588	0.535	0.574	0.521	0.518	0.488	1.107	0.809	1.097	0.835
	720	0.442	0.457	0.477	0.456	0.472	0.490	<u>0.447</u>	<u>0.466</u>	0.521	0.500	0.506	0.507	0.514	0.512	0.643	0.616	0.562	0.535	0.547	0.533	1.181	0.865	1.257	0.889
	Avg	0.408	0.423	0.465	0.455	0.422	0.437	<u>0.413</u>	<u>0.430</u>	0.458	0.450	0.440	0.460	0.496	0.487	0.570	0.537	0.542	0.510	0.491	0.479	1.040	0.795	1.029	0.805
ETT_{h2}	96	0.268	0.328	0.285	0.342	0.289	0.353	<u>0.274</u>	<u>0.336</u>	0.340	0.374	0.358	0.397	0.346	0.388	0.476	0.458	0.340	0.391	0.397	0.437	3.755	1.525	2.626	1.317
	192	0.329	0.375	0.354	0.389	0.383	0.418	<u>0.339</u>	<u>0.379</u>	0.402	0.414	0.429	0.439	0.456	0.452	0.512	0.493	0.430	0.439	0.520	0.504	5.602	1.931	11.12	2.979
	336	<u>0.368</u>	<u>0.409</u>	0.373	0.407	0.448	0.465	0.329	0.380	0.452	0.452	0.496	0.487	0.482	0.486	0.552	0.551	0.485	0.479	0.626	0.559	4.721	1.835	9.323	2.769
	720	0.372	0.420	0.406	0.441	0.605	0.551	<u>0.379</u>	<u>0.422</u>	0.462	0.468	0.463	0.474	0.515	0.511	0.562	0.560	0.500	0.497	0.863	0.672	3.647	1.625	3.874	1.697
	Avg	<u>0.334</u>	<u>0.383</u>	0.381	0.412	0.431	0.446	0.330	0.379	0.414	0.427	0.437	0.449	0.450	0.459	0.526	0.516	0.439	0.452	0.602	0.543	4.431	1.729	6.736	2.191
ETT_{m1}	96	0.272	0.334	0.292	0.346	0.299	0.343	<u>0.290</u>	<u>0.342</u>	0.338	0.375	0.379	0.419	0.505	0.475	0.386	0.398	0.375	0.398	0.374	0.400	0.672	0.571	0.538	0.528
	192	0.310	0.358	0.332	0.372	0.335	0.365	<u>0.332</u>	<u>0.369</u>	0.374	0.387	0.426	0.441	0.553	0.496	0.459	0.444	0.408	0.410	0.400	0.407	0.795	0.669	0.658	0.592
	336	0.352	0.384	<u>0.366</u>	0.394	0.369	<u>0.386</u>	<u>0.366</u>	0.392	0.410	0.411	0.445	0.459	0.621	0.537	0.495	0.464	0.435	0.428	0.438	0.438	1.212	0.871	0.898	0.721
	720	0.383	0.411	0.417	0.421	0.425	0.421	<u>0.416</u>	<u>0.420</u>	0.478	0.450	0.543	0.490	0.671	0.561	0.585	0.516	0.499	0.462	0.527	0.502	1.166	0.823	1.102	0.841
	Avg	0.329	0.372	0.388	0.403	0.357	<u>0.378</u>	<u>0.351</u>	0.380	0.400	0.406	0.448	0.452	0.588	0.517	0.481	0.456	0.429	0.425	0.435	0.437	0.961	0.734	0.799	0.671
ETT_{m2}	96	0.161	0.253	0.173	0.262	0.167	0.269	<u>0.165</u>	<u>0.255</u>	0.187	0.267	0.203	0.287	0.255	0.339	0.192	0.274	0.189	0.280	0.209	0.308	0.365	0.453	0.658	0.619
	192	0.219	0.293	0.229	<u>0.301</u>	<u>0.224</u>	0.303	0.220	0.292	0.249	0.309	0.269	0.328	0.281	0.340	0.280	0.339	0.253	0.319	0.311	0.382	0.533	0.563	1.078	0.827
	336	0.271	0.329	0.286	<u>0.341</u>	0.281	0.342	<u>0.274</u>	0.329	0.321	0.351	0.325	0.366	0.339	0.372	0.334	0.361	0.314	0.357	0.442	0.466	1.363	0.887	1.549	0.972
	720	0.352	0.379	0.378	0.401	0.397	0.421	<u>0.362</u>	<u>0.385</u>	0.408	0.403	0.421	0.415	0.433	0.432	0.417	0.413	0.414	0.413	0.675	0.587	3.379	1.338	2.631	1.242
	Avg	0.251	0.313	0.284	0.339	0.267	0.333	<u>0.255</u>	<u>0.315</u>	0.291	0.333	0.305	0.349	0.327	0.371	0.306	0.347	0.293	0.342	0.409	0.436	1.410	0.810	1.479	0.915
1 st Count		18		0		1		<u>4</u>		0		0		0		0		0		0		0		0	

Results on Few-Shot Learning

Table 3: Few-shot learning on 10% training data. We use the same protocol and notations as in [Tab. 1](#).

Methods		TIME-LLM		GPT4TS		DLinear		PatchTST		TimesNet		FEDformer		Autoformer		Stationary		ETSformer		LightTS		Informer		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT_h1	96	0.448	0.460	<u>0.458</u>	0.456	0.492	0.495	0.516	0.485	0.861	0.628	0.512	0.499	0.613	0.552	0.918	0.639	1.112	0.806	1.298	0.838	1.179	0.792	1.184	0.790
	192	0.484	0.483	<u>0.570</u>	<u>0.516</u>	<u>0.565</u>	0.538	0.598	0.524	0.797	0.593	0.624	0.555	0.722	0.598	0.915	0.629	1.155	0.823	1.322	0.854	1.199	0.806	1.295	0.850
	336	0.589	0.540	<u>0.608</u>	0.535	0.721	0.622	0.657	0.550	0.941	0.648	0.691	0.574	0.750	0.619	0.939	0.644	1.179	0.832	1.347	0.870	1.202	0.811	1.294	0.854
	720	0.700	0.604	<u>0.725</u>	<u>0.591</u>	0.986	0.743	0.762	0.610	0.877	0.641	0.728	0.614	0.721	0.616	0.887	0.645	1.273	0.874	1.534	0.947	1.217	0.825	1.223	0.838
	Avg.	0.556	0.522	<u>0.590</u>	<u>0.525</u>	0.691	0.600	0.633	0.542	0.869	0.628	0.639	0.561	0.702	0.596	0.915	0.639	1.180	0.834	1.375	0.877	1.199	0.809	1.249	0.833
ETT_h2	96	0.275	0.326	<u>0.331</u>	<u>0.374</u>	0.357	0.411	0.353	0.389	0.378	0.409	0.382	0.416	0.413	0.451	0.389	0.411	0.678	0.619	2.022	1.006	3.837	1.508	3.788	1.533
	192	0.374	0.373	<u>0.402</u>	<u>0.411</u>	0.569	0.519	0.403	0.414	0.490	0.467	0.478	0.474	0.474	0.477	0.473	0.455	0.785	0.666	2.329	1.104	3.856	1.513	3.552	1.483
	336	0.406	0.429	0.406	<u>0.433</u>	0.671	0.572	<u>0.426</u>	0.441	0.537	0.494	0.504	0.501	0.547	0.543	0.507	0.480	0.839	0.694	2.453	1.122	3.952	1.526	3.395	1.526
	720	0.427	0.449	<u>0.449</u>	<u>0.464</u>	0.824	0.648	0.477	0.480	0.510	0.491	0.499	0.509	0.516	0.523	0.477	0.472	1.273	0.874	3.816	1.407	3.842	1.503	3.205	1.401
	Avg.	0.370	0.394	<u>0.397</u>	<u>0.421</u>	0.605	0.538	0.415	0.431	0.479	0.465	0.466	0.475	0.488	0.499	0.462	0.455	0.894	0.713	2.655	1.160	3.872	1.513	3.485	1.486
ETT_m1	96	0.346	0.388	0.390	0.404	<u>0.352</u>	0.392	0.410	0.419	0.583	0.501	0.578	0.518	0.774	0.614	0.761	0.568	0.911	0.688	0.921	0.682	1.162	0.785	1.442	0.847
	192	0.373	<u>0.416</u>	0.429	0.423	<u>0.382</u>	0.412	0.437	0.434	0.630	0.528	0.617	0.546	0.754	0.592	0.781	0.574	0.955	0.703	0.957	0.701	1.172	0.793	1.444	0.862
	336	0.413	0.426	0.469	0.439	<u>0.419</u>	<u>0.434</u>	0.476	0.454	0.725	0.568	0.998	0.775	0.869	0.677	0.803	0.587	0.991	0.719	0.998	0.716	1.227	0.908	1.450	0.866
	720	0.485	<u>0.476</u>	0.569	0.498	<u>0.490</u>	0.477	0.681	0.556	0.769	0.549	0.693	0.579	0.810	0.630	0.844	0.581	1.062	0.747	1.007	0.719	1.207	0.797	1.366	0.850
	Avg.	0.404	0.427	0.464	0.441	<u>0.411</u>	<u>0.429</u>	0.501	0.466	0.677	0.537	0.722	0.605	0.802	0.628	0.797	0.578	0.980	0.714	0.971	0.705	1.192	0.821	1.426	0.856
ETT_m2	96	0.177	0.261	<u>0.188</u>	<u>0.269</u>	0.213	0.303	0.191	0.274	0.212	0.285	0.291	0.399	0.352	0.454	0.229	0.308	0.331	0.430	0.813	0.688	3.203	1.407	4.195	1.628
	192	0.241	0.314	<u>0.251</u>	<u>0.309</u>	0.278	0.345	0.252	0.317	0.270	0.323	0.307	0.379	0.694	0.691	0.291	0.343	0.400	0.464	1.008	0.768	3.112	1.387	4.042	1.601
	336	0.274	0.327	0.307	<u>0.346</u>	0.338	0.385	<u>0.306</u>	0.353	0.323	0.353	0.543	0.559	2.408	1.407	0.348	0.376	0.469	0.498	1.031	0.775	3.255	1.421	3.963	1.585
	720	0.417	0.390	<u>0.426</u>	<u>0.417</u>	0.436	0.440	0.433	0.427	0.474	0.449	0.712	0.614	1.913	1.166	0.461	0.438	0.589	0.557	1.096	0.791	3.909	1.543	3.711	1.532
	Avg.	0.277	0.323	<u>0.293</u>	<u>0.335</u>	0.316	0.368	0.296	0.343	0.320	0.353	0.463	0.488	1.342	0.930	0.332	0.366	0.447	0.487	0.987	0.756	3.370	1.440	3.978	1.587
1 st Count		20		<u>3</u>		2		0		0		0		0		0		0		0		0		0	

Results on Zero-Shot Learning

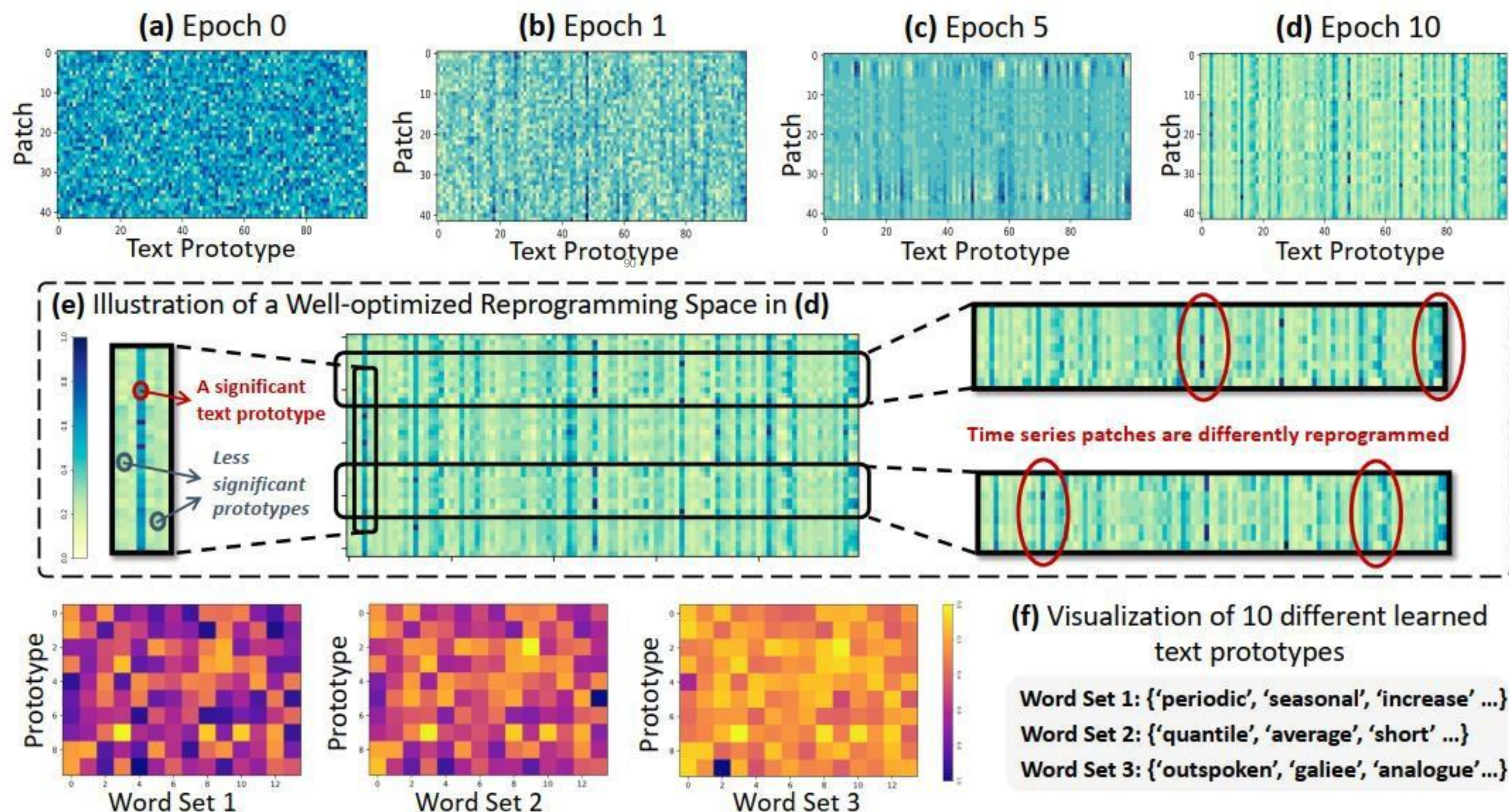
Table 5: Zero-shot learning results. **Red**: the best, Blue: the second best. **Appendix D** shows complete results.

Methods	TIME-LLM		GPT4TS		DLinear		PatchTST		TimesNet		Autoformer	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh1 \rightarrow ETTh2$	0.353	0.387	0.406	0.422	0.493	0.488	<u>0.380</u>	<u>0.405</u>	0.421	0.431	0.582	0.548
$ETTh1 \rightarrow ETTm2$	0.273	0.340	0.325	0.363	0.415	0.452	<u>0.314</u>	<u>0.360</u>	0.327	0.361	0.457	0.483
$ETTh2 \rightarrow ETTh1$	0.479	0.474	0.757	0.578	0.703	0.574	<u>0.565</u>	<u>0.513</u>	0.865	0.621	0.757	0.608
$ETTh2 \rightarrow ETTm2$	0.272	0.341	0.335	0.370	0.328	0.386	<u>0.325</u>	<u>0.365</u>	0.342	0.376	0.366	0.411
$ETTh1 \rightarrow ETTm2$	0.381	0.412	<u>0.433</u>	0.439	0.464	0.475	0.439	<u>0.438</u>	0.457	0.454	0.470	0.479
$ETTh2 \rightarrow ETTm1$	0.268	0.320	0.313	0.348	0.335	0.389	<u>0.296</u>	<u>0.334</u>	0.322	0.354	0.469	0.484
$ETTh1 \rightarrow ETTm1$	0.354	0.400	0.435	0.443	0.455	0.471	<u>0.409</u>	<u>0.425</u>	0.435	0.443	0.423	0.439
$ETTh2 \rightarrow ETTm1$	0.414	0.438	0.769	0.567	0.649	0.537	<u>0.568</u>	<u>0.492</u>	0.769	0.567	0.755	0.591

Ablation Study

Variant	Long-term Forecasting				Few-shot Forecasting			
	ETTh1-96	ETTh1-192	ETTm1-96	ETThm1-192	ETTh1-96	ETTh1-192	ETTm1-96	ETThm1-192
A.1 Llama (Default; 32)	0.362	0.398	0.272	0.310	0.448	0.484	0.346	0.373
A.2 Llama (8)	0.389	0.412	0.297	0.329	0.567	0.632	0.451	0.490
A.3 GPT-2 (12)	0.385	0.419	0.306	0.332	0.548	0.617	0.447	0.509
A.4 GPT-2 (6)	0.394	0.427	0.311	0.342	0.571	0.640	0.468	0.512
B.1 w/o Patch Reprogramming	0.410	0.412	0.310	0.342	0.498	0.570	0.445	0.487
B.2 w/o Prompt-as-Prefix	0.398	0.423	0.298	0.339	0.521	0.617	0.432	0.481
C.1 w/o Dataset Context	0.402	0.417	0.298	0.331	0.491	0.538	0.392	0.447
C.2 w/o Task Instruction	0.388	0.420	0.285	0.327	0.476	0.529	0.387	0.439
C.3 w/o Statistical Context	0.391	0.419	0.279	0.347	0.483	0.547	0.421	0.461

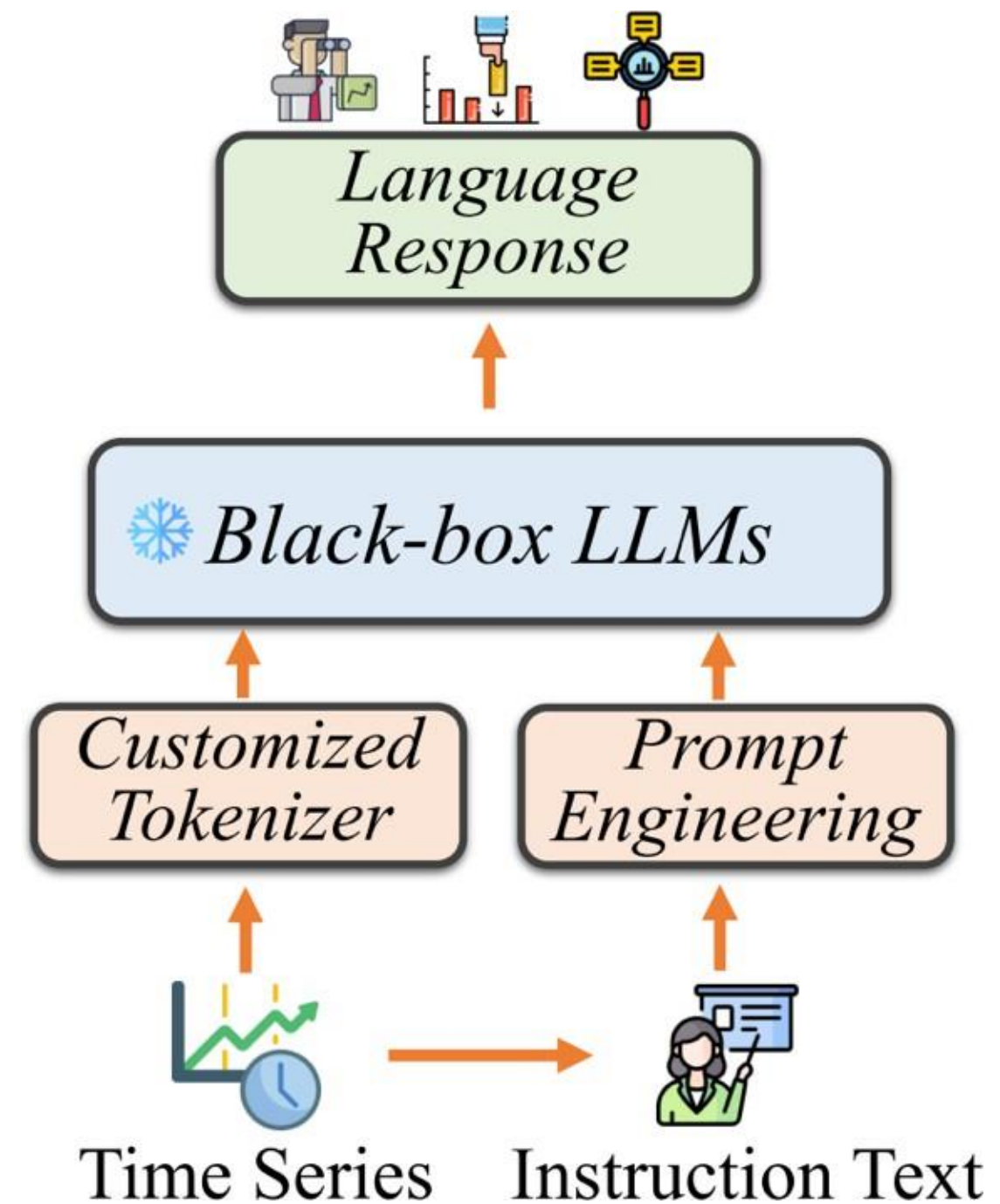
Visualization on Reprogramming



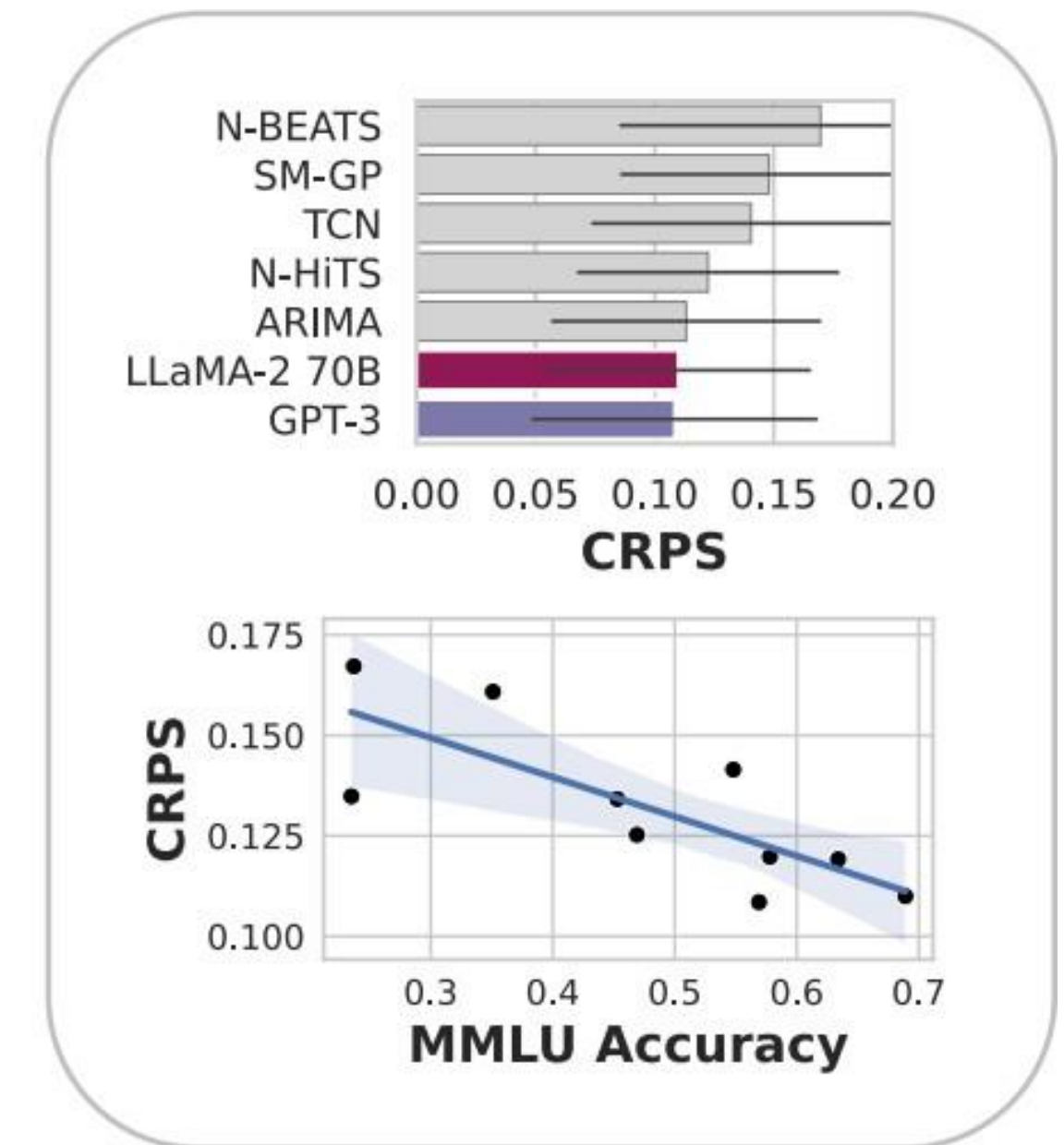
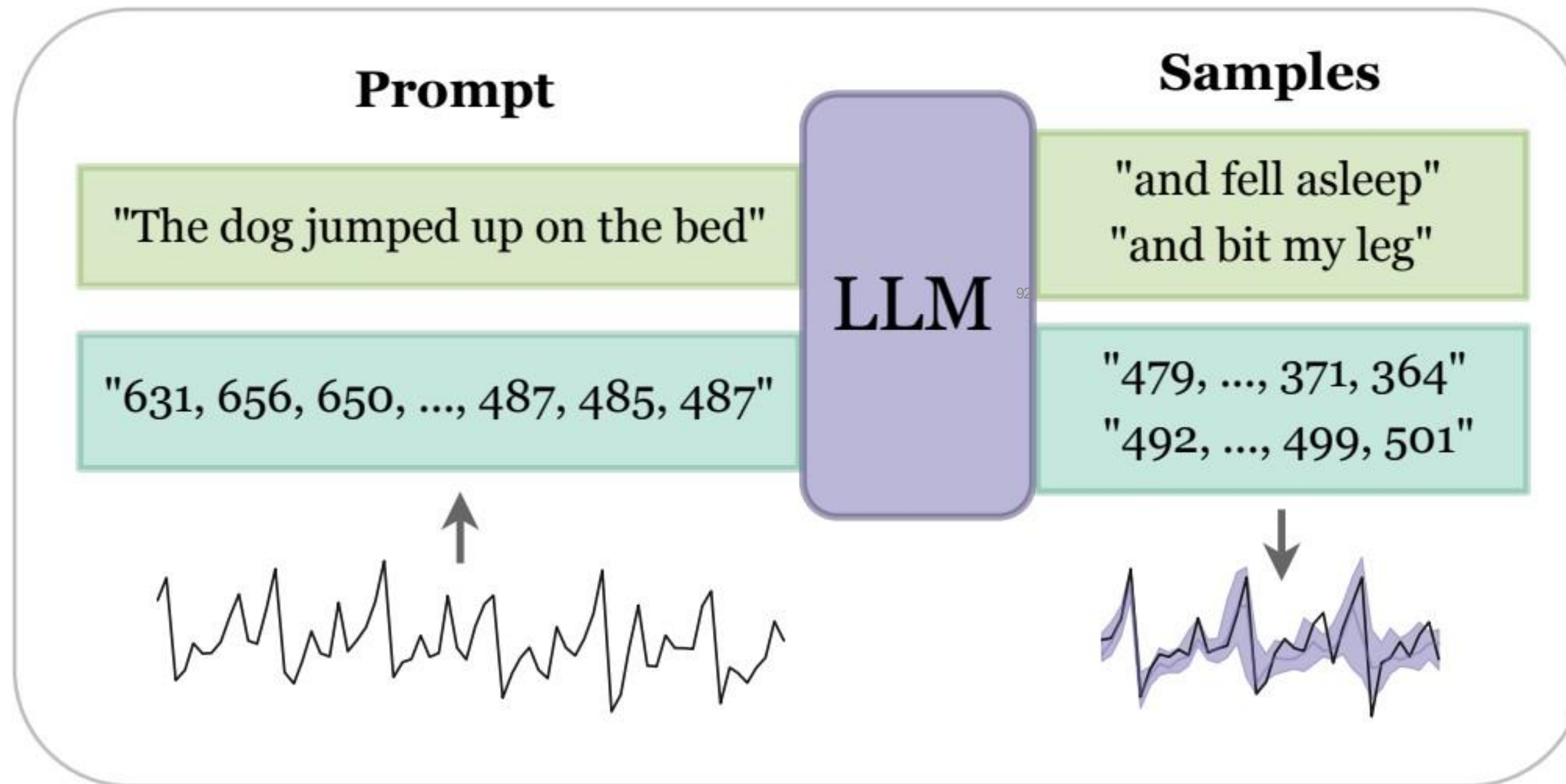
Non-Tuning-based LLM Predictors

