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

1.1 Motivation

Forecasting

- Predicting the future
  - Allows preparation
- Many applications
  - Electricity prices
  - Finance
- Long term forecasting?
  - Obviously more difficult than short term
  - Time constrained tasks

Long Term Forecasting

- What defines long term?
  - Historical horizon
  - Forecasting horizon
  - Both exceed 96 time steps
    - Hourly time step $\longrightarrow$ 4 days
  - Time series

Variable Correlation

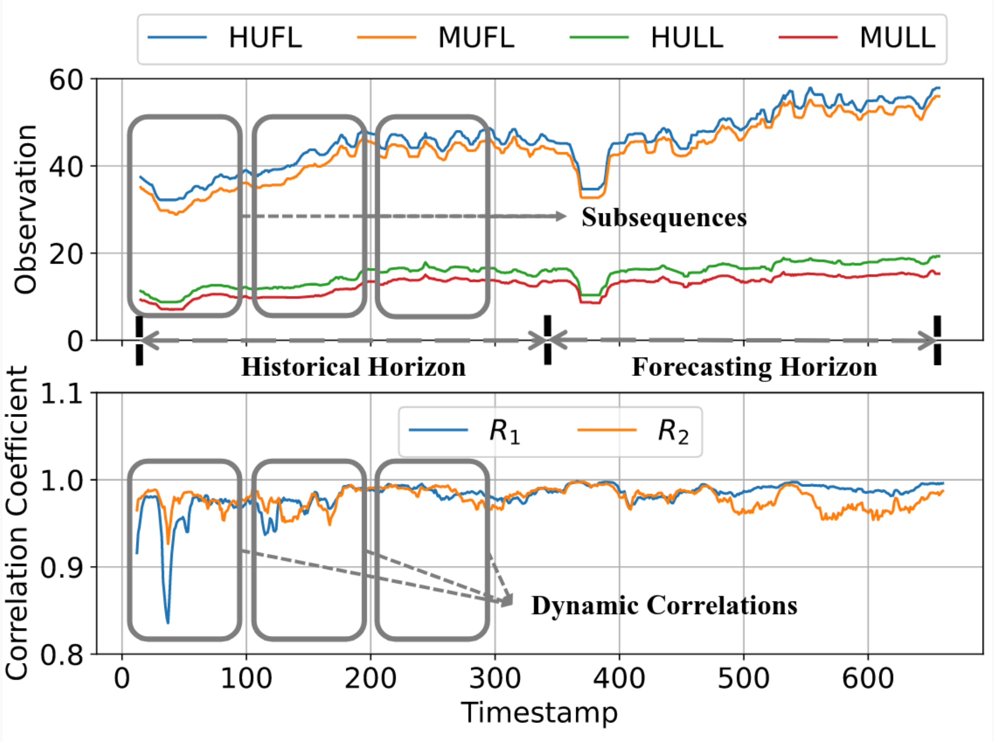
- Complex systems have many variables
- A increases and B increases $\longrightarrow$ Positive
- A increases and B decreases $\longrightarrow$ Negative
- A increases and B is stagnant $\longrightarrow$ None
- These impact forecasting accuracy
  - Patterns in the data

Dynamic Correlations

- Are variable correlations stable over time?
  - No
- Correlations are dynamic over time
  - Seasons
  - Sensor drift
- We often consider average
  - Especially hurtful in time series
  - Predictions are bad in periods

Dynamic Correlations

- Are variable correlations stable over time?
  - No
- Correlations are dynamic over time
  - Seasons
  - Sensor drift
- We often consider average
  - Especially hurtful in time series
  - Predictions are bad in periods



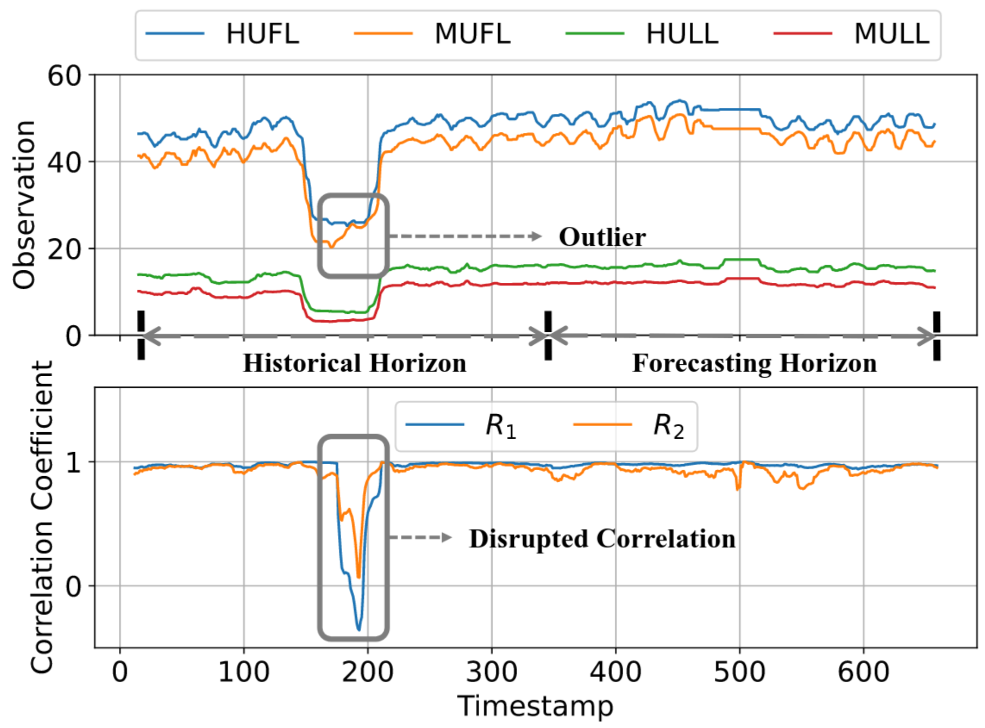
(a) Dynamic correlations. The Average $R_1 = 0.995$ and $R_2 = 0.990$.

