A Survey on Deep Learning Advances for Time Series Forecasting

@GeminiLight 知

Content

Introduction

Time Series

Given $\boldsymbol{X}_1, \boldsymbol{X}_2, \dots, \boldsymbol{X}_t$

Find $\boldsymbol{Y}_{t+\Delta}$ $\boldsymbol{X}_{t+\Delta}$ $x_{i,t+\Delta}$

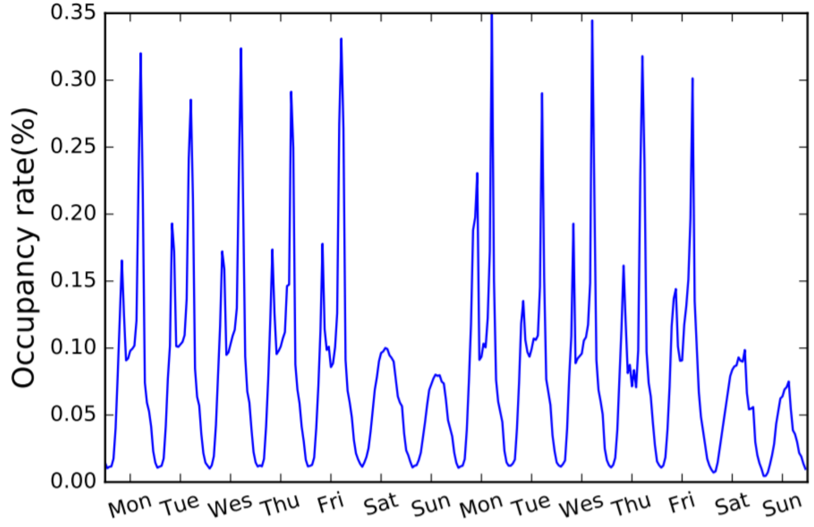
$x_{i,t+\Delta}, \dots, x_{i,t+\Delta},$

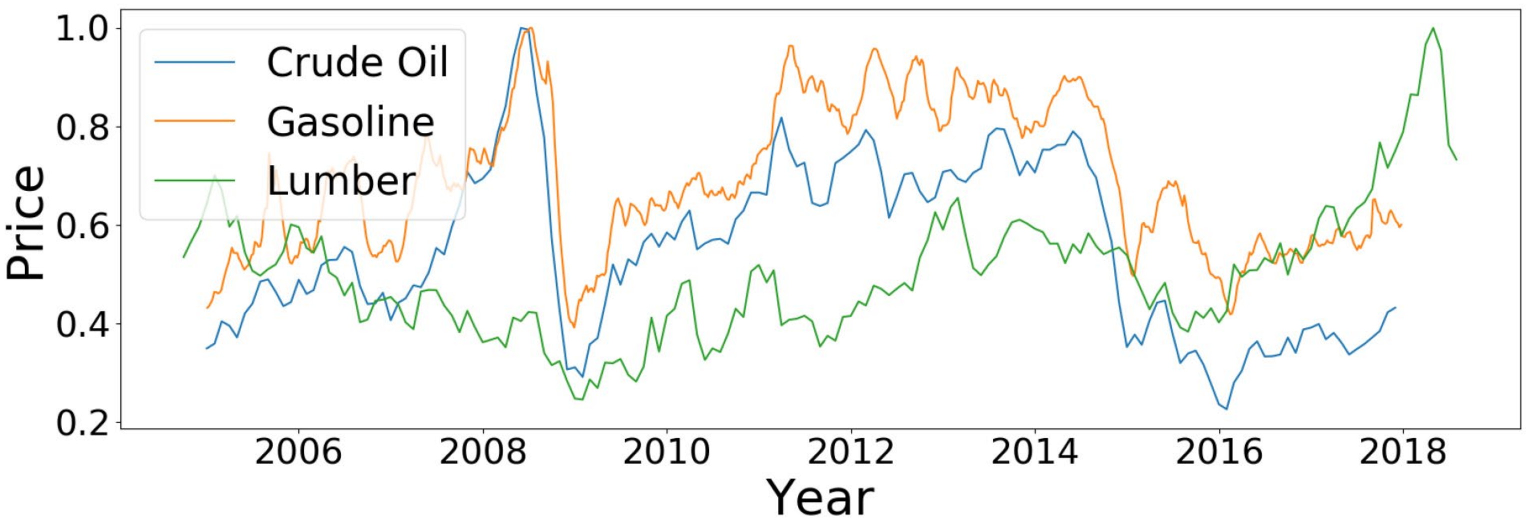
Characteristic

- High dimensionality
- Much noise
- Insufficient or unavailable

Types of Trends

- Trend
- Seasonal
- Cyclical
- Irregular





Introduction

Categories

Forecasting step

One-step | Multi step

Outputting results

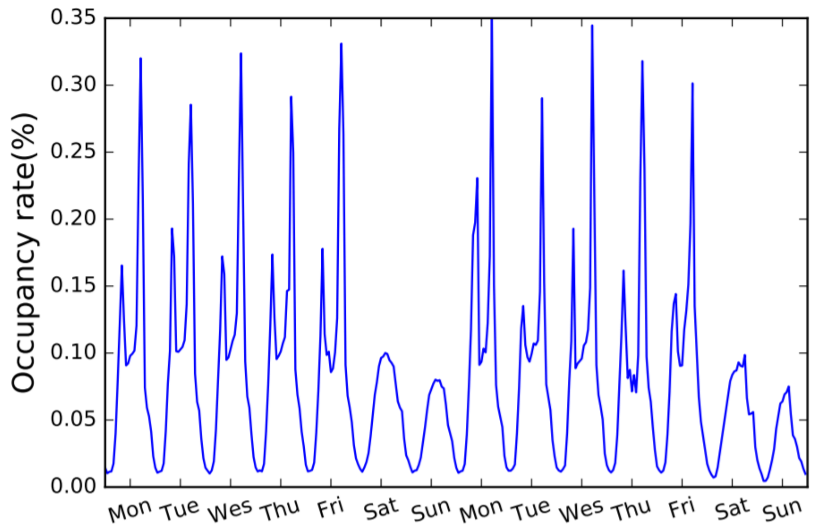
Single point | Probability

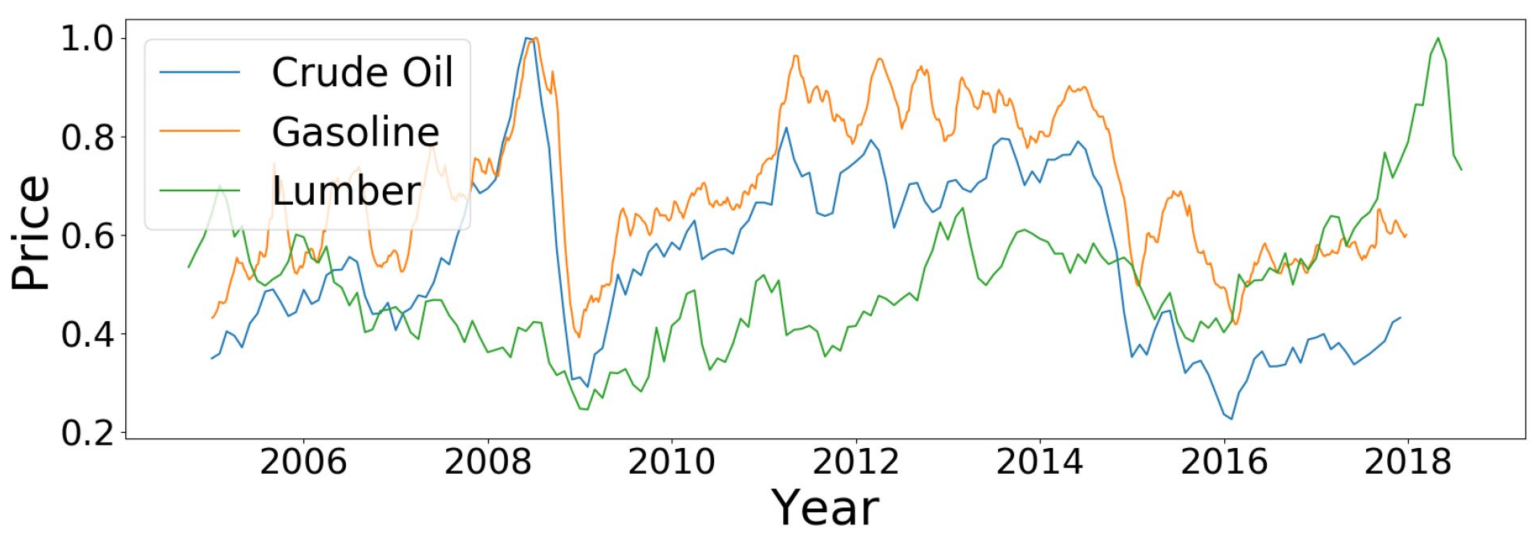
Inputting variables

Autoregressive | Covariate

Forecasting target

Univariate | Multivariate





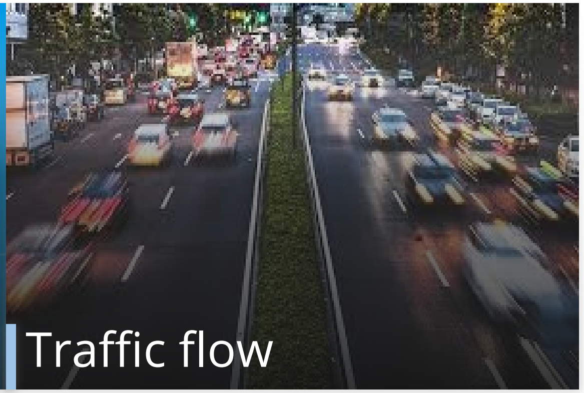
Introduction

Applications

Production Sales

Stock price

Weather forecast



Traffic flow

Implementation principle

Statistical methods	Machine learning
Statistical learning	Deep learning

- Support and can handle multivariate inputs
- Capture complex nonlinear relationships
- May not require a scaled or stationary time series as input

Classical

AR (Auto Regressive)

$$x_t = \emptyset_1 x_{t-1} + \emptyset_2 x_{t-2} + \cdots + \emptyset_p x_{t-p} + u_t$$

$\emptyset_i$: Autoregressive coefficient
u_t: White noise

MA (Moving Average)

$$x_t = u_t + \emptyset_1 u_{t-1} + \emptyset_2 u_{t-2} + \cdots + \emptyset_3 u_{t-q}$$

$\emptyset_i$: Moving regression coefficient
u_i: White noise

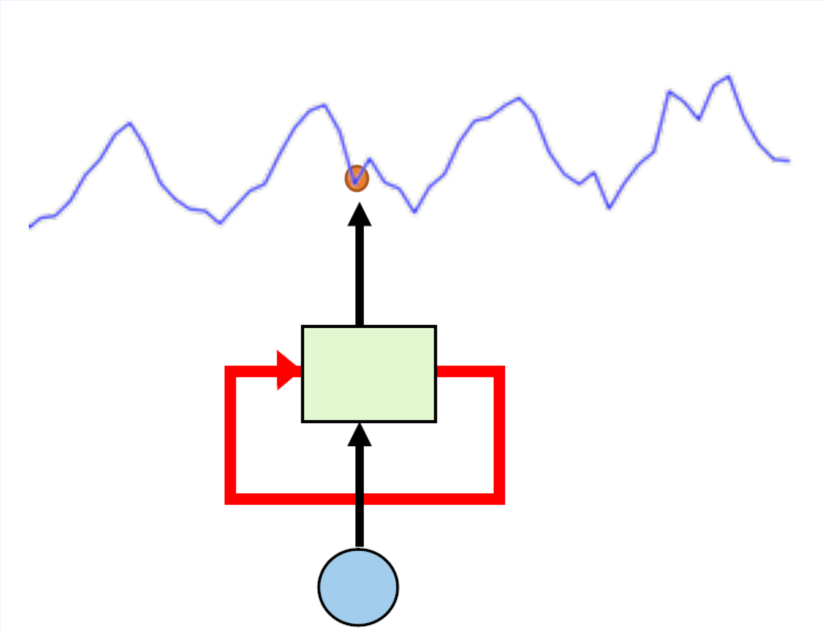
ARMA (Auto Regressive and Moving Average)

$$x_t = u_t + \emptyset_1 u_{t-1} + \emptyset_2 u_{t-2} + \cdots + \emptyset_q u_{t-q} + \vartheta_1 x_{t-1} + \vartheta_2 x_{t-2} + \cdots + \vartheta_p x_{t-p}$$

AR: the relationship between current data and later data
MA: random perturbation (noise)

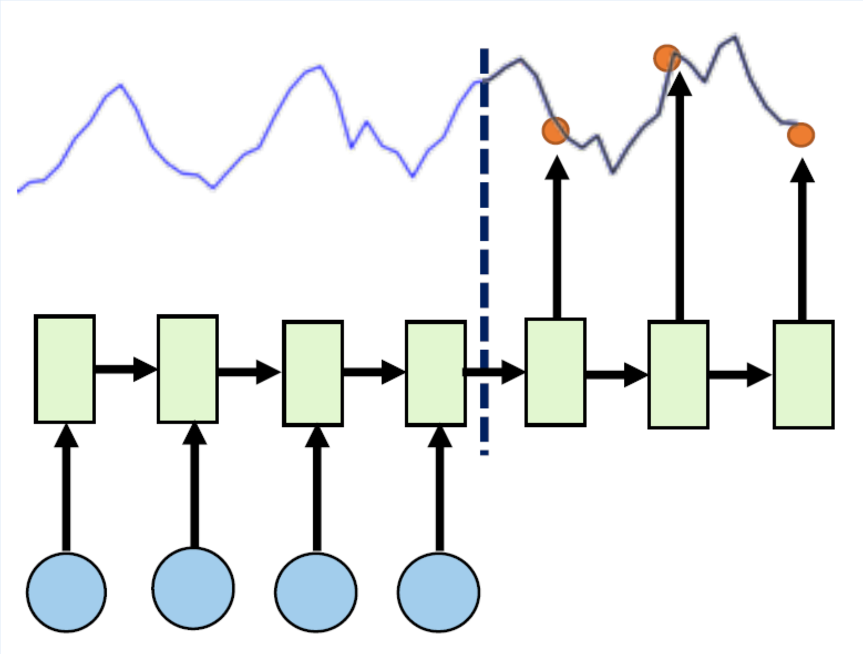
Basic Model

Canonical



One-to-One $f_t\colon x_t \mapsto z_t$

Seq2Seq

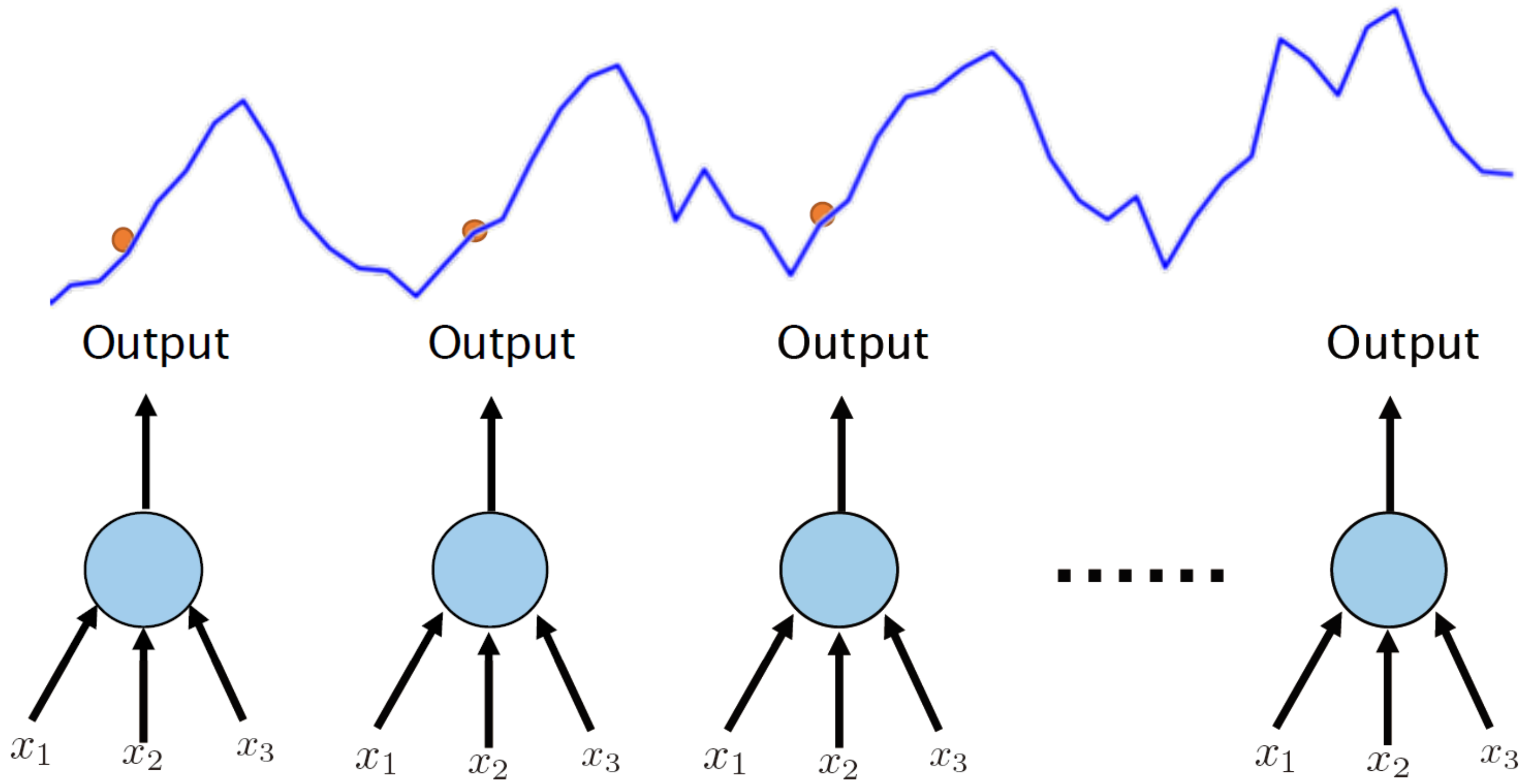


Many-to-Many

$$f\colon \{z_1, \cdots, z_{T_e}\} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$

MLP

Sequential relationship?



$$z_t = \text{DEEP-NET}(\boldsymbol{x}_t)$$

MLP

CNN

Long-Term Sequential relationship?

