Memformer

A Memory Guided Transformer for Time Series Forecasting

Yunyao Cheng, Chenjuan Guo, Bin Yang, Haomin Yu, Kai Zhao, Christian S. Jensen

February 2025

Proceedings of the VLDB Endowment, Volume 18, Issue 2

Presented by **Andreas Gottschalk Krath**

1. Introduction

2. Methodology

Instance normalization

- Normalize within historical horizon only
- Mitigates the issue of internal covariate shift
- Allows model to effectively grasp the intricate temporal dynamics inherent in time series

$$H' = (H - \mu)/\sqrt{(\sigma^2 + c)}, \text{where}$$

H is the historical horizon

μ is the mean

σ is the variance

c ensures numerical stability

Instance normalization

- Normalize within historical horizon only
- Mitigates the issue of internal covariate shift
- Allows model to effectively grasp the intricate temporal dynamics inherent in time series

$H' = (H - \mu)/\sqrt{(\sigma^2 + c)}$, where

H is the historical horizon

μ is the mean

σ is the variance

c ensures numerical stability

poral dynamics inherent in time series. Instance normalization is defined as $\mathbf{H}' = (\mathbf{H} - \mu)/\sqrt{(\sigma^2 + \text{constant})}$, where $\mathbf{H}'$ denotes the preprocessed feature, μ and σ denote the mean and variance of the sample, respectively, and "constant" is a small positive real number included to ensure numerical stability.

2.2 Preprocessing

Instance normalization

- Normalize within historical horizon only
- Mitigates the issue of internal covariate shift
- Allows model to effectively grasp the intricate temporal dynamics inherent in time series

$$H' = (H - \mu)/\sqrt{(\sigma^2 + c)}, \text{ where}$$

H is the historical horizon

μ is the mean

σ is the variance

c ensures numerical stability

poral dynamics inherent in time series. Instance normalization is defined as $\mathbf{H}' = (\mathbf{H} - \mu)/\sqrt{(\sigma^2 + \text{constant})}$, where $\mathbf{H}'$ denotes the preprocessed feature, μ and σ denote the mean and variance of the sample, respectively, and "constant" is a small positive real number included to ensure numerical stability.

- Mistake in variance?
  - σ is conventional notation for standard deviation
  - σ^2 is conventional notation for variance

What is going on?

What is going on?

- Explored code to find answer
- `data_provider/data_loader.py`
  - Only place anything related to loading data happens
  - `Dataset_ETT_hour`, `Dataset_ETT_minute`, `Dataset_Custom`, `Dataset_Pred`

What is going on?

- Explored code to find answer
- `data_provider/data_loader.py`
  - Only place anything related to loading data happens
  - `Dataset_ETT_hour`, `Dataset_ETT_minute`, `Dataset_Custom`, `Dataset_Pred`

```
from sklearn.preprocessing import StandardScaler
class ...:
    def __read_data__(self):
        self.scalar = StandardScaler()
        if self.scale:
            self.scaler.fit(train_data.values)
            data = self.scaler.transform(df_data.values)
```

What is going on?

- Explored code to find answer
- `data_provider/data_loader.py`
  - Only place anything related to loading data happens
  - `Dataset_ETT_hour`, `Dataset_ETT_minute`, `Dataset_Custom`, `Dataset_Pred`

```
from sklearn.preprocessing import StandardScaler
class ...:
    def __read_data__(self):
        self.scalar = StandardScaler()
        if self.scale:
            self.scaler.fit(train_data.values)
            data = self.scaler.transform(df_data.values)
```

- They fit on training data
- Normalize entire dataset with μ and σ from training data

What are they actually doing?

Preprocessing

$H' = (H - \mu)/\sqrt{(\sigma^2 + c)}$, where

H is the historical horizon

μ is the mean

σ is the variance

c ensures numerical stability

StandardScaler

$z = (x - \mu)/\sigma$, where

x is the sample

μ is the mean

σ is the standard deviation

What are they actually doing?

Preprocessing

$H' = (H - \mu)/\sqrt{(\sigma^2 + c)}$, where

H is the historical horizon

μ is the mean

σ is the variance

c ensures numerical stability

StandardScaler

$z = (x - \mu)/\sigma$, where

x is the sample

μ is the mean

σ is the standard deviation

- We know that $\sqrt{\sigma^2} = \sigma$

What are they actually doing?