- Non-tuning-based predictors, suitable for closed-source models, involve **preprocessing TS data to fit LLM input spaces**
- Examples
 - LLMTime for time series analysis
 - LLM-Mob for human mobility

91

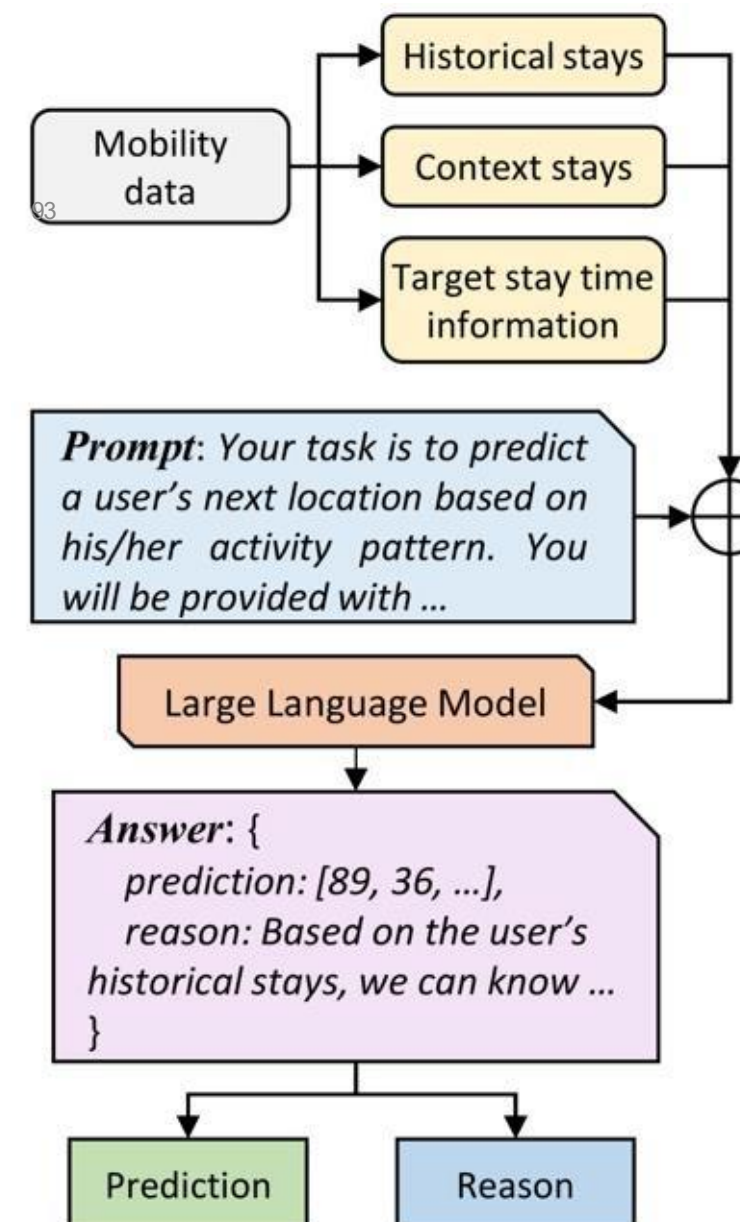
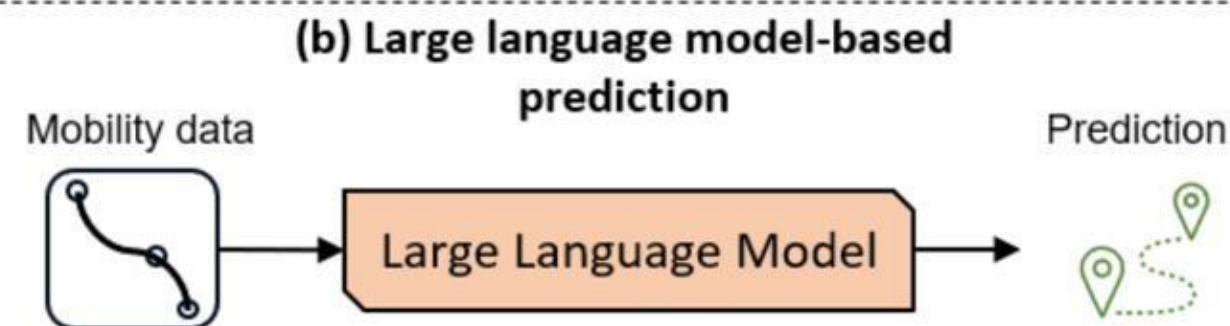
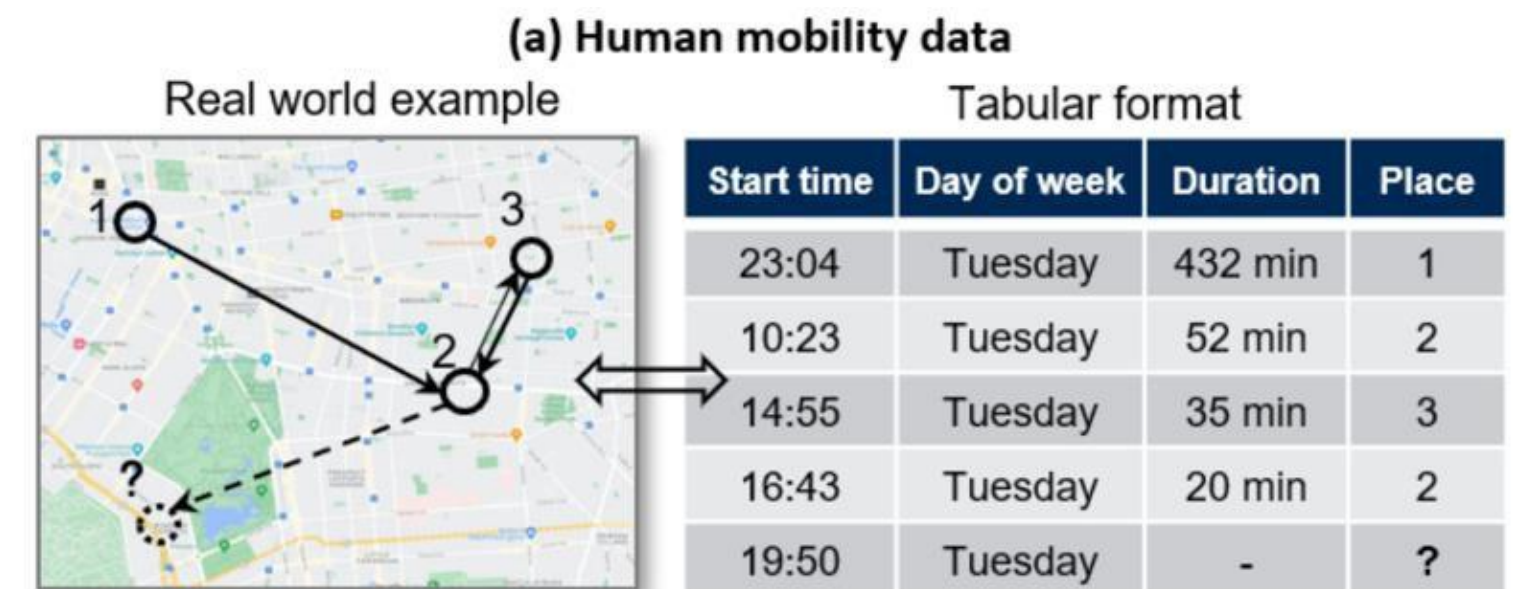


Example: LLMTime



Example: LLM-Mob

- LLM-Mob leverages the language understanding and reasoning capabilities of LLMs for analyzing human mobility data



Instruction | Specify the task Your task is to predict a user's next location based on his/her activity pattern.

Data | Describe the data You will be provided with `<history>` which is a list containing this user's historical stays, then `<context>` which provide contextual information about where and when this user has been to recently. Stays in both `<history>` and `<context>` are in chronological order. Each stay takes on such form as (start_time, day_of_week, duration, place_id). The detailed explanation of each element is as follows:
start_time: the start time of the stay in 12h clock format.
day_of_week: indicating the day of the week.
duration: an integer indicating the duration (in minute) of each stay. Note that this will be None in the `<target_stay>` introduced later.
place_id: an integer representing the unique place ID, which indicates where the stay is.
Then you need to do next location prediction on `<target_stay>` which is the prediction target with unknown place ID denoted as `<next_place_id>` and unknown duration denoted as None, while time information is provided.

Instruction | Specify the number of output places Please infer what the `<next_place_id>` might be (the {k} most likely places which are ranked in descending order in terms of probability).

Instruction | Guide the model to "think" Please consider the following aspects:
1. the activity pattern of this user that you learned from `<history>`, e.g., repeated visits to certain places during certain times;
2. the context stays in `<context>`, which provide more recent activities of this user;
3. the temporal information (i.e., start_time and day_of_week) of target stay, which is important because people's activity varies during different time (e.g., nighttime versus daytime) and on different days (e.g., weekday versus weekend).

Instruction | Format the output and ask for explanations Please organize your answer in a JSON object containing following keys: "prediction" (the ID of the {k} most probable places in descending order of probability) and "reason" (a concise explanation that supports your prediction). Do not include line breaks in your output.

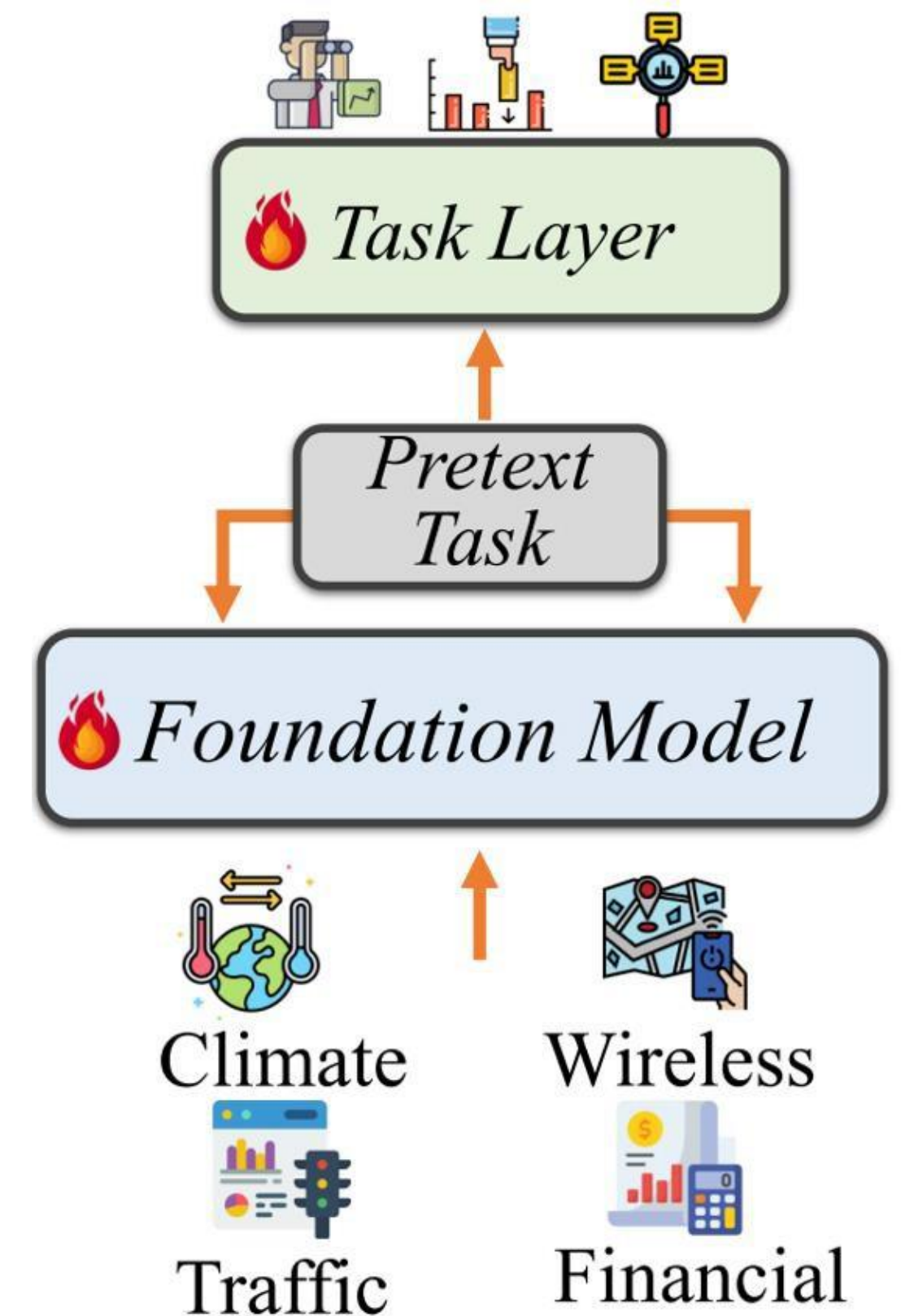
Data | Provide the data The data are as follows:
`<history>`: {historical_stays}
`<context>`: {context_stays}
`<target_stay>`: {target_stay}

Other LLM-as-Predictors

- Beyond the previously discussed methods, another significant approach in TS analysis involves **building foundation models from scratch**

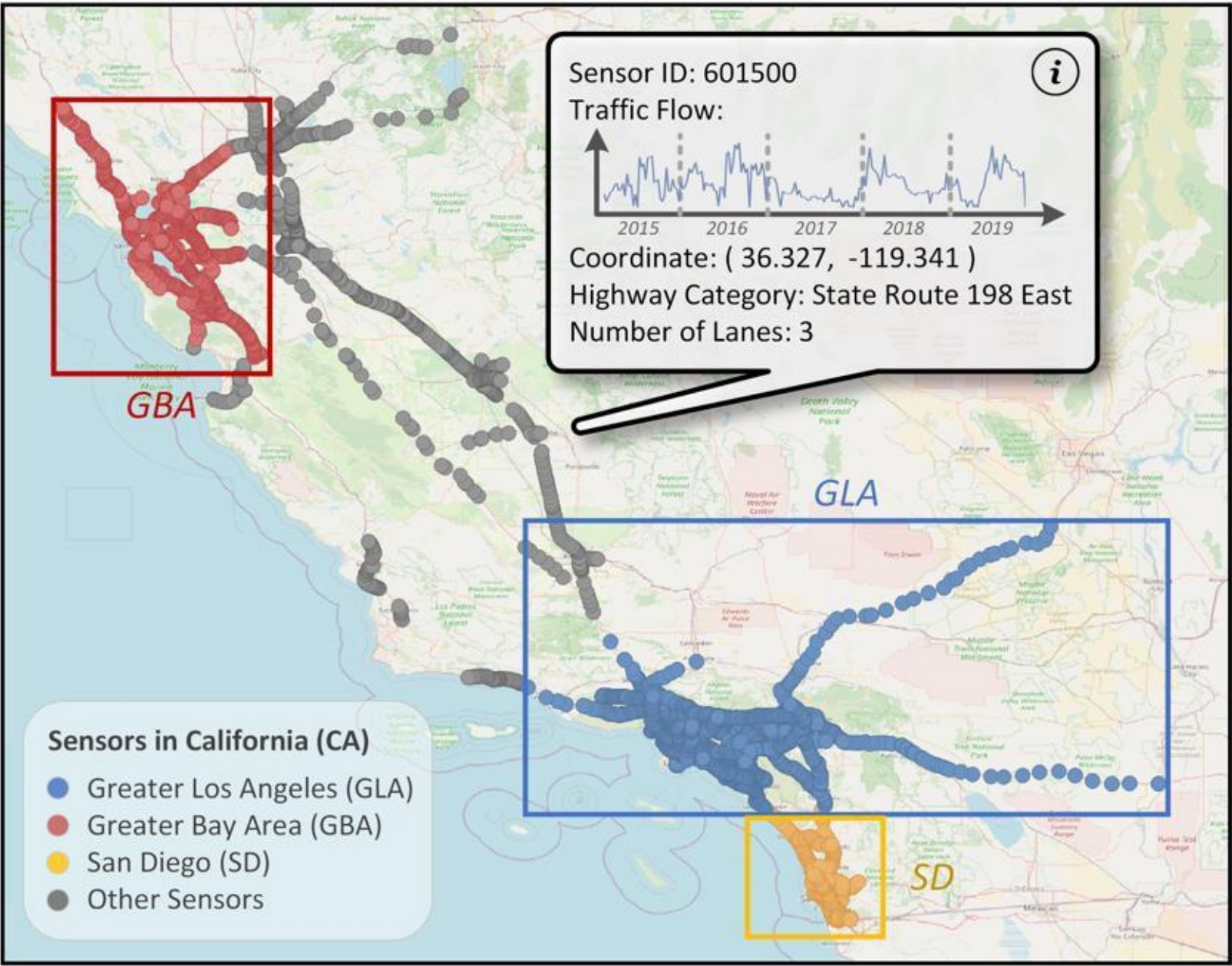
A major challenge:

Large-scale datasets!



LargeST: A Large-Scale Traffic Dataset

- We crawled, processed, and released a large-scale traffic dataset
 - Scalability, Generalizability, Robustness, etc.



95

Dataset	Nodes	Edges	Degree	Meta	Time Range	Frames	Data Points
PeMSD7(M)	228	1,664	7.3	6	05/01/2012 – 06/30/2012	12,672	2.89M
PeMSD7(L)	1,026	14,534	14.2	0	05/01/2012 – 06/30/2012	12,672	13.00M
METR-LA	207	1,515	7.3	3	03/01/2012 – 06/27/2012	34,272	7.09M
PEMS-BAY	325	2,369	7.3	3	01/01/2017 – 06/30/2017	52,116	16.94M
PEMS03	358	546	1.5	1	09/01/2018 – 11/30/2018	26,208	9.38M
PEMS04	307	338	1.1	0	01/01/2018 – 02/28/2018	16,992	5.22M
PEMS07	883	865	1.0	0	05/01/2017 – 08/06/2017	28,224	24.92M
PEMS08	170	276	1.6	0	07/01/2016 – 08/31/2016	17,856	3.04M
CA	8,600	201,363	23.4	9	01/01/2017 – 12/31/2021	525,888	4.52B
GLA	3,834	98,703	25.7	9	01/01/2017 – 12/31/2021	525,888	2.02B
GBA	2,352	61,246	26.0	9	01/01/2017 – 12/31/2021	525,888	1.24B
SD	716	17,319	24.2	9	01/01/2017 – 12/31/2021	525,888	0.38B

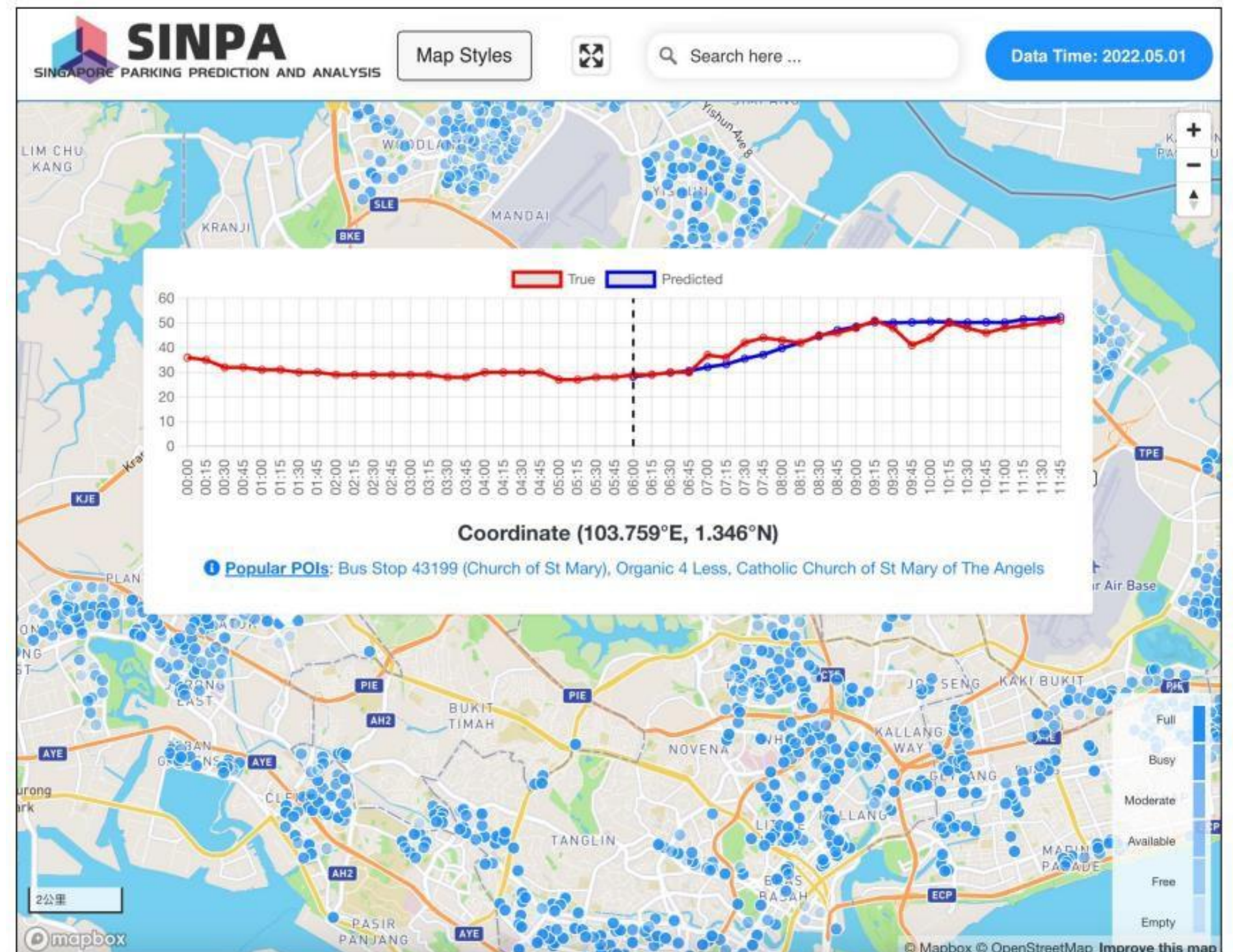


SINPA: A Large-Scale Carpark Dataset

- We crawled, processed, and released a large-scale parking dataset

Data	Attributes	Values
Parking Availability	Time Spans	2020/07/01 - 2021/06/30
	Time Granularity	15 minutes
	# Parking lots	1,687
	# Samples	14,600
	Missing Rate	1.59%
Meteorology	Temperature (mean)	27.72
	Humidity (mean)	81.18
	Windspeed (mean)	4.62
Time-related	Time of Day	24
	Day of the Week	7
	Holiday	2
Geolocation	# Pln_Area Categories	36
Utilization Type	# Categories	10
Road Networks	# Segments	15,291