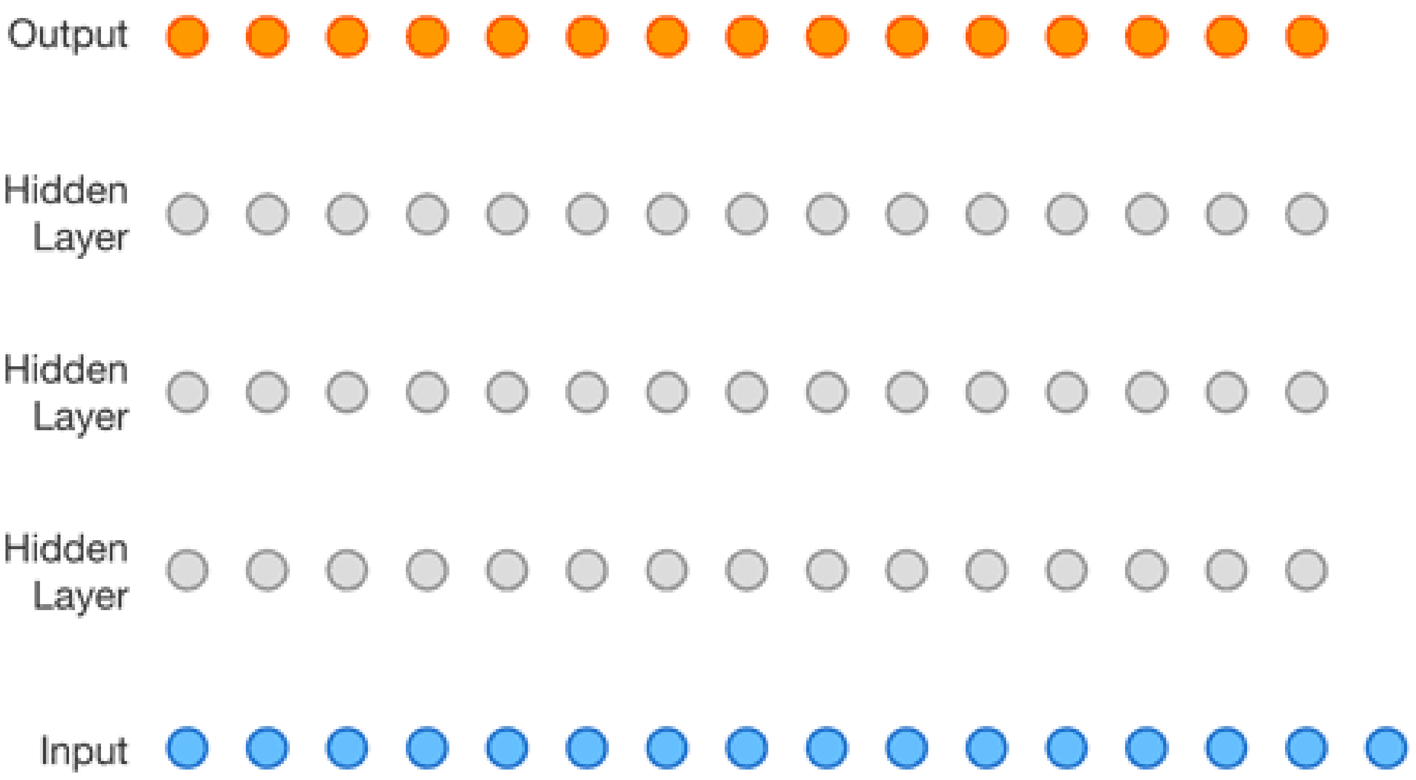


Image from blog by DeepMind

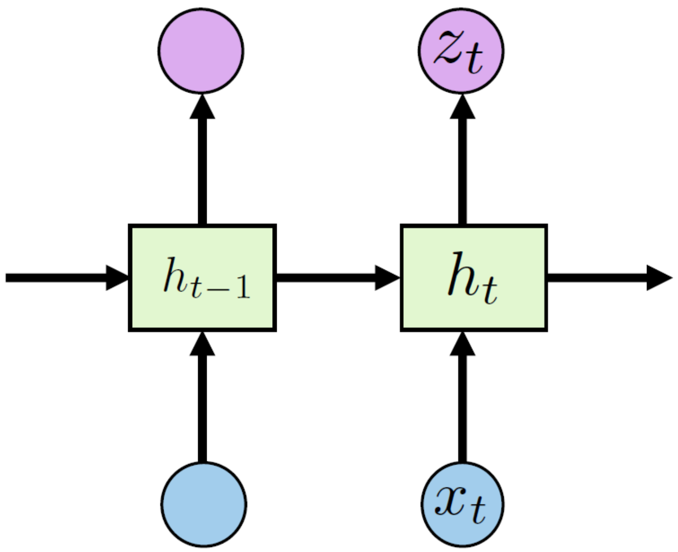
Yet there has been a lack of empirical evidence showing that this type of models can actually capture the temporal dependencies by discovering the latent hierarchical structure of the sequence

RNN

RNN

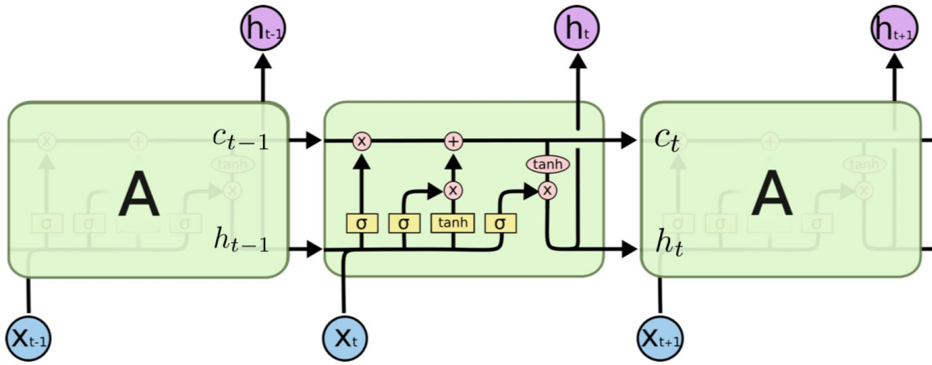
Today = yesterday's information + new knowledge

Recurrent Neural Network (RNN)



$$h_t = \sigma(\theta_0 h_{t-1} + \theta_1 x_t)$$
$$z_t = \sigma(\theta h_t)$$

Long Short-Term Memory (LSTM)



$$c^t = z^f \odot c^{t-1} + z^i \odot z$$
$$h^t = z^o \odot \tanh(c^t)$$
$$y^t = \sigma(W' h^t)$$

$$C_t = \alpha_t \cdot C_{t-1} + \beta_t \times \sigma(\theta_0 h_{t-1} + \theta_1 x_t)$$

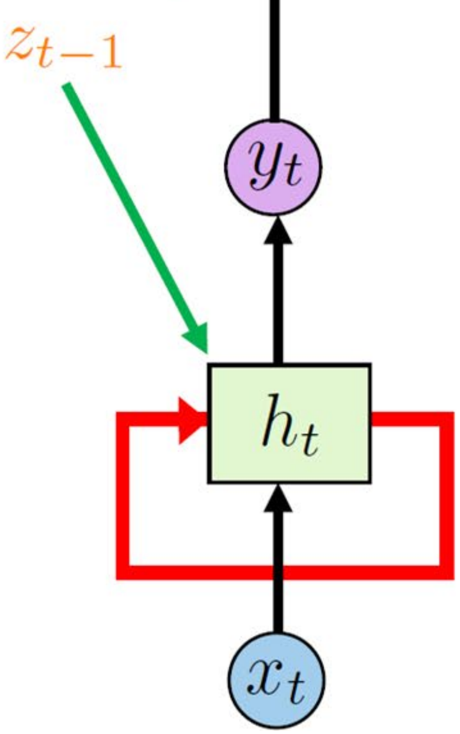
current state = forget gate × old stuff + input gate × new stuff

Exponential Smoothing $s_i = (1-\alpha)s_{i-1} + \alpha x_i$

hidden state: Change faster

cell state: Change slowly

DeepAR

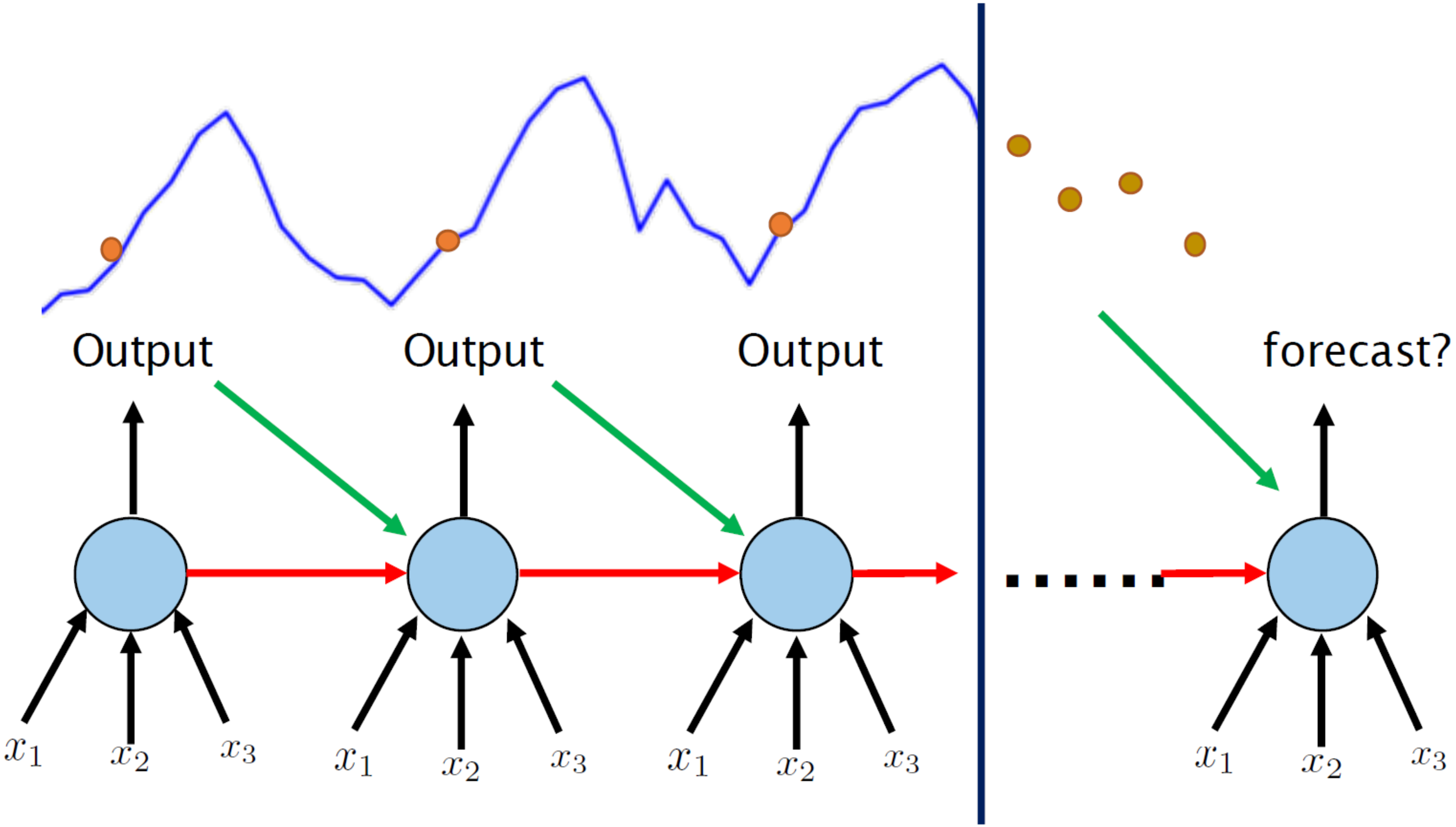


Similar to Autoregressive Input

$$\begin{pmatrix} h_{t-1} \\ x_t \\ z_{t-1} \end{pmatrix}$$

Previous state
Current features
Last observed value

DeepAR



David Salinas et al. “DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks”. *arXiv*, 2017