Preprocessing

$H' = (H - \mu)/\sqrt{(\sigma^2 + c)}$, where

H is the historical horizon

μ is the mean

σ is the variance

c ensures numerical stability

StandardScaler

$z = (x - \mu)/\sigma$, where

x is the sample

μ is the mean

σ is the standard deviation

- We know that $\sqrt{\sigma^2} = \sigma$
- Essentially same formula, except constant

What are they actually doing?

Preprocessing

$H' = (H - \mu)/\sqrt{(\sigma^2 + c)}$, where

H is the historical horizon

μ is the mean

σ is the variance

c ensures numerical stability

StandardScaler

$z = (x - \mu)/\sigma$, where

x is the sample

μ is the mean

σ is the standard deviation

- We know that $\sqrt{\sigma^2} = \sigma$
- Essentially same formula, except constant
- Fit on training data, normalize entire dataset $\longrightarrow$ global normalization

What are they actually doing?

Preprocessing

$H' = (H - \mu)/\sqrt{(\sigma^2 + c)}$, where

H is the historical horizon

μ is the mean

σ is the variance

c ensures numerical stability

StandardScaler

$z = (x - \mu)/\sigma$, where

x is the sample

μ is the mean

σ is the standard deviation

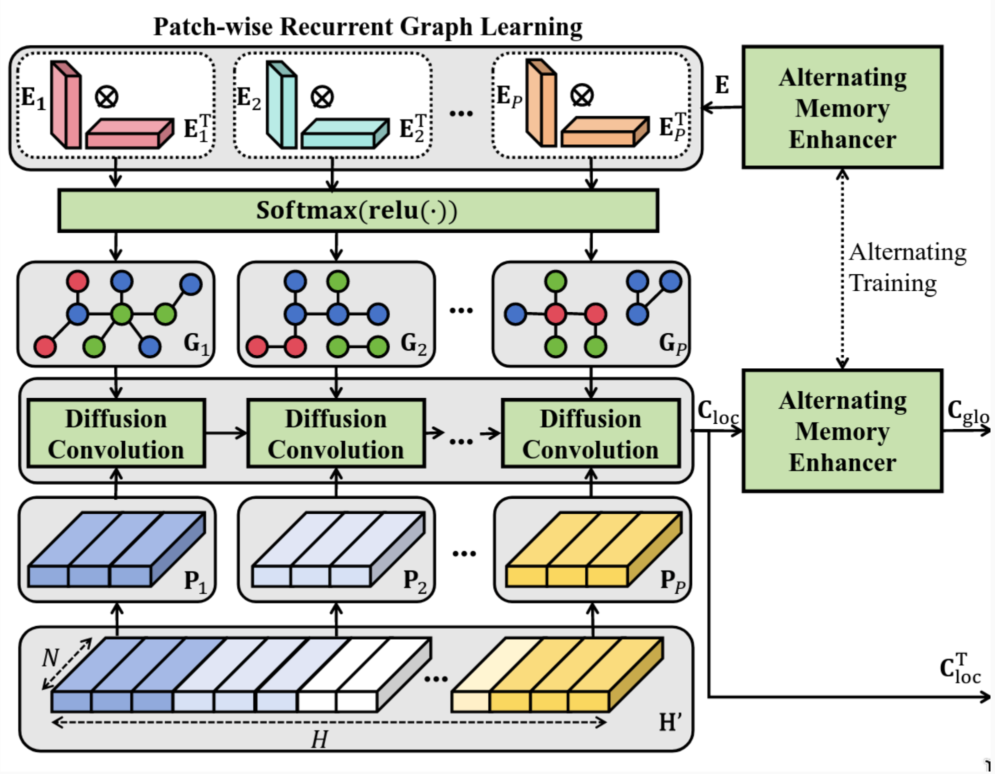
- We know that $\sqrt{\sigma^2} = \sigma$
- Essentially same formula, except constant
- Fit on training data, normalize entire dataset $\longrightarrow$ global normalization
- None of the stated benefits of instance normalization