96



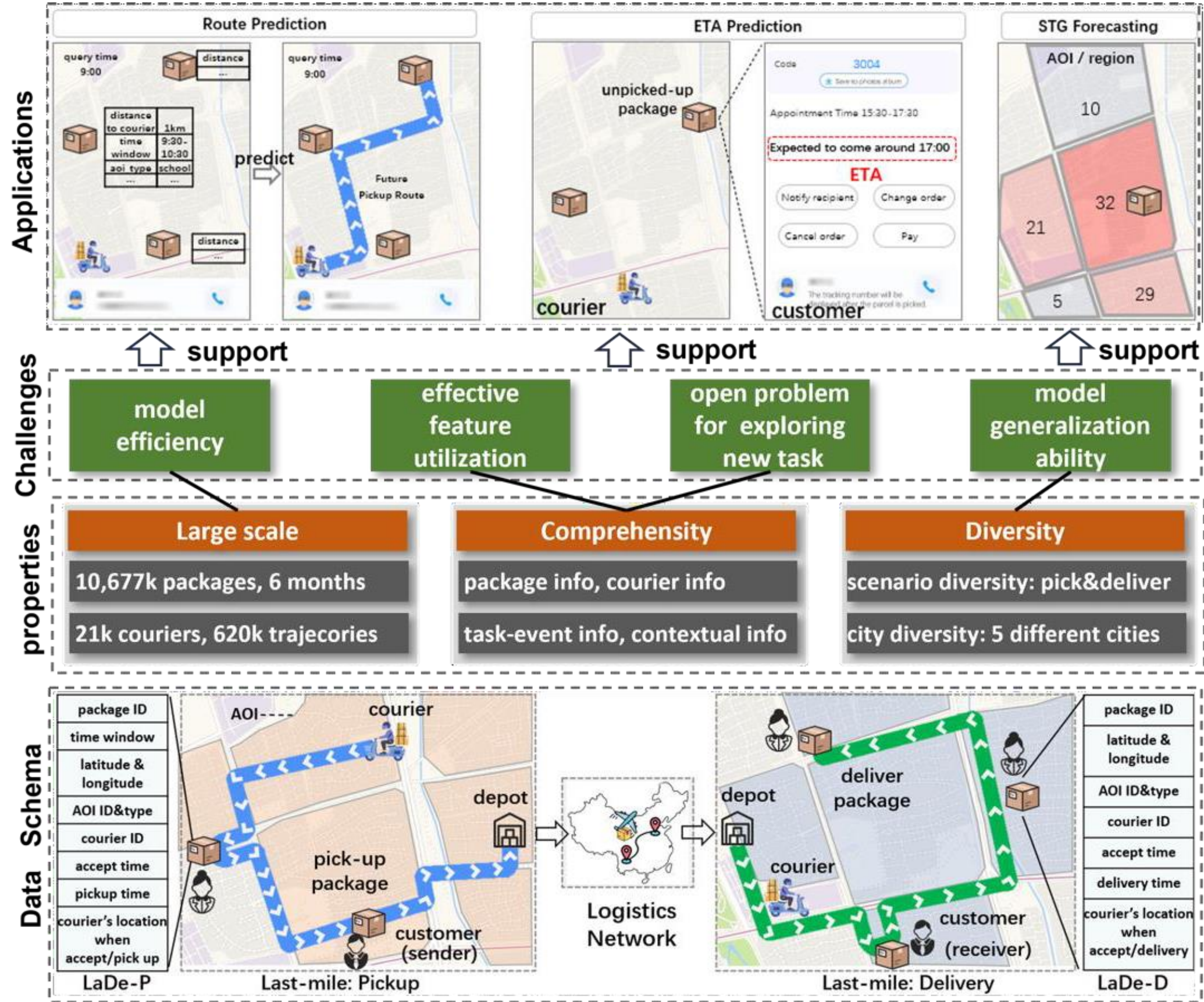
LaDe: The First Last-Mile Delivery Dataset

- We released **The First Last-Mile Delivery Dataset from Industry**

- Large scale
- Comprehensivity
- Diversity

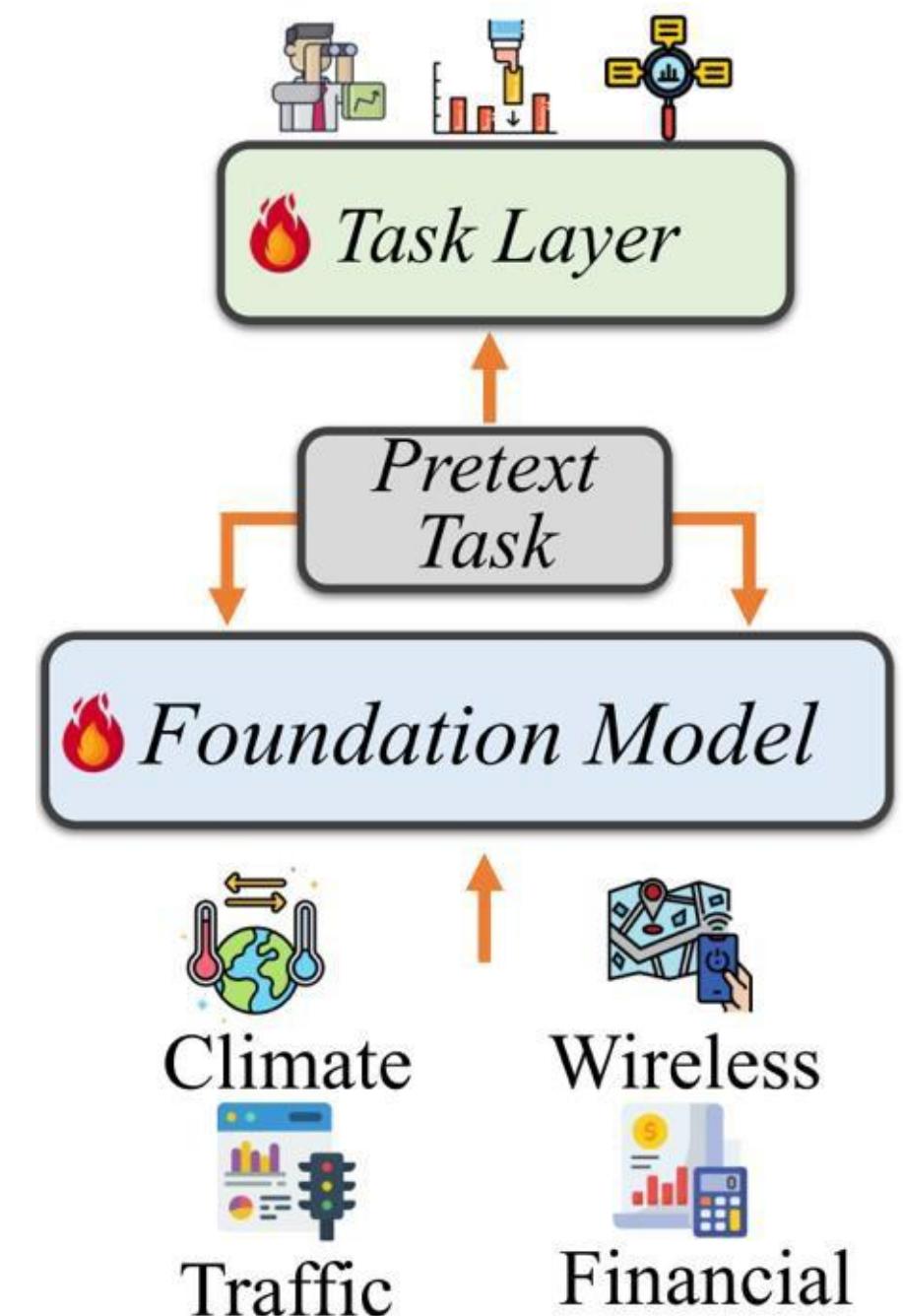
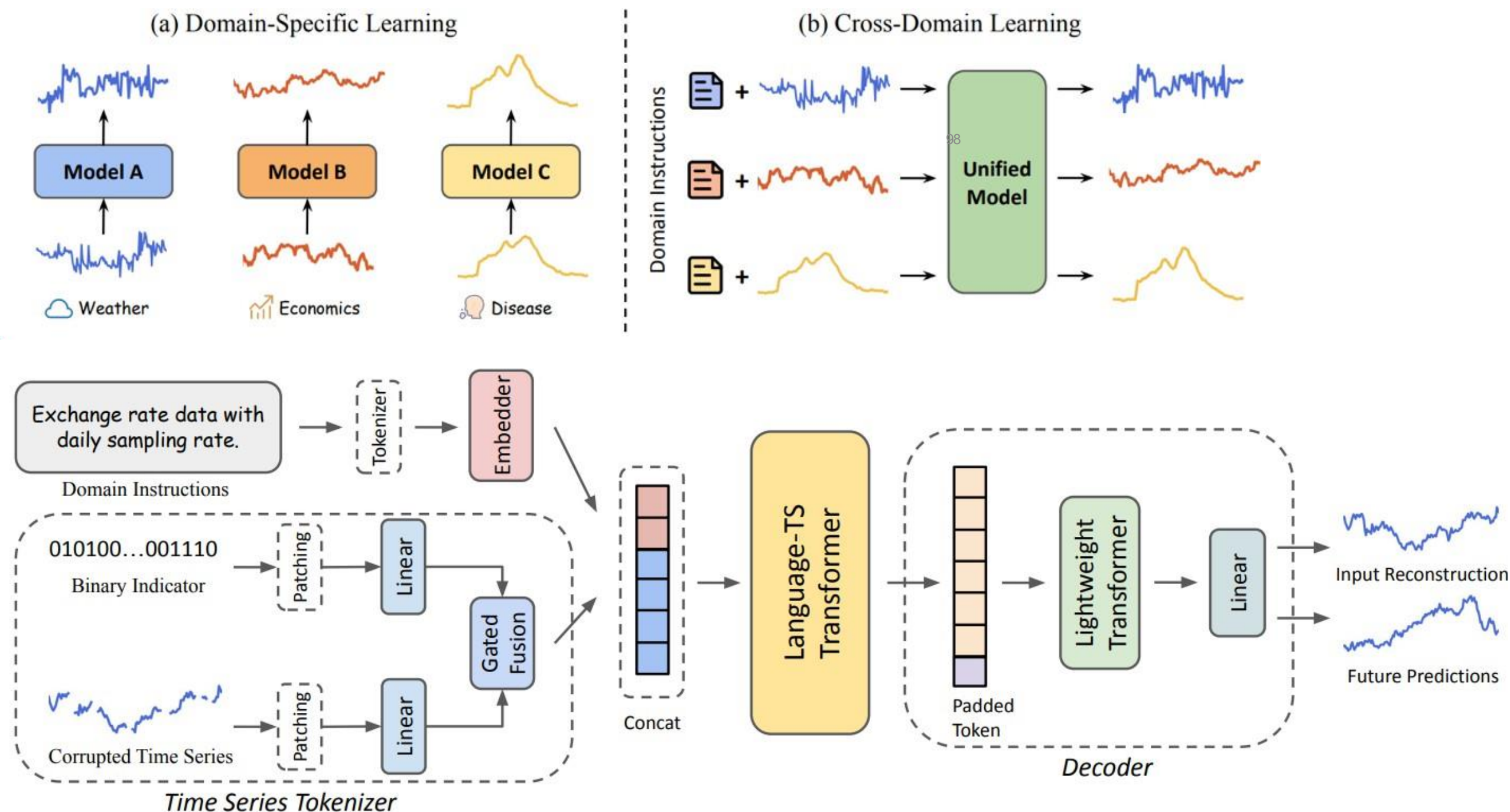


97



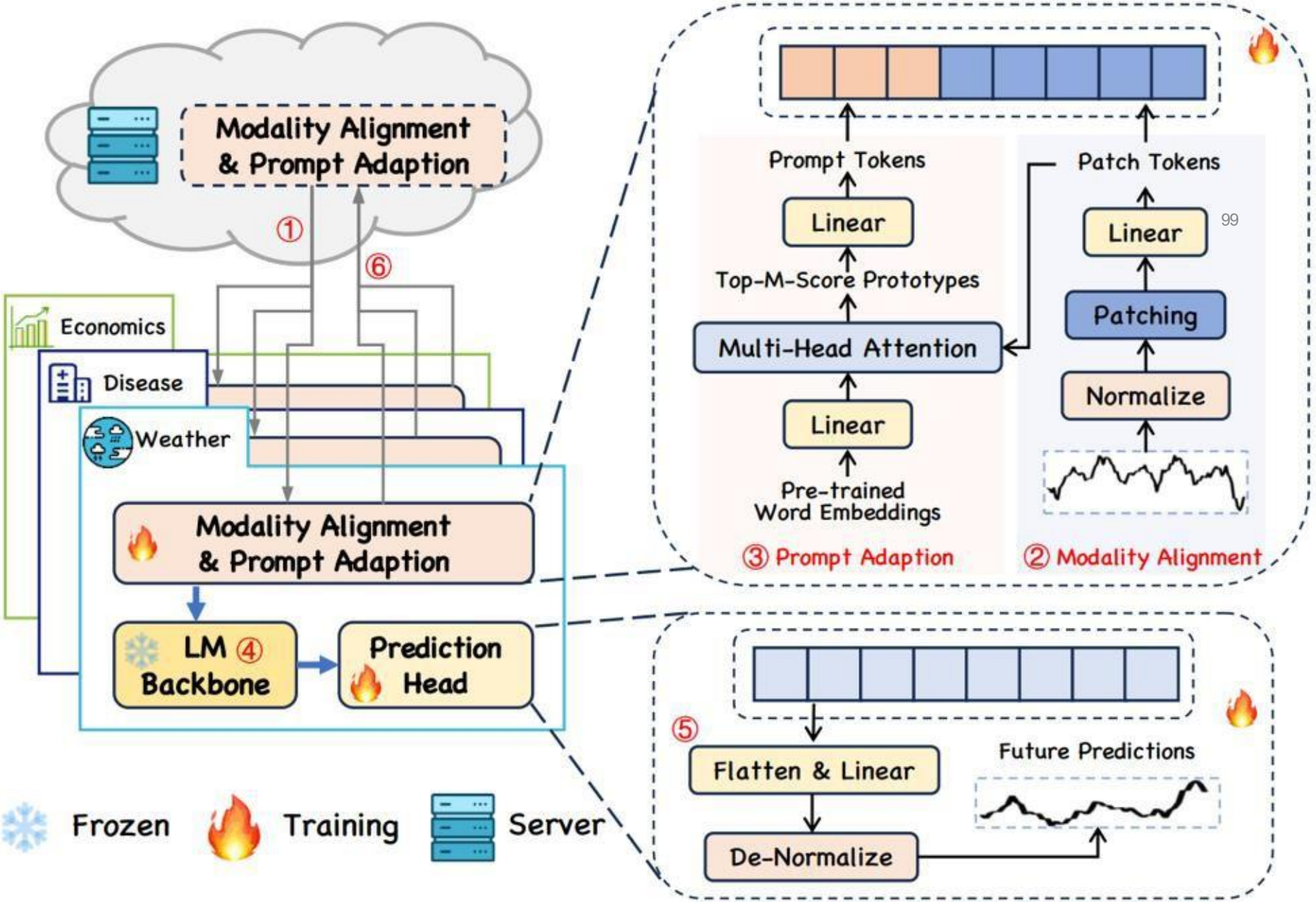
UniTime: Building Time Series Foundation Models

- The prerequisite of training a **Foundation Model** for time series is **training a model on cross-domain time series all at once**

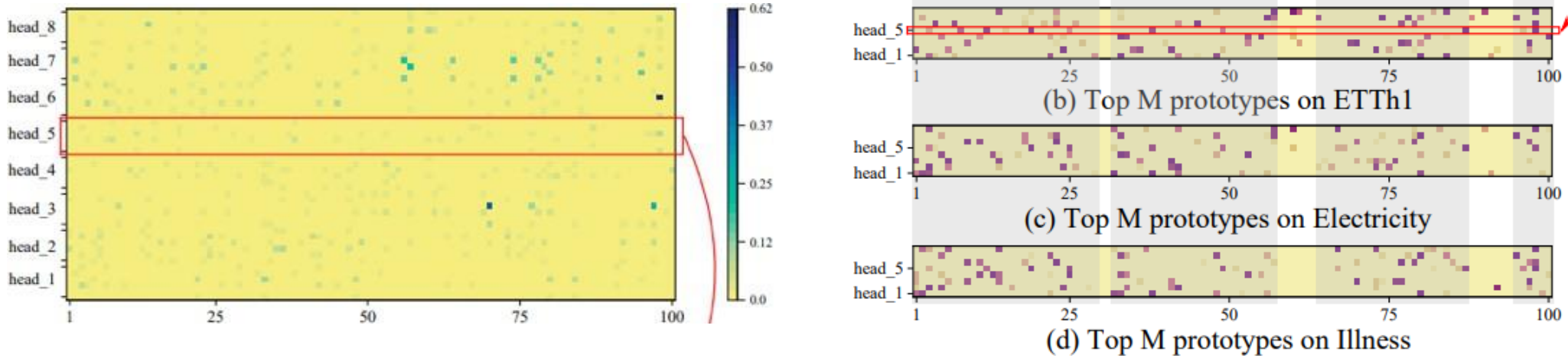


Time-FFM: Federated TS Learning

- What if the input data come from **difference sources**?



Type	TY1				TY2				TY3			
Method	TIME-FFM	FedIT	FedAdapter ^H	FedAdapter ^P	UniTime	GPT4TS	PatchTST	TimesNet	DLinear	FEDformer	Autoformer	Informer
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.442	0.434	0.481	0.461	0.488	0.467	0.503	0.479	0.442	0.448	0.502	0.461
ETTh2	0.382	0.406	0.374	0.396	0.373	0.398	0.380	0.403	0.378	0.403	0.386	0.406
ETTm1	0.399	0.402	0.644	0.517	0.643	0.511	0.640	0.516	0.385	0.399	0.551	0.483
ETTm2	0.286	0.332	0.297	0.341	0.295	0.340	0.298	0.342	0.293	0.334	0.321	0.356
Electricity	0.216	0.299	0.390	0.478	0.408	0.489	0.334	0.420	0.216	0.305	0.251	0.338
Weather	0.270	0.288	0.282	0.310	0.282	0.308	0.287	0.309	0.253	0.276	0.293	0.309
Exchange	0.338	0.391	0.389	0.423	0.382	0.419	0.380	0.417	0.364	0.404	0.421	0.446
ILI	2.107	0.924	4.423	1.448	5.247	1.621	5.251	1.600	2.137	0.929	3.678	1.372
Average	0.555	0.434	0.910	0.547	1.015	0.569	1.009	0.561	0.559	0.437	0.800	0.521
1 st Count	8	1	1	0	3	0	0	0	4	0	1	0



LLM-as-Agent

- Our standpoint is that LLMs can serve as the central hub for understanding and advancing time series analysis in three principal ways
 - **LLM-as-Enhancers**: augmenting TS data and existing approaches with enhanced external knowledge and analytical prowess
 - **LLM-as-Predictors**: utilizing their extensive internal knowledge and emerging reasoning abilities to benefit a range of downstream tasks, e.g., forecasting
 - **LLM-as-Agents**: transcending conventional roles to actively engage in and transform time series analysis

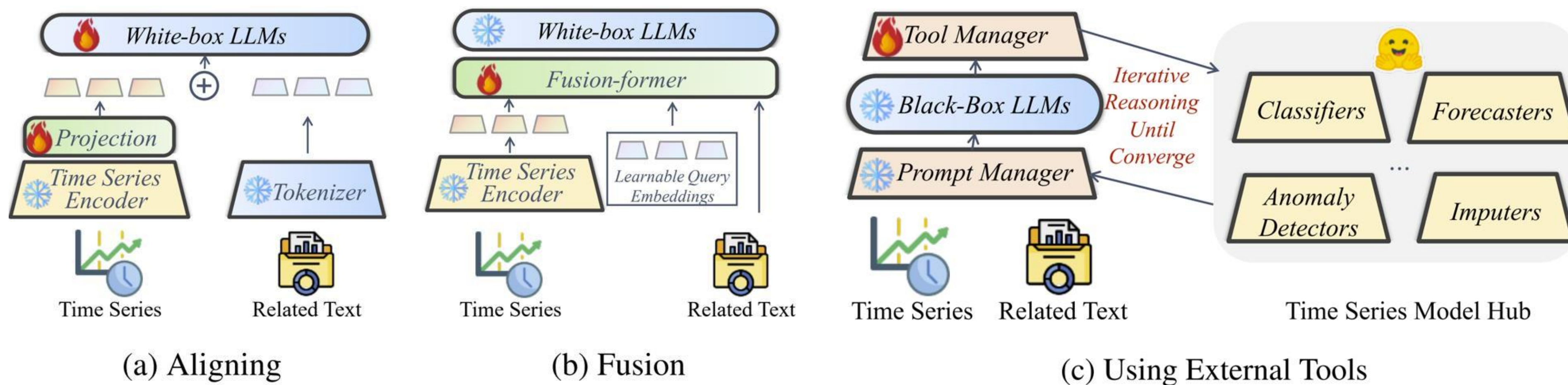
Background of LLM-as-Agent

- **Tuning-based LLM-as-Predictors** utilize LLMs as robust model checkpoints, attempting to adjust certain parameters for specific domain applications.
 - However, this approach often **sacrifices the interactive capabilities of LLMs** and may not fully exploit the benefits offered by LLMs, such as in-context learning or chain-of-thought.
- **Non-tuning-based LLM-as-Predictors**, integrating TS data into textual formats or developing specialized tokenizers
 - Facing limitations due to LLMs' primary training on linguistic data, hindering their comprehension of complex temporal knowledge and patterns not easily captured in language

A new promising paradigm rises:
LLM-as-Agents!

LLM-as-Agent for Time Series Analysis

- Different directions for incorporating time series knowledge to LLMs



LLM-as-Agent for Time Series Analysis

- We empirically verify that LLM can serve as Effective Analytical Agent

Data format description:

The Human Activity Recognition database was built from the recordings of 30 study participants performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors. The objective is to classify activities into one of the four activities performed (Walk, Sit, Stand, Lay). For each record in the dataset the following is provided:

1. Triaxial acceleration from the accelerometer (total¹⁰ acceleration) and the estimated body acceleration.
2. Triaxial Angular velocity from the gyroscope.

Few shot data instances:

Here are some classification examples:

```
tBodyAcc-mean()-X tBodyAcc-mean()-Y tBodyAcc-mean()-Z angle(X,gravityMean) angle(Y,gravityMean) angle(Z,gravityMean) Activity
0.257178 -0.02329 -0.01465 -0.72001 0.276801 -0.05798 STANDING
0.277235 -0.02568 -0.11843 -0.8159 0.166483 -0.09444 SITTING
0.300905 -0.02361 -0.09689 0.774053 -0.47774 -0.50714 LAYING
```

Task description:

Now we have the following data instances, please justify their Activity:

```
0.269066 -0.02769 -0.11018 -0.6805 0.302289 -0.06727
0.278291 -0.01454 -0.10524 -0.81653 0.170525 -0.09102
0.272059 -0.02127 -0.10064 0.767363 -0.4718 -0.51501
```



Certainly, based on a general understanding of the patterns in the provided Human Activity Recognition dataset, here are hypothetical guesses for the activities corresponding to the given instances:

1. The activity for the first instance (0.269066, -0.02769, -0.11018, -0.6805, 0.302289, -0.06727) might be **Walk**.
2. The activity for the second instance (0.278291, -0.01454, -0.10524, -0.81653, 0.170525, -0.09102) could be **Sit**.
3. The activity for the third instance (0.272059, -0.02127, -0.10064, 0.767363, -0.4718, -0.51501) may be **Lay**.



Correct answer of instances 2, 3 !



LLM-as-Agent for Time Series Analysis

- LLM agent can provide **interpretability** and **truthfulness**

Please tell me the reason.



It's important to note that these are speculative guesses based on general patterns observed in accelerometer and gyroscope data for common human activities. The reasoning behind these guesses is influenced by typical characteristics associated with certain activities:

Instance 1 The relatively consistent acceleration values along with angular velocity may suggest a pattern associated with walking. **Walking often involves rhythmic and repetitive motion.**

Instance 2 The lower acceleration values and certain angular velocity characteristics might resemble the patterns seen during sitting. **Sitting typically involves less movement compared to other activities.**

Instance 3 The specific combination of acceleration and angular velocity features could resemble the patterns observed during lying down or laying. **Laying down is associated with lower acceleration and distinct angular velocity patterns.**



Interpretability !

Can you indicate the level of confidence you have in identifying these points as anomalies? Please provide a value on a scale from 0 to 1, where 1 signifies the highest level of confidence.