Disrupted Correlations

- System errors
- External influence
- What happens with outliers?
  - Affect correlation $\longrightarrow$ accuracy
- Many models are sensitive to outliers
  - Numeric difference dominates training
  - Reason for a lot of preprocessing
    - Normalization
    - Clipping
    - Pruning

Disrupted Correlations

- System errors
- External influence
- What happens with outliers?
  - Affect correlation ⟶ accuracy
- Many models are sensitive to outliers
  - Numeric difference dominates training
  - Reason for a lot of preprocessing
    - Normalization
    - Clipping
    - Pruning



(b) Disrupted correlation. The Average $R_1 = 0.908$ and $R_2 = 0.963$.

1.2 Problem

Challenge 1

- Capture dynamic correlations
- Mitigate disrupted correlations
- Existing solutions struggle with the latter
  - Capture dynamic and disrupted
  - Reduces model robustness

Challenge 2

- Local information 🤝 global information
- Global information is *all* local information
- Local information *affects* global information
- Existing solutions struggle with combining
  - Only local
  - Only global

1.3 Contributions

Memformer

- Transformer
- Patch-wise recurrent graph learning
  - Captures dynamic correlations
- Global attention
  - Mitigates disrupted correlations
- Adresses challenge 1

Alternating Memory Enhancer

- Memory network
- Associates local and global information
- Adresses challenge 2

Experiments

- Proof

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

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

H is the historical horizon

μ is the mean

σ is the variance

c ensures 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

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

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

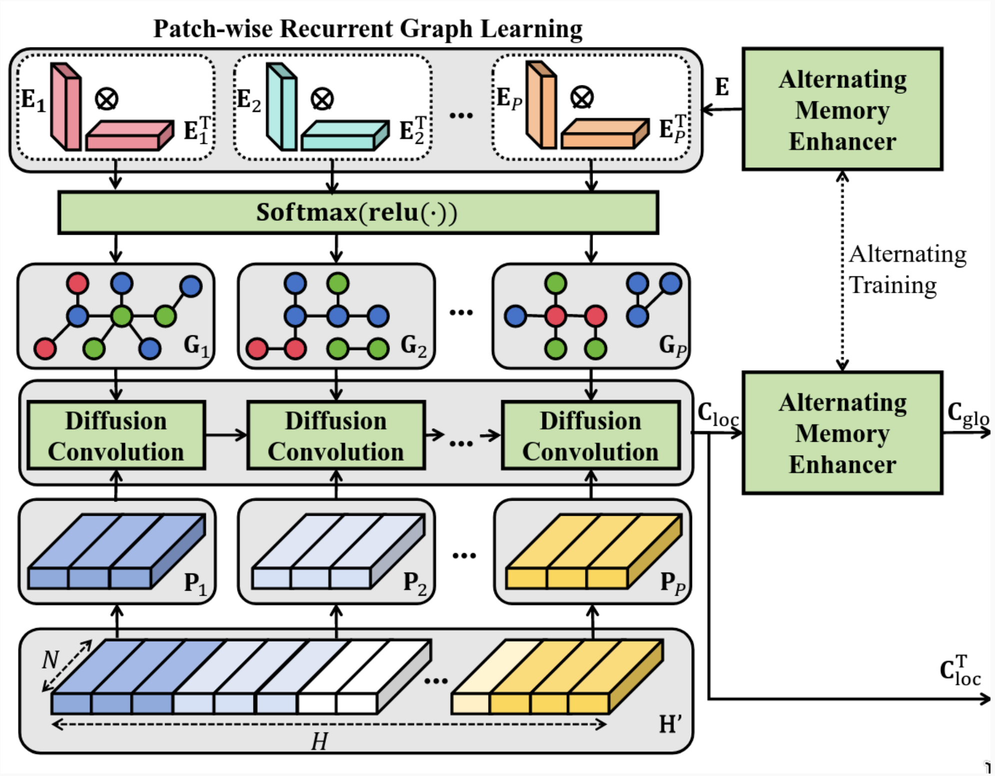
- We know that $\sqrt{\sigma^2} = \sigma$
- Essentially same formula, except constant
- Fit on training data, normalize entire dataset $\longrightarrow$ global normalization
- None of the stated benefits of instance normalization
  - Mitigate internal covariate shift
  - Grasp intricate temporal dynamics in TS