Architecture

Upper part → dynamic correlation

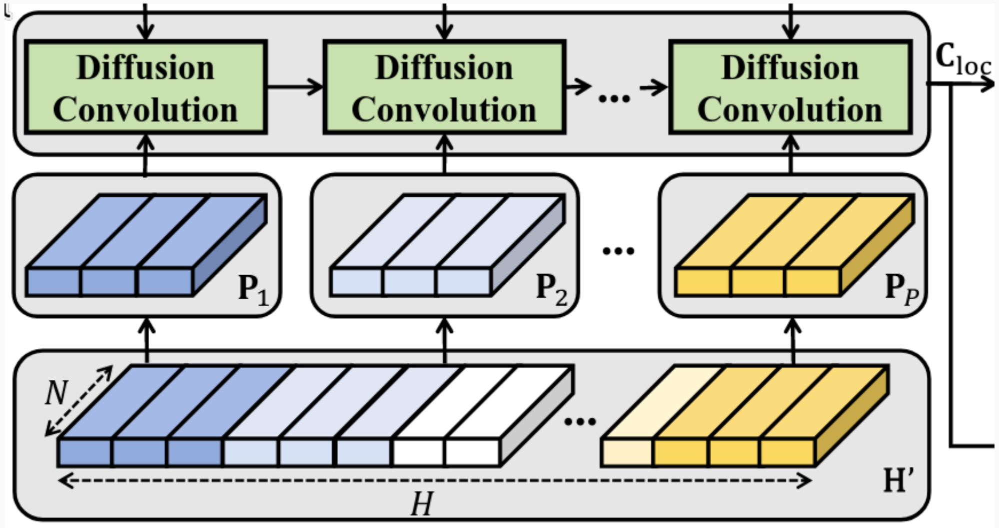
Lower part → normalized data

Output → enriched input features



Normalized Data

- Normalized as described earlier
  - Not what the paper actually states



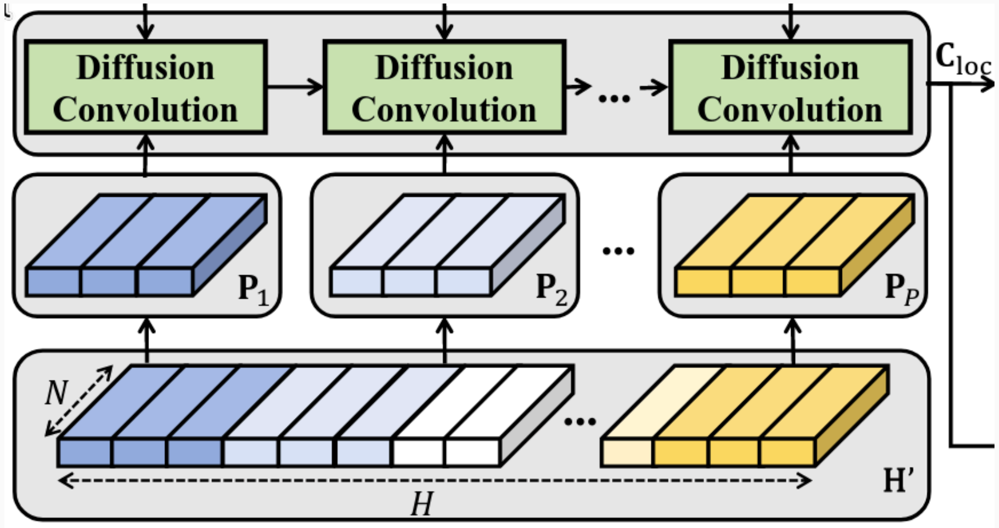
2.3 Patch-wise Recurrent Graph Learning

Normalized Data

- Normalized as described earlier
  - Not what the paper actually states

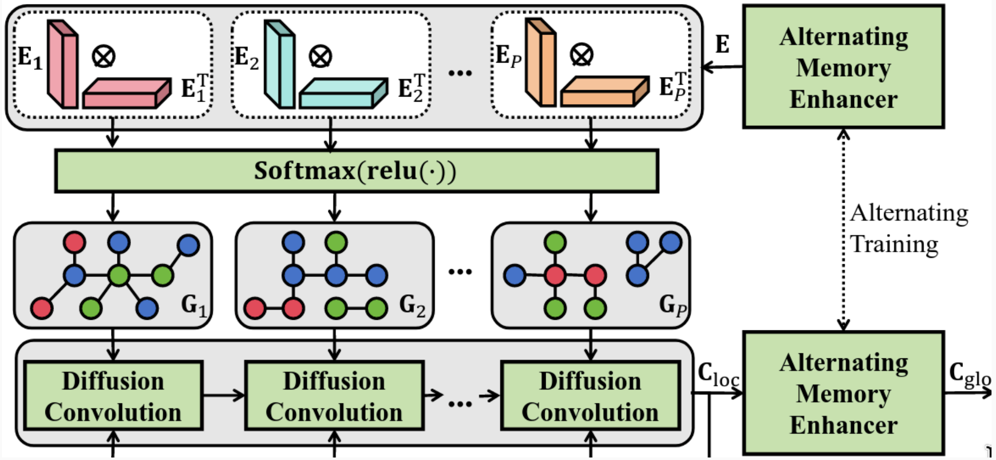
Patches

- H' is split into p patches
- Stride S
- Size T
- If $S \geq T$ patches are disjoint
- If $S < T$ patches overlap