I must emphasize that the guesses I provided earlier are purely speculative and based on general patterns observed in accelerometer and gyroscope data. Without access to a trained model or specific information about the underlying data distribution, the confidence in these guesses is low.

If I were to assign a confidence level on a scale from 0 to 1, where 1 signifies the highest confidence, it would be approximately 0.3. **This low confidence level reflects the speculative nature of the guesses and the lack of a precise understanding of the data distribution and model specifics. For accurate predictions, a trained model with associated confidence scores would be necessary.**



Truthfulness !

Limitations of LLM Agent for Time Series

- Difficulty in Understanding Complex Patterns
- Bias and Task Preferences
- **Hallucination**

Data format description:

The Electric Transformer Temperature (ETT) serves as a vital indicator for long-term electric power management. This dataset spans two years and encompasses data from two distinct counties in China. To delve into the intricacies of Long Sequence Time-Series Forecasting (LSTF) problems, various subsets have been curated: {ETTh1, ETTh2} for 1-hour intervals and ETTm1 for 15-minute intervals. Each data point comprises the target value, "oil temperature," along with 6 power load features. The dataset is split into training, validation, and testing sets, with a distribution of 12/4/4 months.

Task description:

Now, I would like to provide you with some instances from the dataset. Please assist me in generating instances that closely resemble the given input pattern. The instances are as follows:

Few shot data instances:

date, HUFL, HULL, MUFL, MULL, LUFL, LULL, OT

2016-07-01 00:00:00,5.827000141143799,2.009000062942505,1.5989999771118164,0.4620000123977661,4.203000068664552,1.3400000333786009,30.5310001373291
2016-07-01 01:00:00,5.692999839782715,2.075999975204468,1.4919999837875366,0.4259999990463257,4.142000198364259,1.371000051498413,27.78700065612793
2016-07-01 00:00:00,5.827000141143799,2.009000062942505,1.5989999771118164,0.4620000123977661,4.203000068664552,1.3400000333786009,30.5310001373291
2016-07-01 01:00:00,5.692999839782715,2.075999975204468,1.4919999837875366,0.4259999990463257,4.142000198364259,1.371000051498413,27.78700065612793
2016-07-01 02:00:00,5.1570000648498535,1.741000056266785,1.2790000438690186,0.35499998927116394,3.776999950408936,1.218000054359436,27.78700065612793
2016-07-01 03:00:00,5.0900001525878915,1.9420000314712524,1.2790000438690186,0.3910000026226044,3.806999921798706,1.2790000438690186,25.04400062561035
2016-07-01 04:00:00,5.357999801635742,1.9420000314712524,1.4919999837875366,0.4620000123977661,3.868000030517578,1.2790000438690186,21.947999954223643
2016-07-01 05:00:00,5.625999927520752,2.1429998874664307,1.5279999971389768,0.5329999923706055,4.051000118255615,1.371000051498413,21.173999786376953

Certainly, here are instances from the dataset that closely resemble the given input pattern:

2016-07-01 01:00:00,5.692999839782715,2.075999975204468,1.4919999837875366,0.4259999990463257,4.142000198364259,1.371000051498413,27.78700065612793
2016-07-01 02:00:00,5.1570000648498535,1.741000056266785,1.2790000438690186,0.35499998927116394,3.776999950408936,1.218000054359436,27.78700065612793
2016-07-01 03:00:00,5.0900001525878915,1.9420000314712524,1.2790000438690186,0.3910000026226044,3.806999921798706,1.2790000438690186,25.04400062561035
2016-07-01 04:00:00,5.357999801635742,1.9420000314712524,1.4919999837875366,0.4620000123977661,3.868000030517578,1.2790000438690186,21.947999954223643

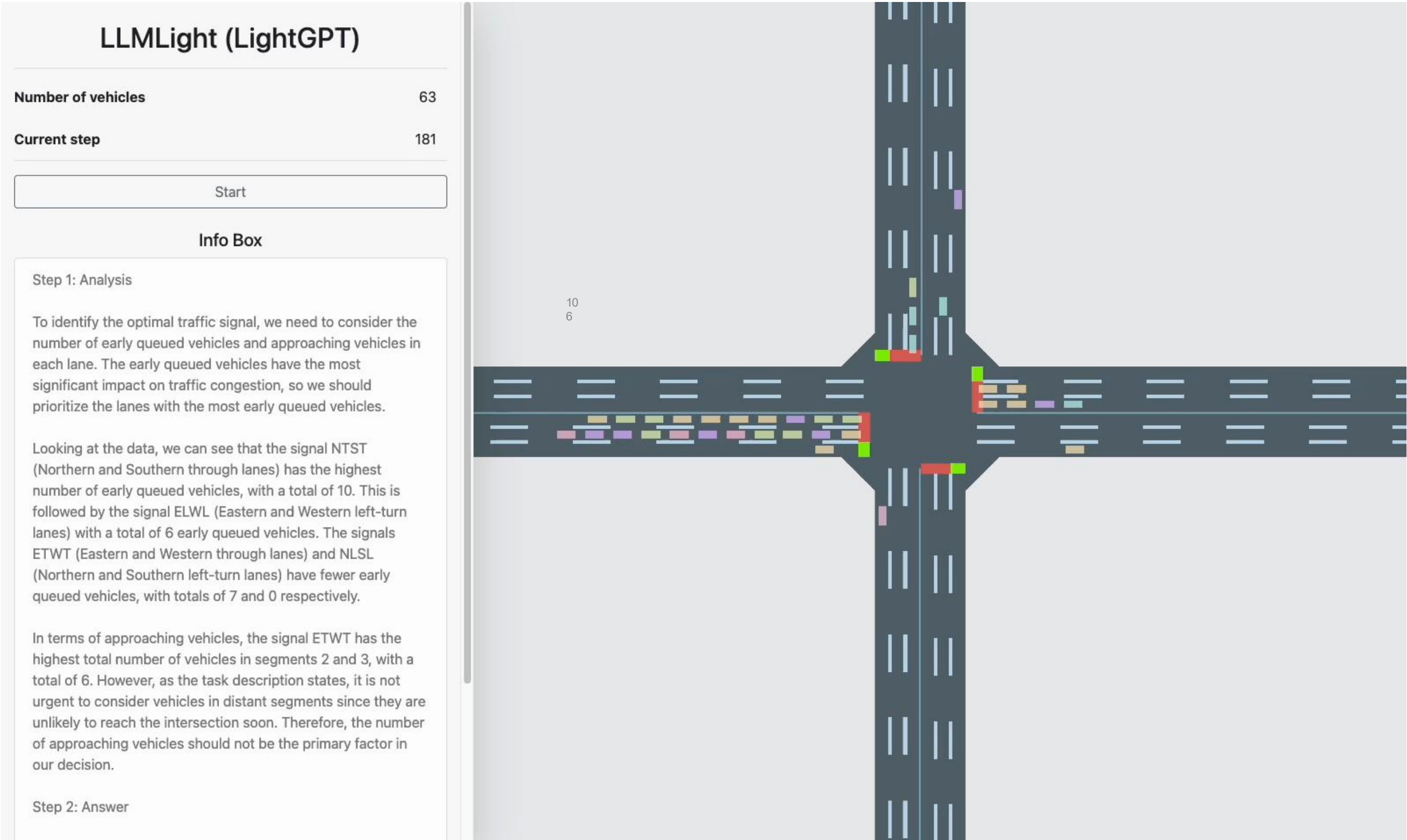
These instances continue the hourly trend of oil temperature and power load features, maintaining the structure and characteristics of the provided dataset.



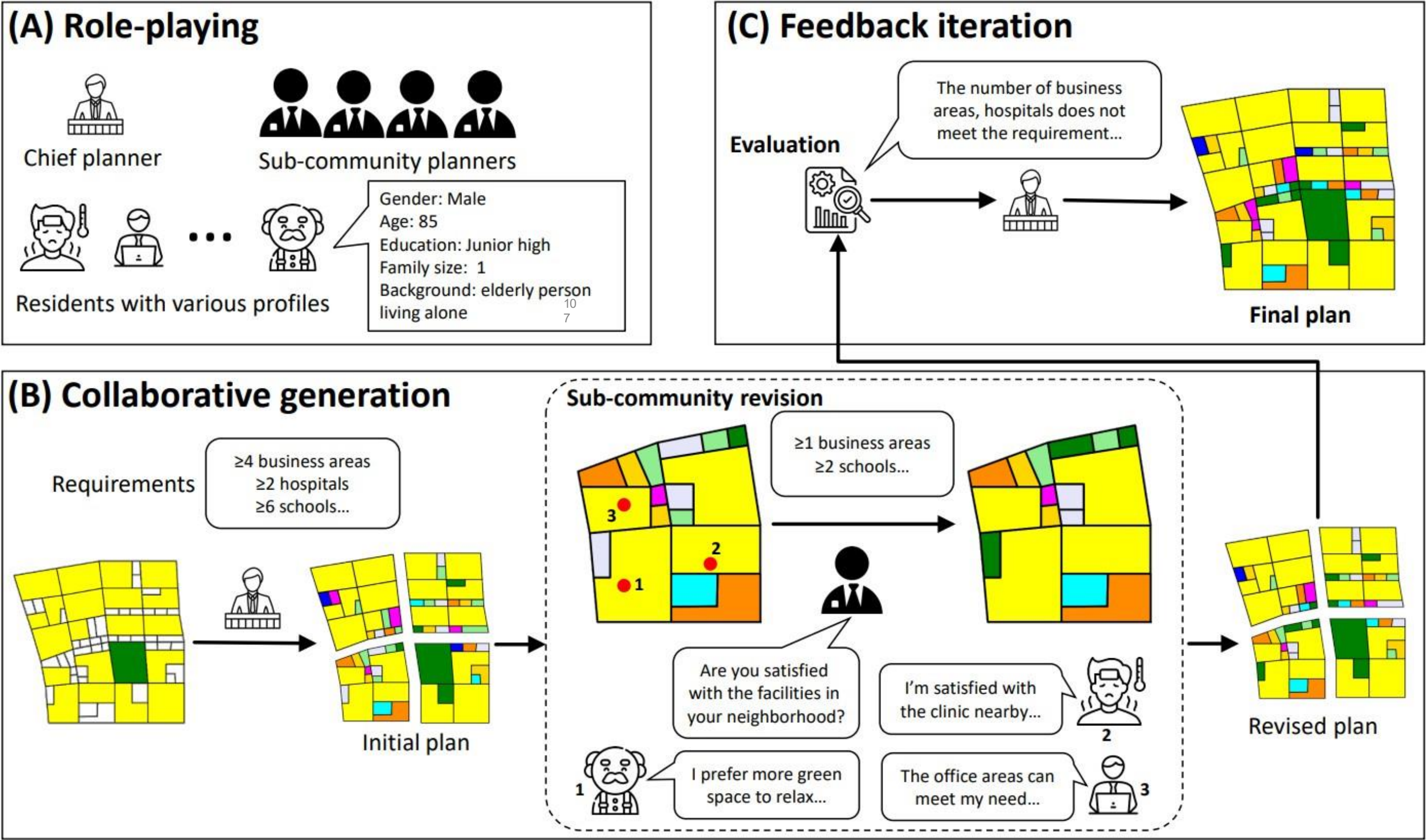
Hallucination !



LLM Agents for Traffic Light Control



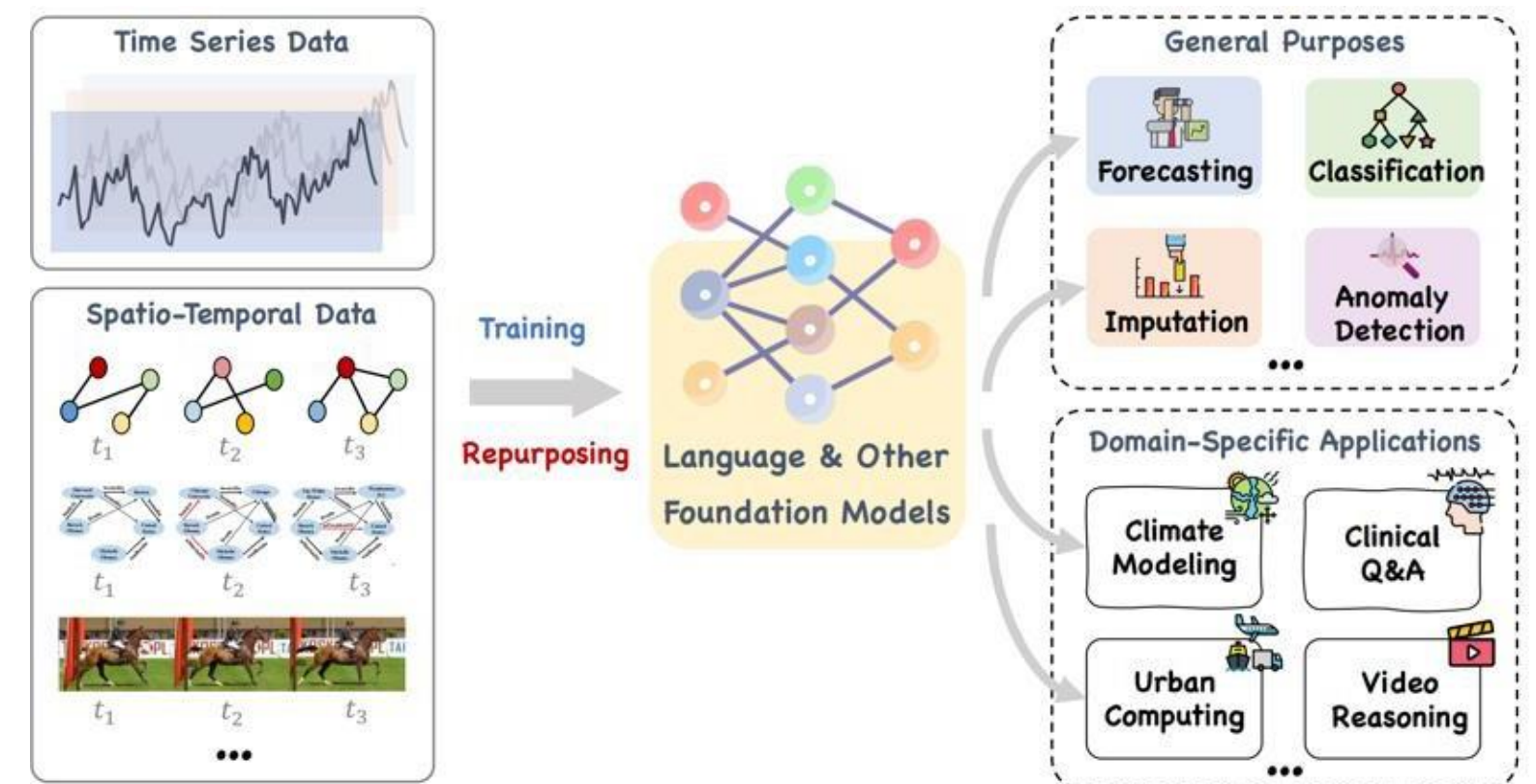
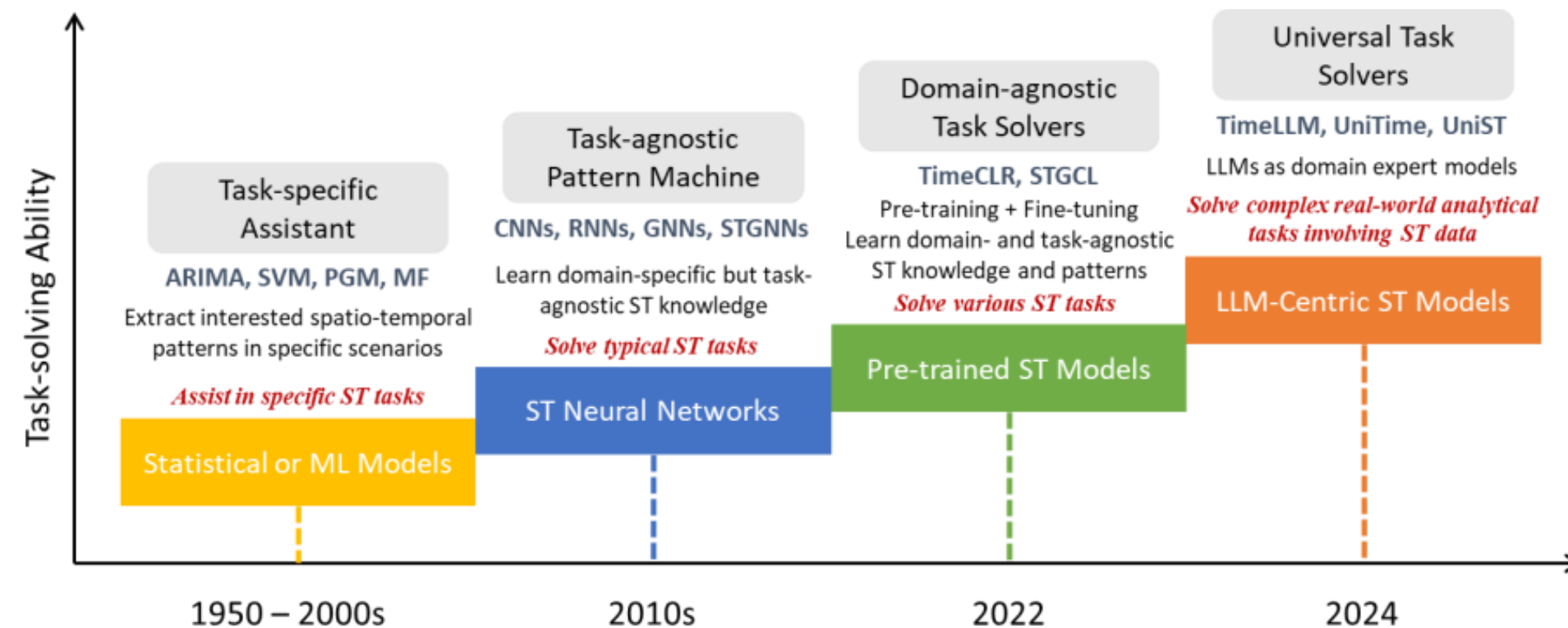
LLM Agents for Urban Planning



Summary

- Deep learning for time series analysis
- When spatio-temporal data meet LLMs
 - LLM-as-Enhancer
 - LLM-as-Predictor
 - LLM-as-Agent

10
8



Content

1 Spatio-temporal Data and Tasks

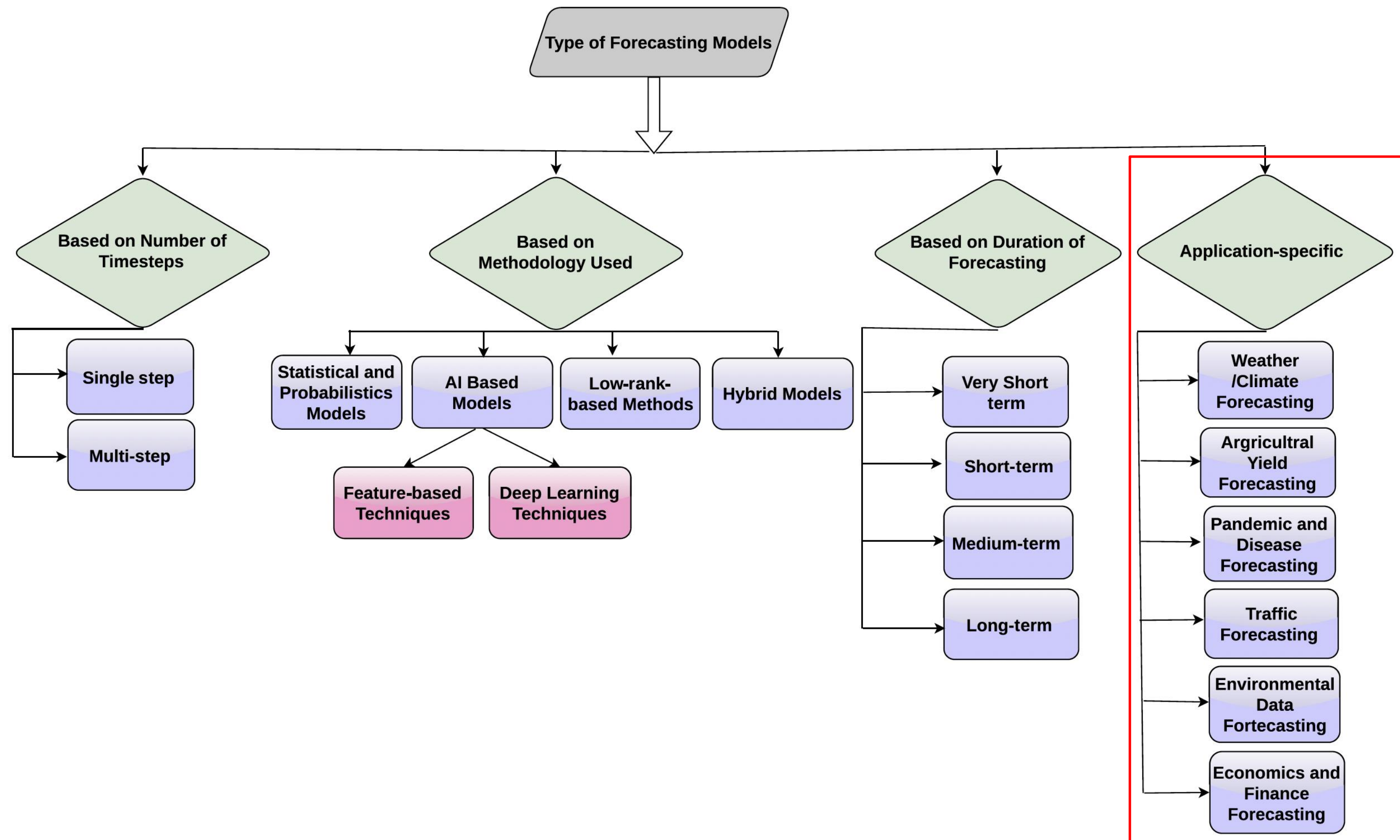
2 Problem Definitions and Evaluation Metrics

3 Spatio-temporal Modeling

4 Applications

5 Challenges and future works

Applications



Application - Transportation

With the rapid growth of transportation data from sensors (e.g., loop detectors, cameras, GPS), there is an urgent need to use deep learning to model the complex spatiotemporal correlations for tasks like:

- Traffic flow prediction
- Traffic incident detection
- Traffic congestion prediction

Transportation data can appear in various spatiotemporal forms:

- ST Raster: Traffic flow matrices (sensor \times time)
- Graphs: Sensor networks modeled by road connections
- Time Series: Single-road traffic histories

Modeling Approaches:

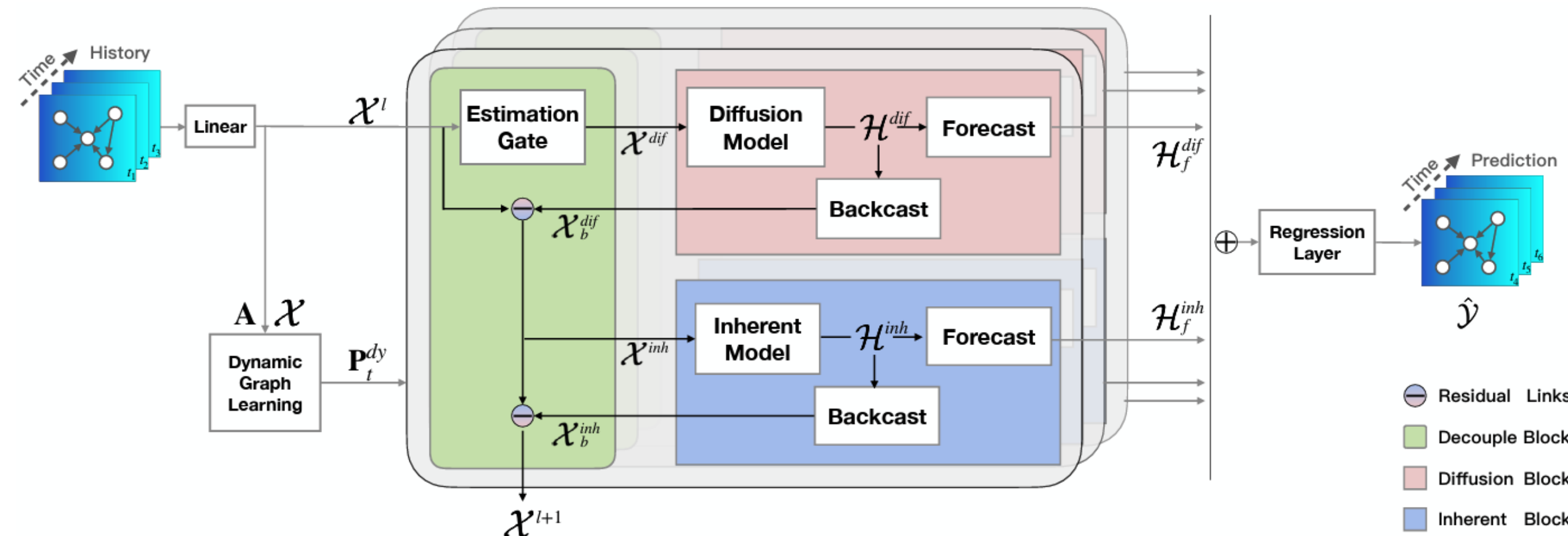
- GraphCNNs for sensor network graphs
- RNN/LSTM for single-road time series

Additionally, transportation data is influenced by external factors (e.g., weather, holidays, events), so models must effectively fuse external features with traffic data for better prediction accuracy.