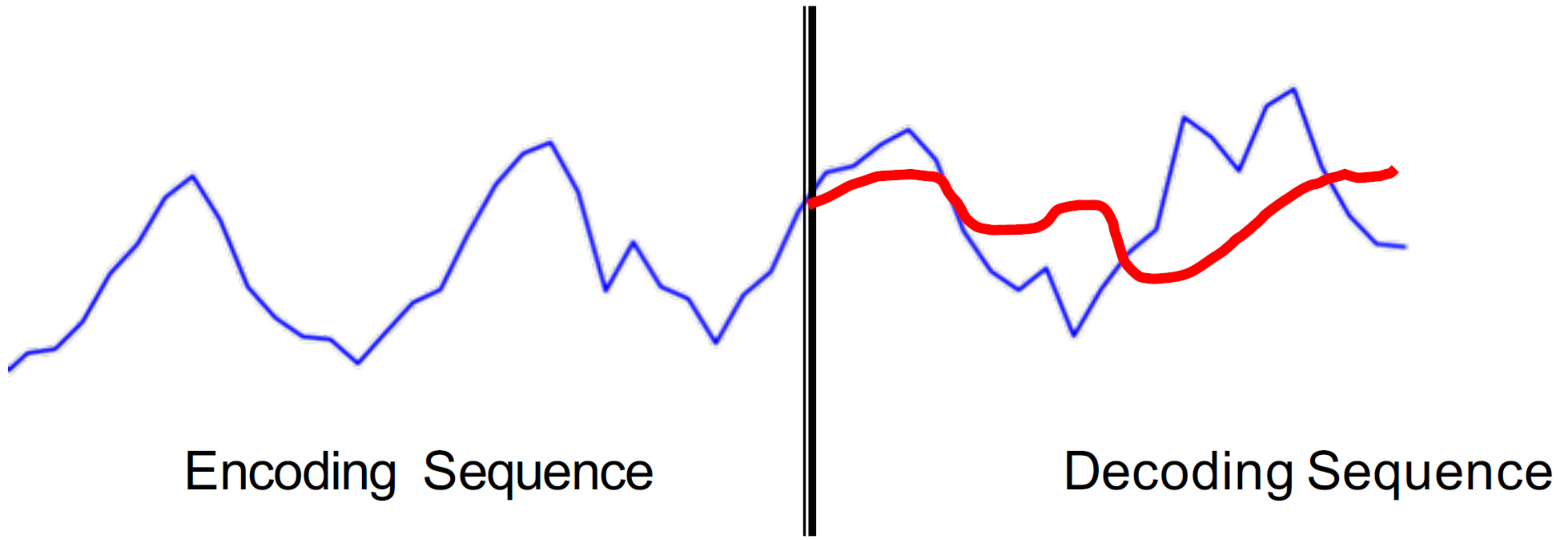
Seq2Seq

Encoder

$$f_{encoder}: \{z_1, \cdots, z_{T_e}\} \mapsto \boldsymbol{h}_{T_e}$$

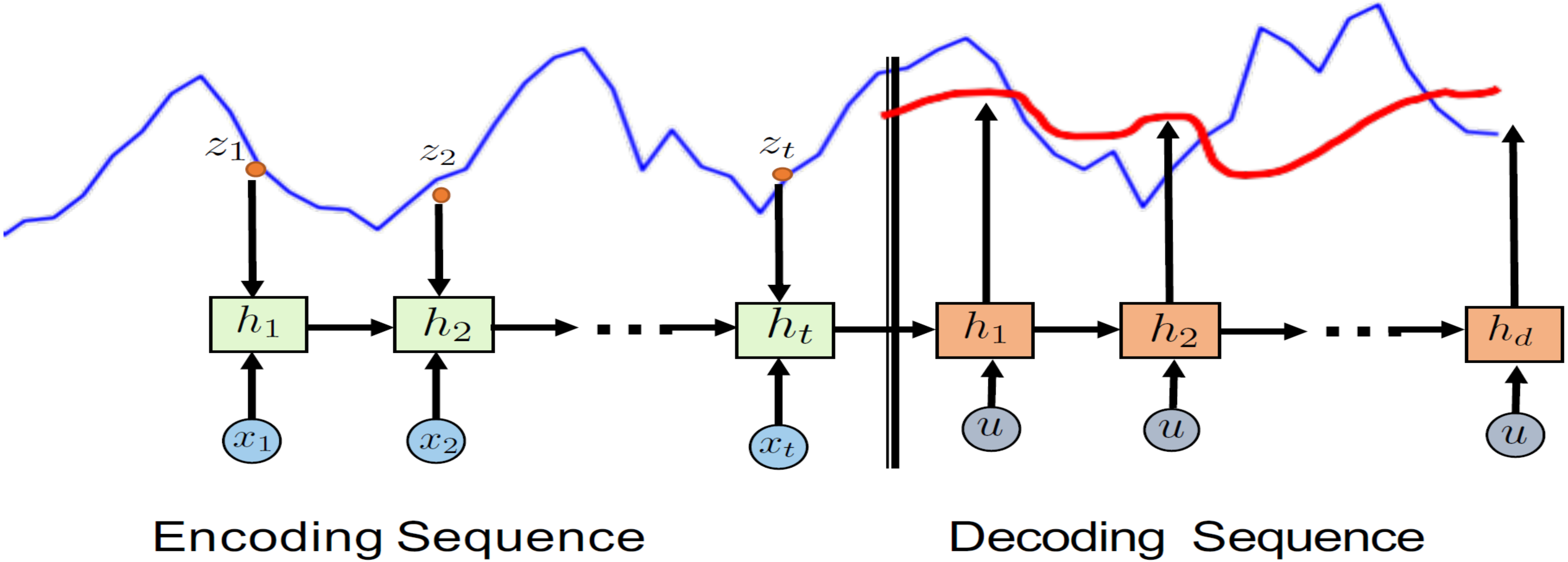
Decoder

$$f_{decoder}: \boldsymbol{h}_{T_e} \mapsto \{z_{T_e+1}, \cdots, z_{T_e+T_d}\}$$



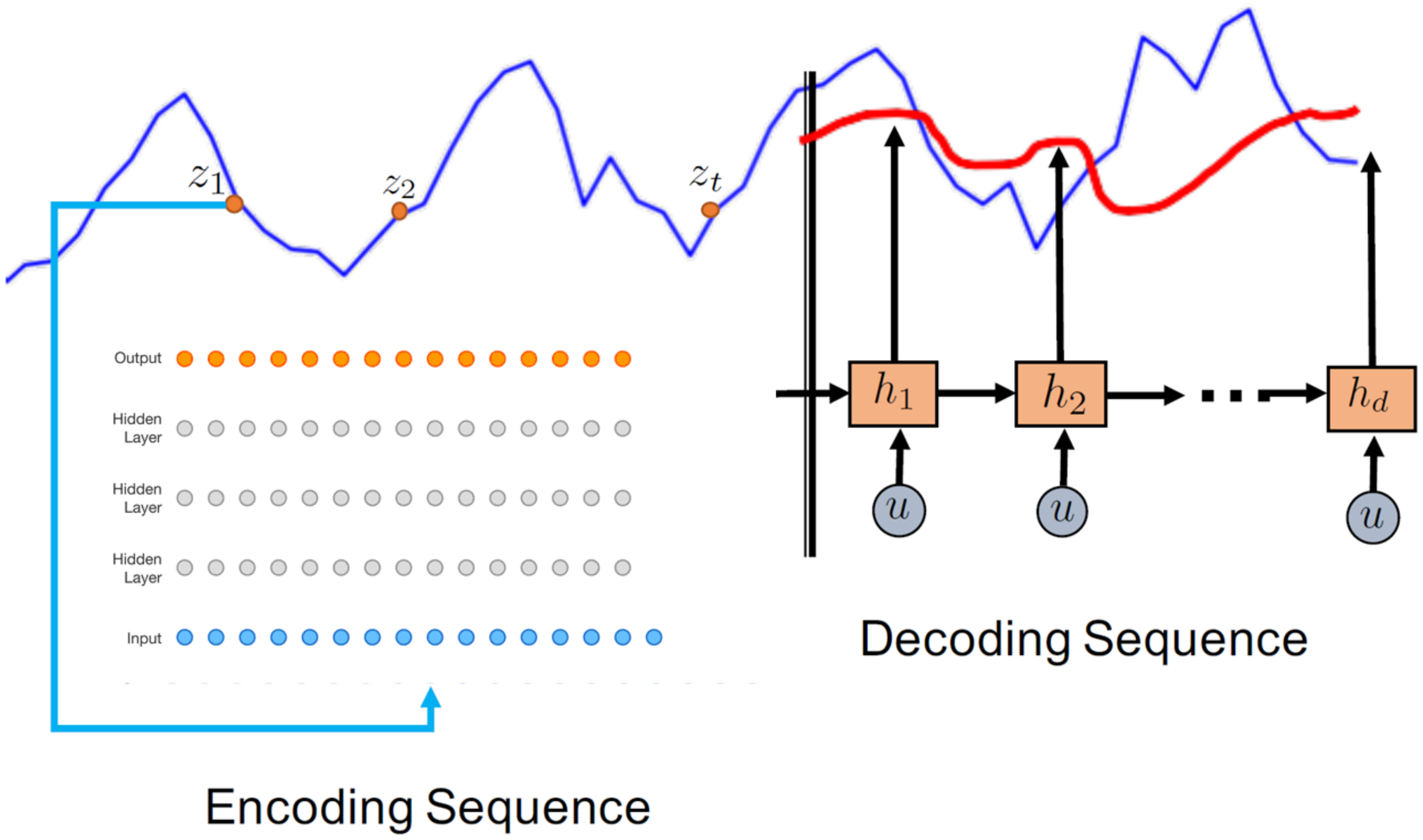
Seq2Seq

RNN-RNN

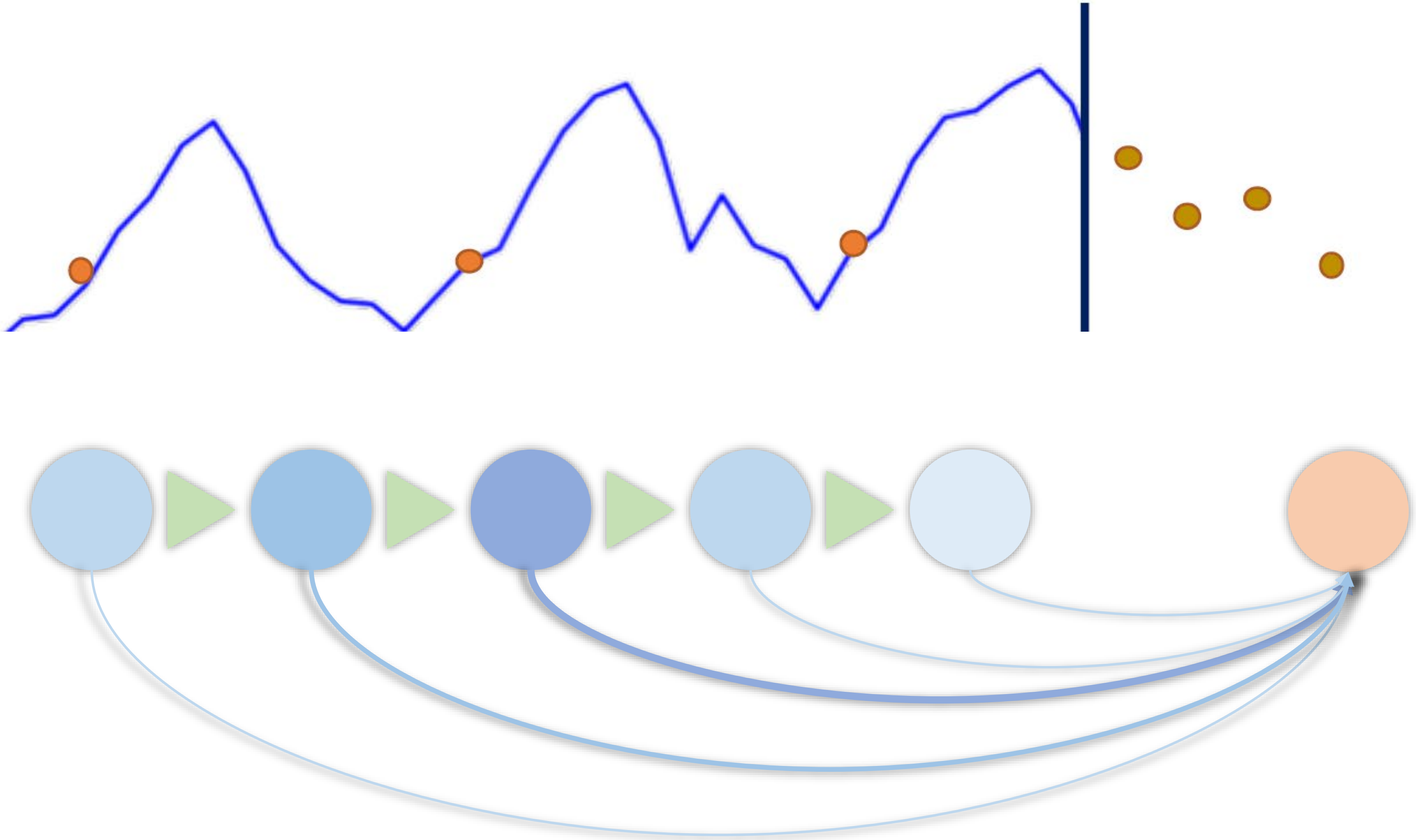


Seq2Seq

CNN-RNN

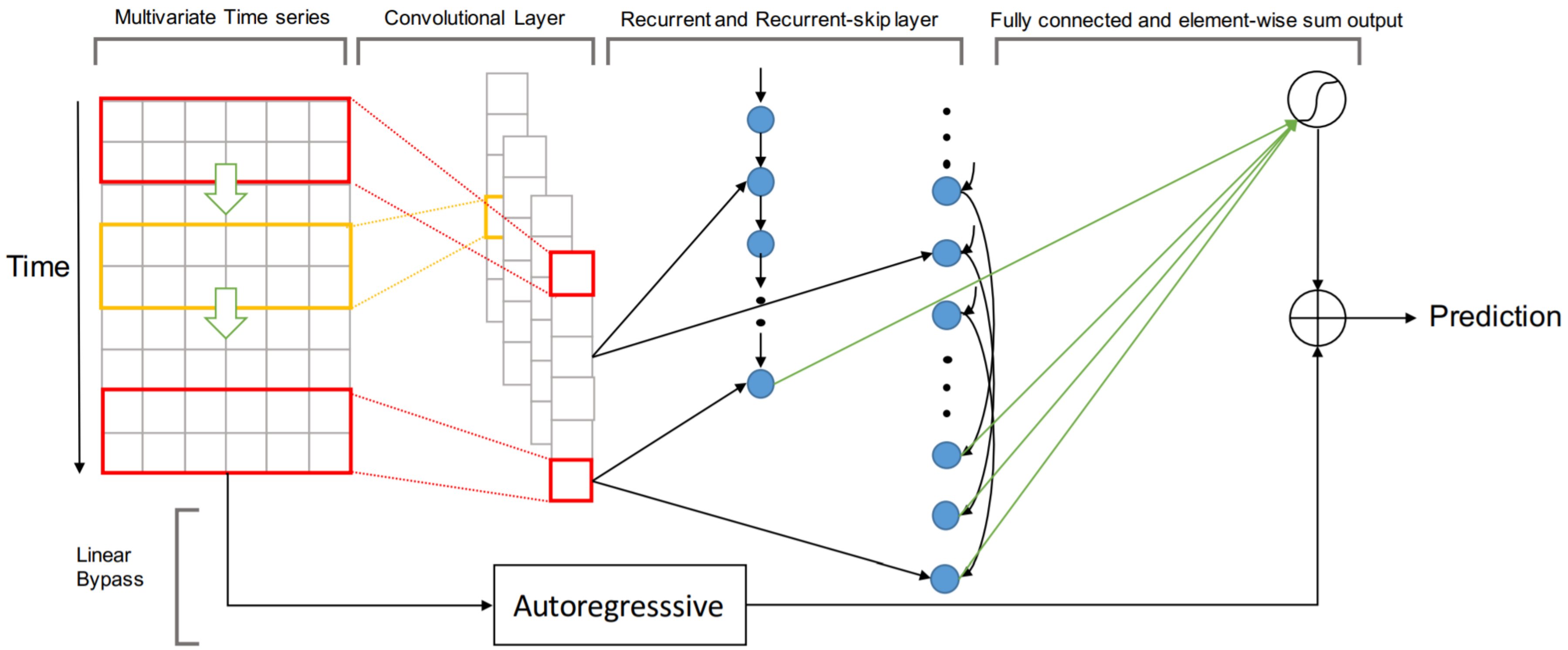


Attention



LSTNet

Input → CNN & AR → RNN & RNN-Skip → FC & Sum → Output



Non-linear

Neural Network

Recurring patterns

Linear

Autoregressive

Focus on local scaling

Long- and Short-term Time-series network (LSTNet)

Guokun Lai et al. "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks". In *SIGIR*, 2018 CCF-A

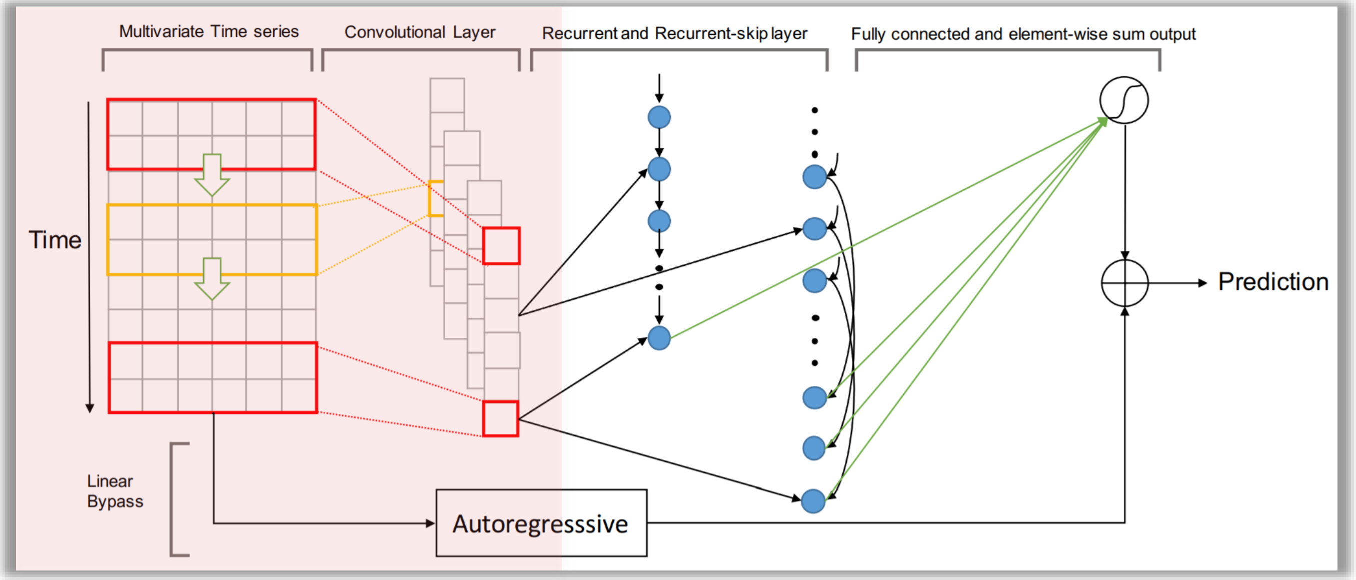
LSTNet

Input → CNN & AR → RNN & RNN-Skip → FC & Sum → Output

CNN

$$h_k = RELU(W_k * X + b_k)$$

Extract short-term patterns in the time dimension as well as local dependencies between variables



Autoregressive

CNN & RNN <- Non-linear nature

the scale of outputs is not sensitive to the scale of inputs

$$h_{t,i}^{L} = \sum_{k=0}^{q^{ar}-1} W_k^{ar} \boldsymbol{y}_{t-k,i} + b^{ar}$$

Final Prediction

$$\hat{Y}_t = h_t^D + h_t^L$$

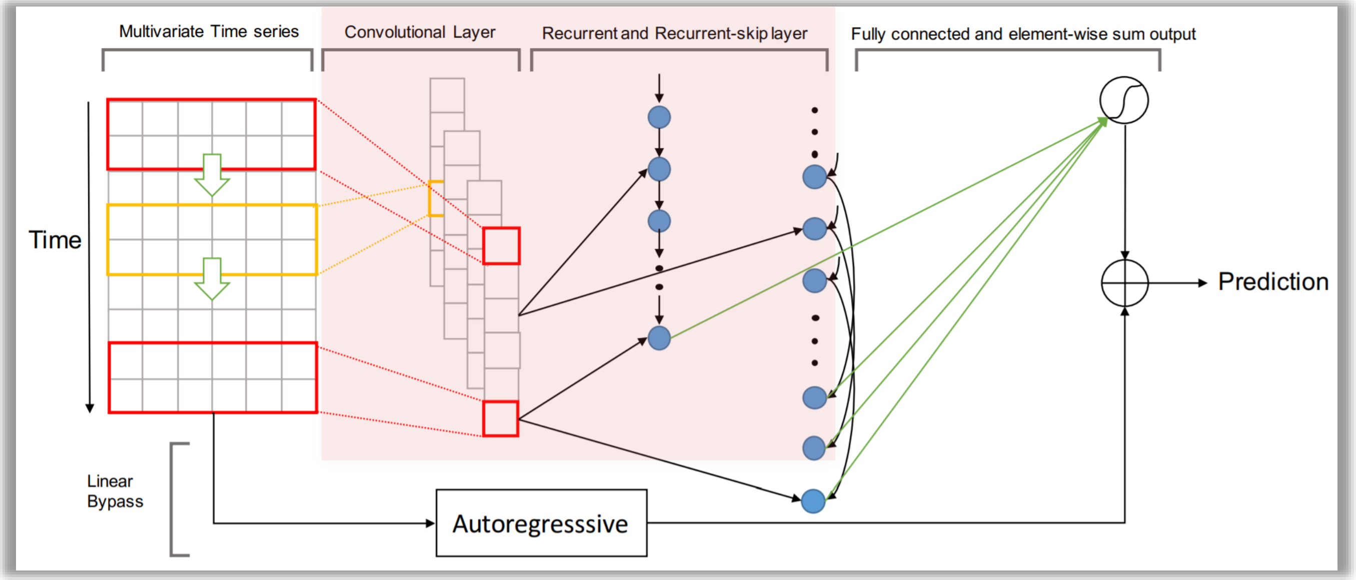
LSTNet

Input → CNN & AR → RNN & RNN-Skip → FC & Sum → Output

Recurrent Component

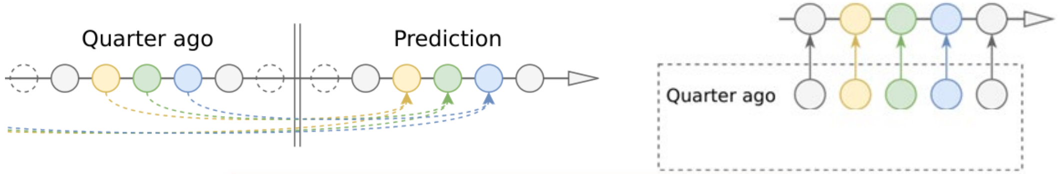
h_t^R

Gated Recurrent Unit (GRU)



Recurrent-skip Component

GRU and LSTM usually fail to capture very long-term correlation in practice

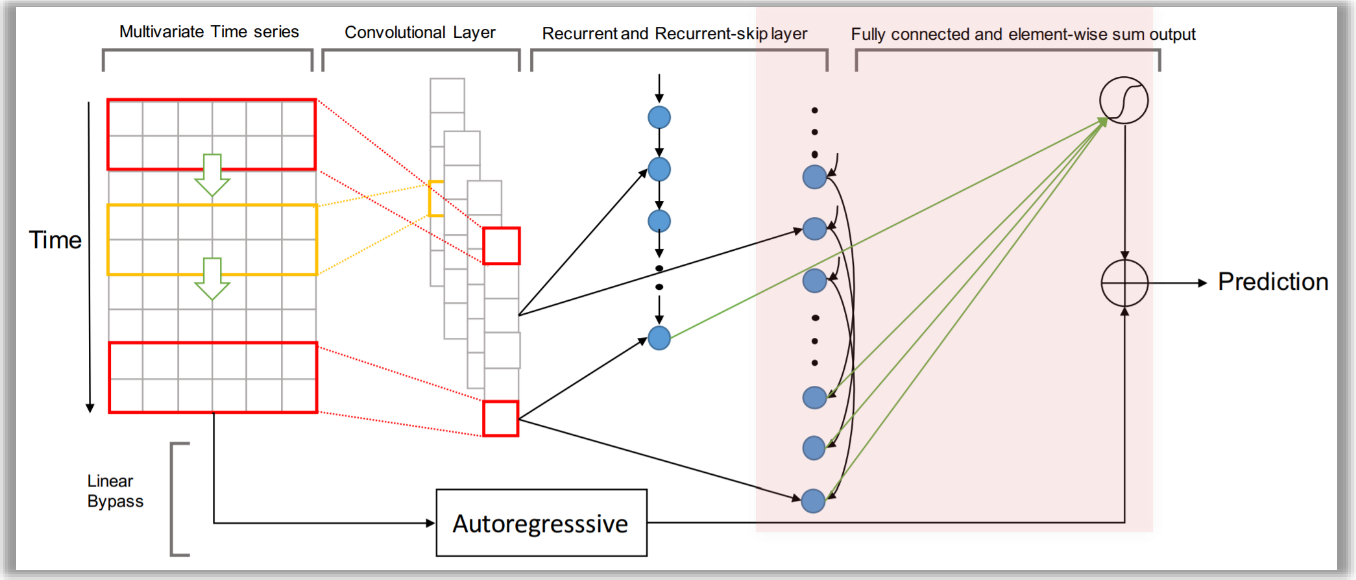


$$\underbrace{h_{t-p+1}^S, h_{t-p+2}^S \ldots, h_t^S}_{\text{period}}$$

LSTNet

Input → CNN & AR → RNN & RNN-Skip → FC & Sum → Output

MLP aggregation



Recurrent Component h_t^R

Recurrent-skip Component

$$h^S_{t-p+1}, h^S_{t-p+2} \ldots, h^S_t$$

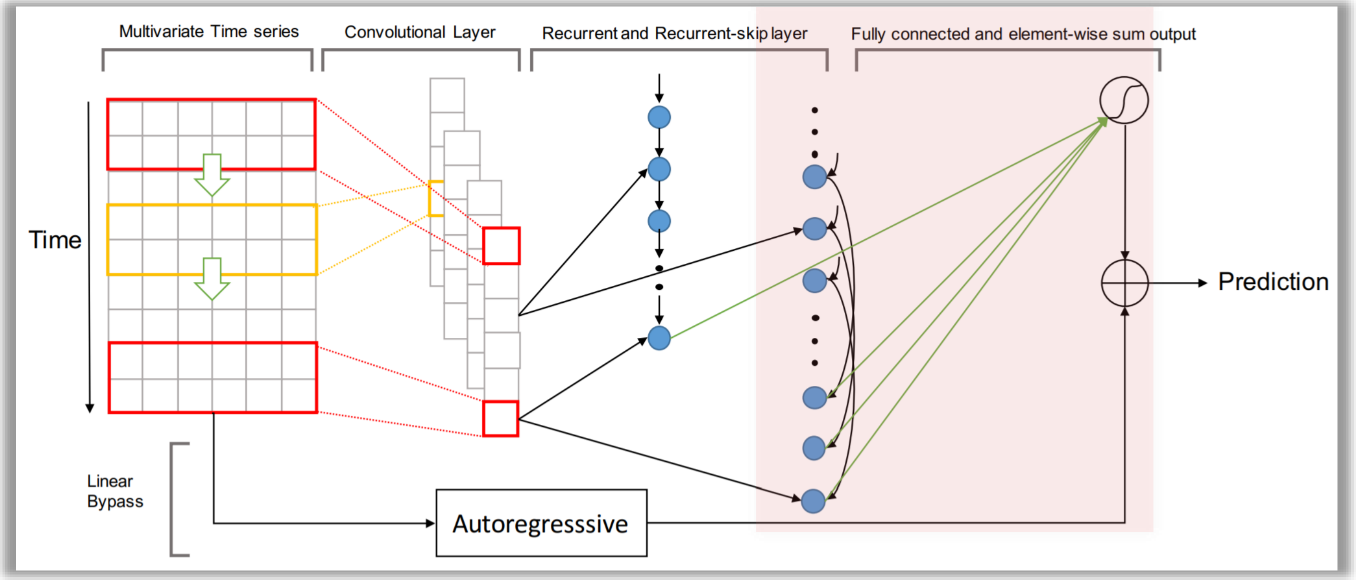
$$h_t^D = W^R h_t^R + \sum_{i=0}^{p-1} W_i^S h_{t-i}^S + b$$

LSTNet

Input → CNN & AR → RNN & RNN-Skip → FC & Sum → Output

period length is dynamic?