Architecture

Upper part → dynamic correlation

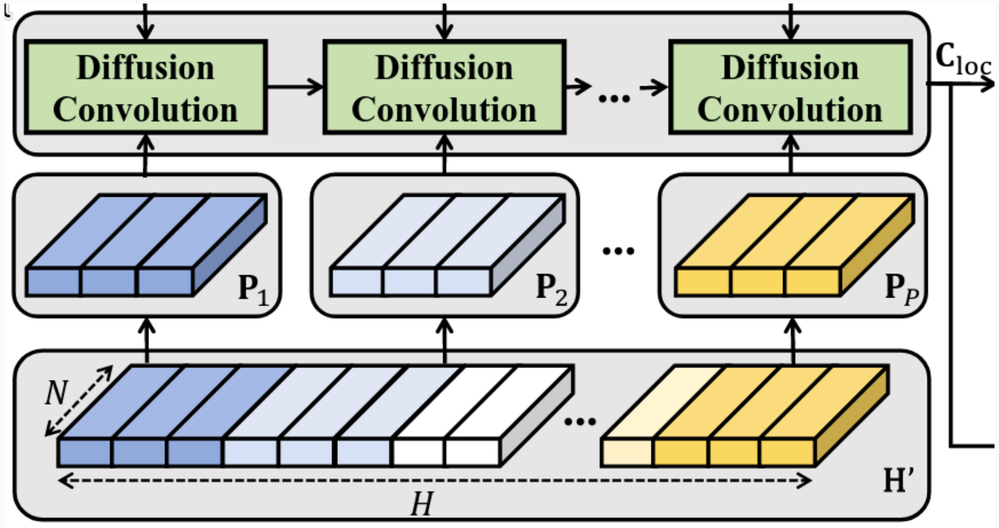
Lower part → normalized data

Output → enriched input features



Normalized Data

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

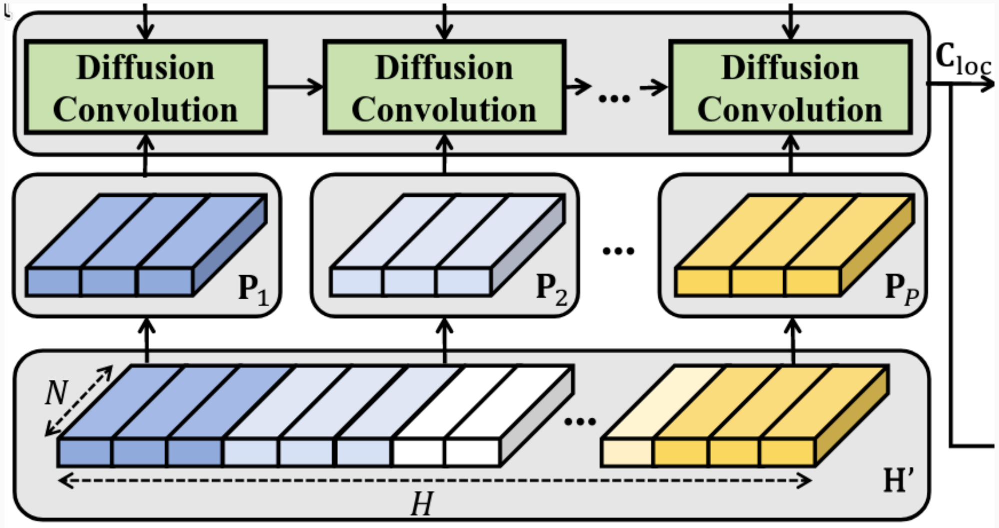


Normalized Data

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

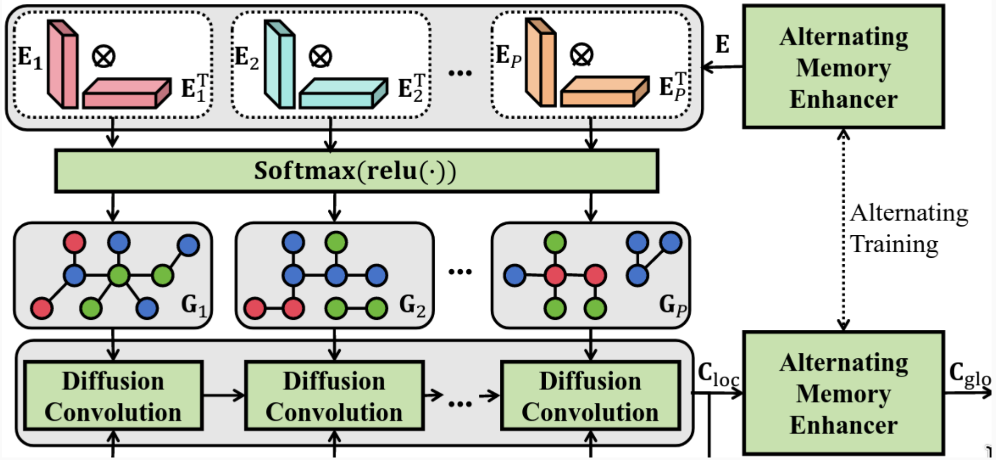
Patches

- 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

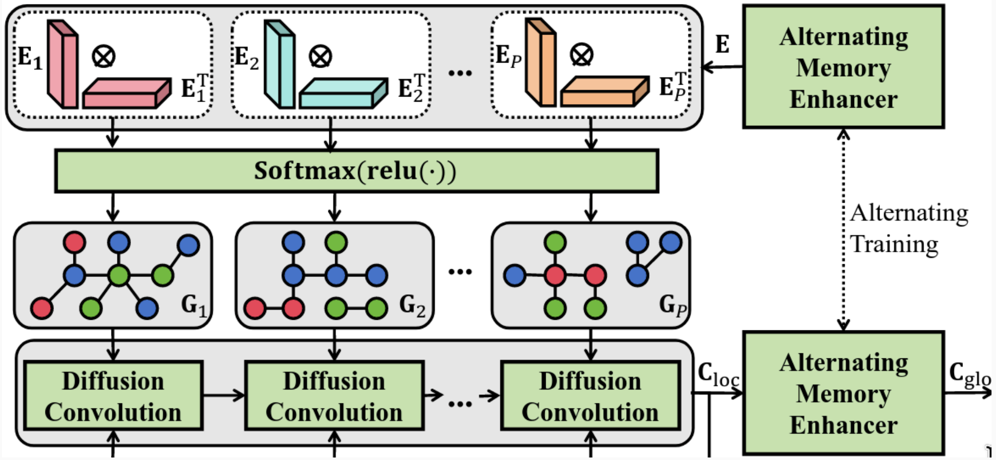


AME

- 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



AME

