# Fast and Accurate Online Coupled Matrix-Tensor Factorization via Frequency Regularization

*Yong-chan Park, SeungJoo Lee, and U Kang*

*Data Mining Lab*

*Dept. of CSE*

*Seoul National University*

August 9–13, 2026
KDD2026
Jeju, Korea

# Outline

**Introduction** — Why online CMTF?

**Proposed Method** — Adaptive forgetting, frequency regularization, incremental updates

**Experiments** — Speed, accuracy, scalability, and ablations

**Conclusion** — Main takeaways

# Introduction

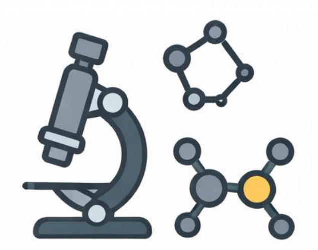
- **Motivation:** multi-source data arrive continuously

- **Applications**
  - Modern applications generate multi-source streams

Recommender system

Social network analysis

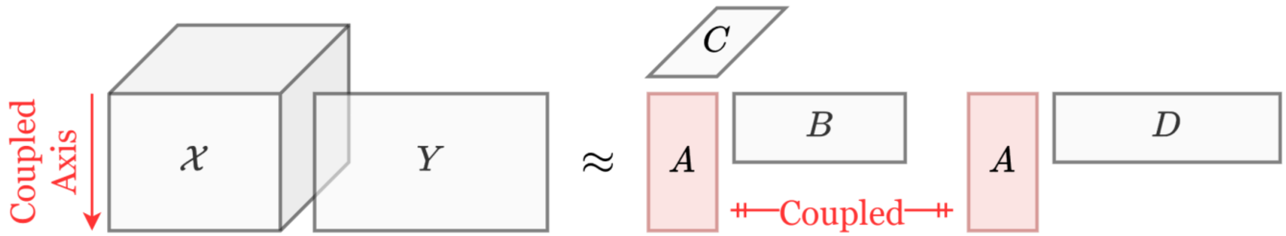
Multimodal healthcare



Multi-omics analysis

# Introduction

- **Background:** Coupled Matrix-Tensor Factorization (CMTF)
  - CMTF jointly factorizes a tensor $\mathcal{X}$ and its coupled matrix $Y$ to effectively find latent features




- **Key benefits**
  - Integrates heterogeneous data sources

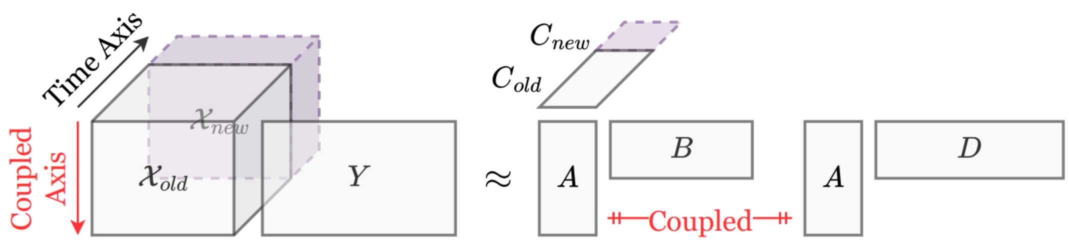
# Introduction

- **Why static CMTF is not enough**
  - Static CMTF becomes expensive in streams
  - When new data arrive:
    - old tensor + new slices → <u>re-factorize everything</u>? 🤔

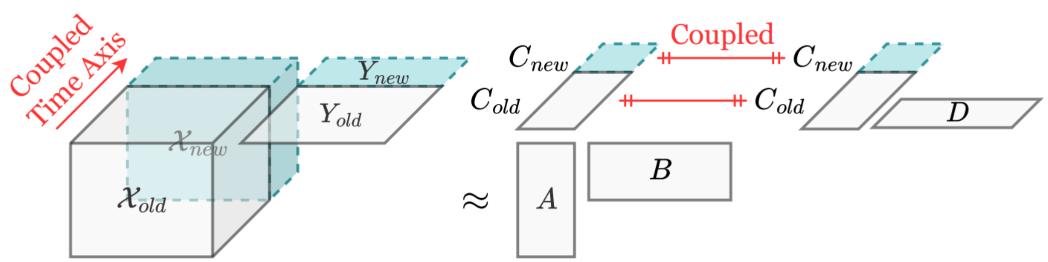
- **Problems:**
  - Repeated computation
  - Slow update
  - Poor adaptation to temporal changes