Temporal Attention



Recurrent Component h_t^R

$$H_t^R = [h_{t-q}^R, \ldots, h_{t-1}^R]$$

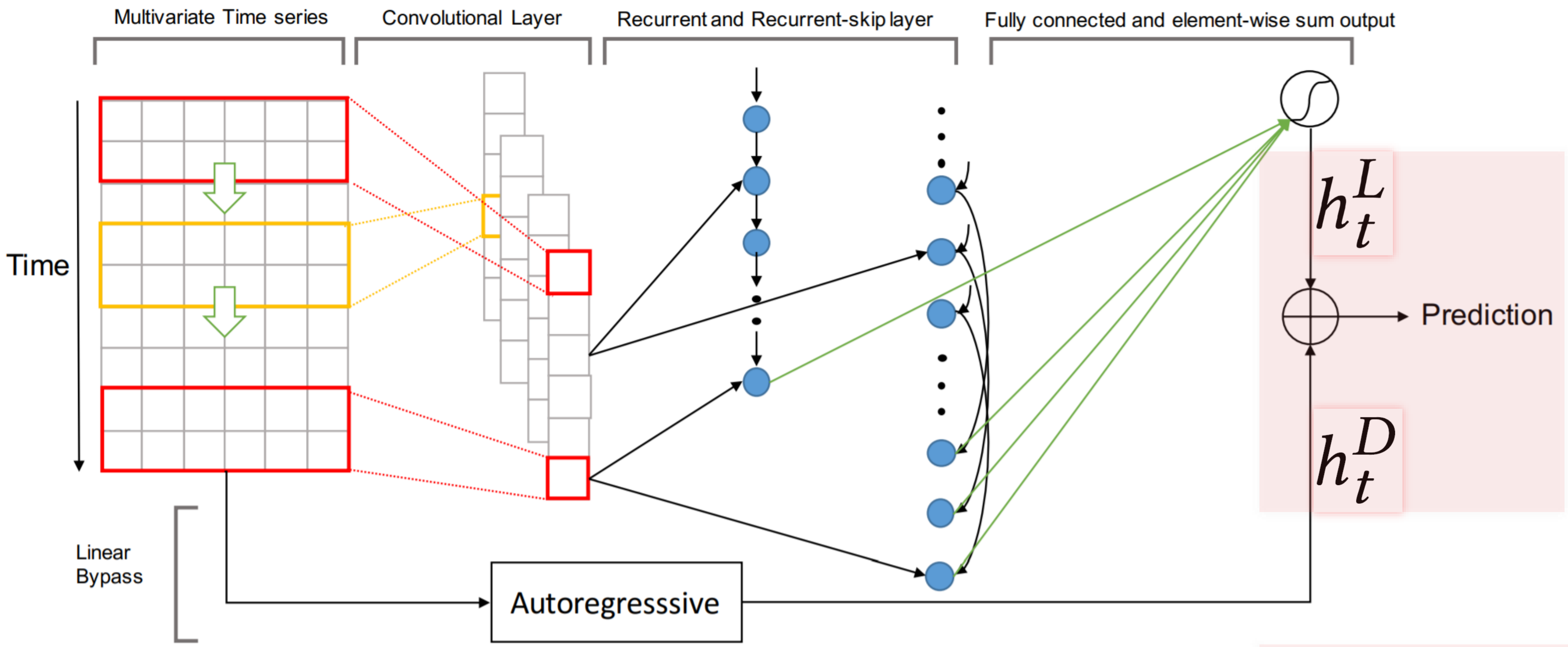
$$\boldsymbol{\alpha}_t = AttnScore(H_t^R, h_{t-1}^R)$$

$$\boldsymbol{c}_t = H_t \boldsymbol{\alpha}_t$$

$$h_t^D = W[\boldsymbol{c}_t; h_{t-1}^R] + b$$

LSTNet

Input → CNN & AR → RNN & RNN-Skip → FC & Sum → Output



Non-linear

Neural Network

Recurring patterns

Linear

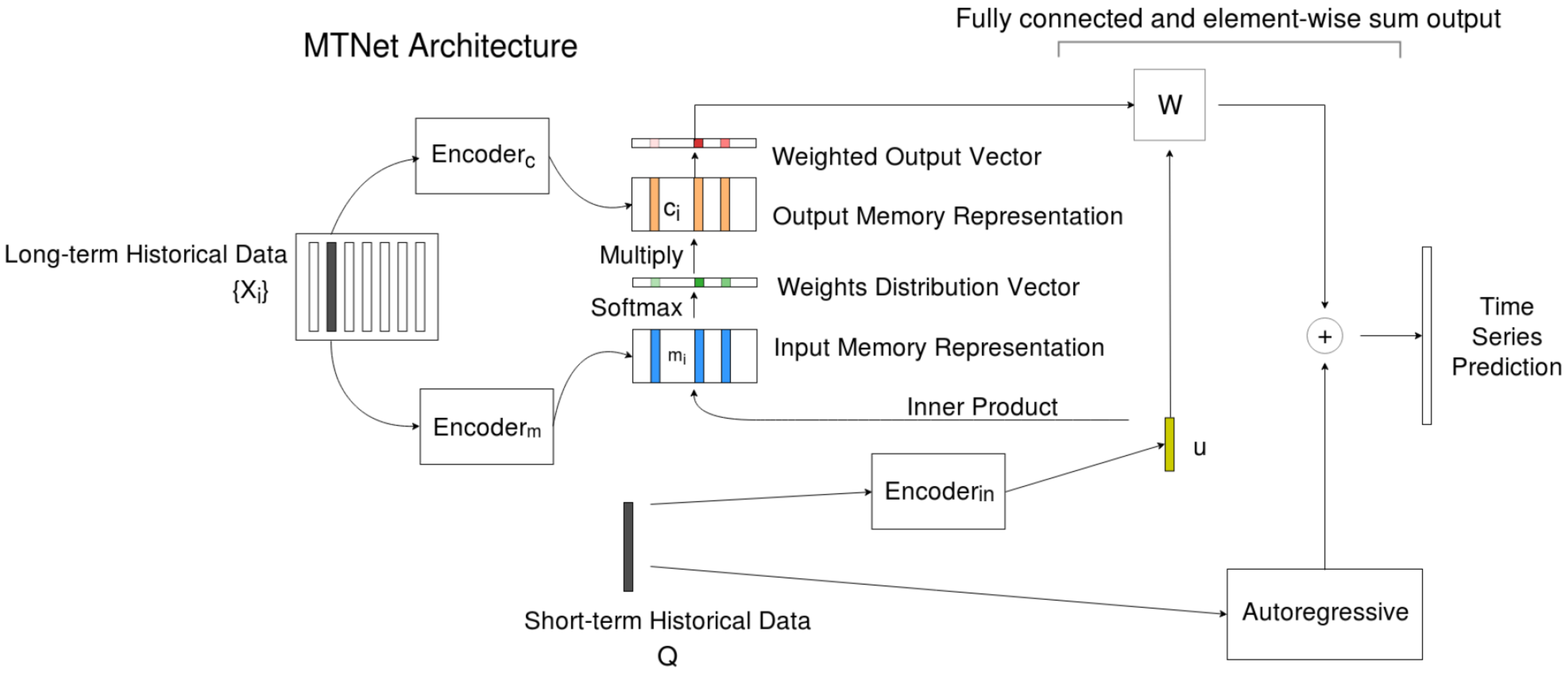
Autoregressive

Focus on local scaling

Long- and Short-term Time-series network (LSTNet)

Final Prediction $\hat{Y}_t = h_t^D + h_t^L$

MTNet



Memory Time-series Network (MTNet)

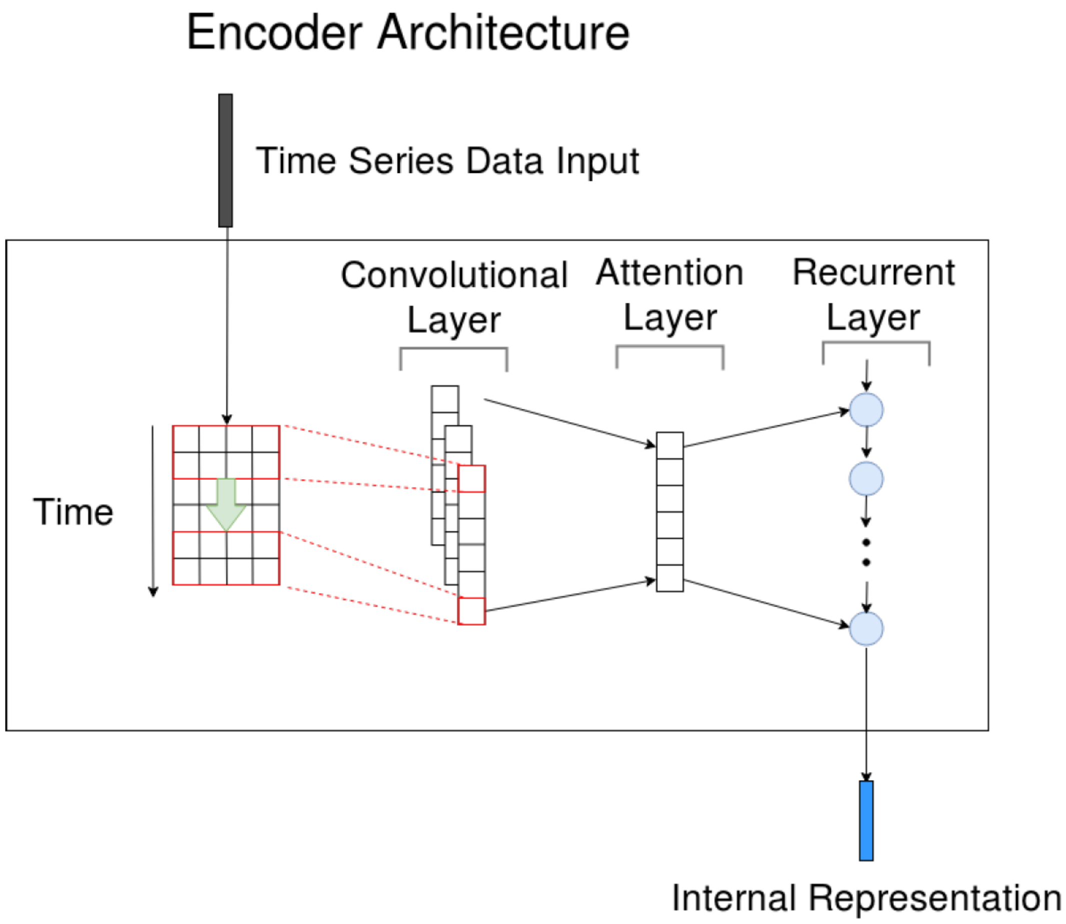
MTNet

Encoder

a set of long-term time series $\boldsymbol{X}$

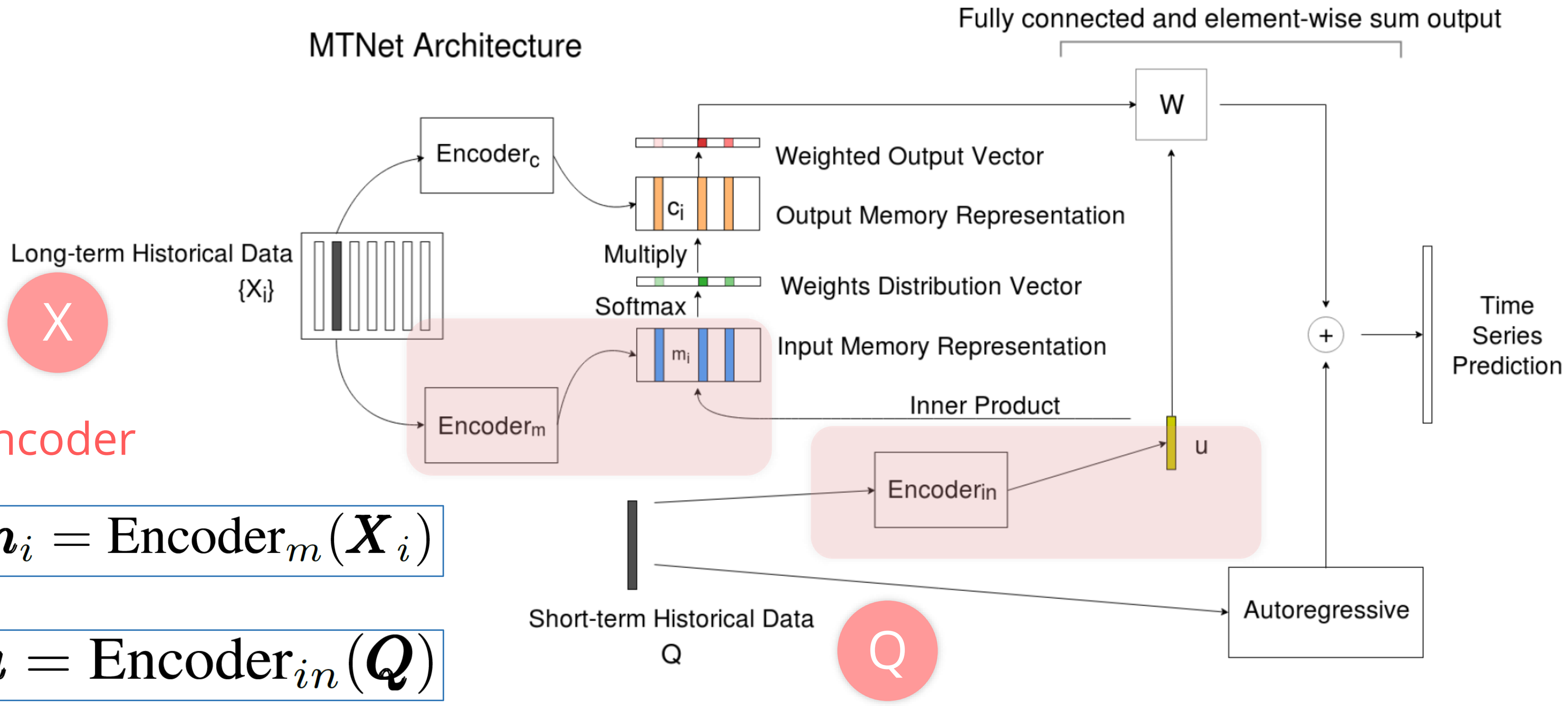
$$\{\boldsymbol{X}_i\} = \boldsymbol{X}_1, \cdots, \boldsymbol{X}_n$$

a short-term historical time series $\boldsymbol{Q}$



Memory Time-series Network (MTNet)

MTNet



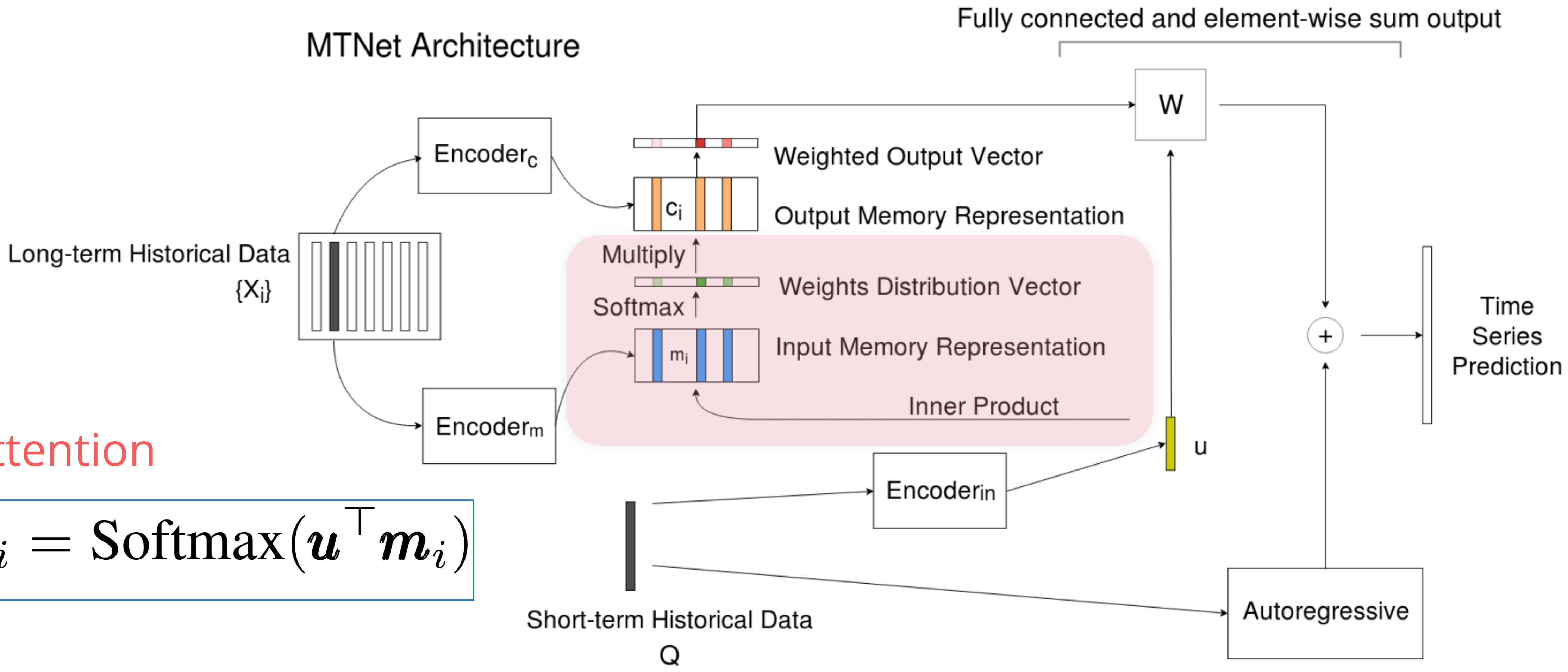
Encoder

$$\boldsymbol{m}_i = \text{Encoder}_m(\boldsymbol{X}_i)$$

$$\boldsymbol{u} = \text{Encoder}_{in}(\boldsymbol{Q})$$

Memory Time-series Network (MTNet)

MTNet

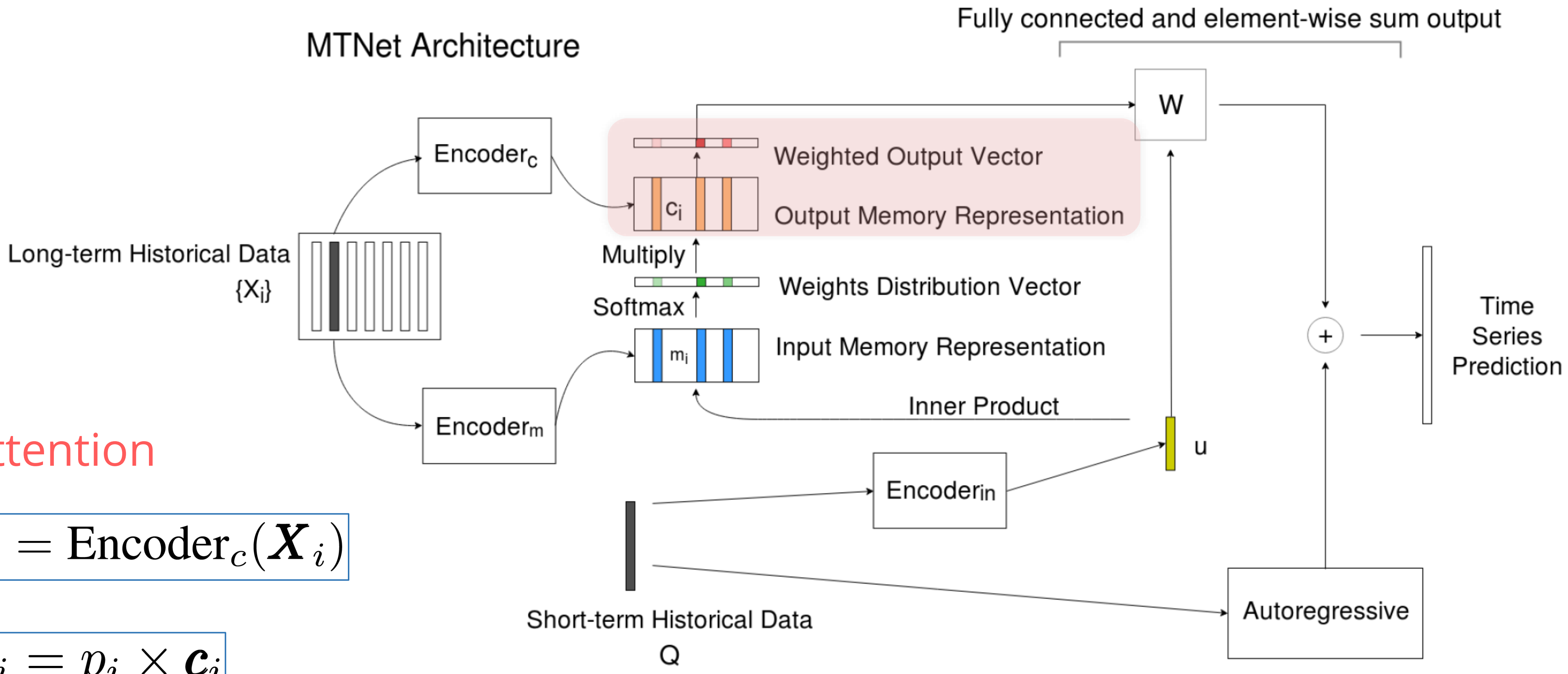


Attention

$$p_i = \mathrm{Softmax}(\boldsymbol{u}^\top \boldsymbol{m}_i)$$

Memory Time-series Network (MTNet)