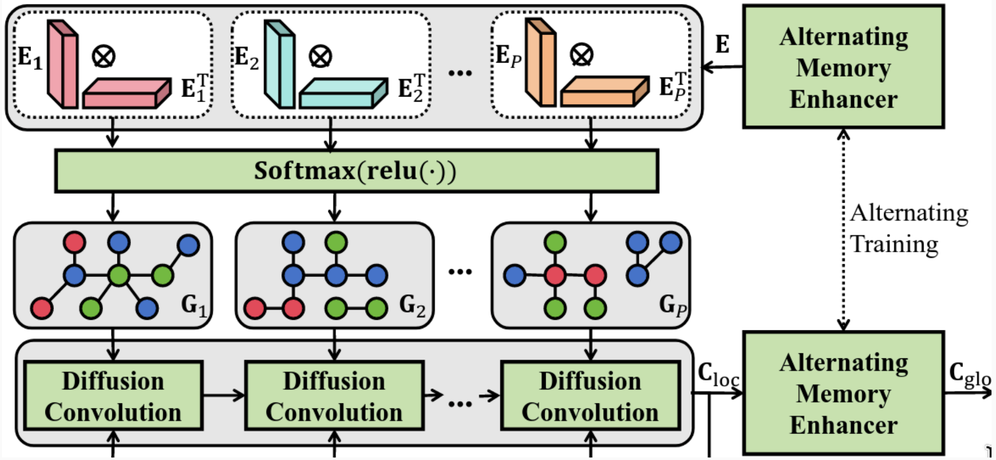
- Provides local memory embedding
  - These are learnable parameters
- Consistant local memory for patch P_i
- Matrix product of $E_i \otimes E_i^T$
  - Similarity matrix for variables in P_i

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



2.3 Patch-wise Recurrent Graph Learning

Diffusion Convolution

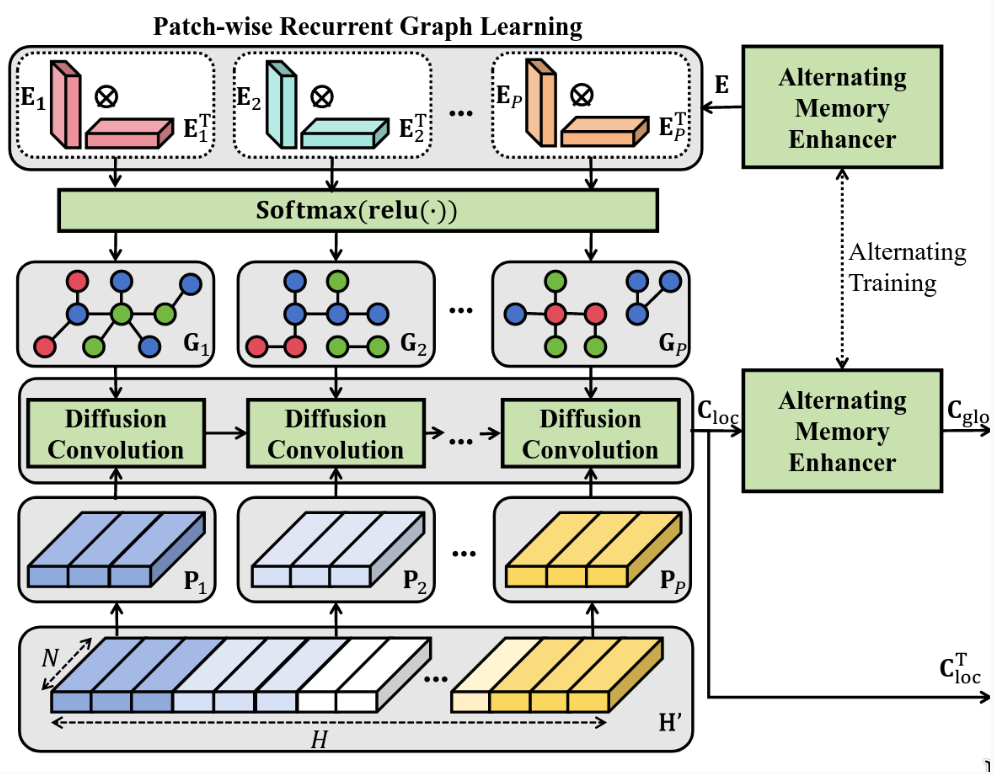
- Normalized data is adjusted based on connections in graph
- Numeric values "diffuse" into neighbours
  - Not only immediate neighbours
- Spatially relates data based on connections