Application - Transportation

Introduce a novel framework that decouples traffic data into diffusion and inherent components to better model complex spatial-temporal dependencies in traffic forecasting.

- Decoupled Spatial-Temporal Framework (DSTF): Separates traffic data into diffusion signals (capturing spatial dependencies) and inherent signals (capturing temporal patterns).
- Dynamic Graph Learning Module: Learns dynamic characteristics of traffic networks over time.
- Residual Decomposition Mechanism: Enhances the model's ability to capture complex patterns by decomposing residuals.



Application - On-Demand Service

With the rise of on-demand service platforms (e.g., Uber, DiDi, Mobike, GoGoVan), a large volume of spatiotemporal (ST) data is generated, involving customer locations and service times.

To better meet real-time demand and optimize services, accurate demand-supply prediction across locations and times is crucial.

Deep Learning Approaches:

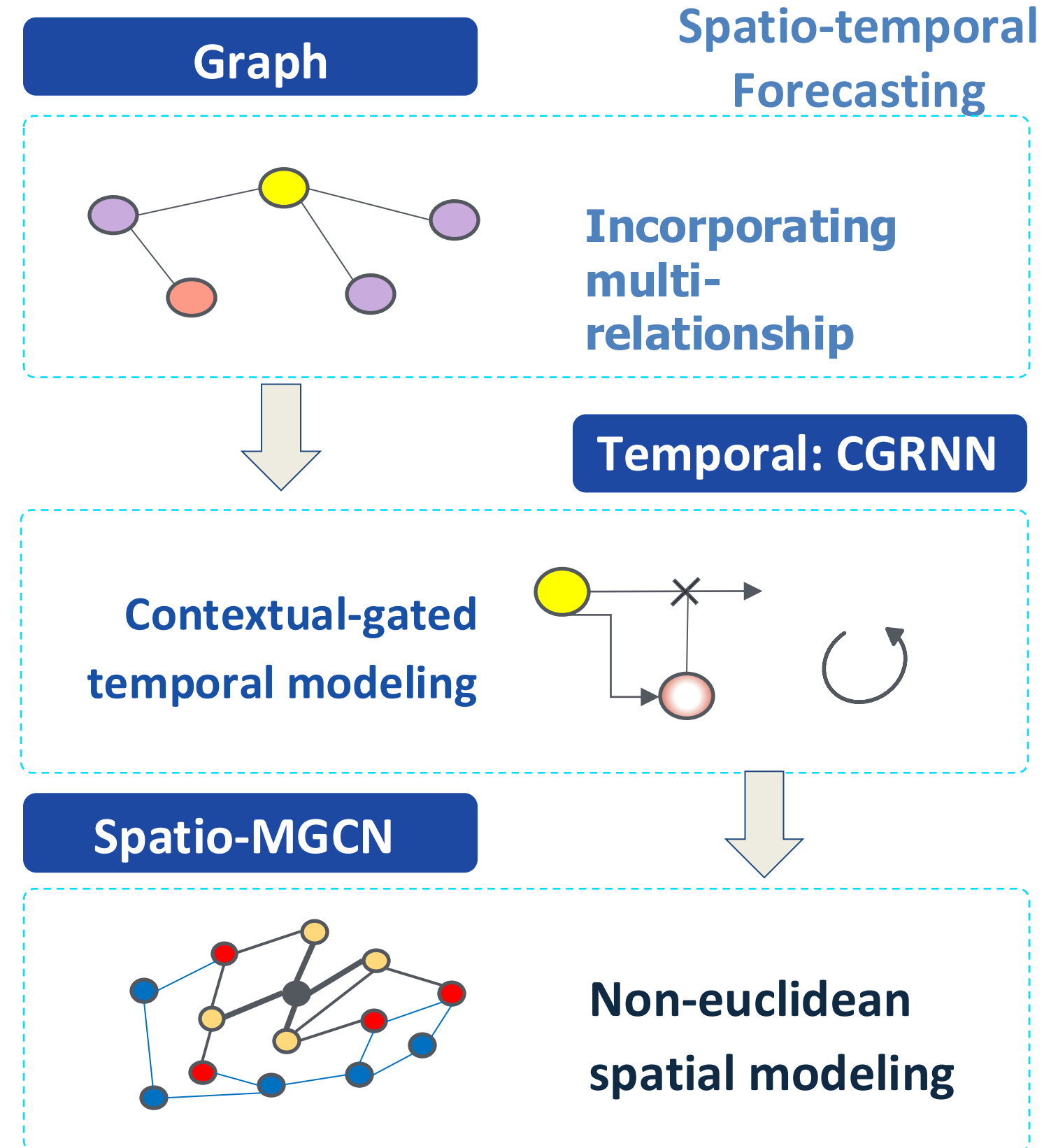
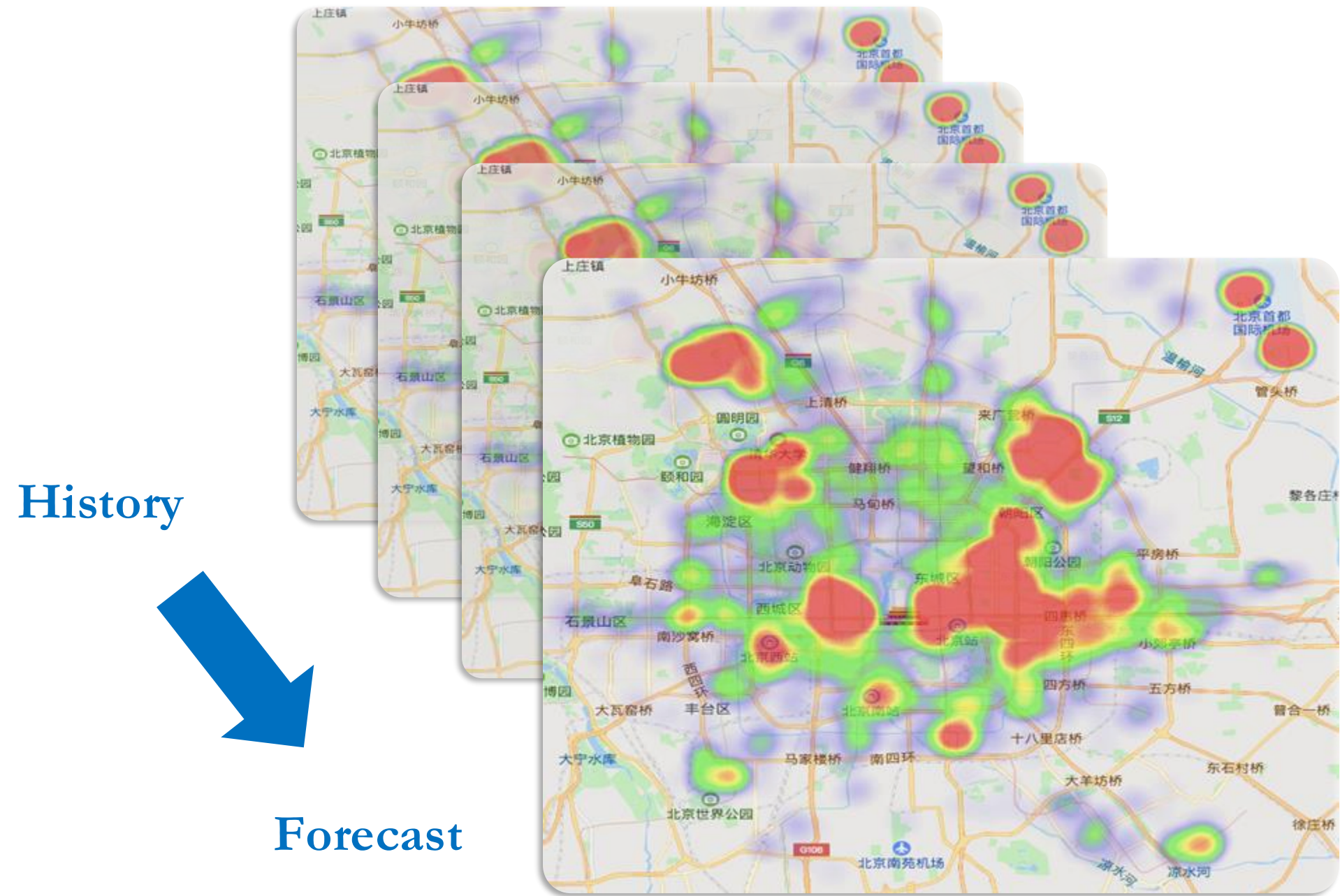
- Dockless bike-sharing: Deep learning methods predict demand-supply distributions.
- Bike-sharing systems: Graph CNN models forecast hourly bike demand at stations by modeling bike flow as a graph.
- Taxi services: LSTM models predict area-specific taxi demand.
- Ride-hailing platforms: ResNet models predict supply-demand patterns.

Modeling Strategy:

- Represent demand-supply across city regions as spatial maps or raster tensors.
- Apply CNNs, RNNs, or hybrid deep models for feature extraction and future prediction.

Application - On-Demand Service

ST-MGCN: Spatiotemporal Multi-Graph Convolution Network:



Application - On-Demand Service

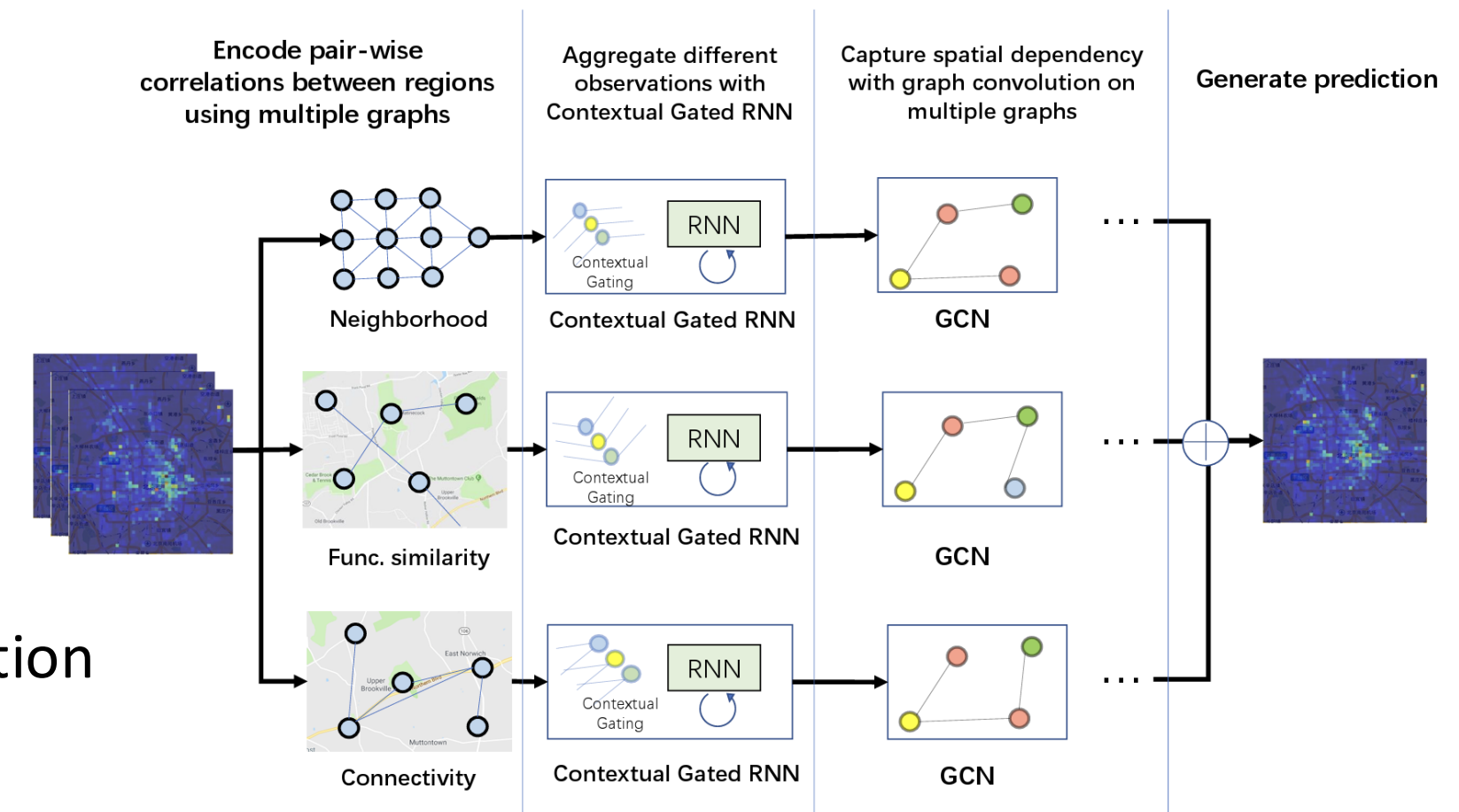
ST-MGCN: Spatiotemporal Multi-Graph Convolution Network:

Key Idea:

Model non-Euclidean spatial dependencies and global temporal dynamics simultaneously for accurate region-level ride-hailing demand forecasting.

Model Architecture:

- Multi-Graph Construction:
Build three graphs to capture different spatial relationships:
Neighborhood Graph (adjacent regions)
Functional Similarity Graph (similar POI surroundings)
Transportation Connectivity Graph (road network links)
- Multi-Graph Convolution:
Perform graph convolutions over multiple graphs to aggregate information from spatially correlated regions (both near and distant).
- Contextual Gated RNN (CGRNN):
Augments RNN with global context-aware gating.
Dynamically reweights different time steps based on global demand patterns.
- Prediction Head:
Outputs future region-level ride-hailing demand after spatial and temporal aggregation.



Application - On-Demand Service

ST-MGCN: Spatiotemporal Multi-Graph Convolution Network:

Graph Generation

Neighborhood

$$A_{N,ij} = \begin{cases} 0, & v_i \text{ and } v_j \text{ are adjacent} \\ 1, & \text{otherwise} \end{cases}$$

POI similarity

$$A_{S,ij} = \text{sim}(P_{v_i}, P_{v_j})$$

Road connectivity

$$A_{C,ij} = \max(0, \text{conn}(v_i, v_j) - A_{N,ij})$$

Temporal modeling

CGRNN

- Use 1-layer GCN to invoke context information
- Use spatial global pooling to get temporal gate
- Apply gate to input signal
- Aggregate gated signal by share-weight RNN

Spatial modeling

MGCN

- Use stacked GCN layer to extract spatial information
- The locality is determined by graph Laplacian and convolution degree
- A proper way to extract spatial information under arbitrary relationship

Application - On-Demand Service

ST-MGCN: Experiments

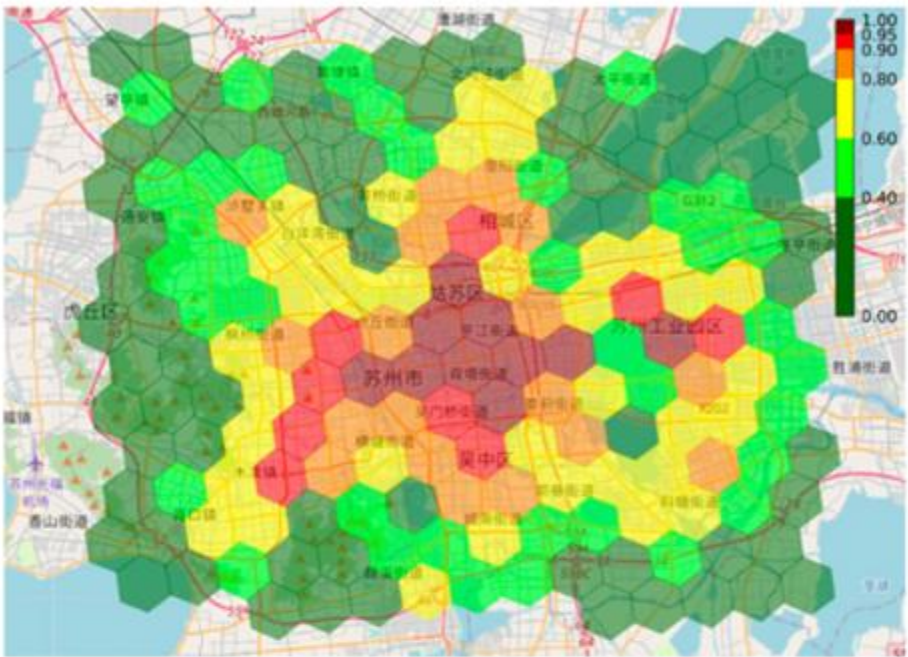
Method	Beijing		Shanghai	
	RMSE	MAPE(%)	RMSE	MAPE(%)
HA	16.14	23.9	17.15	34.8
LASSO	14.24±0.14	23.8±0.8	10.62±0.06	22.9±0.8
Ridge	14.24±0.11	23.8±0.9	10.61±0.04	23.1±0.8
VAR	13.32±0.17	22.4±1.6	10.54±0.18	23.7±1.4
STAR	13.16±0.22	22.2±1.9	10.52±0.21	23.2±1.4
GBM	13.66±0.16	23.1±1.5	10.25±0.11	23.4±1.2
STResNet	11.77±0.95	14.8±6.0	9.87±0.94	14.9±6.0
DMVST-Net	11.62±0.48	12.3±5.5	9.61±0.44	13.8±1.2
ST-GCN	11.62±0.36	10.1±5.1	9.29±0.31	11.2±1.3
<i>ST-MGCN</i>	10.78±0.25	8.8±3.5	8.30±0.16	9.3±0.9

Application - On-Demand Service

OD-CED: Spatio-temporal Prediction of Fine-Grained Origin-Destination Matrices with Applications in Ridesharing

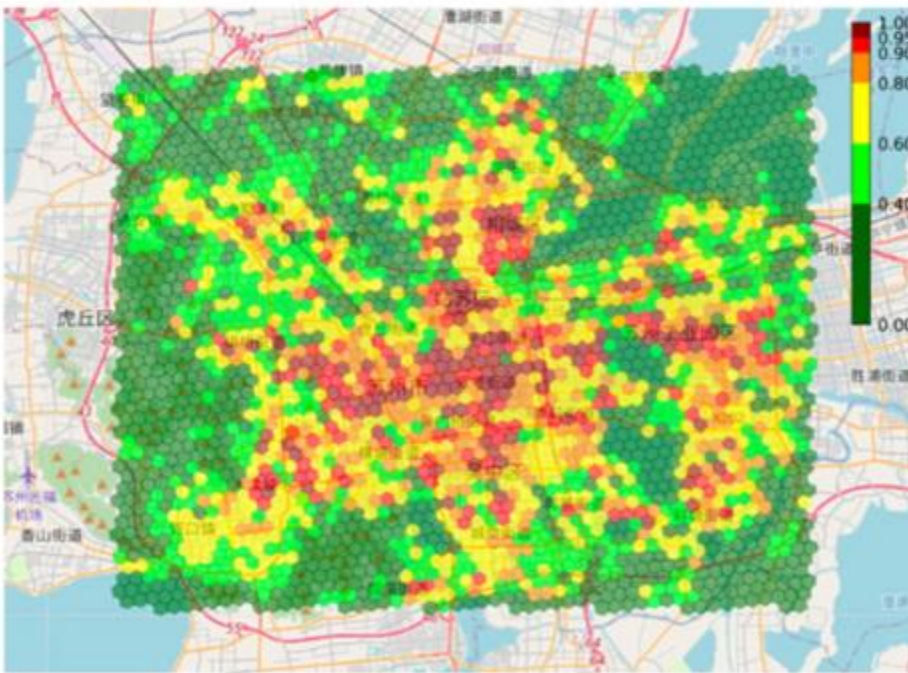
Challenges:

- 1. **Scalability** – OD matrices grow exponentially with more spatial divisions.
- 2. **Data Sparsity** – Over 90% of fine-grained OD flows have zero demand.
- 3. **Semantic & Geographical Dependencies** – Travel demand is influenced by both regional function (e.g., residential vs. commercial) and spatial proximity.



(a)

$$\begin{matrix} & g_1 & g_2 & g_3 & \cdot & g_j & \cdot & g_{183} \\ g_1 & \left[\begin{array}{ccccccc} 9 & 0 & 12 & \cdot & \cdot & \cdot & 0 \end{array} \right. \\ g_2 & \left[\begin{array}{ccccccc} 0 & 23 & 0 & \cdot & \cdot & \cdot & 2 \end{array} \right. \\ g_3 & \left[\begin{array}{ccccccc} 1 & 8 & 9 & \cdot & \cdot & \cdot & 4 \end{array} \right. \\ \cdot & \left[\begin{array}{ccccccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \end{array} \right. \\ g_i & \left[\begin{array}{ccccccc} \cdot & \cdot & \cdot & \cdot & x_{i,j} & \cdot & 6 \end{array} \right. \\ \cdot & \left[\begin{array}{ccccccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \end{array} \right. \\ g_{183} & \left[\begin{array}{ccccccc} 5 & 3 & 7 & 0 & 0 & 2 & 0 \end{array} \right. \end{matrix}$$



(b)

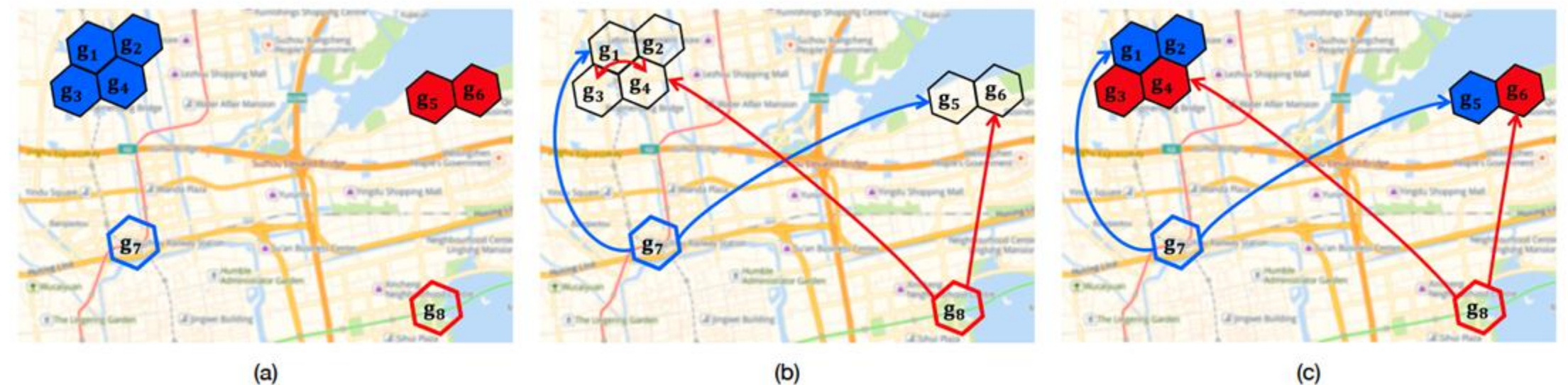
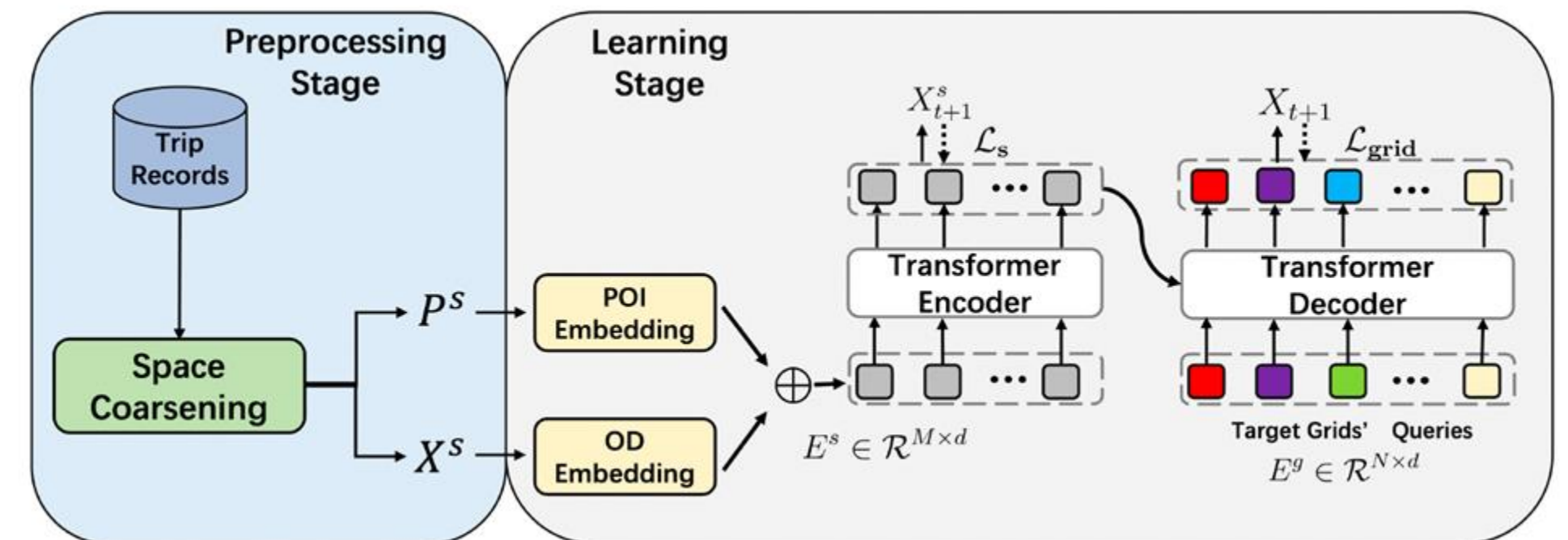
$$\begin{matrix} & g_1 & g_2 & g_3 & \cdot & g_j & \cdot & g_{2531} \\ g_1 & \left[\begin{array}{ccccccc} 1 & 0 & 2 & \cdot & \cdot & \cdot & 0 \end{array} \right. \\ g_2 & \left[\begin{array}{ccccccc} 0 & 0 & 0 & \cdot & \cdot & \cdot & 2 \end{array} \right. \\ g_3 & \left[\begin{array}{ccccccc} 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 \end{array} \right. \\ \cdot & \left[\begin{array}{ccccccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \end{array} \right. \\ g_i & \left[\begin{array}{ccccccc} \cdot & \cdot & \cdot & \cdot & x_{i,j} & \cdot & 0 \end{array} \right. \\ \cdot & \left[\begin{array}{ccccccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \end{array} \right. \\ g_{2531} & \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \right. \end{matrix}$$

Application - On-Demand Service

OD-CED: Spatio-temporal Prediction of Fine-Grained Origin-Destination Matrices with Applications in Ridesharing

OD-CED Model: A Novel OD Prediction Framework

- **Space Coarsening Module:** Merges fine-grained cells into super-cells to mitigate sparsity.
- **Encoder-Decoder Architecture:** Captures semantic and geographical dependencies effectively.
- **Permutation-Invariant OD Embedding:** Learns robust representations of OD flows.