MTNet



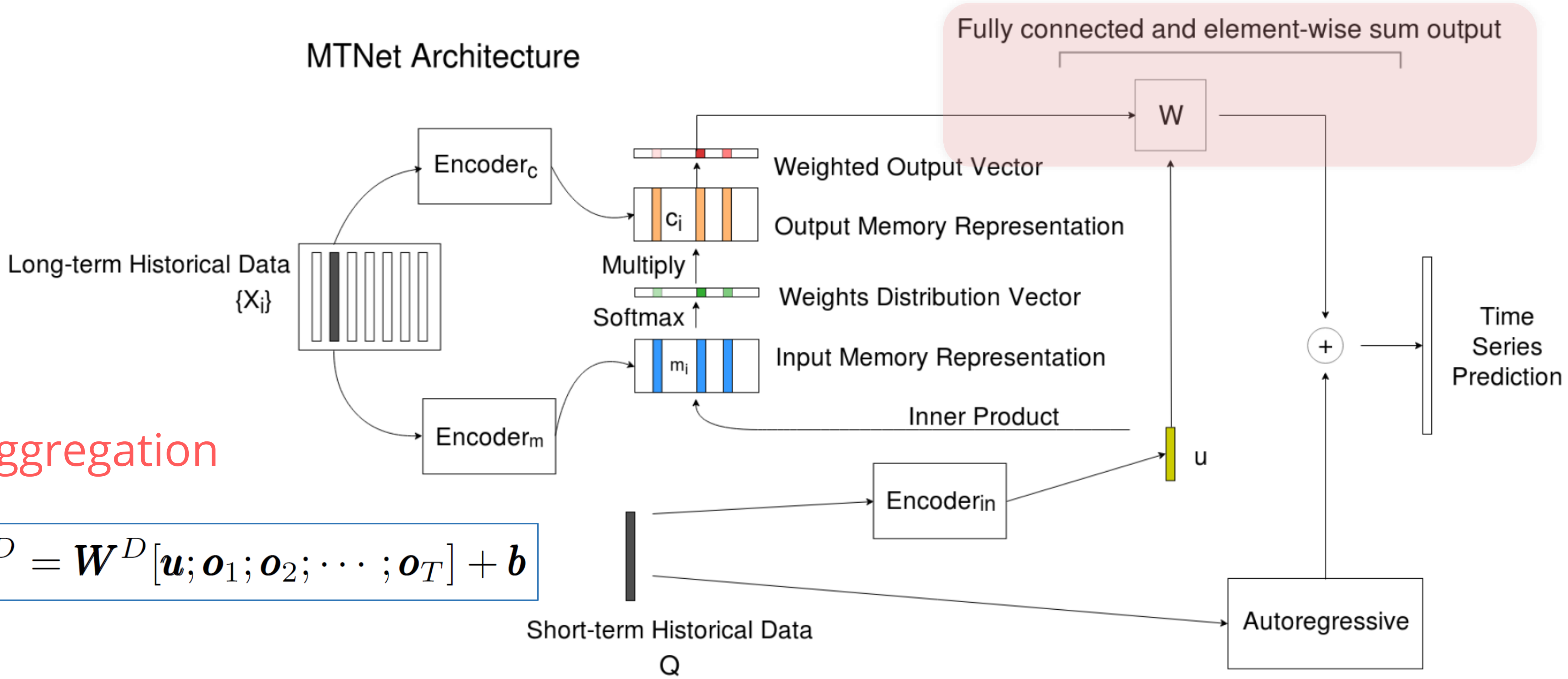
Attention

$$\boxed{\boldsymbol{c}_i = \mathrm{Encoder}_c(\boldsymbol{X}_i)}$$

$$\boxed{\boldsymbol{o}_i = p_i \times \boldsymbol{c}_i}$$

Memory Time-series Network (MTNet)

MTNet



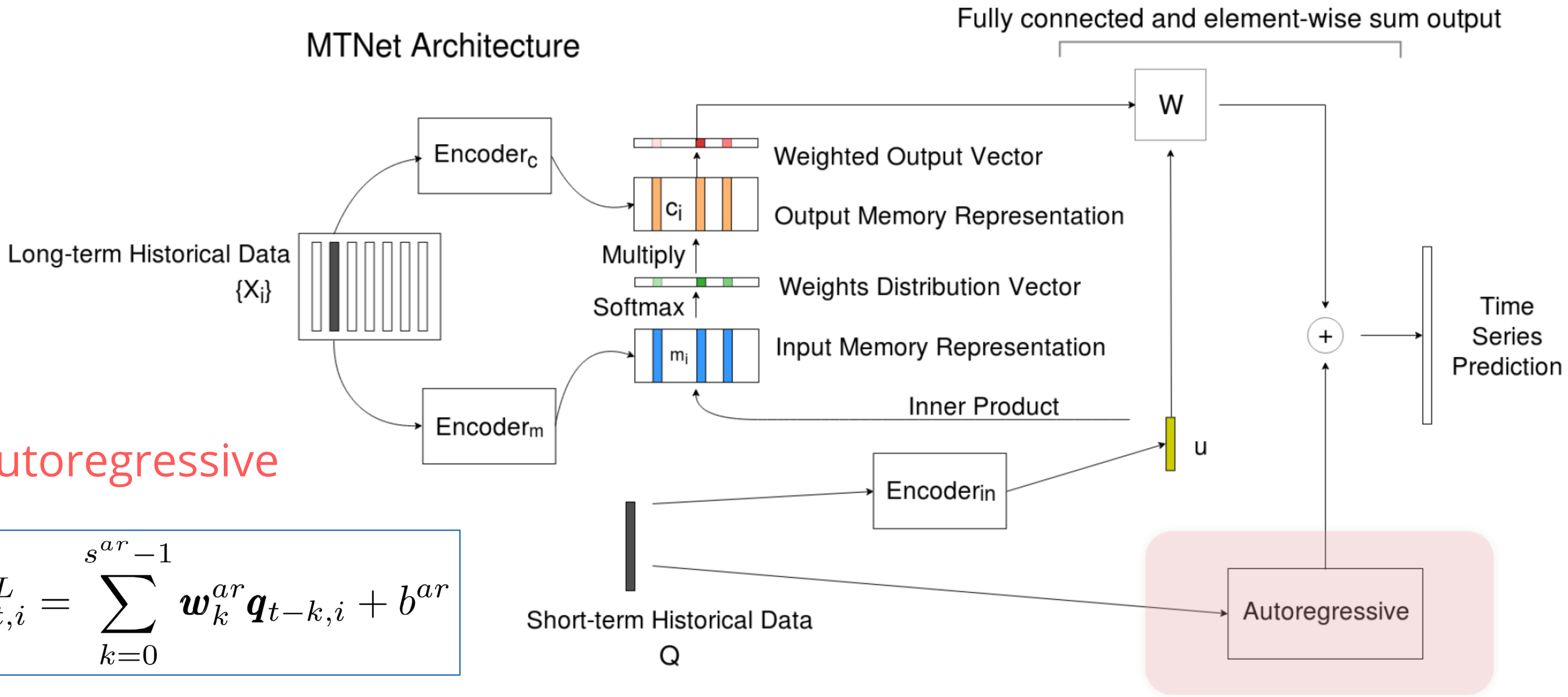
Aggregation

$$y_t^D = W^D[u; o_1; o_2; \cdots ; o_T] + b$$

Memory Time-series Network (MTNet)

Yen-Yu Chang et al. "A Memory-Network Based Solution for Multivariate Time-Series Forecasting". In *AAAI*, 2019 CCF-A

MTNet



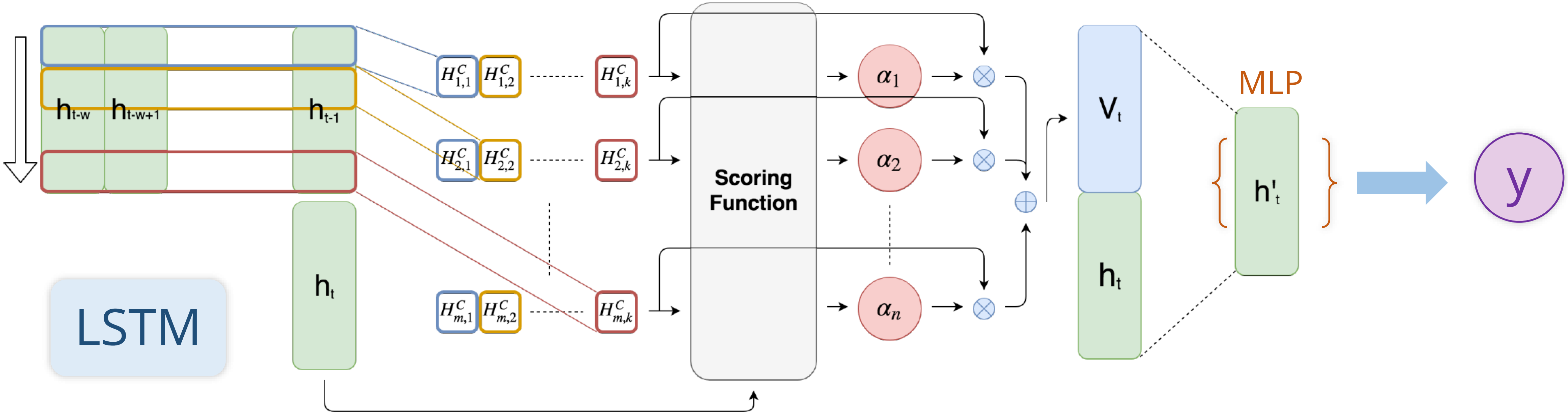
Autoregressive

$$
\boldsymbol{y}_{t,i}^{L} = \sum_{k=0}^{s^{ar}-1} \boldsymbol{w}_{k}^{ar} \boldsymbol{q}_{t-k,i} + b^{ar}
$$

Memory Time-series Network (MTNet)

Yen-Yu Chang et al. "A Memory-Network Based Solution for Multivariate Time-Series Forecasting". In *AAAI*, 2019 CCF-A

TPA-LSTM



Temporal Pattern Attention

Time steps

Attention

Inputting Variables

Transformer

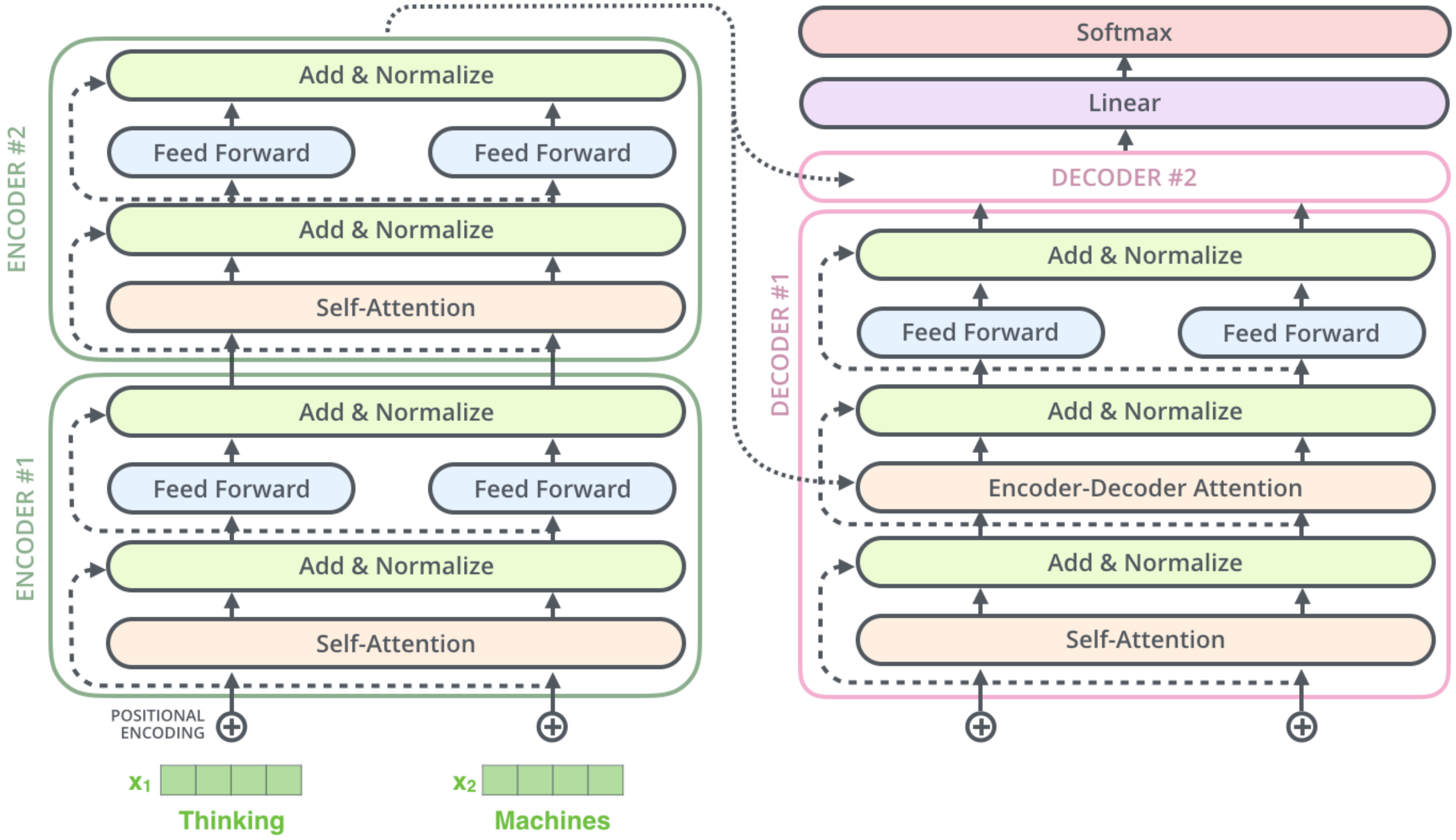
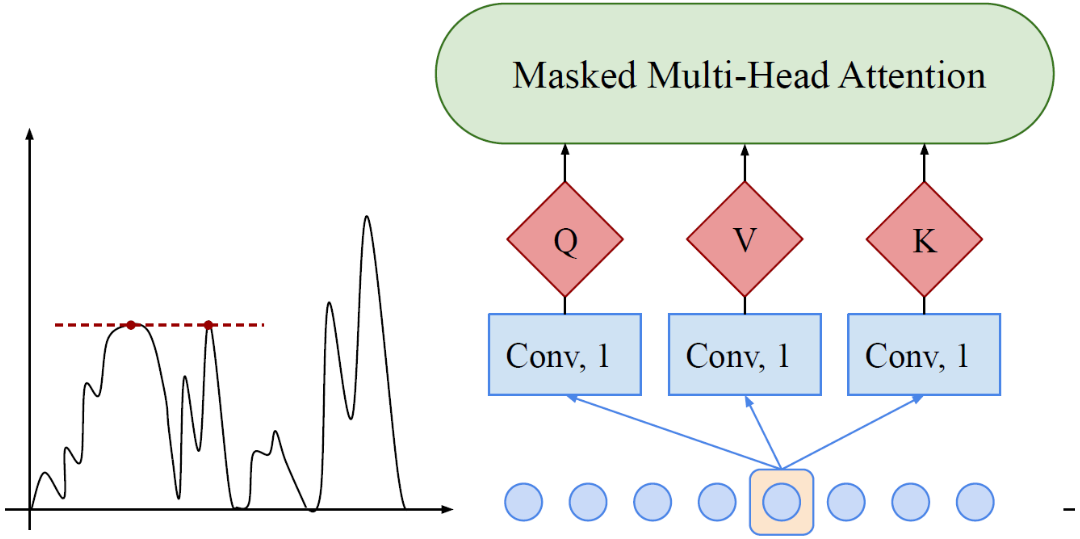


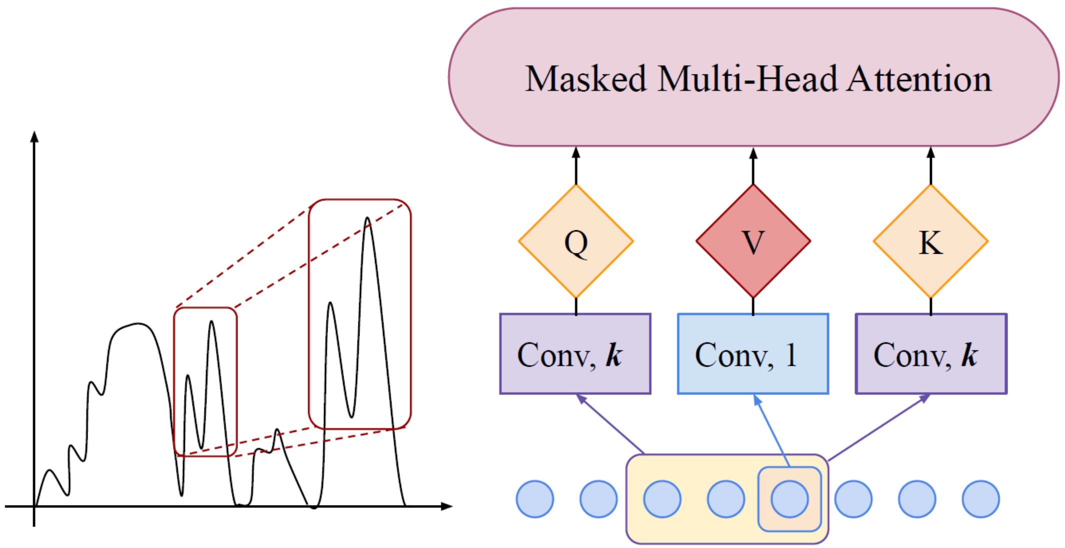
Image from Blog by Jalammar

Enhancing the locality of Transformer

Canonical self-attention

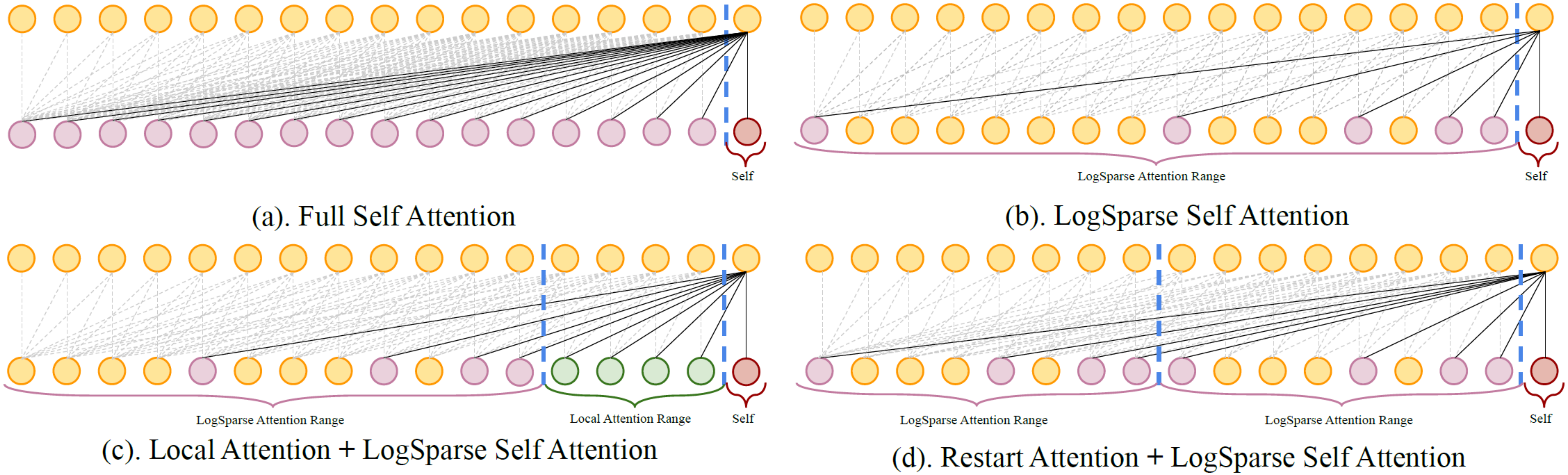


Convolutional self-attention



Shiyang Li et al. "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting". In *NeurIPS*, 2019 CCF-A