Gated Recurrent Unit

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

Output

- Input features enriched with local information
- Spatial $\longrightarrow$ dynamic correlations
- Temporal $\longrightarrow$ GRU

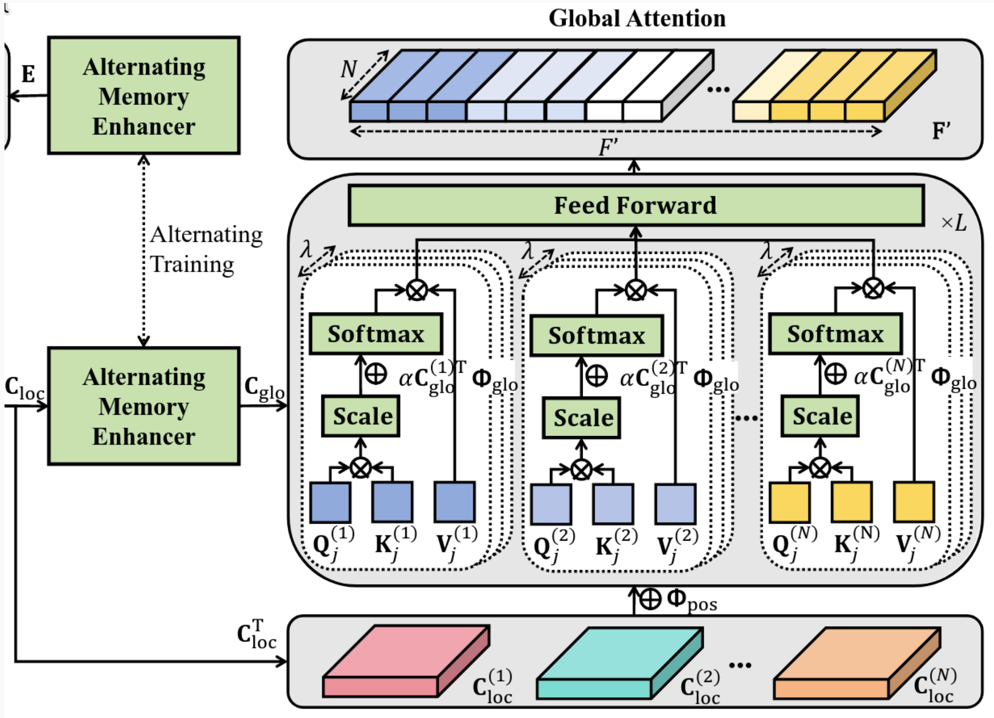


Motivation

- Patch-wise correlations are sensitive
  - Outliers dominate
- Constrain locally enriched features
  - Mitigate disrupted correlations

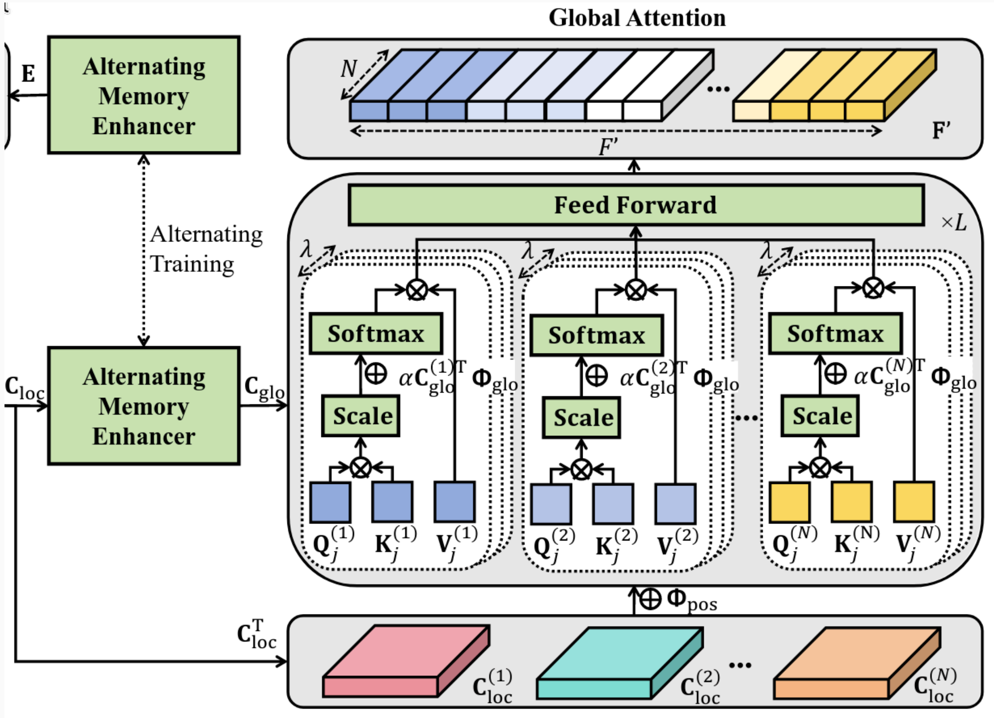
Input

- Transpose locally enriched features
  - Isolate variables
  - Diffusion earlier
- Linear transformation
  - Positional encoding
- Converted to Q, K, V matrices
  - Learnable parameters