Application - On-Demand Service

OD-CED: Experiments

Dataset Performance Comparison (City-C & City-S):

- **City-C:**
 - RMSE reduced from **1.255 (GEML)** → **0.905 (OD-CED)** (~28% improvement).
 - wMAPE reduced from **0.667 (GEML)** → **0.411 (OD-CED)** (~39% improvement).
- **City-S:**
 - RMSE reduced from **1.146 (GEML)** → **0.740 (OD-CED)** (~35% improvement).
 - wMAPE reduced from **0.605 (GEML)** → **0.323 (OD-CED)** (~47% improvement).

Training Time Comparison (per epoch on V100 GPU):

- OD-CED: **22.12s**
- STGCN: **28.81s**
- GEML (state-of-the-art): **39.63s**
- CSTN & MRSTN: **1200+ seconds**
- OD-CED is **2x faster than GEML** and **over 50x faster** than CNN-based methods.

Method	City-C			City-S		
	wMAPE	RMSE	CPC	wMAPE	RMSE	CPC
HA	0.813	1.442	0.348	0.821	1.435	0.355
OLSR	0.822	1.419	0.324	0.816	1.351	0.333
LASSO	0.807	1.424	0.359	0.813	1.349	0.337
CSTN	0.782	1.370	0.354	0.721	1.217	0.451
MRSTN	0.788	1.380	0.351	0.766	1.253	0.464
GEML	0.667	1.255	0.540	0.605	1.146	0.597
STGCN	0.681	1.337	0.488	0.596	1.210	0.674
OD-CED	0.411	0.905	0.776	0.323	0.740	0.889

	CSTN	MRSTN	GEML	STGCN	OD-CED
# of Params (in millions)	0.54M	0.67M	2.9M	1.6M	0.1M
Training Time (in seconds)	1222.13s	1602.14s	39.63s	28.81s	22.12s

Application - Meso Level Supply-Demand Forecasting

Causal Probabilistic Spatio-Temporal Fusion Transformers in Two-Sided Ride-Hailing Markets

Goal: Predicting supply and demand in ride-hailing platforms using a **causal, interpretable**, and **scalable** forecasting framework

Algorithm 3: A collaborative causal spatio-temporal fusion transformer (**CausalTrans**).

Authors: Wang et al.,

Journal: *ACM Transactions on Spatial Algorithms and Systems*, 2024

Application - Meso Level Supply-Demand Forecasting

Collaborative Problem

$$P(x_v(t+1:t+\tau_{\max})|x_v(:t), z_v(:t+\tau_{\max}))$$
$$P(y_v(t+1:t+\tau_{\max})|y_v(:t), x_v(:t+\tau_{\max}), z_v(:t+\tau_{\max}))$$

Where:

$x_v(t)$: demand at time t in grid v ;

$y_v(t)$: supply at time t in grid v ;

$z_v(t)$: external covariates (e.g., weather, holiday) at time t in grid v ;

τ_{\max} : pre-specified time length

$v \in V$.

Probabilistic Forecasting

- Given $q \in Q = \{10\%, 50\%, 90\%\}$, then quantile loss QL_q at each point q is:

$$QL_q(x_t, \hat{x}_{t-\tau}^q) = \{q - I(x_t \leq \hat{x}_{t-\tau}^q)\} * (x_t - \hat{x}_{t-\tau}^q)$$

- Then final quantile loss is:

$$Loss_Q = \sum_{x_t \in \Omega} \sum_{q \in Q} \sum_{\tau=1}^{\tau_{\max}} \frac{QL_q(x_t, \hat{x}_{t-\tau}^q)}{M * \tau_{\max}}$$

- We introduce quantile risk as a key metric:

$$Risk_q = \frac{2 \sum_{x_t \in \tilde{\Omega}} \sum_{\tau=1}^{\tau_{\max}} QL_q(x_t, \hat{x}_{t-\tau}^q)}{\sum_{x_t \in \tilde{\Omega}} \sum_{\tau=1}^{\tau_{\max}} |x_t|}$$

where $\tilde{\Omega}$ is the best dataset.

Application - Meso Level Supply-Demand Forecasting

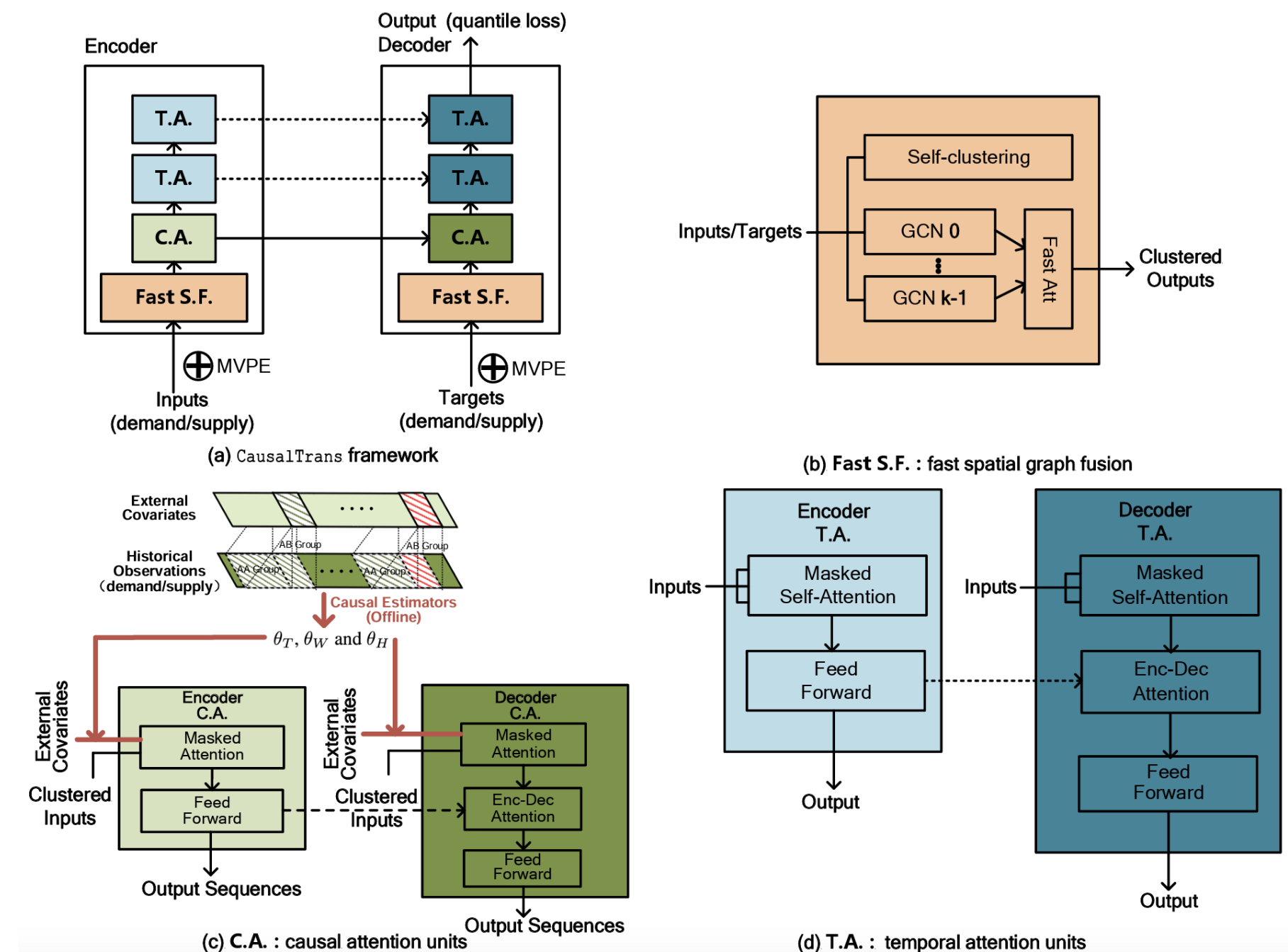
The overview of **CausalTrans** framework:

(a). The framework consists of three essential components: **Fast S.F.** (*fast graph spatial fusion*), **C.A.** (*causal attention*), and **T.A.** (*temporal attention*). Demand and supply are trained separately in sequence.

(b). The **Fast S.F.** consists of self-clustering with GAT and fast attention.

(c). The **C.A.** applies offline trained causal weights θ to online treatments evaluations.

(d). The **T.A.** aims to keep ordering self-attentions.



Application - Meso Level Supply-Demand Forecasting

CausalTrans - Causal Attention Mechanism

We transfer the weights of external covariates to causal weights by **HTE methods** (e.g. double machine learning).

Algorithm 1 Causal Attention Algorithm with DML

Input: Given demand matrix $x(:t)$ at a grid v before time t , three kinds of treatments includes weekday and hour slots $T(:t) = \{W(:t), H(:t)\}$, weather vectors $W(:t)$, and holidays one-hot vectors $H(:t)$

Output: causal effect coefficients θ_T for $T(:t)$, θ_W for $W(:t)$, and θ_H for $H(:t)$

```

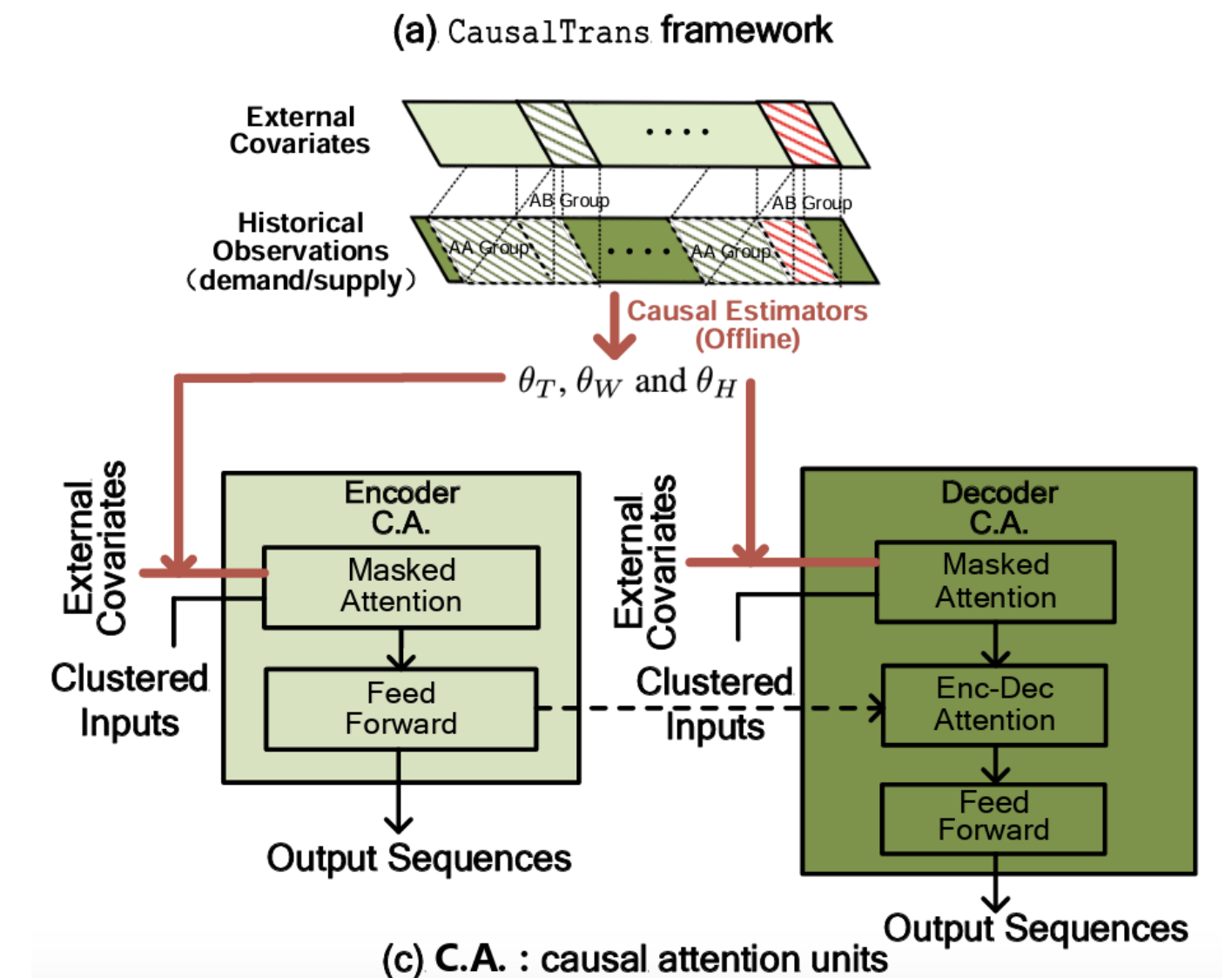
1: Take  $\theta_T$  as an example, and suppose that a AA group and AB group on  $T(:t)$  is  $T_{AA} = T_{AB} = \{$ 
2: for all  $\{T_w(t_0), T_w(t_1)\} \in \{Mon, Tue, \dots, Sun\}, \{T_h(t_0), T_h(t_1)\} \in \{1, \dots, 24\}$  do
3:   if  $T_w(t_0) = T_w(t_1), T_h(t_0) = T_h(t_1), \mathcal{P}_{T-Test}(x(t_0), x(t_1)) < 0.05$  then
4:     for all  $t'_0 \in \{ : t_0 \}$  and  $t'_1 \in \{ : t_1 \}$  do
5:       Calculate 1st-order differences  $\tilde{x}(t'_0 : t_0)$  and  $\tilde{x}(t'_1 : t_1)$ 
6:       if  $\mathcal{P}_{KPSS}(\tilde{x}(t'_0 : t_0)), \mathcal{P}_{KPSS}(\tilde{x}(t'_1 : t_1))$  and  $\mathcal{P}_{T-Test}(\tilde{x}(t'_0 : t_0), \tilde{x}(t'_1 : t_1)) > 0.05$  then
7:          $T_{AA}.append([(x(t'_0 : t_0), x(t'_1 : t_1))])$ 
8:          $T_{AB}.append([(x(t_0), x(t_1))])$ 
9:       end if
10:    end for
11:  end if
12: end for
13: Do DML on  $T_{AA}$  and  $T_{AB}$  datasets and estimate treatment coefficients  $\theta_T$ 
14: Repeat from Step 2 and estimate  $\theta_W$  and  $\theta_H$  by different DML.
15: return  $\theta_T, \theta_W$ , and  $\theta_H$ 
  
```

(a) causal attention algorithm

step 1: external covariates: weather, holidays and subsidy;

step 2: build various of control groups and treat groups;

step 3: do **DML** and get causal attention or weights.



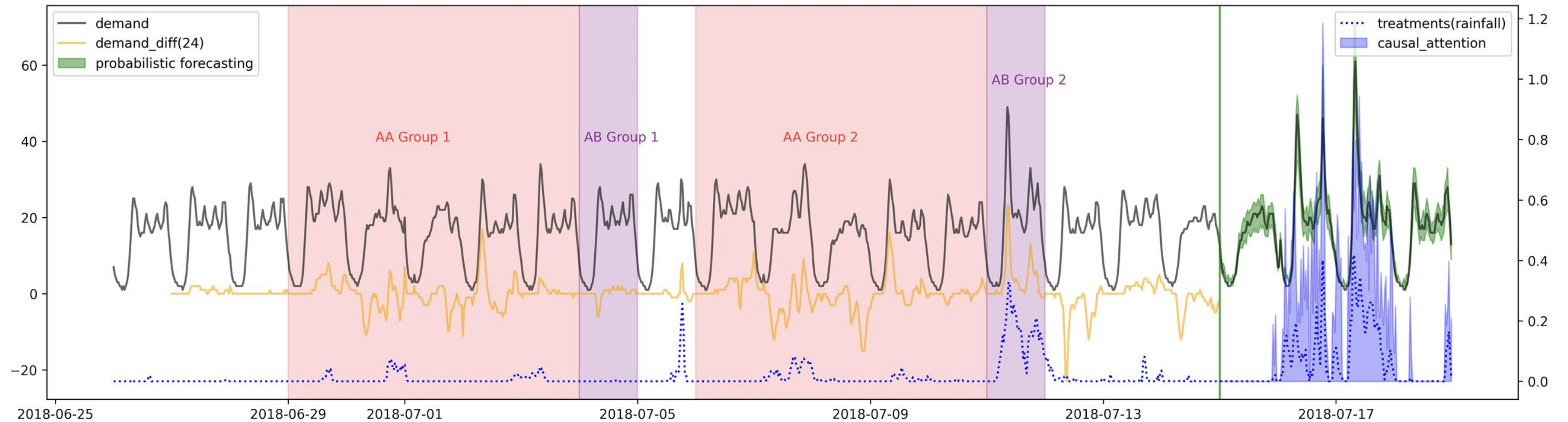
(b) how to work in ConvTrans

step 1: offline training causal attention;

step 2: add above weights in multi-head attention

Application - Meso Level Supply-Demand Forecasting

CausalTrans - Causal Attention Visualization



- “AA group 1” and “AA group 2” are regarded as comparable contexts;
- “AB group 1” and “AB group 2” is control group and treatment group;
- Do **DML** and get causal attention weights.

Application - Meso Level Supply-Demand Forecasting

CausalTrans - Experiment

(a) Risk_(50%) losses on the retail and ride-hailing datasets.

	ConvTrans	Seq2Seq	MQRNN	DeepAR	DMVST	ST-MGCN	TFT	CausalTrans
Retail	0.429 [◇]	0.411 [◇]	0.379 [◇]	0.386	0.403	0.395	0.354 [◇]	0.352(-0.6%)
Ride-hailing (1d, city A, Demand)	0.573	0.550	0.495	0.499	0.524	0.482	0.450	0.434(-3.7%)
Ride-hailing (1d, city A, Supply)	0.482	0.453	0.428	0.422	0.443	0.421	0.415	0.393(-5.3%)
Ride-hailing (1d, city B, Demand)	0.470	0.455	0.405	0.400	0.422	0.404	0.370	0.361(-2.5%)
Ride-hailing (1d, city B, Supply)	0.426	0.404	0.388	0.384	0.388	0.378	0.357	0.341(-4.5%)
Ride-hailing (7d, city A, Demand)	0.756	0.717	0.653	0.663	0.664	0.677	0.689	0.613(-6.2%)
Ride-hailing (7d, city A, Supply)	0.612	0.569	0.516	0.519	0.536	0.575	0.583	0.468(-9.3%)
Ride-hailing (7d, city B, Demand)	0.693	0.627	0.574	0.571	0.590	0.588	0.576	0.539(-5.6%)
Ride-hailing (7d, city B, Supply)	0.568	0.519	0.499	0.501	0.503	0.525	0.528	0.454(-9.0%)

(b) Risk_(90%) losses on the retail and ride-hailing datasets.

	ConvTrans	Seq2Seq	MQRNN	DeepAR	DMVST	ST-MGCN	TFT	CausalTrans
Retail	0.192 [◇]	0.157 [◇]	0.152 [◇]	0.156	0.156	0.155	0.147 [◇]	0.143(-2.8%)
Ride-hailing (1d, city A, Demand)	0.238	0.208	0.205	0.205	0.208	0.195	0.192	0.164(-14.6%)
Ride-hailing (1d, city A, Supply)	0.212	0.177	0.164	0.162	0.173	0.165	0.160	0.142(-11.3%)
Ride-hailing (1d, city B, Demand)	0.208	0.176	0.159	0.158	0.170	0.157	0.155	0.145(-6.5%)
Ride-hailing (1d, city B, Supply)	0.205	0.197	0.157	0.188	0.169	0.151	0.149	0.139(-6.7%)
Ride-hailing (7d, city A, Demand)	0.324	0.306	0.276	0.289	0.286	0.280	0.297	0.244(-11.6%)
Ride-hailing (7d, city A, Supply)	0.259	0.233	0.207	0.204	0.237	0.248	0.237	0.173(-15.2%)
Ride-hailing (7d, city B, Demand)	0.288	0.269	0.241	0.240	0.252	0.255	0.238	0.216(-9.3%)
Ride-hailing (7d, city B, Supply)	0.214	0.184	0.177	0.179	0.168	0.197	0.204	0.153(-8.9%)

- Use grid search to optimize hyperparameters;
- **DeepAR** outperforms **Seq2Seq** and **MQRNN** because of Poisson and weather covariates;
- **CausalTrans** outperforms other methods primarily due to causal estimator **DML**;
- **CausalTrans** achieves lower losses on supply than demand based on both causal relationship;
- Long-term prediction focuses on unbiased distribution estimation.

Application - Climate & Weather

Weather and climate data capture atmospheric and oceanic conditions (e.g., temperature, wind, pressure, precipitation, air quality) via sensors at fixed or mobile locations.

Due to strong **spatiotemporal correlations** in climate data, **spatiotemporal deep modeling (STDM)** techniques are widely used for **short-term and long-term forecasting**.

Deep Learning Approaches:

- **Air quality inference:** Predict urban air pollution.
- **Precipitation prediction:** Forecast rain using remote sensing.
- **Wind speed prediction:** Model anemometer readings.
- **Extreme weather detection:** Identify severe weather events.

Data Types:

- **Spatial maps:** e.g., radar reflectivity images
- **Time series:** e.g., wind speed readings
- **Event data:** e.g., extreme weather occurrences

Example Models:

- **Attention models** for air quality prediction
- **CNNs** for detecting extreme weather and precipitation forecasting

Application - Neuroscience

Various brain imaging technologies — such as **fMRI**, **EEG**, **MEG**, and **fNIRS** — are widely used in neuroscience research. These technologies differ significantly in spatial and temporal resolution:

- **fMRI**: Millions of spatial locations, lower temporal resolution (~2 seconds per measurement)
- **EEG**: Tens of locations, very high temporal resolution (~1 millisecond)

Data Representation:

- Brain imaging data (fMRI, EEG) are naturally represented as **spatial maps** or **rasters**, making them suitable for **DL** analysis.

Deep Learning Applications:

- **Disease classification and diagnosis**: e.g., Autism Spectrum Disorder, amnestic Mild Cognitive Impairment, Schizophrenia
- **Brain function network classification**
- **Brain activation pattern classification**

Example Models:

- **LSTM** for Autism Spectrum Disorder detection
- **CNN** for diagnosing amnestic Mild Cognitive Impairment
- **FNN** for classifying Schizophrenia

Application 1: BrainGNN

BrainGNN is a **graph neural network (GNN)** specifically designed for analyzing **functional MRI (fMRI)** brain data. It **predicts cognitive states or disease status** while providing **interpretability** by identifying important brain regions and connections.