  - Common elements for adjacent patches



AME

- Provides local memory embedding
  - These are learnable parameters
- Consistant local memory for patch P_i
- Matrix product of $E_i \otimes E_i^T$
  - Similarity matrix for variables in P_i

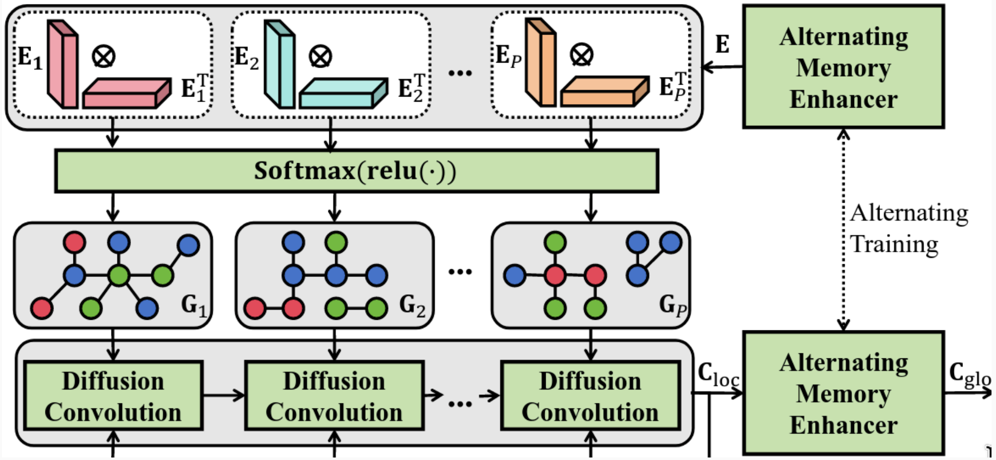


AME

- Provides local memory embedding
  - These are learnable parameters
- Consistant local memory for patch P_i
- Matrix product of $E_i \otimes E_i^T$
  - Similarity matrix for variables in P_i

ReLU + Softmax

- ReLU eliminates negative values
  - Removes negative correlations
- Softmax scales into influence scores