Breaking the memory bottleneck of Transformer



(a). Full Self Attention

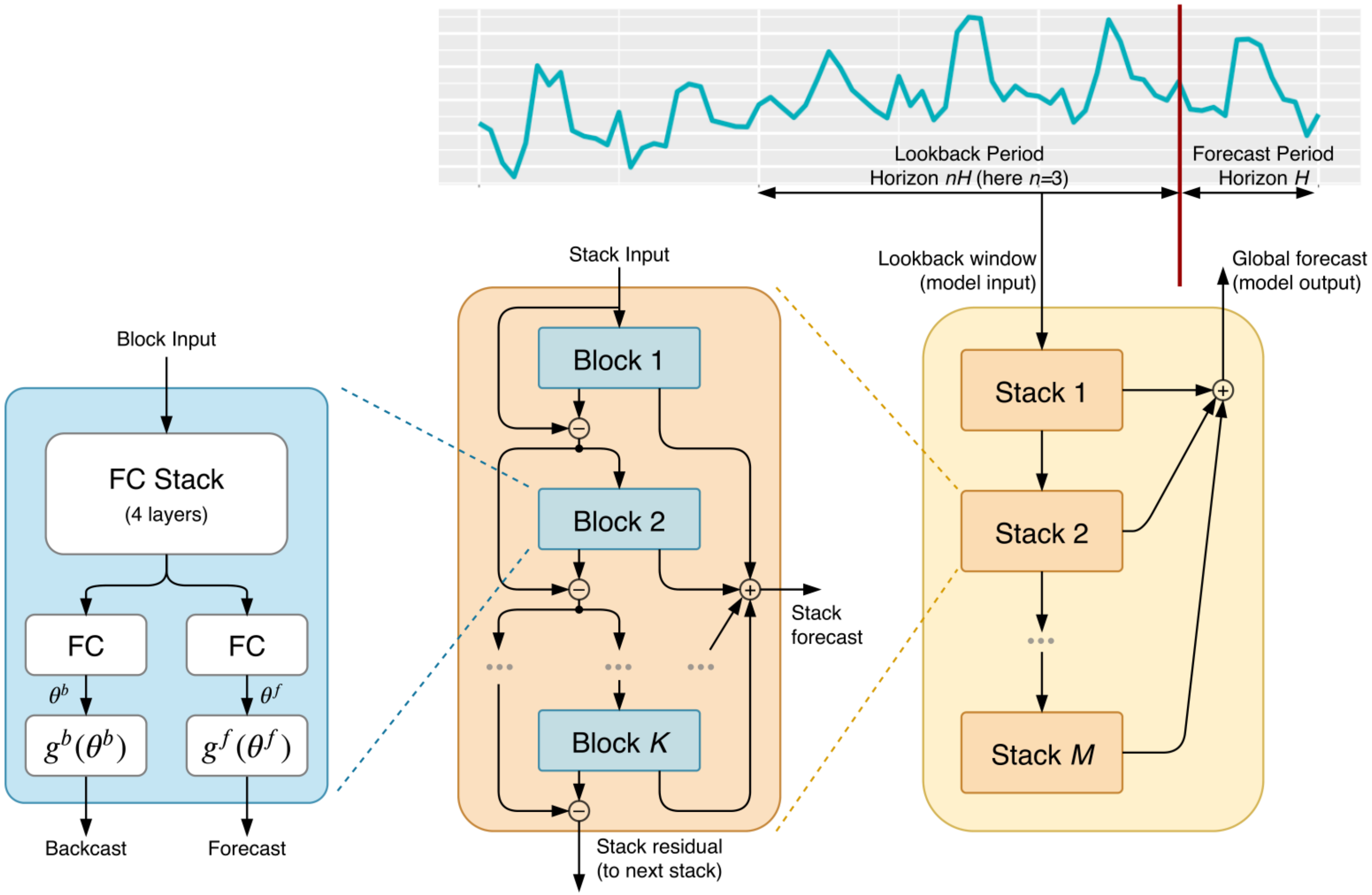
(b). LogSparse Self Attention

(c). Local Attention + LogSparse Self Attention

(d). Restart Attention + LogSparse Self Attention

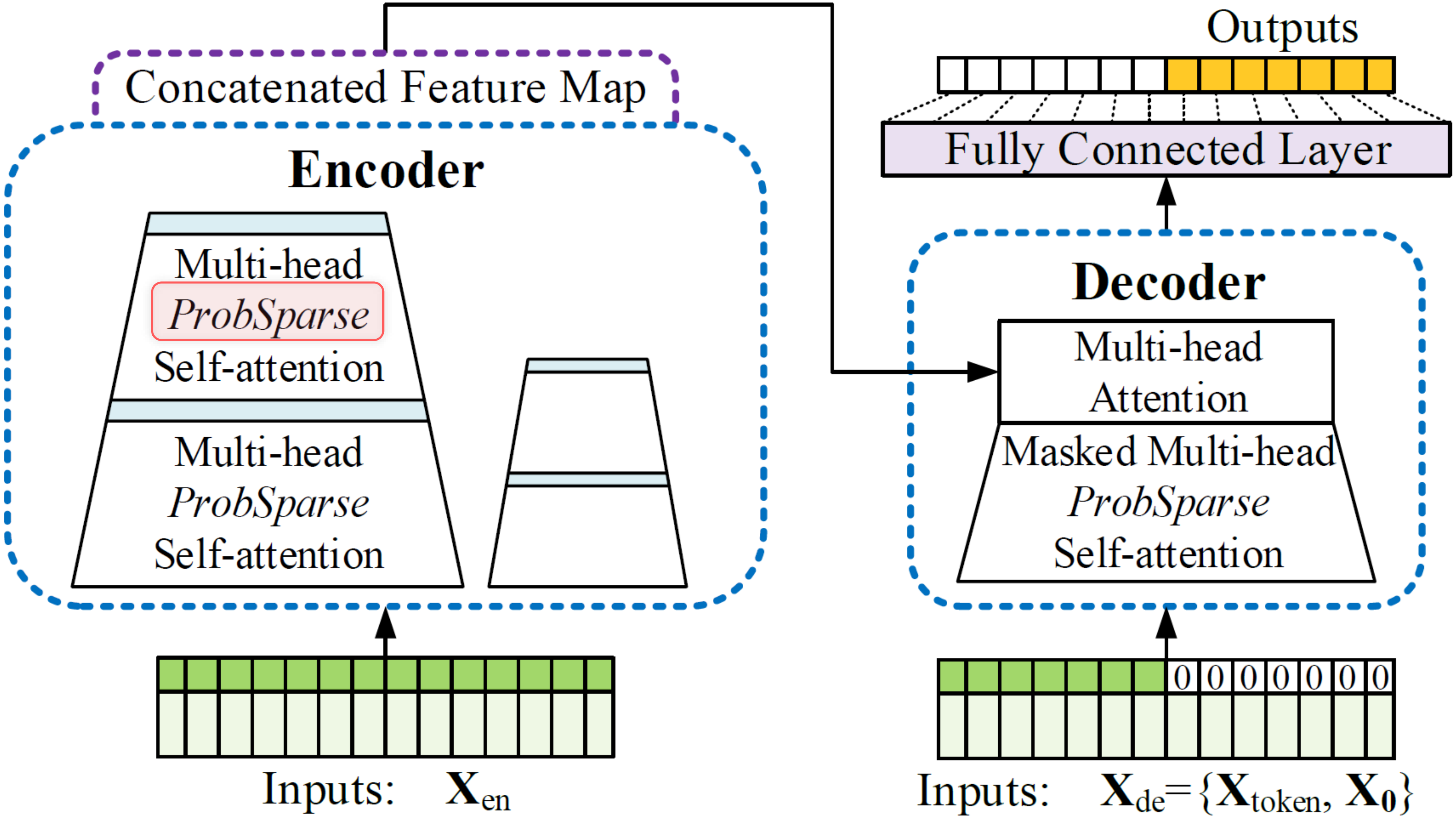
Shiyang Li et al. "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting". In *NeurIPS*, 2019 CCF-A

N-BEATS



Boris N. Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In *ICLR*, 2020 CCF-A

Informer



Haoyi Zhou et al. "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting". In *AAAI*, 2021 CCF-A

Self-attention Distilling operation

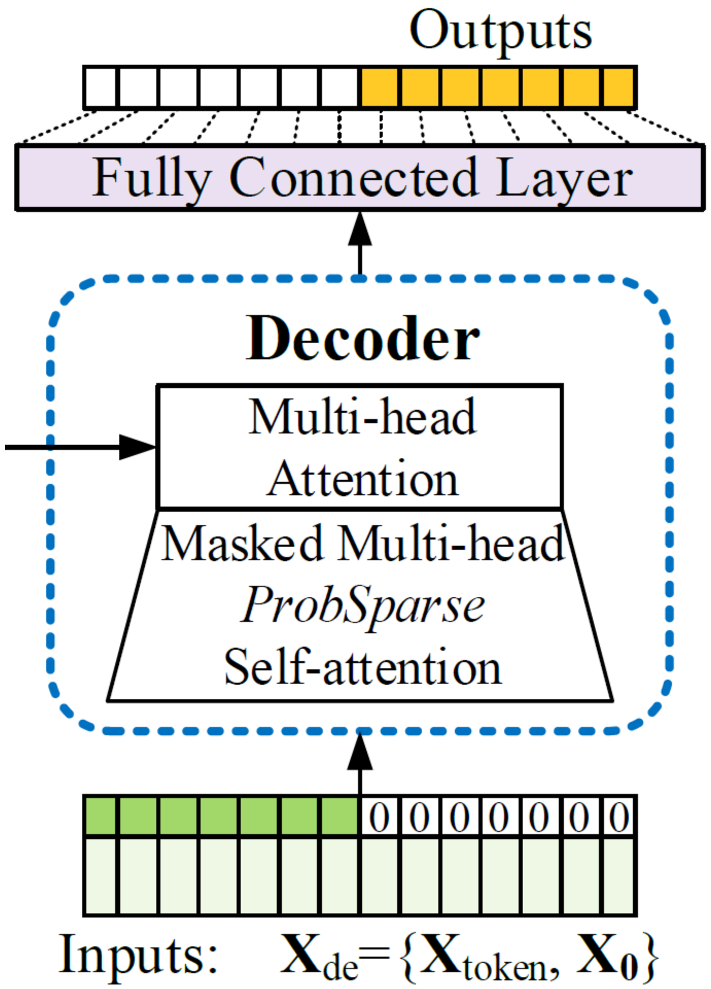
$$\mathbf{X}_{j+1}^{t} = \mathrm{MaxPool}\left(\mathrm{ELU}\left(\mathrm{Conv1d}([\mathbf{X}_{j}^{t}]_{\mathrm{AB}}) \right) \right)$$

Start token is efficiently applied in NLP's "dynamic decoding"

$$\mathbf{X}_{\mathrm{de}} = \{\mathbf{X}_{5d}, \mathbf{X}_{\mathbf{0}}\}$$

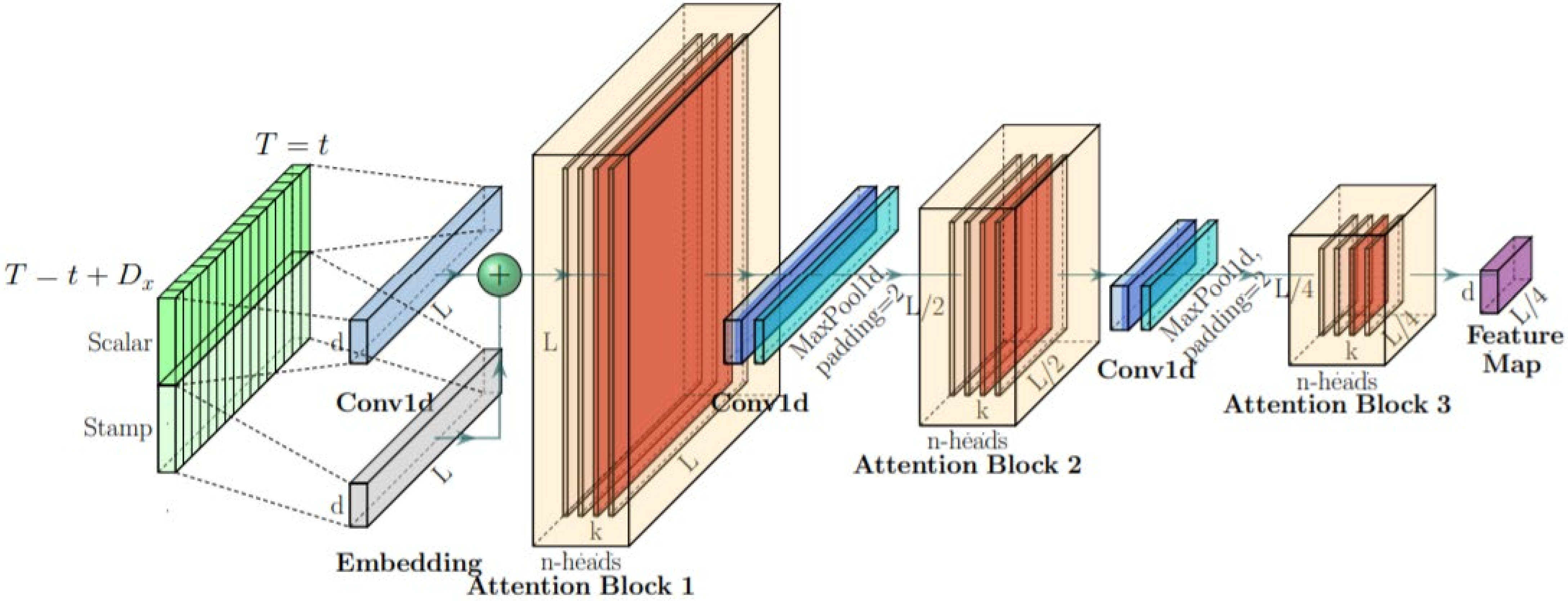
$\mathbf{X}_{5d}$ start token

$\mathbf{X}_{\mathbf{0}}$ contains target sequence's time stamp, i.e., the context at the target week

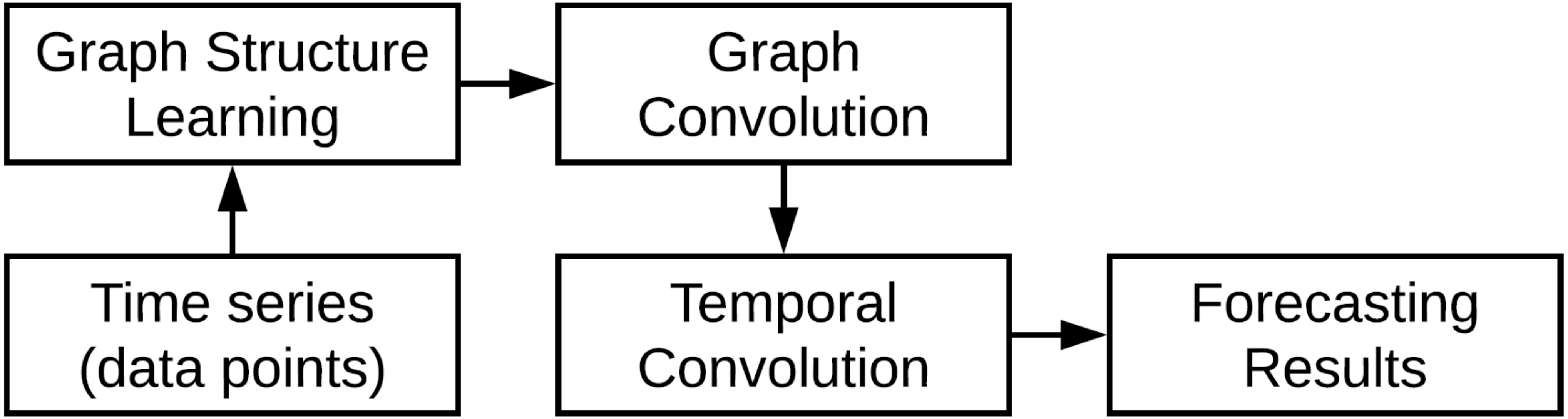


Generative-style decoder

$$\mathbf{X}_{j+1}^{t} = \mathrm{MaxPool}\left(\mathrm{ELU}(\mathrm{Conv1d}([\mathbf{X}_{j}^{t}]_{\mathrm{AB}})) \right)$$



MTGNN



Graph learning module spatial

extracts a sparse graph adjacency matrix adaptively based on data.

$$X = \{S_{t_1}, S_{t_2}, \ldots, S_{t_P}\}$$

Graph convolution module spatial

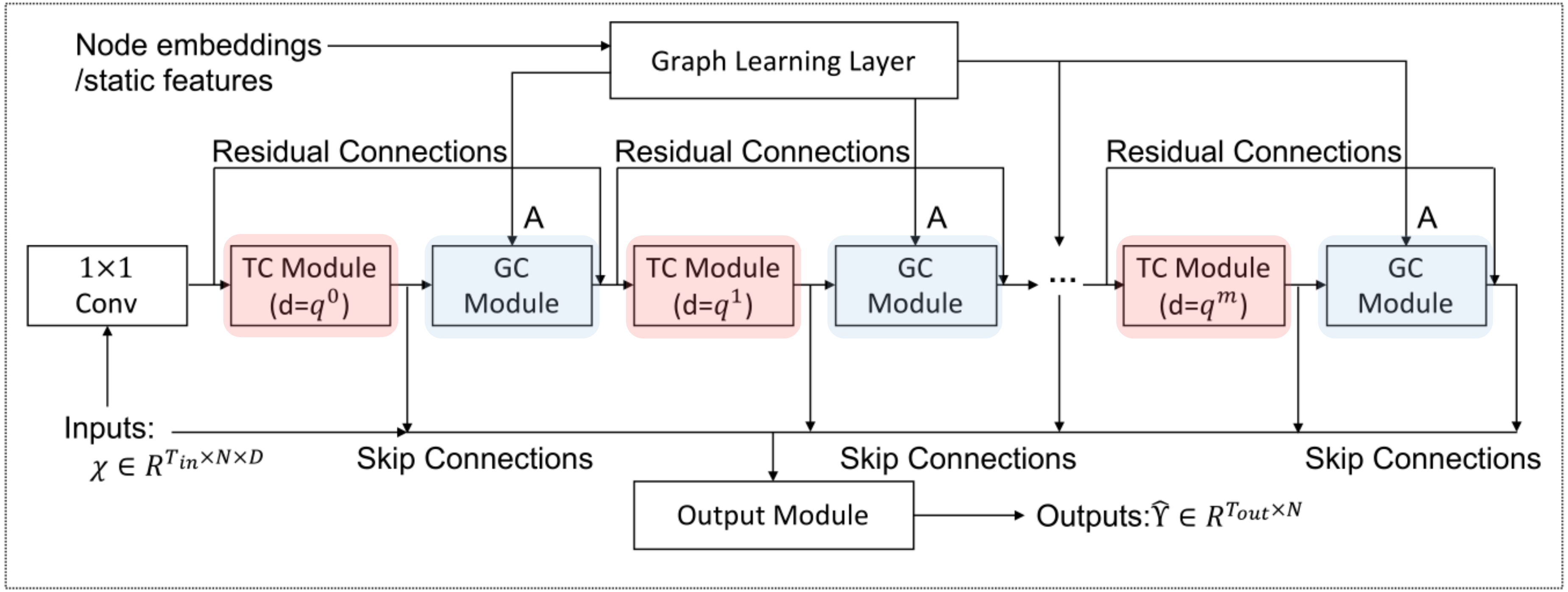
address the spatial dependencies among variables

Temporal convolution module temporal

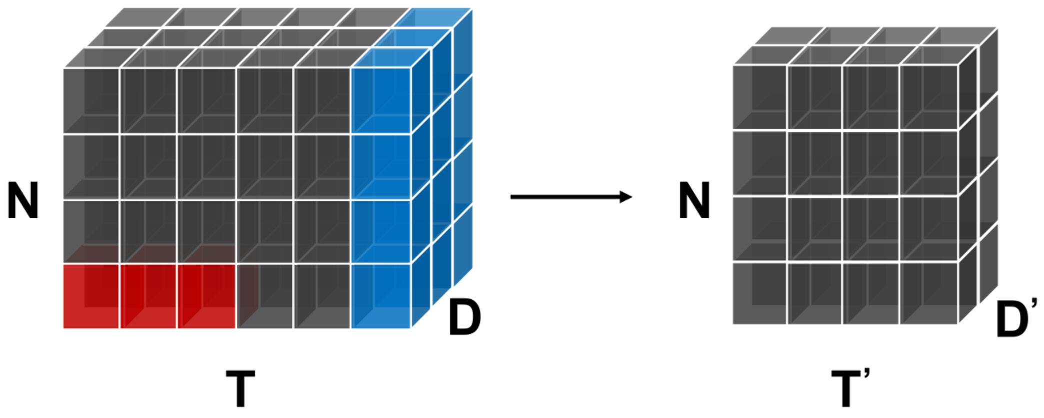
capture temporal patterns by modified 1D convolutions

Zonghan Wu et al. “Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks”. In *KDD*, 2020 CCF-A