Model Highlights:

Node-Level Pooling:

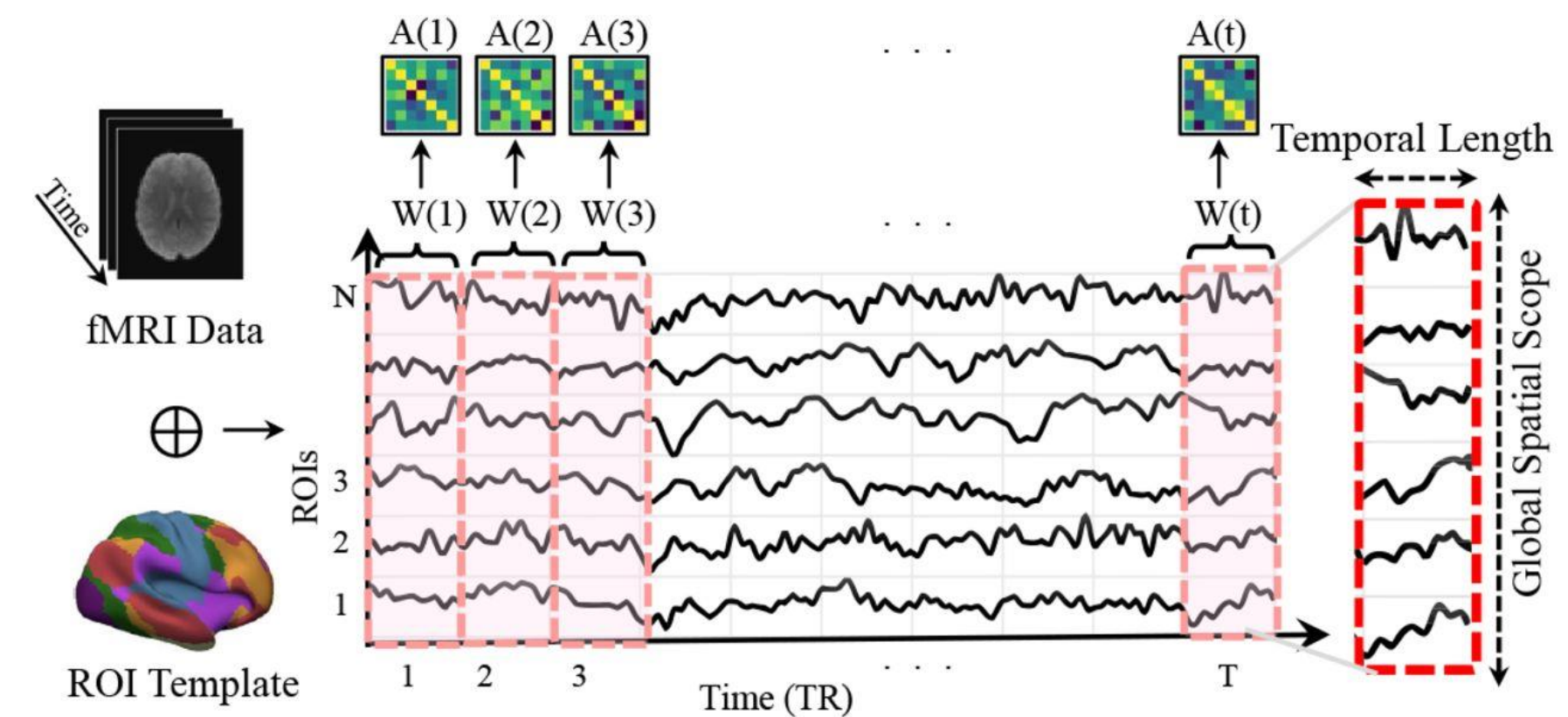
- Groups similar brain regions (ROIs) into clusters based on learned features.

ROI Selection Layer:

- Automatically selects important brain regions contributing to the prediction.

Attention Mechanism:

- Highlights key functional connections between selected ROIs.



Dynamic Graph Transformer for Brain Disorder Diagnosis

Application

Introduce a model that **jointly learns spatial and temporal features** from resting-state fMRI (rs-fMRI) using a combination of **graph convolution** and **recurrent neural networks (RNNs)**.

Model Highlights:

Spatial Graph Convolution:

- Models **functional connectivity** between brain regions at each time step, treating brain ROIs as graph nodes.

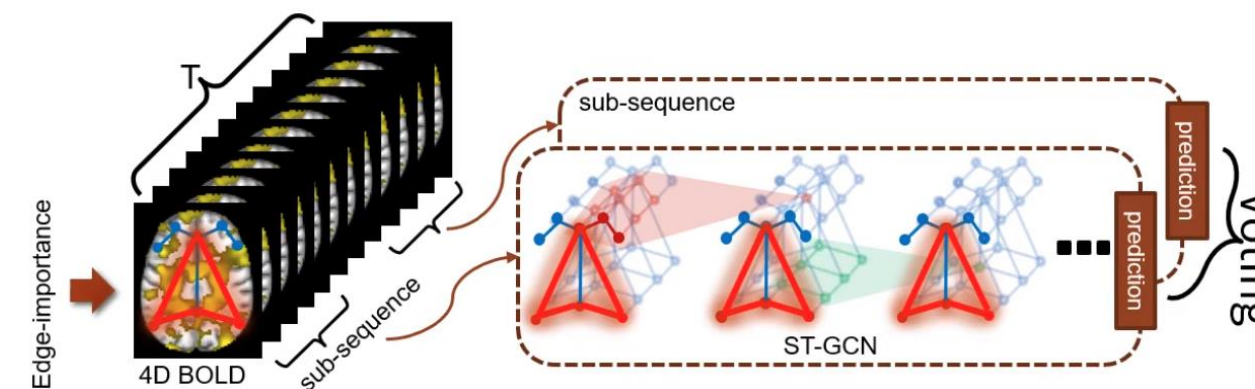
Temporal RNN:

- Captures the **evolution of brain connectivity over time** by applying an RNN (such as GRU) on node embeddings.

End-to-End Training:

- Learns both spatial (graph structure) and temporal (dynamic activity) representations directly from raw fMRI sequences.

- Our proposal: ST-GCN
 - › Consider both temporal dependency and functional connectivity
 - › Train on short sub-sequences
 - › Learns the importance of graph edges to the prediction



Application

Total Activation (TA) is a method that **deconvolves fMRI signals** to recover the underlying **neural activity-inducing signals** by applying **spatio-temporal regularization**.

Model Highlights:

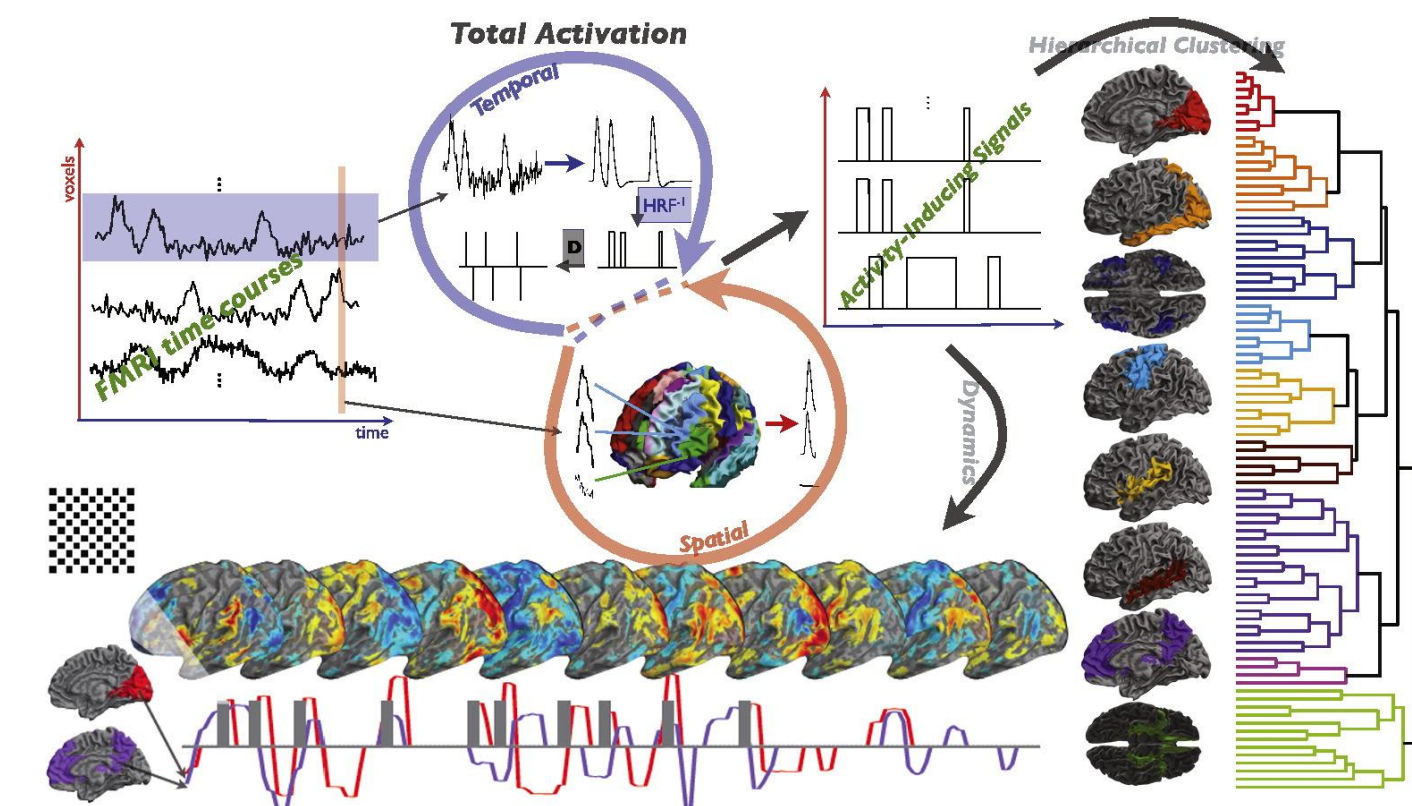
•Temporal Regularization:

- Promotes **piecewise constant activation patterns** over time (temporal sparsity).

•Spatial Regularization:

- Enforces **spatial smoothness** across neighboring voxels (nearby brain regions).

•Solves an optimization problem balancing **data fidelity** with **spatio-temporal priors**.



Total activation: fMRI deconvolution through spatio-temporal regularization

Content

1 Spatio-temporal Data and Tasks

2 Problem Definitions and Evaluation Metrics

3 Spatio-temporal Modeling

4 Applications

5 Challenges and future works

Challenges

- **Latent graph learning**

What to do when the underlying graph is not known?

- **Learning in non-stationary environments**

What to do when the environment changes?

- **Scalability**

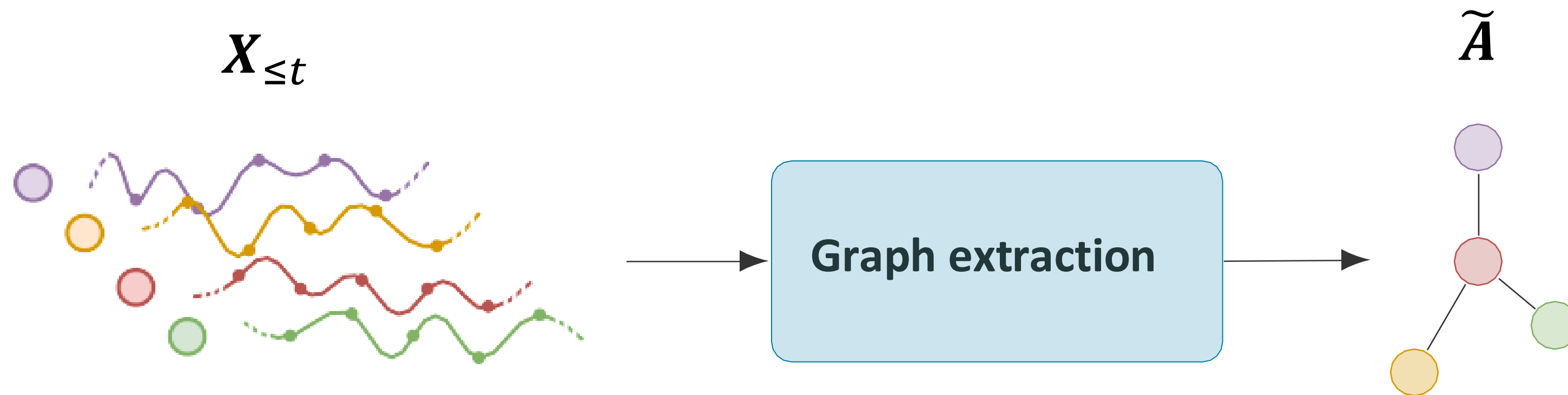
How to deal with large collections of time series?

- **Dealing with missing data**

How to deal with missing observations within the time series?

Learning and adjacency matrix

- ✗ Relational information is **not** always **available**
- ✗ or might be **ineffective**<sup>13
4</sup> in capturing spatial dynamics.
- ✓ **Relational** architectural **biases** can nonetheless be exploited
→ **extract a graph** from the time series or node attributes

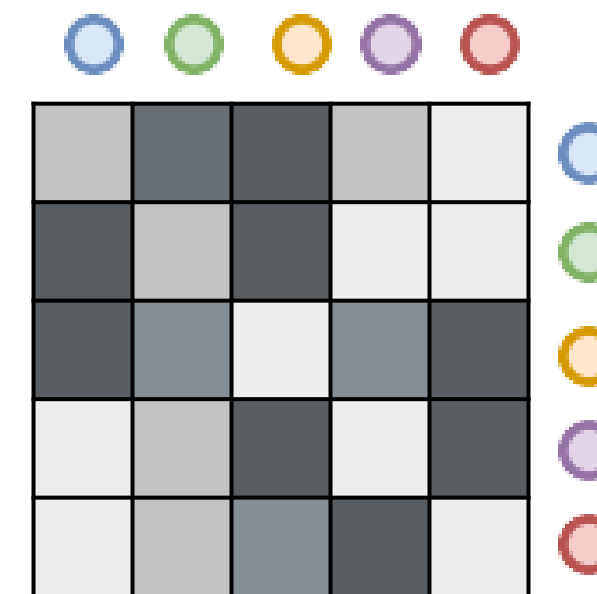


- It can be interpreted as **regularizing** a **spatial attention** operator.

Time-series similarities

Probably, the simplest approach to extract a graph from the time series is by computing **time series similarity scores**.

- Pearson correlation
- Correntropy
- Granger causality
- Kernels for time series
- ...



→ Thresholding might be necessary to obtain binary and sparse graphs.

Latent graph learning

An integrated approach: learn the **relations** end-to-end with the downstream task

- as a function of the input data,
- as trainable parameters of the model,
- or both.

This problem is known as latent graph learning (or latent graph inference).

Direct approach

A direct approach consists in learning \tilde{A} as function $\xi(\cdot)$ of edge scores $\Phi \in R^{N \times N}$ as

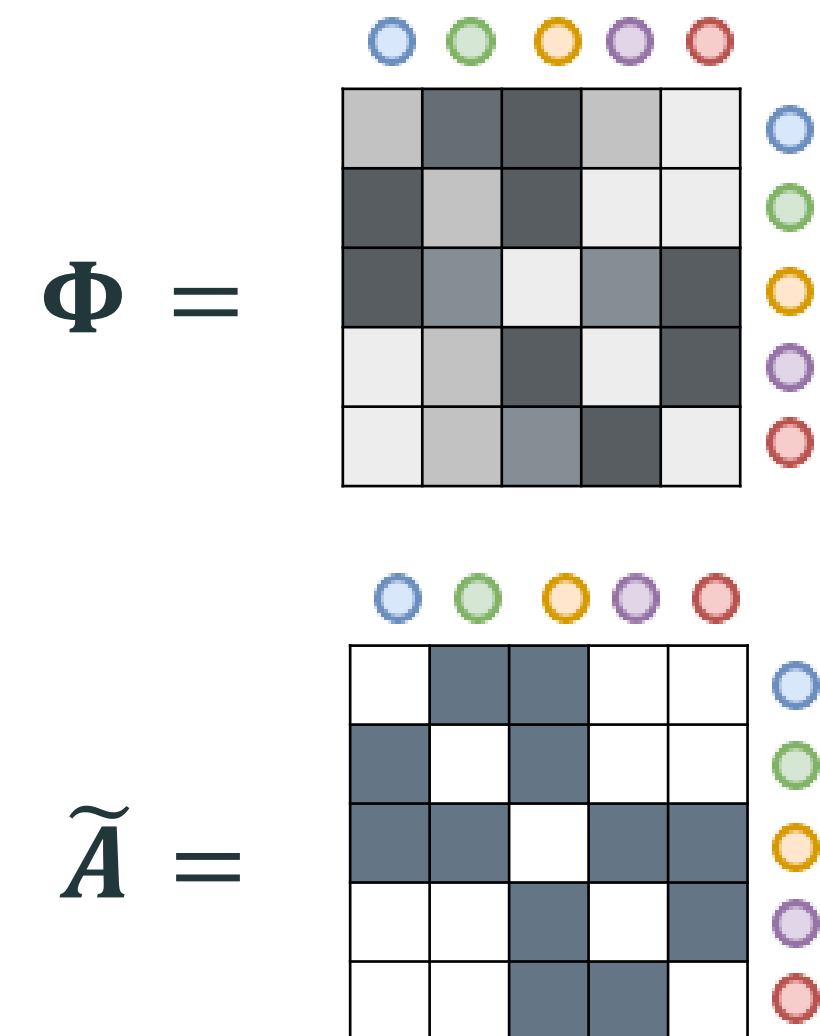
$$\tilde{A} = \xi(\Phi)$$

Edge scores Φ

- can be a table of **learnable** model **parameters**,
- obtained as a **function** of the **inputs** and/or other parameters.

Function $\xi(\cdot)$ is a nonlinear activation

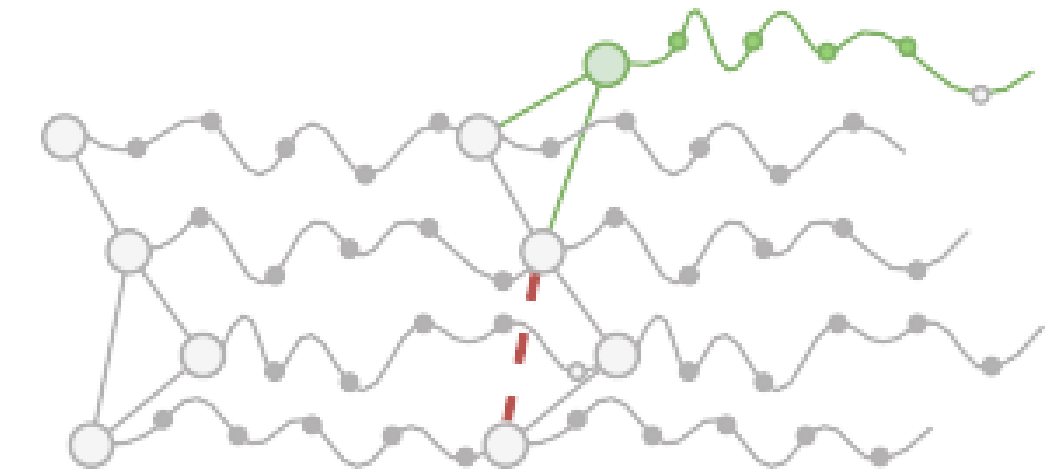
- it can be exploited to **make \tilde{A} sparse**.



Inductive learning

In real-world applications, one often needs to

- operate under **changes** in the network **connectivity**
- make predictions for **newly added nodes**
- **transfer** the model to **different** sensor **networks** (collections of time series)



Useful in **several tasks**, like, forecasting, missing data imputation, and virtual sensing.

Attention: Performance can easily **degrade** if the **data distribution** of target nodes

- **deviates** from that at **training nodes**
- **changes over time**.

Transferability of STGNNs

Global STGNNs are **inductive** and can directly be used in the above settings, provided that the training and target data are similar enough.

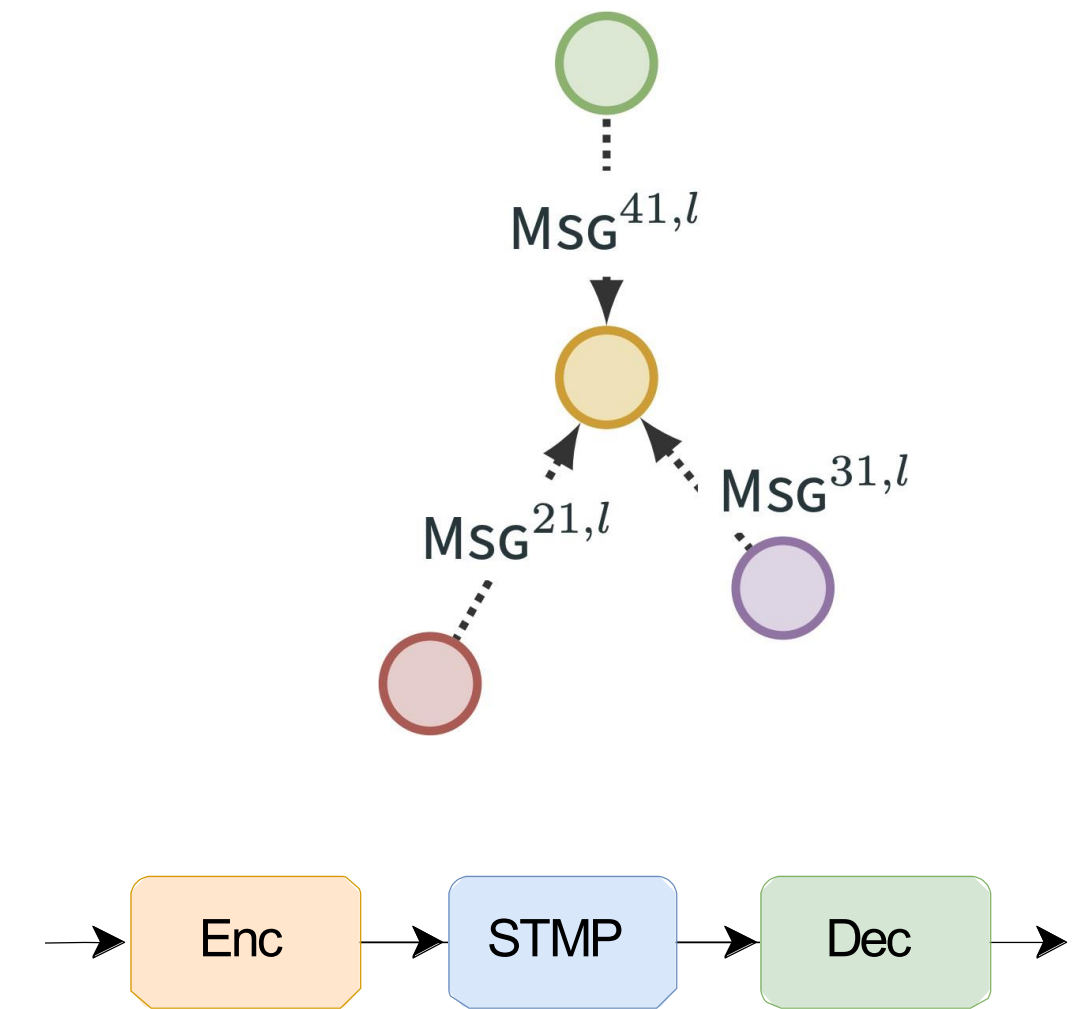
- MP operates on **generic neighborhoods**
- MP **parameters** are **shared** across nodes

Otherwise, STGNNs **need** to be **adjusted**

- **fine-tuning** (a subset of) the weights of the model on the new data
- exploiting **transfer learning** strategies

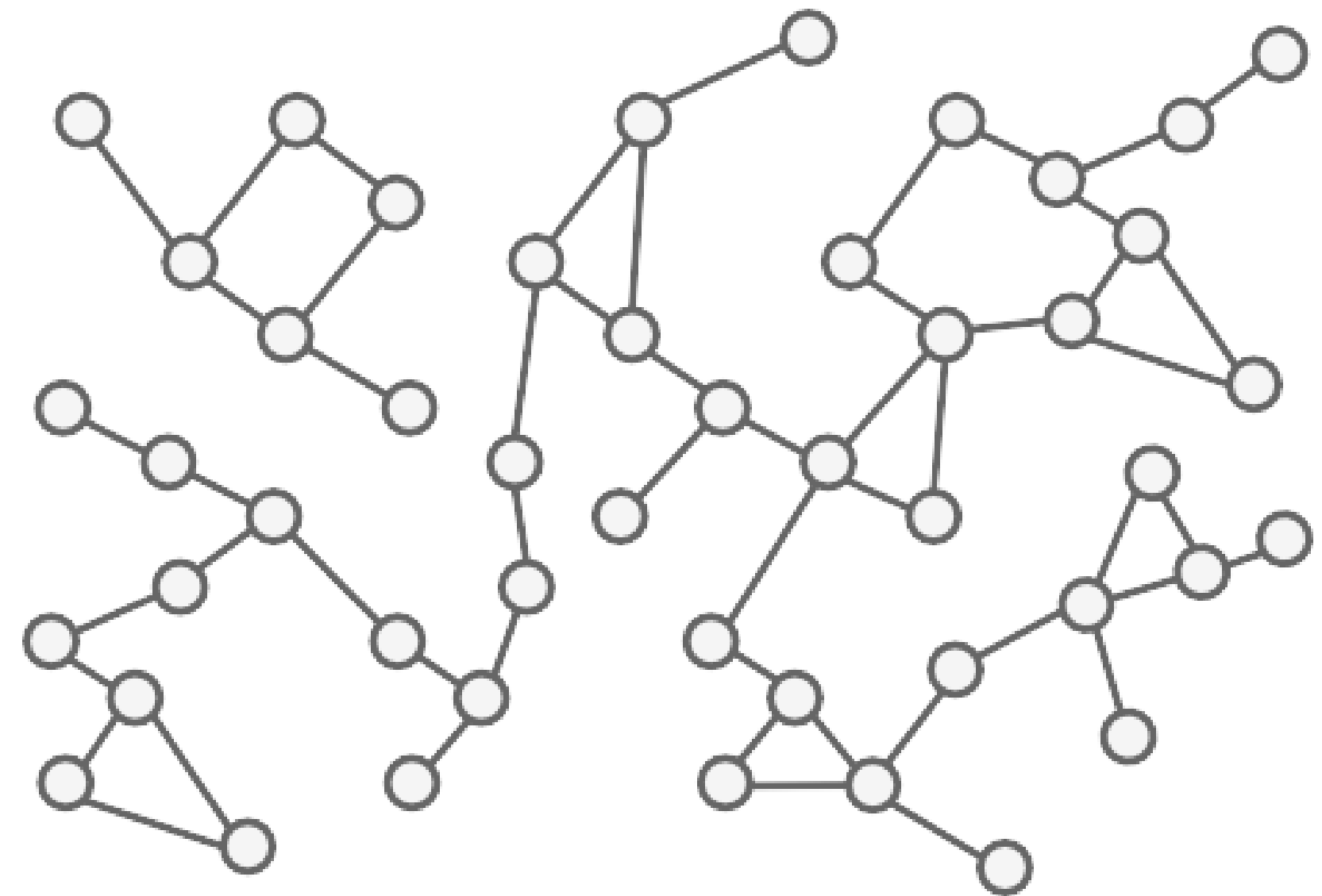
Attention: Global-local STGNNs reduce the cost of **transfer learning**

- **sharing** most of the **parameters** and finetuning node-specific parameters only
- node **embeddings** can be **regularized** to facilitate the learning further.