# Introduction

- **Two streaming CMTF settings**
  - Single-Tensor Streaming: Tensor grows; matrix is fixed
  - Matrix-Tensor Streaming: Both tensor and coupled matrix grow




(a) Single-Tensor Streaming




(b) Matrix-Tensor Streaming

# Introduction

- **Challenges of online CMTF**
  - 1. Time-evolving data
    - Past data may become less relevant
  - 2. Noisy temporal factors
    - Time factors are vulnerable to noise and overfitting
  - 3. Efficient updates
    - Full retraining is too costly

# Proposed Method

- **Overview of FOCAL**
  - 1. Adaptive forgetting
    - Learn how much history to keep.
  - 2. Frequency regularization
    - Guide temporal factors using spectral patterns.
  - 3. Incremental factor updates
    - Reuse old computations and update only what changes.

# Proposed Method

## Idea 1: Adaptive Forgetting

- FOCAL separates old and new data:

$$\mathcal{L} = \underbrace{\lambda \|\mathcal{X}_{old} - [\![A, B, C_{old}]\!]\|_F^2}_{\textit{old error}} + \underbrace{\|\mathcal{X}_{new} - [\![A, B, C_{new}]\!]\|_F^2}_{\textit{new error}} + \|Y - AD^\top\|_F^2$$

  - $\lambda$: forgetting factor
  - $\|\cdot\|_F$: Frobenius norm

$\mathcal{X}_{old} \in \mathbb{R}^{I \times J \times K_{old}}$
$\mathcal{X}_{new} \in \mathbb{R}^{I \times J \times K_{new}}$
$Y \in \mathbb{R}^{I \times L}$
$A \in \mathbb{R}^{I \times R}$
$B \in \mathbb{R}^{J \times R}$
$C_{old} \in \mathbb{R}^{K_{old} \times R}$
$C_{new} \in \mathbb{R}^{K_{new} \times R}$
$D \in \mathbb{R}^{L \times R}$

- $\lambda$ controls memory
  - Small $\lambda$: focus more on recent data
  - Large $\lambda$: preserve more historical data

recent focus — $\lambda$ — history focus

# Proposed Method

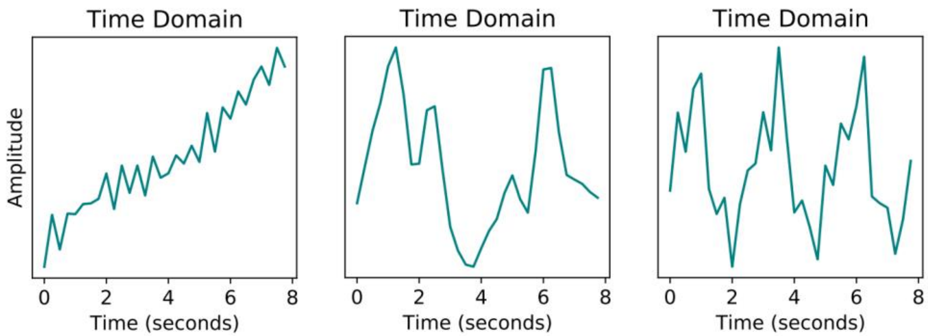
- **Idea 1: Adaptive Forgetting**
  - How $\lambda$ adapts online?
    - FOCAL updates memory from local error
  - At each update:
    - $\lambda \leftarrow \lambda + \eta(\bar{e}_{loc} - e_{loc})$
      - $e_{loc}$: local error
      - $\bar{e}_{loc}$: average local error
      - $\eta$: learning rate
    - High error now ($\bar{e}_{loc} < e_{loc}$) → recent focus ($\lambda \downarrow$)
    - Low error now ($\bar{e}_{loc} > e_{loc}$) → history focus ($\lambda \uparrow$)

recent focus history focus

$\leftarrow \lambda \rightarrow$

# Proposed Method

- **Idea 2: Frequency Regularization**
  - Temporal factors should capture:
    - Long-term trends
    - Smooth local changes
    - Seasonal patterns