MTGNN

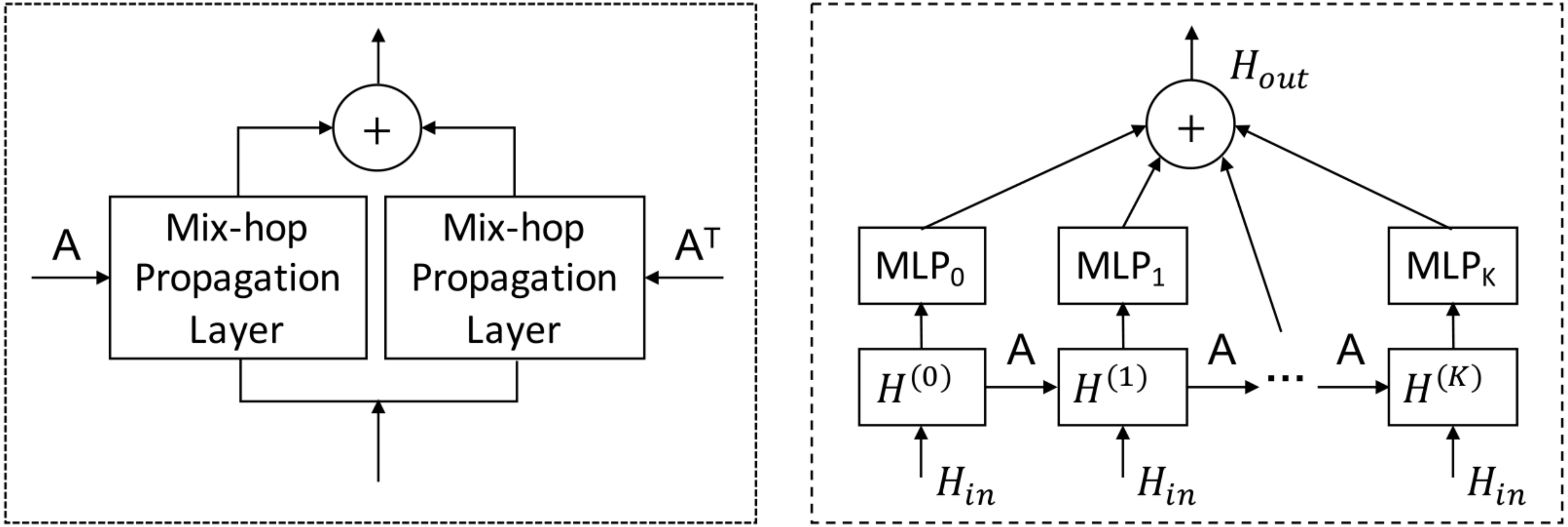


Node embeddings
/static features
Graph Learning Layer
Residual Connections
Residual Connections
Residual Connections
A
A
A
1×1
Conv
TC Module
(d=q^0)
GC
Module
TC Module
(d=q^1)
GC
Module
…
TC Module
(d=q^m)
GC
Module
Inputs:
$\chi \in R^{T_{in} \times N \times D}$
Skip Connections
Skip Connections
Skip Connections
Output Module
Outputs:$\hat{Y} \in R^{T_{out} \times N}$



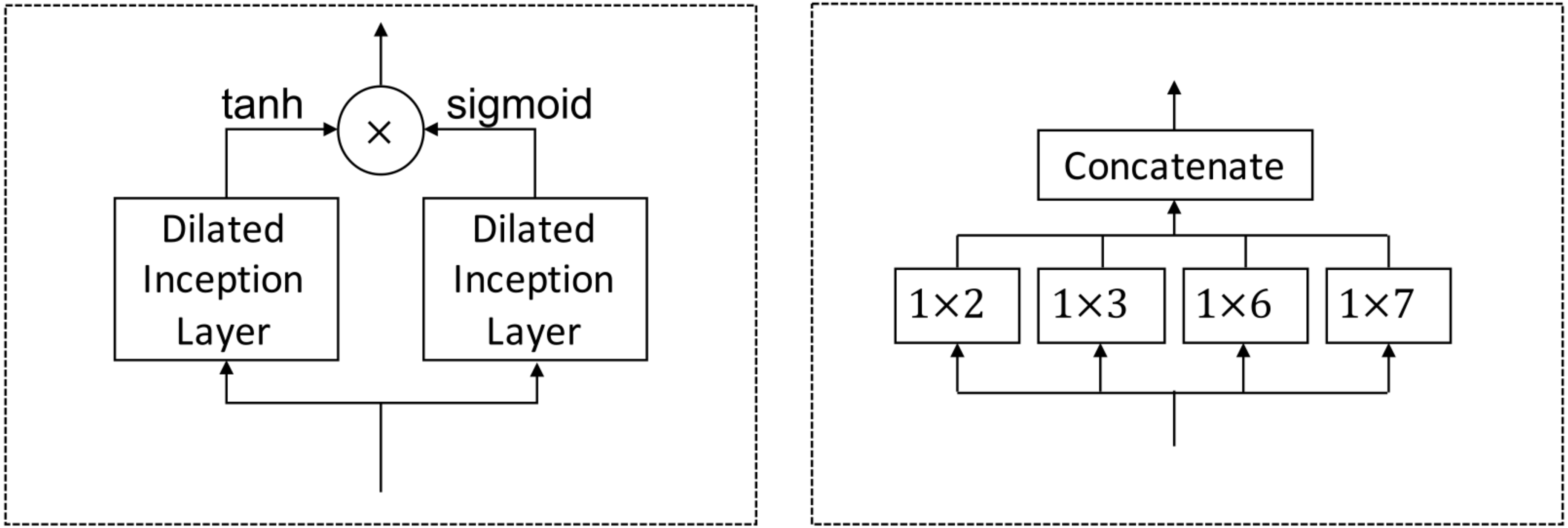
N
D
T
N
D'
T'

GC module



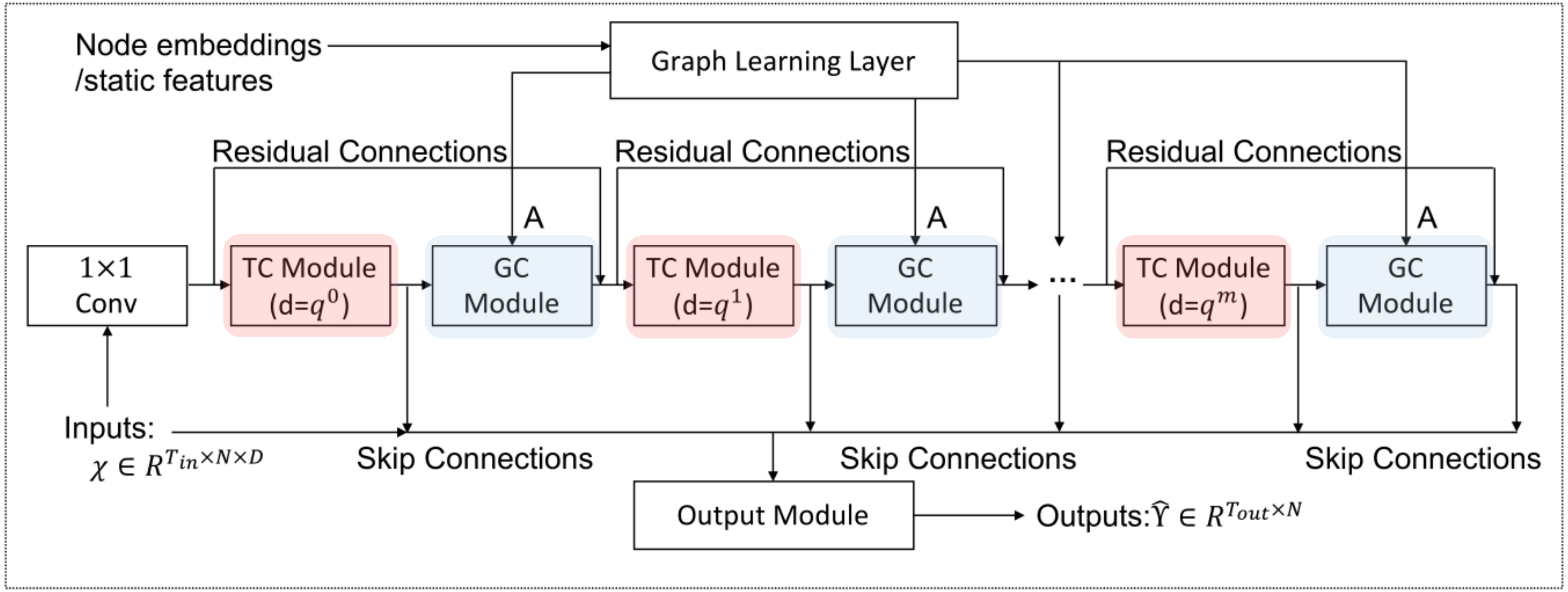
(a) GC module **(b) Mix-hop propagation layer**

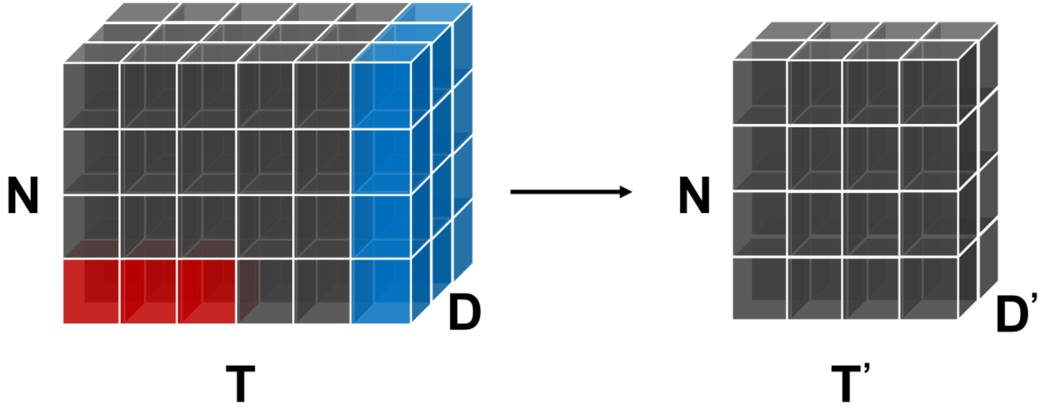
TC module



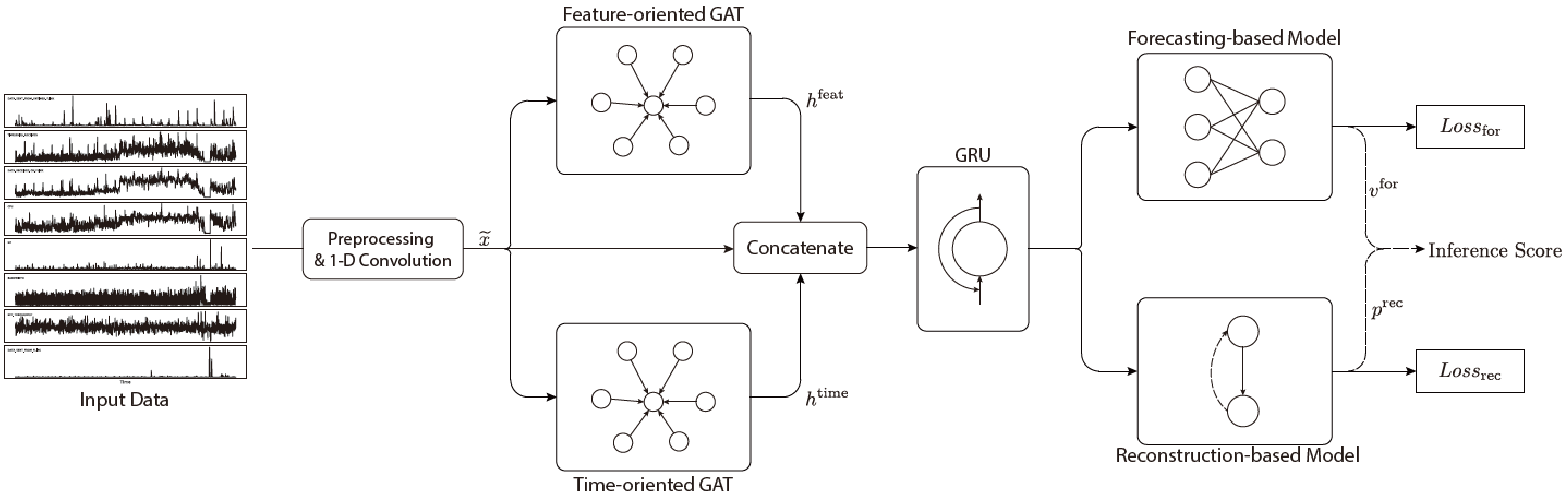
(a) TC module **(b) Dilated inception layer**

MTGNN


Node embeddings
/static features
Graph Learning Layer
Residual Connections
Residual Connections
Residual Connections
A
A
A
1×1
Conv
TC Module
(d=q^0)
GC
Module
TC Module
(d=q^1)
GC
Module
...
TC Module
(d=q^m)
GC
Module
Inputs:
$\chi \in R^{T_{in} \times N \times D}$
Skip Connections
Skip Connections
Skip Connections
Output Module
Outputs: $\hat{Y} \in R^{T_{out} \times N}$

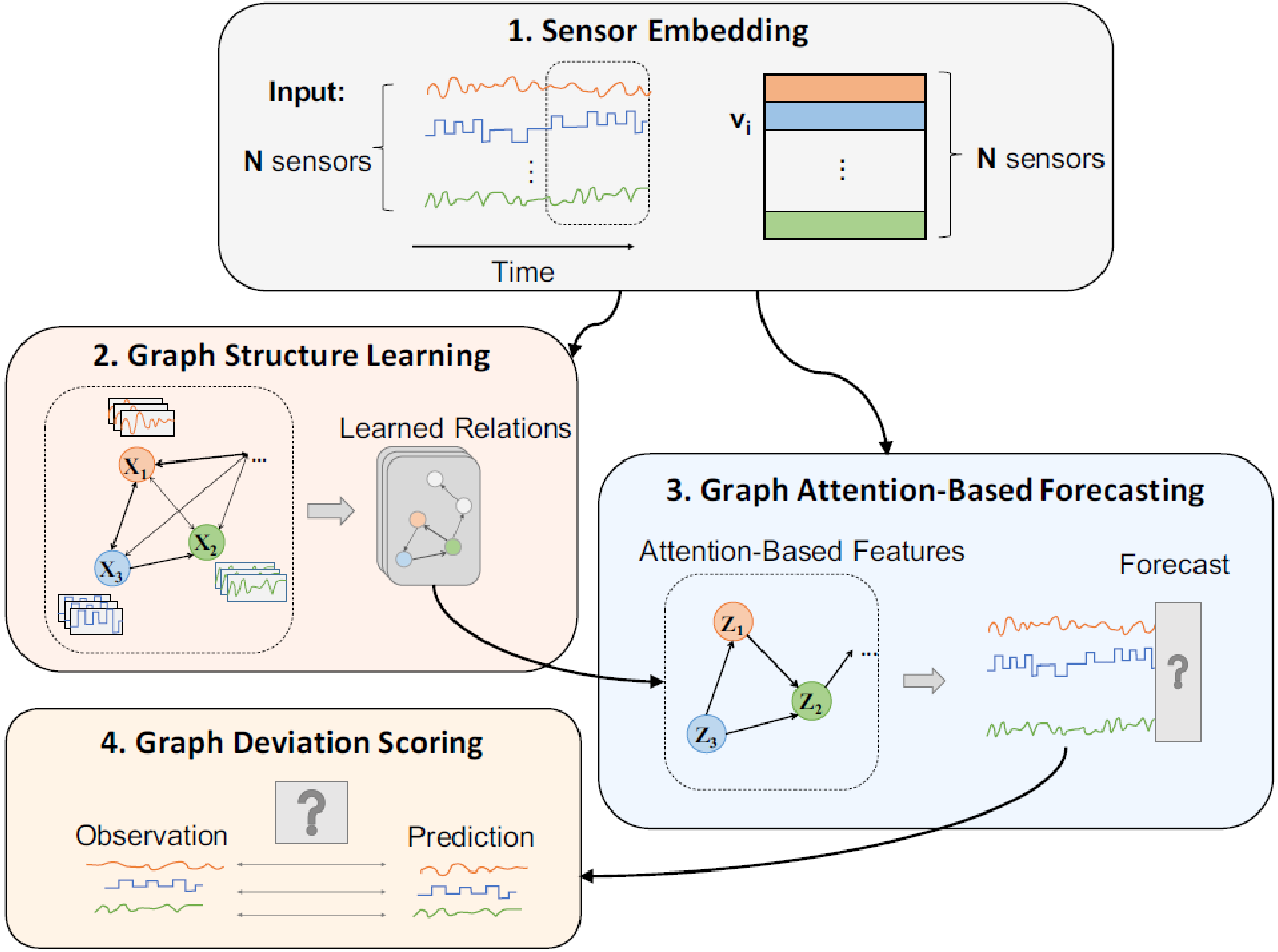

N
D
T
N
D'
T'

MTAD-GAT



Input Data
Preprocessing & 1-D Convolution
Feature-oriented GAT
Time-oriented GAT
h^{feat}
h^{time}
Concatenate
GRU
Forecasting-based Model
Reconstruction-based Model
$Loss_{for}$
$Loss_{rec}$
v^{for}
p^{rec}
Inference Score

GDN

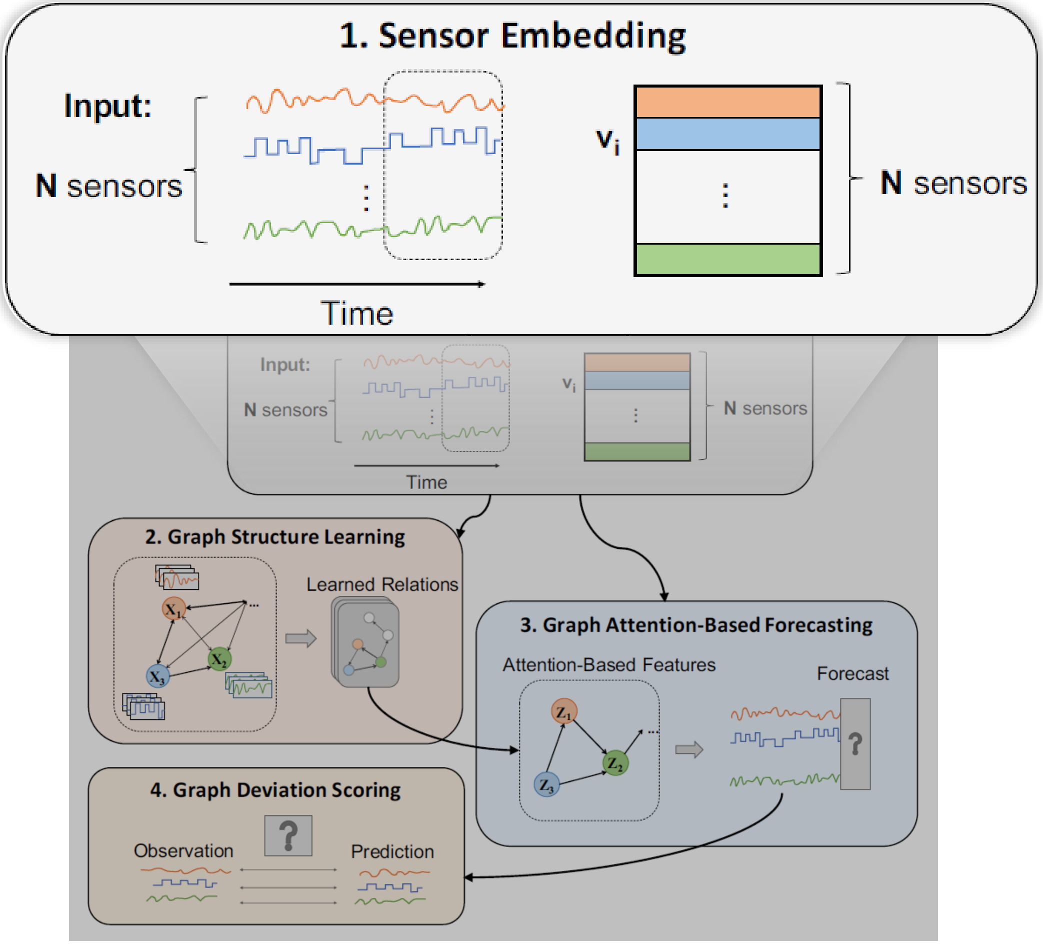

1. Sensor Embedding
Input:
N sensors
Time
v_i
N sensors
2. Graph Structure Learning
Learned Relations
X_1
X_2
X_3
3. Graph Attention-Based Forecasting
Attention-Based Features
Forecast
Z_1
Z_2
Z_3
4. Graph Deviation Scoring
Observation
Prediction