The scalability feature

- **Graph-based processing** allows us to learn a single model...
- ...able to deal with a **large collection** of time series...
- ...while accounting for the most relevant
 - **relational information.**



The scalability issue

Spatiotemporal data span – as the name suggests – **two dimensions**:

- the **spatial** dimension, corresponding to the number of time series (sensors).
- the **time** dimension, corresponding to the number of time steps (number of observations acquired per sensor).

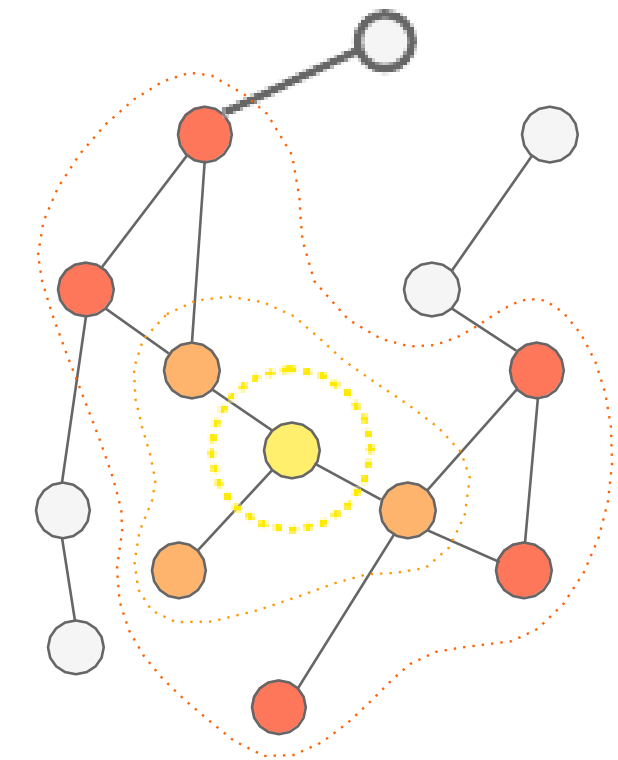
In the real world, dealing with **thousands of sensors** acquiring data at **high sampling rates** is quite common (e.g., smart cities).

- A large amount of data needs to be **processed at once**.
- In particular, to account for **long-range** spatiotemporal dependencies.

Graph subsampling

Computations can be reduced by training on **subgraphs** of the full network, e.g., by

- sampling the **K -th order neighborhood** of a subset of nodes;
- **rewiring** the graph to reduce the total number of edges.

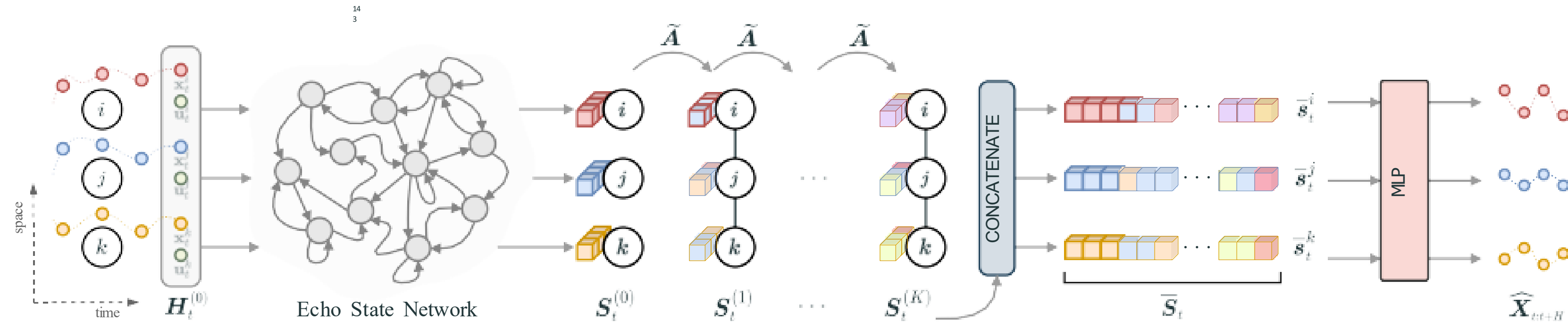


Mostly adapted from methods developed in **static graph processing** (e.g., 2 references below).

- Subsampling might break long-range spatiotemporal dependencies.
- The learning signal may be noisy.

SGP: Scalable Graph Predictor

Extracted representations can be sampled uniformly across time and space during training.



- ✓ The cost of a training step is independent of W, N and $|E_t|$.
- ✓ Performance matches state of the art.
- ✗ More storage space is required, as the number of extracted features is much higher than d_x .
- ✗ More reliant on hyperparameter selection than end-to-end approaches.

The problem of missing data

So far, we assumed to deal with **complete sequences**, i.e., to have valid observations associated with each node (sensor) and time step.

However, time series collected by real-world sensor networks often have **missing data**, due to:

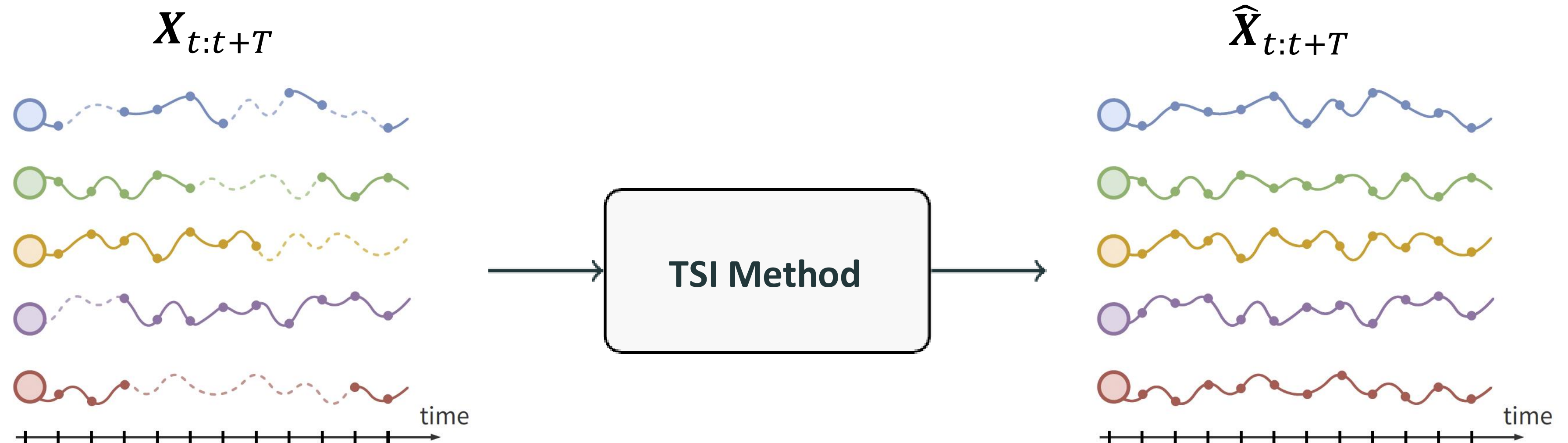
- faults, of either transient or permanent nature;
- asynchronicity among the time series;
- communication errors...

Most forecasting methods operate on complete sequences.

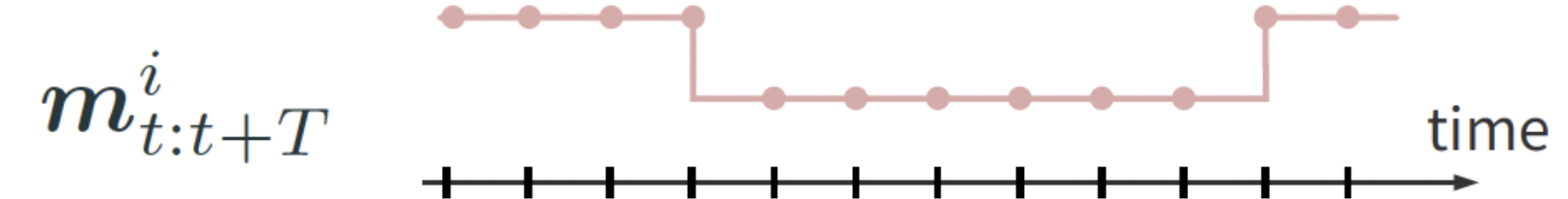
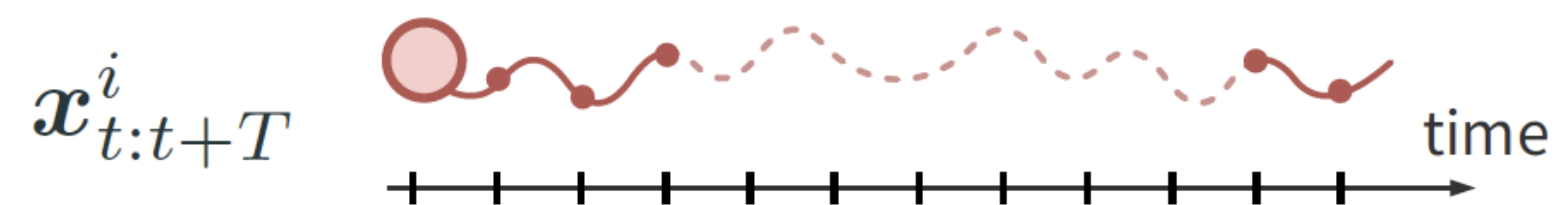
→ We need a way to **impute**, i.e., *reconstruct*, missing data.

Time series imputation

The problem of reconstructing missing values in a sequence of data is often referred to as [time series imputation](#) (TSI).



Time series imputation



Time series imputation

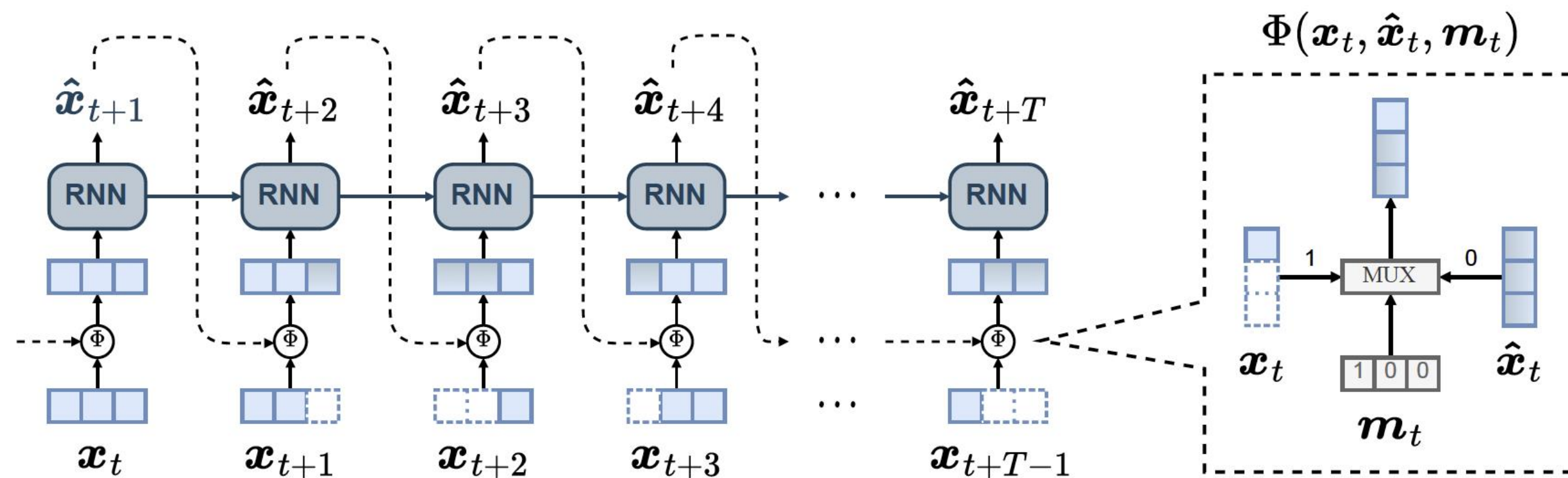
Given a window of $T \geq 1$ observations $\mathbf{X}_{<T}$ with missing values, the **time series imputation** problem consists in estimating the missing observations in the sequence

$$\mathbf{x}_t^i \sim p(\mathbf{x}_t^i | \mathcal{X}_{<T}) \quad \forall i, t \text{ such that } m_t^i = 0$$

with $\mathcal{X}_{<T} = \{\mathbf{x}_t^i | \mathbf{x}_t^i \in \mathbf{X}_{<T} \text{ and } m_t^i = 1\}$ being the **observed set**.

Deep learning for TSI

Besides standard statistical methods, deep learning approaches have become a popular alternative, in particular, **autoregressive models** (e.g., RNNs).



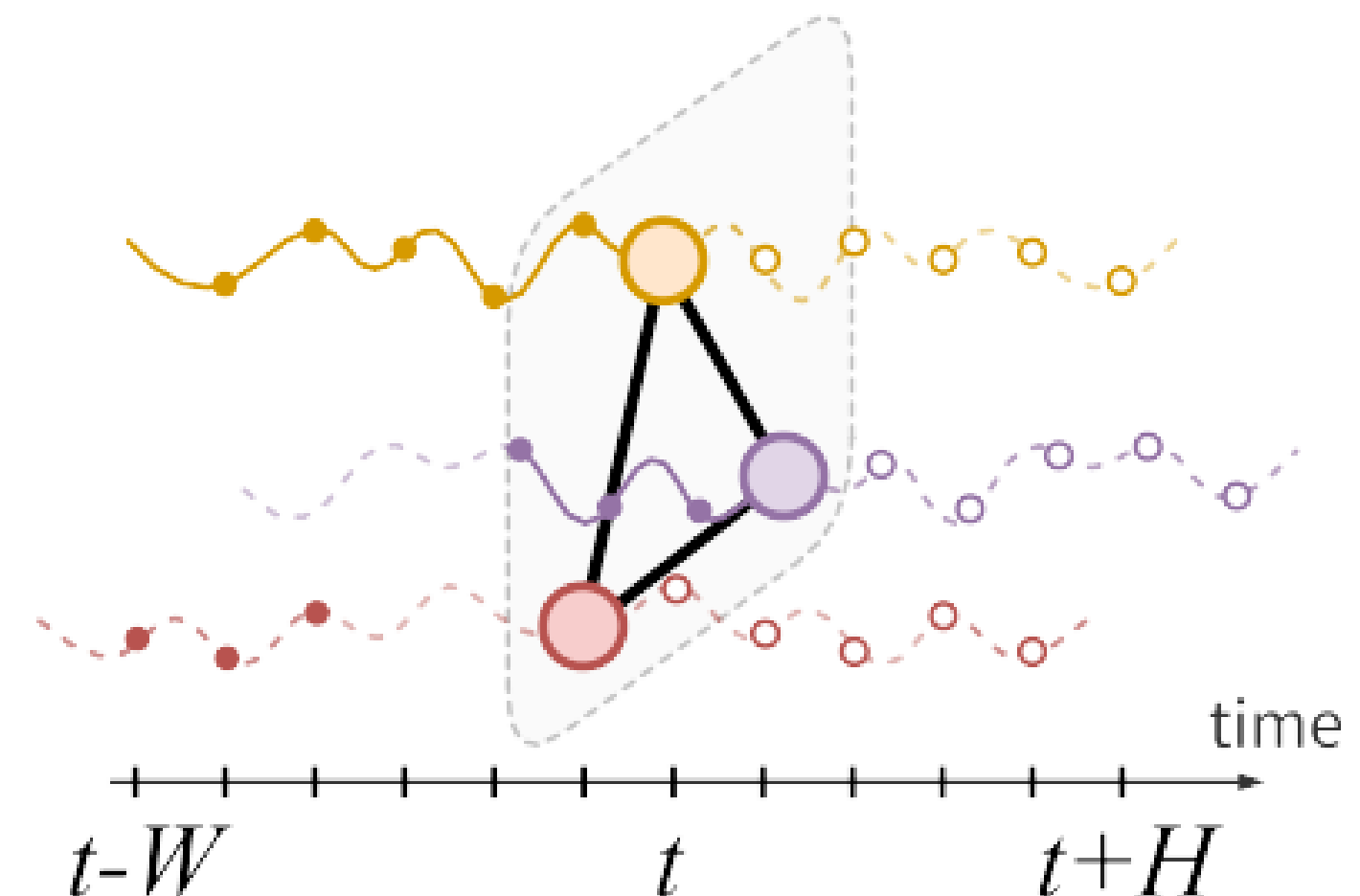
- ✓ Effective in exploiting past (and future, with bidirectional models) **node** observations...
- ✗ ...but struggle in capturing **nonlinear space-time dependencies**.

Forecasting from Partial Observations

A more direct approach to the problem is to **avoid the reconstruction step** and consider forecasting architecture that can **directly deal with irregular observations**.

The mechanisms used in imputation models can be adapted to build forecasting architectures.

- Such models can be used to **jointly impute** missing observations and **forecast** future values.



References

1. Li, W., & Law, K. L. E. (2024). Deep learning models for time series forecasting: A review. *IEEE Access*, 12, 92306–92327.
2. Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.
3. Benidis, K., Rangapuram, S.S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L. and Aubet, F.X., 2022. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6), pp.1-36.
4. Cini, A., Marisca, I., Zambon, D., & Alippi, C. (2023). Graph deep learning for time series forecasting. arXiv preprint arXiv:2310.15978.
5. Cini, A., Zambon, D., & Alippi, C. (2023). Sparse graph learning from spatiotemporal time series. *Journal of Machine Learning Research*, 24(115), 1–38.
6. Cini, A., Marisca, I., Zambon, D., & Alippi, C. (2023). Taming local effects in graph-based spatiotemporal forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*.
7. Cini, A., Marisca, I., Bianchi, F. M., & Alippi, C. (2023). Scalable spatiotemporal graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6), 7218–7226.
8. Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4), 12–25.
9. Gadgil, S., Zhao, Q., Pfefferbaum, A., Sullivan, E. V., Adeli, E., & Pohl, K. M. (2020). Spatio-temporal graph convolution for resting-state fMRI analysis. arXiv preprint arXiv:2003.10613.
10. Geng, X., Li, Y., Wang, L., Zhang, L., Yang, Q., Ye, J., & Liu, Y. (2019). Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 3656–3663)*.
11. Gruver, N., et al. (2023). Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems (NeurIPS)*.
12. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)* (pp. 1025–1035).
13. Huang, S., Wang, D., Wu, X., & Tang, A. (2019). DSANet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)* (pp. 2129–2132).
14. Jin, G., Liang, Y., et al. (2023). Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.
15. Jin, M., Wen, Q., Liang, Y., Zhang, C., Xue, S., Wang, X., Zhang, J., Wang, Y., Chen, H., Li, X., Pan, S., Tseng, V. S., Zheng, Y., Chen, L., & Xiong, H. (2023). Large models for time series and spatio-temporal data: A survey and outlook. arXiv preprint.
16. Jin, M., et al. (2024). Position: What can large language models tell us about time series analysis. *Proceedings of the 41st International Conference on Machine Learning (ICML)*.
17. Jin, M., et al. (2024). Time-LLM: Time series forecasting by reprogramming large language models. *Proceedings of the International Conference on Learning Representations (ICLR)*.
18. Karahanoglu, F. I., & Van De Ville, D. (2013). Total activation: fMRI deconvolution through spatio-temporal regularization. *NeuroImage*, 73, 121–134.
19. Lai, S., et al. (2024). LLMLight: Large language models as traffic signal control agents. arXiv preprint.
20. Li, W., & Law, K. L. E. (2024). Deep learning models for time series forecasting: A review. *IEEE Access*, 12, 92306–92327.
21. Li, Y., Li, Y., & Li, Y. (2020). Modeling temporal patterns with dilated convolutions for time-series forecasting. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(4), 1–30.
22. Li, X., Zhou, Y., Gao, L., Zhang, Y., Shen, D., & Huang, H. (2021). BrainGNN: Interpretable brain graph neural network for fMRI analysis. *Medical Image Analysis*, 74, 102233.
23. Liang, Y., Ouyang, K., Yan, H., Wang, Y., Tong, Z., & Zimmermann, R. (2021). Modeling trajectories with neural ordinary differential equations. *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, 207–213.
24. Liang, Y., Ouyang, K., Wang, Y., & Zimmermann, R. (2022). TrajFormer: Efficient trajectory classification with transformers. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM)*, 2129–2132.
25. Liang, Y., Ouyang, K., Wang, Y., & Zimmermann, R. (2021). Fine-grained urban flow prediction. *Proceedings of the Web Conference 2021 (WWW)*, 1629–1639.

1. Liang, Y., Xia, Y., Ke, S., Wang, Y., Zhang, J., Zheng, Y., & Zimmermann, R. (2023). AirFormer: Predicting nationwide air quality in China with transformers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(12), 13789–13797.
2. Liang, Y., et al. (2024). Foundation models for time series analysis: A tutorial and survey. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.
3. Liang, Y., et al. (2023). Exploring large language models for human mobility prediction under public events. *arXiv preprint*.
4. Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., & Xu, Q. (2021). SCINet: Time series modeling and forecasting with sample convolution and interaction. *arXiv preprint arXiv:2106.09305*.
5. Liu, X., Liang, Y., et al. (2023). LargeST: A benchmark dataset for large-scale traffic forecasting. *Advances in Neural Information Processing Systems (NeurIPS)*.
6. Liu, X., Liang, Y., et al. (2024). UniTime: A language-empowered unified model for cross-domain time series forecasting. *Proceedings of the Web Conference 2024 (WWW)*.
7. Liu, Q., Liang, Y., et al. (2024). Time-FFM: Towards LM-empowered federated foundation model for time series forecasting. *arXiv preprint*.
8. Mallick, T., Balaprakash, P., Rask, E., & Macfarlane, J. (2020). Transfer learning with graph neural networks for short-term highway traffic forecasting. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)* (pp. 10338–10345).
9. Panagopoulos, G., Nikolentzos, G., & Vazirgiannis, M. (2021). Transfer graph neural networks for pandemic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5), 3961–3969.
10. Raschka, S. (2023, February 9). Understanding and coding the self-attention mechanism of large language models from scratch. Retrieved from [https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html:contentReference\[oaicite:41\]{index=41}](https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html:contentReference[oaicite:41]{index=41})
11. Rong, Y., Huang, W., Xu, T., & Huang, J. (2020). DropEdge: Towards deep graph convolutional networks on node classification. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
12. Wang, X., et al. (2024). Where would I go next? Large language models as human mobility predictors. *arXiv preprint*.
13. Wu, L., Wen, H., Liang, Y., et al. (2024). LaDe: The first comprehensive last-mile delivery dataset from industry. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.
14. Xia, Y., Liang, Y., et al. (2023). Deciphering spatio-temporal graph forecasting: A causal lens and treatment. *Advances in Neural Information Processing Systems (NeurIPS)*.
15. Xu, X., et al. (2023). Temporal data meets LLM: Explainable financial time series forecasting. *arXiv preprint*.
16. Yan, Y., Liang, Y., et al. (2024). UrbanCLIP: Learning text-enhanced urban region profiling with contrastive language-image pretraining from the web. *Proceedings of the Web Conference 2024 (WWW)*.
17. Yang, R., Dai, R., Gao, S., Tang, X., Zhou, F., & Zhu, H. (2025). Spatio-temporal prediction of fine-grained origin-destination matrices with applications in ridesharing. *arXiv preprint arXiv:2503.24237*.
18. Zhang, X., Roy Chowdhury, R., Gupta, R. K., & Shang, J. (2024). Large language models for time series: A survey. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 821–829).
19. Zhang, Q., et al. (2023). Spatio-temporal graph learning with large language model. *OpenReview*.
20. Zhang, H., Liang, Y., et al. (2024). Predicting carpark availability in Singapore with cross-domain data: A new dataset and a data-driven approach. *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*.
21. Zhou, F., Luo, S., Qie, X., Ye, J., & Zhu, H. (2024). Causal probabilistic spatio-temporal fusion transformers in two-sided ride-hailing markets. *ACM Transactions on Spatial Algorithms and Systems*, 10(3), Article 28.
22. Zhou, T., Niu, P., Wang, X., Sun, L., & Jin, R. (2023). One fits all: Power general time series analysis by pretrained LM. *Advances in Neural Information Processing Systems (NeurIPS)*.
23. Zhou, Z., et al. (2024). Large language model empowered participatory urban planning. *arXiv preprint*.
24. Zhu, Y., Liang, Y., et al. (2024). ControlTraj: Controllable trajectory generation with topology-constrained diffusion model. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

How to succeed in this course?

```
from __future__ import print_function
import torch
import torch.nn as nn
import torch.nn.functional as F

# the neural network
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 16, 5)
        self.conv2 = nn.Conv2d(16, 16, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Practice



Explore



Visualize

Discuss



Ask