Attention

- Relatively conventional implementation
  - Query and Key to find importance
  - Weight Value by importance
- Global information is new
- Adding global information after softmax
  - Bias probabilities
  - Global information affects parameters



Output

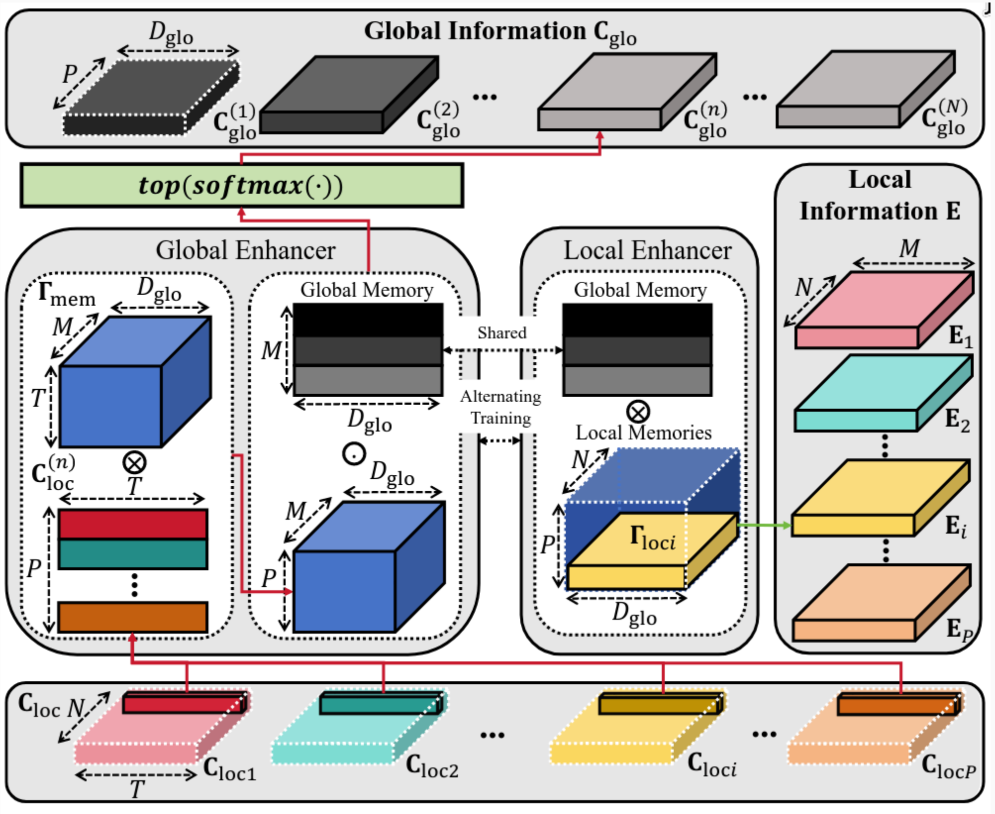
- The final "representation" of data
- **F'** is not a forecast
  - Final feature representation
- Linear layer maps to forecasting horizon

Overview

- Input
  - Locally correlated features
- Outputs
  - Local information E
  - Global information C_{glo}
- Shared global memory

Hyperparameters

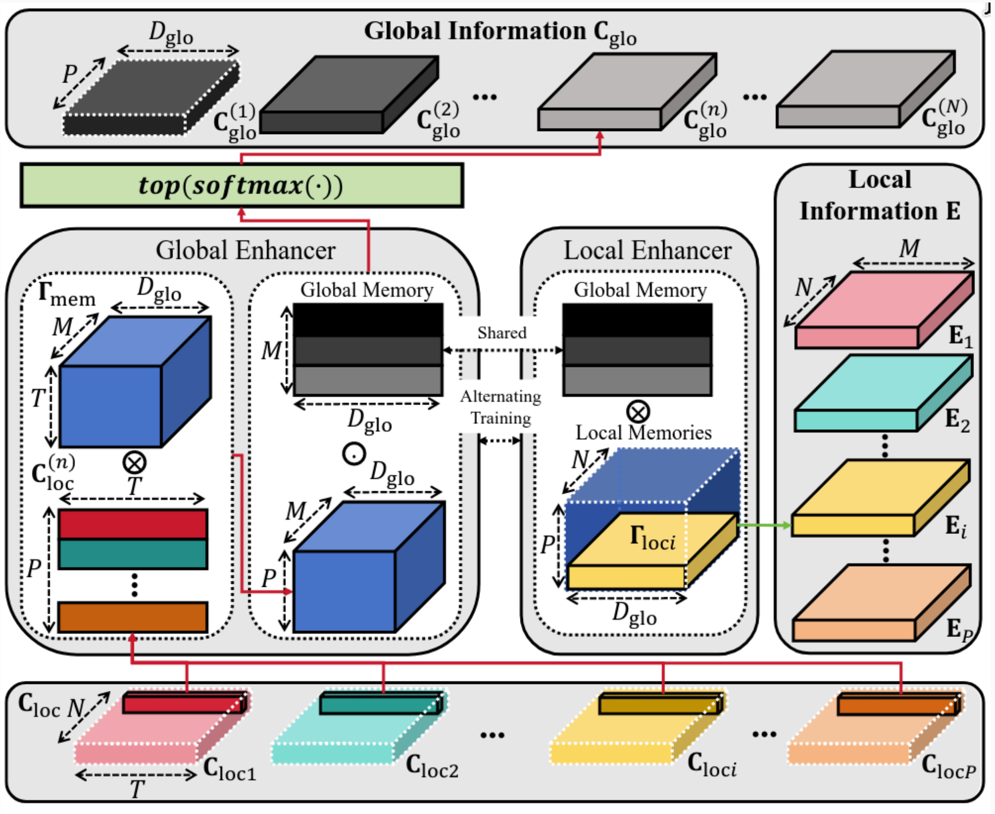
- $M \longrightarrow$ number of high level patterns
  - Spikes, seasons, stable
- $D_{\text{glo}} \longrightarrow$ richness of patterns