2.3 Patch-wise Recurrent Graph Learning

AME

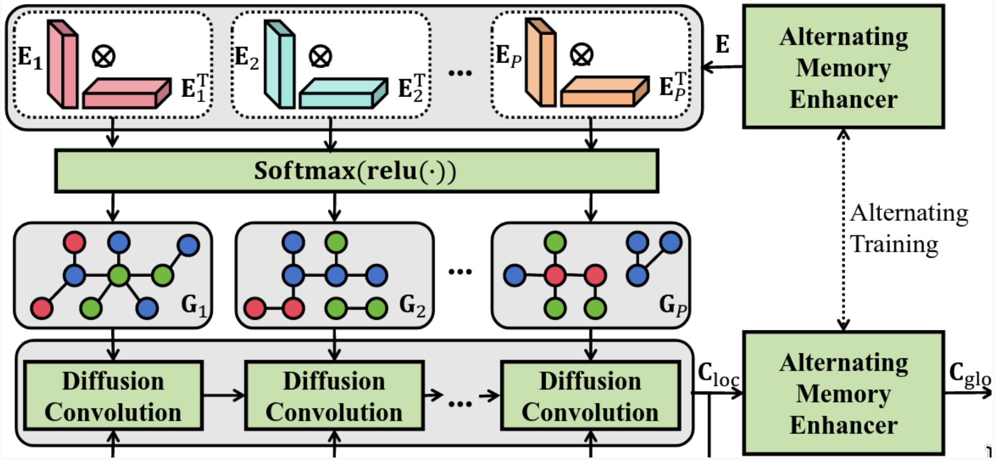
- Provides local memory embedding
  - These are learnable parameters
- Consistant local memory for patch P_i
- Matrix product of $E_i \otimes E_i^T$
  - Similarity matrix for variables in P_i

ReLU + Softmax

- ReLU eliminates negative values
  - Removes negative correlations
- Softmax scales into influence scores

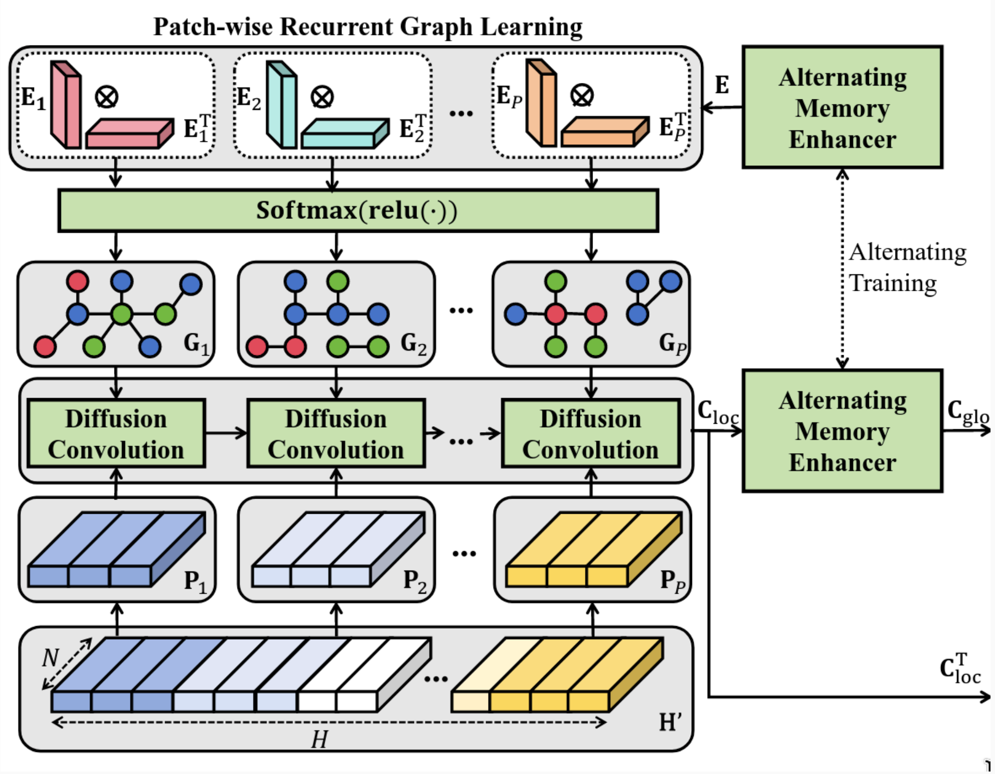
Graph

- Translates influence scores into graph
- Captures connection between variables
  - Dynamic correlations



Diffusion Convolution

- Normalized data is adjusted based on connections in graph
- Numeric values “diffuse” into neighbours
  - Not only immediate neighbours
- Spatially relates data based on connections



Gated Recurrent Unit

- Forwards information from P_i to P_{i+1}
- Temporally relates data in a sequence

Output

- Enriched input features
- Spatial $\longrightarrow$ dynamic correlations
- Temporal $\longrightarrow$ GRU

3. Results