GDN

1. Sensor Embedding

Capture the unique characteristics of each sensor

$$\mathbf{v_i} \in \mathbb{R}^d,\ for\, i \in \{1,2,\cdots,N\}$$



USE

1) For structure learning

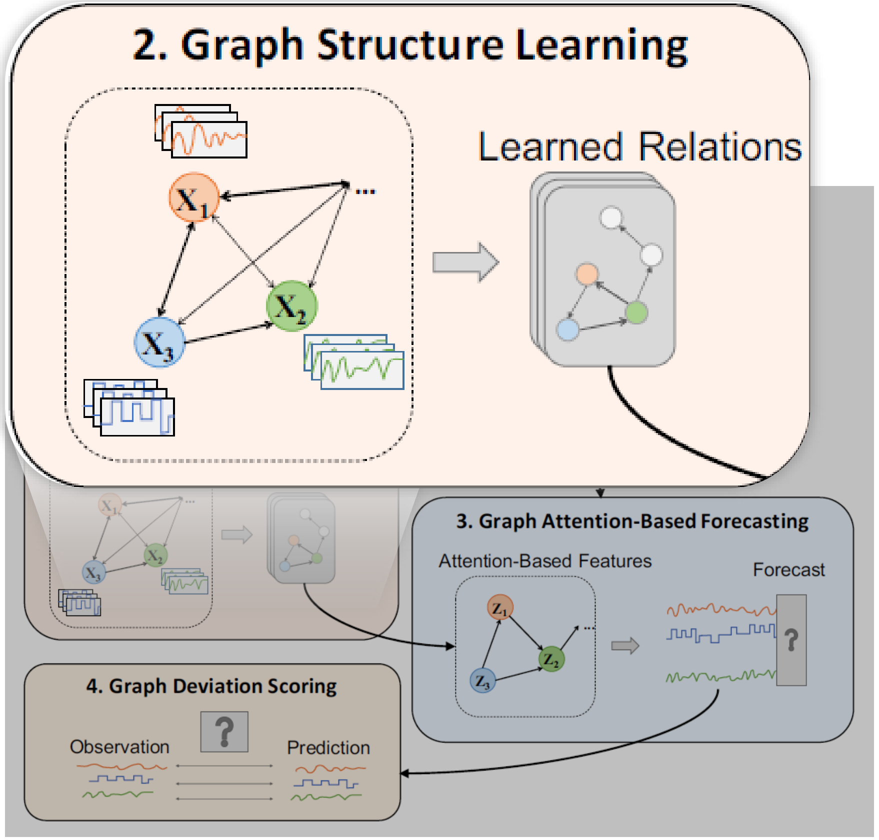
To determine which sensors are related to one another

2) In attention mechanism

To perform attention over neighbors in a way that allows heterogeneous effects for different types of sensors

2. Graph Structure Learning

learns a graph structure representing dependence relationships between sensors



Directed graph	
edges	sensors
nodes	dependency relationships

For each sensor i

1) candidate relations

$$\mathcal{C}_i \subseteq \{1,2,\cdots,N\} \setminus \{i\}$$

2) compute the similarity

$$e_{ji} = \frac{\mathbf{v_i}^\top \mathbf{v_j}}{\| \mathbf{v_i} \| \cdot \| \mathbf{v_j} \|} \quad for\, j \in \mathcal{C}_i$$

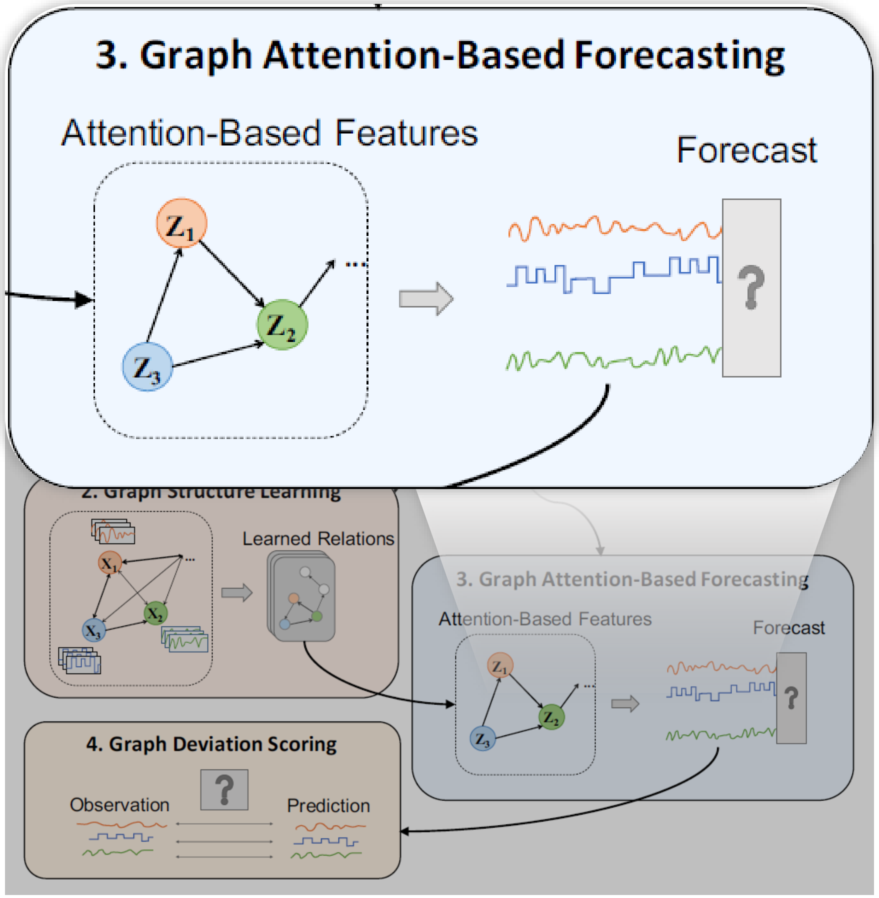
3) select edges

$$A_{ji} = 1\{j \in \mathrm{Top\,K}(\{e_{ki}: k \in \mathcal{C}_i\})\}$$

Get Adjacency matrix A

3. Graph Attention-Based Forecasting

forecasts future values of each sensor
based on a graph attention function over its neighbors



1) Input | 2) Feature Extractor | 3) Output

a sliding window of size w
over the historical time series data

$$x^{(t)} := \left[s^{(t-w)}, s^{(t-w+1)}, \cdots, s^{(t-1)}\right]$$

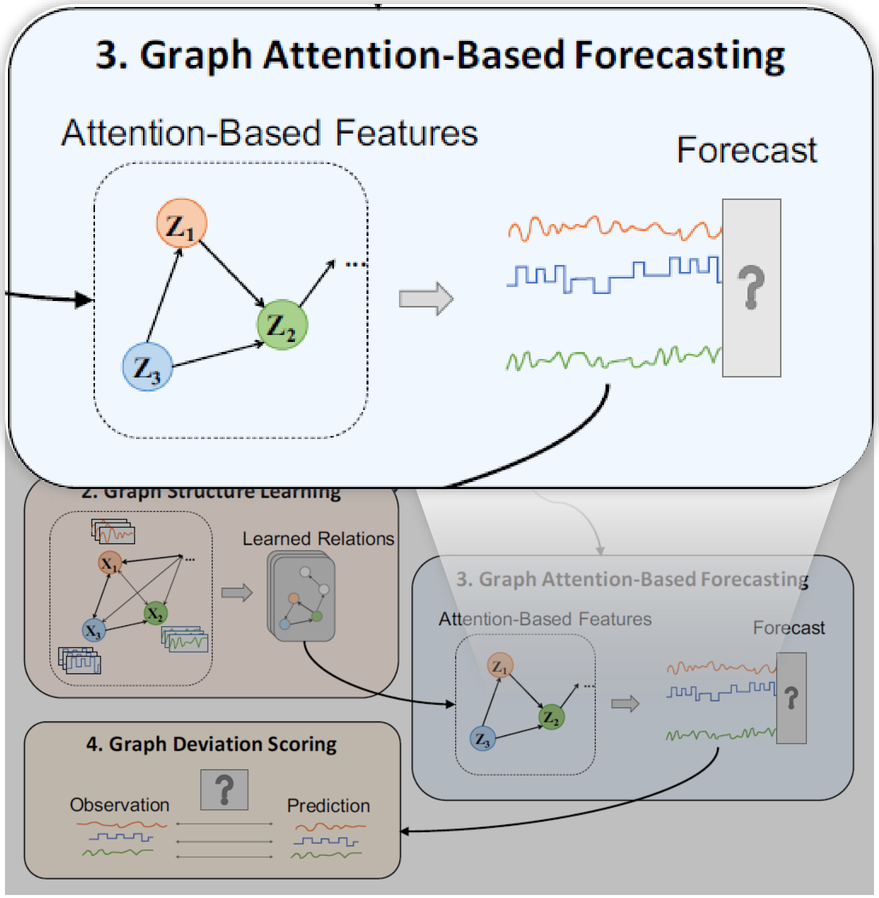
target output

sensor data at the current time tick， i.e. $s^{(t-w)}$

3. Graph Attention-Based Forecasting

forecasts future values of each sensor

based on a graph attention function over its neighbors



1) Input | 2) Feature Extractor | 3) Output

Aggregated representation $\mathbf{z}_i$

$$\mathbf{z}_i^{(t)} = \mathrm{ReLU}\left(\alpha_{i,i}\mathbf{W}\mathbf{x}_i^{(t)} + \sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\mathbf{W}\mathbf{x}_j^{(t)}\right)$$

compute attention coefficients

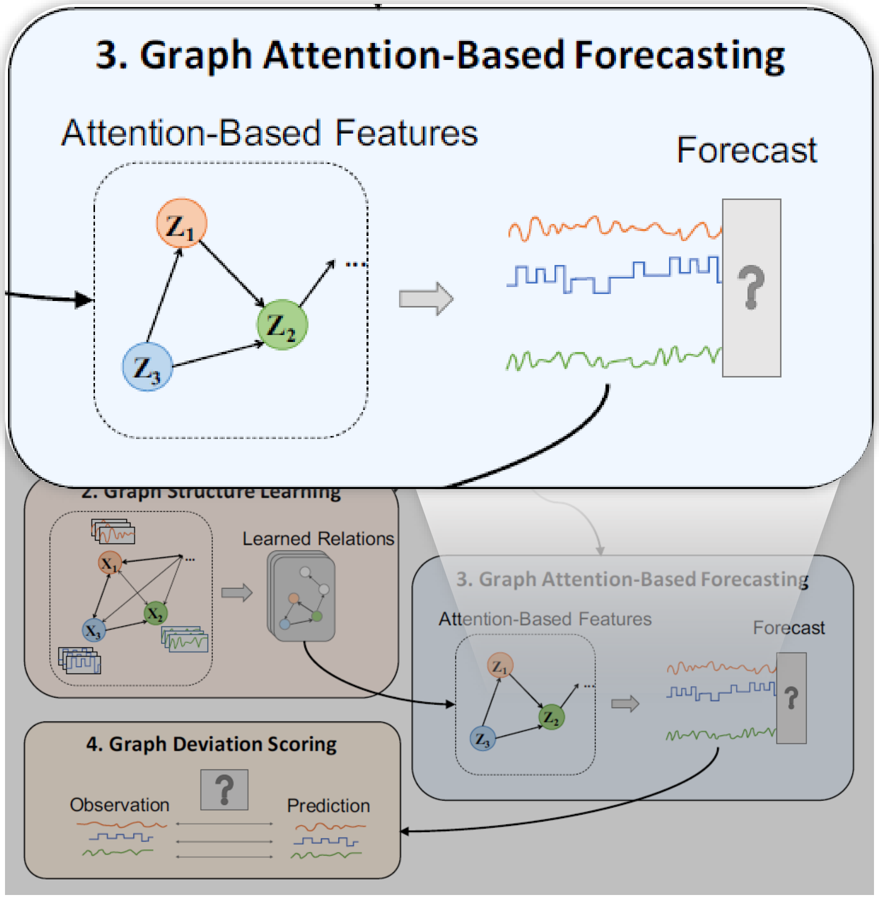
$$\mathbf{g}_i^{(t)} = \mathbf{v}_i \oplus W\mathbf{x}_i^{(t)}$$

$$\pi(i,j) = LeakyReLU\left(\mathbf{a}^\top\left(\mathbf{g}_i^{(t)} \oplus \mathbf{g}_j^{(t)}\right)\right)$$

$$\alpha_{i,j} = \frac{\exp(\pi(i,j))}{\sum_{k\in\mathcal{N}(i)\cup\{i\}} \exp(\pi(i,k))}$$

3. Graph Attention-Based Forecasting

forecasts future values of each sensor
based on a graph attention function over its neighbors



1) Input | 2) Feature Extractor | 3) Output

Representations of N nodes $\left\{\mathbf{z}_1^{(t)}, \cdots, \mathbf{z}_N^{(t)}\right\}$

Target output $\hat{\mathbf{s}}^{(\mathbf{t})} = f_\theta\left(\left[\mathbf{v}_1 \circ \mathbf{z}_1^{(t)}, \cdots, \mathbf{v}_N \circ \mathbf{z}_N^{(t)}\right]\right)$

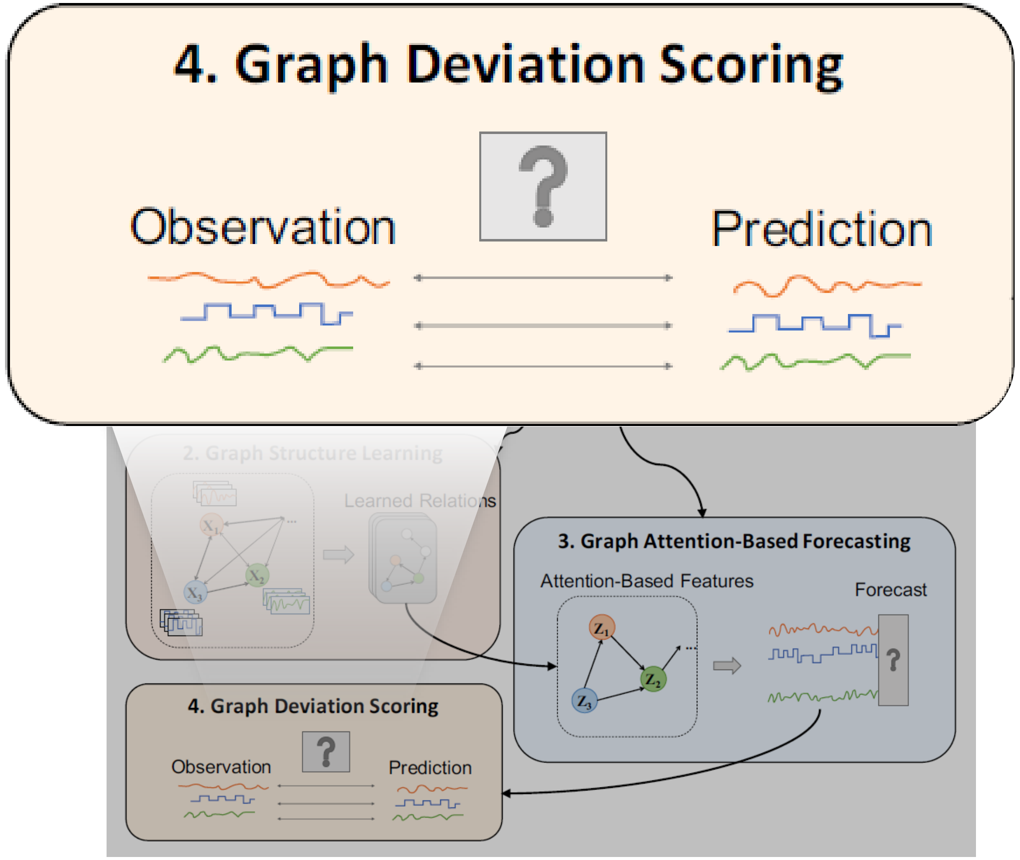
output dimensionality N

Mean Squared Error

$$L_{\mathrm{MSE}} = \frac{1}{T_{train} - w} \sum_{t=w+1}^{T_{train}} \| \hat{\mathbf{s}}^{(\mathbf{t})} - \mathbf{s}^{(\mathbf{t})} \|_2^2$$

4. Graph Deviation Scoring

identifies deviations from the learned relationships, and localizes and explains these deviations.



Anomaly Detection

1) Computing an error value

$$\mathrm{Err}_i(t) = \left|\mathbf{s}_\mathbf{i}^{(\mathbf{t})} - \hat{\mathbf{s}}_\mathbf{i}^{(\mathbf{t})}\right|$$

2) Perform a robust normalization

$$a_i(t) = \frac{\mathrm{Err}_i(t) - \tilde{\mu}_i}{\tilde{\sigma}_i}$$

3) Aggregate over sensors

$$\mathrm{A}(t) = \max_i a_i(t)$$

4) Generate the smoothed scores

$$A_s(t) = \mathrm{SMA}(\mathrm{A}(t))$$

5) labelled as an anomaly

If $A_s(t) > \mathrm{THRESHOLD}$

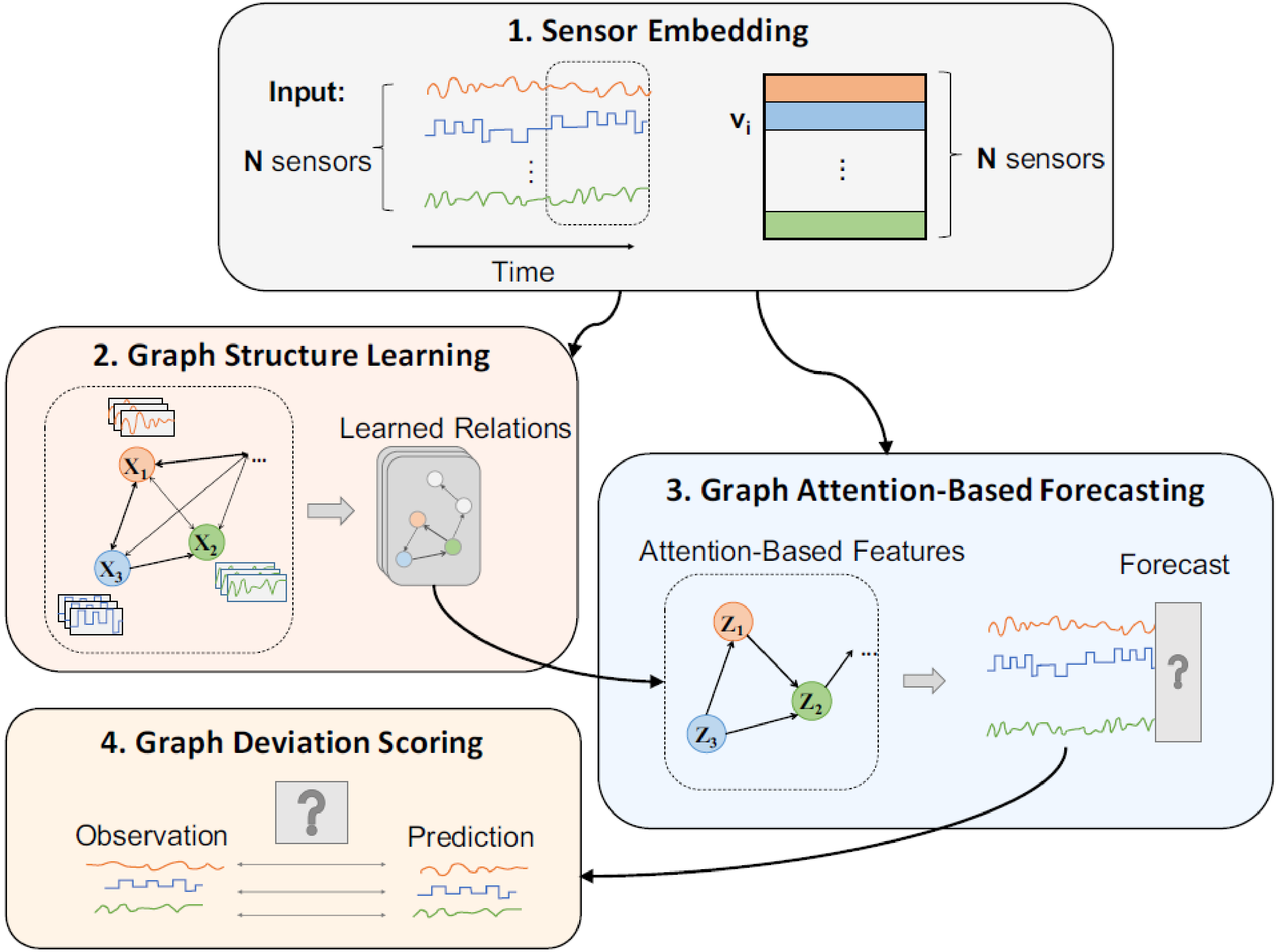
t time
i sensor

$\tilde{\mu}_i$ median
$\tilde{\sigma}_i$ IQR2

max function

Ailin Deng et al. "Graph Neural Network-Based Anomaly Detection in Multivariate Time Series". In *AAAI*, 2021 CCF-A

GDN


1. Sensor Embedding
Input:
N sensors
Time
v_i
N sensors
2. Graph Structure Learning
Learned Relations
3. Graph Attention-Based Forecasting
Attention-Based Features
Forecast
4. Graph Deviation Scoring
Observation
Prediction

Summary

DL for TS

- CNN: CONV | Pooling
- RNN: LSTM | GRU
- Transformer: Seq2Seq | Attention
- GNN: GCN | GAT
- Other...: GAN | DRL

Thanks

A Survey on Deep Learning Advances for Time Series Forecasting

@GeminiLight