2.5 Alternating Memory Enhancer

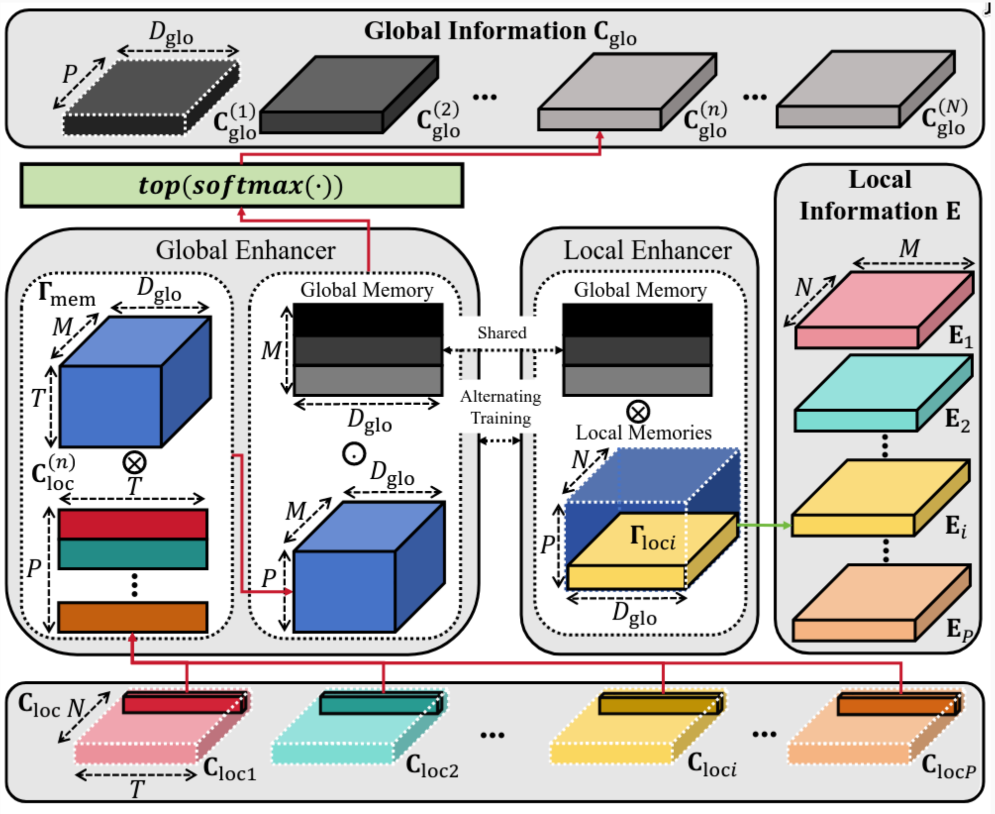
Local Enhancer

- Local memory regions $\Gamma_{\text{loc}i}$
  - One for each patch
- $P_i \longleftrightarrow \Gamma_{\text{loc}i} \longrightarrow \Gamma_{\text{loc}i} \longleftrightarrow E_i$
- Not directly identical
  - E contains global memories
  - Defined by C_{loc}
- Memories are **not** information



Global Enhancer

- Learns from locally correlated features
- Γ_{mem} is a large trainable tensor
  - Produces inquiry tensor
  - Recognizes patterns in data
    - The M high level patterns
- Inquiry tensor
  - Prevalence of patterns in local data
  - Similarity scores with global memory
- Probability distribution
  - Importance of pattern
- Top k most important patterns
  - Stored in C_{glo}
  - Scaled based on importance
    - Weighted sum



Alternating Training

- Local information E requires
  - Local memories
  - Global memories
- Updating both simultaneously
  - Unstable training
  - Issues converging
- LE and GE alternate training
  - Split adjustment of memories

LE Training

- Local memories > global memories
  - More parameters $\longrightarrow$ longer convergence
- Balance convergence
  - Different learning rates
  - LE training more

2.5 Alternating Memory Enhancer

Alternating Training

- Local information E requires
  - Local memories
  - Global memories
- Updating both simultaneously
  - Unstable training
  - Issues converging
- LE and GE alternate training
  - Split adjustment of memories

LE Training

- Local memories > global memories
  - More parameters ⟶ longer convergence
- Balance convergence
  - Different learning rates
  - LE training more

Algorithm 1 AME alternating training

Input: Historical horizon and ground truth $\mathbf{H}, \mathbf{F}$; local and global memories $\boldsymbol{\Gamma}_{\text{loc}}, \boldsymbol{\Gamma}_{\text{glo}}$; local training step ϵ; learning rates $\eta_{\text{loc}}, \eta_{\text{glo}}$ for local and global enhancers

Output: Local and global information $\mathbf{E}, \mathbf{C}_{\text{glo}}$; learned local and global memories $\boldsymbol{\Gamma}_{\text{loc}}, \boldsymbol{\Gamma}_{\text{glo}}$, tensor $\boldsymbol{\Gamma}_{\text{mem}}$, and bias $\mathbf{b}_{\text{mem}}$

```
Initialisation: Initializing local and global memories Γ_loc, Γ_glo, tensor Γ_mem, and bias b_mem randomly
while Γ_loc, Γ_glo, Γ_mem, and b_mem are not converged do
    for iteration = 0 to ϵ do
        H' ← Preprocessing(H)
        E ← A_loc(Γ_loc, Γ_glo)
        C_loc ← G_Θ(H', E)
        C_glo ← A_glo(C_loc, Γ_glo)
        F' ← T_Φ(C_loc, C_glo)
        F̂ ← LinearHead(F')
        Γ_loc ← Γ_loc − η_loc ∇_{Γ_loc} L(F̂, F)
    end for
    H' ← Preprocessing(H)
    E ← A_loc(Γ_loc, Γ_glo)
    C_loc ← G_Θ(H', E)
    C_glo ← A_glo(C_loc, Γ_glo)
    F' ← T_Φ(C_loc, C_glo)
    F̂ ← LinearHead(F')
    Γ_glo ← Γ_glo − η_glo ∇_{Γ_glo} L(F̂, F)
    Γ_mem ← Γ_mem − η_glo ∇_{Γ_mem} L(F̂, F),
    b_mem ← b_mem − η_glo ∇_{b_mem} L(F̂, F)
end while
```

3. Results

4. Critique

4.1 Notation

Preprocessing

- As mentioned earlier
- Unconventional notation
- Obscures details

Inconsistencies

- C_{glo} is global memory
- C_{loc} is locally correlated features
- E is local memory

4.1 Notation

Preprocessing

- As mentioned earlier
- Unconventional notation
- Obscures details

Inconsistencies

- C_{glo} is global memory
- C_{loc} is locally correlated features
- E is local memory

Symbol Reuse

- $\mathbf{F}$ is the ground truth
- F is the dimensionality of $\mathbf{F}$
- $\mathbf{F'}$ is the encoding output
- F' is the dimensionality of $\mathbf{F'}$
- Confusing statements and diagrams

Preprocessing

- As mentioned earlier
- Unconventional notation
- Obscures details

Inconsistencies

- C_{glo} is global memory
- C_{loc} is locally correlated features
- E is local memory

Symbol Reuse

- $\mathbf{F}$ is the ground truth
- F is the dimensionality of $\mathbf{F}$
- $\mathbf{F'}$ is the encoding output
- F' is the dimensionality of $\mathbf{F'}$
- Confusing statements and diagrams

$\mathbf{F'} \in \mathbb{R}^{F' \times N}$, where F' is the temporal dimension of the representation.

Preprocessing

- As mentioned earlier
- Unconventional notation
- Obscures details

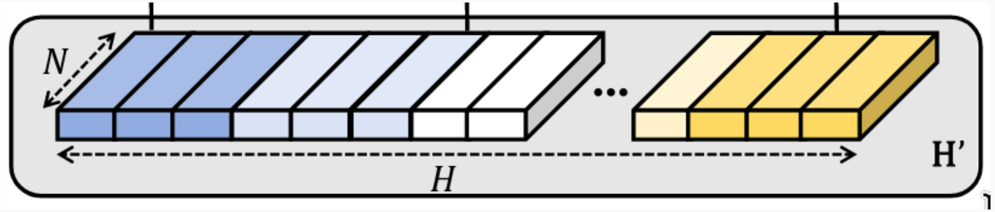
Inconsistencies

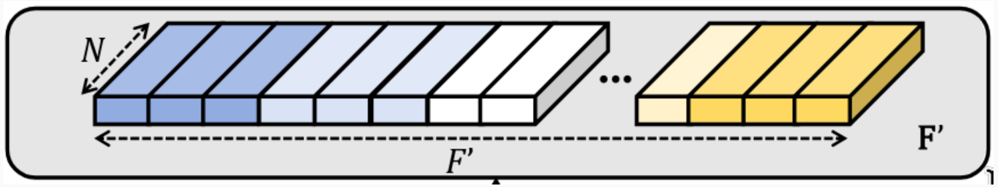
- C_{glo} is global memory
- C_{loc} is locally correlated features
- E is local memory

Symbol Reuse

- $\mathbf{F}$ is the ground truth
- F is the dimensionality of $\mathbf{F}$
- $\mathbf{F'}$ is the encoding output
- F' is the dimensionality of $\mathbf{F'}$
- Confusing statements and diagrams

$\mathbf{F'} \in \mathbb{R}^{F' \times N}$, where F' is the temporal dimension of the representation.





5. Praise

Colors

- Help understanding and data flow
  - Preprocessing ⟶ final encoding
  - Minor inconsistencies
    - Attention

Dimensionality

- Squares ⟶ 2-dimensional
- Cubes ⟶ 3-dimensional
- Transposed ⟶ lying down
- Slices of shapes
  - M slices of global memory
  - P slices of local memory