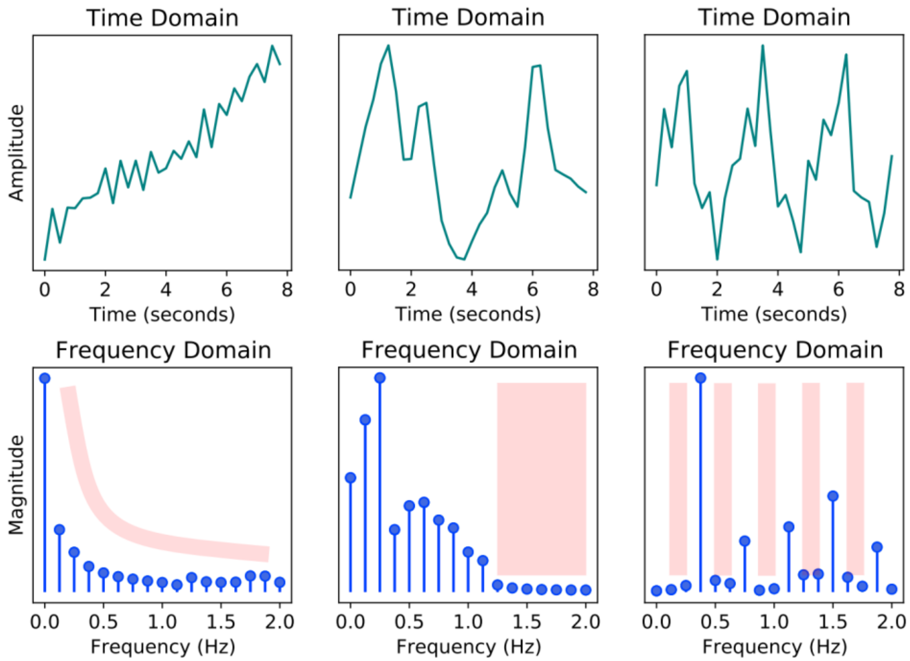



(a) Global trends (b) Local smoothness (c) Seasonality

  - But without regularization, they may fit noise

# Proposed Method

- **Idea 2: Frequency Regularization**
  - Temporal patterns become clear in frequency space
    - Long-term trends ↔ Strong low frequencies
    - Smooth local changes ↔ Suppress high frequencies
    - Seasonal patterns ↔ Periodic peaks

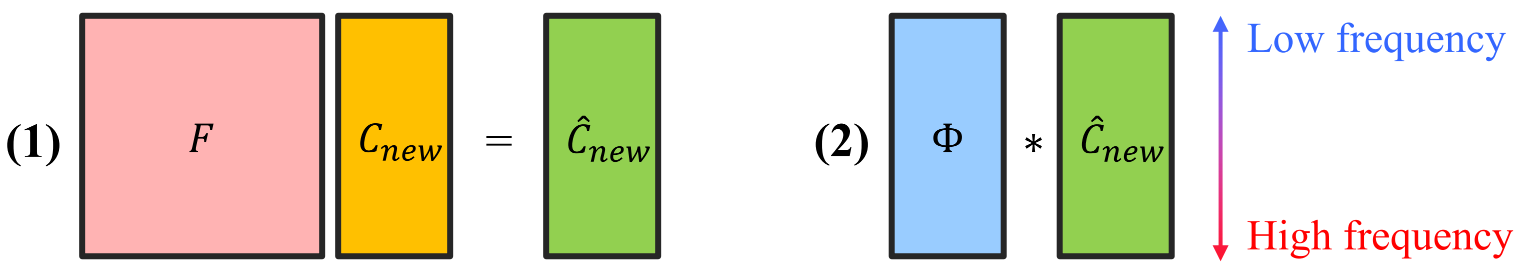


# Proposed Method

## Idea 2: Frequency Regularization

- FOCAL regularizes the temporal factors in frequency space
  - $\mathcal{L}_f = \|\Phi * (FC_{new})\|_F^2$
    - $\Phi$: learned frequency mask
    - $C_{new}$: new temporal factor
    - $F$: Fourier matrix
    - $*$ : Hadamard product

$\Phi \in \mathbb{R}^{K_{new} \times R}$
$F \in \mathbb{R}^{K_{new} \times K_{new}}$
$C_{new} \in \mathbb{R}^{K_{new} \times R}$



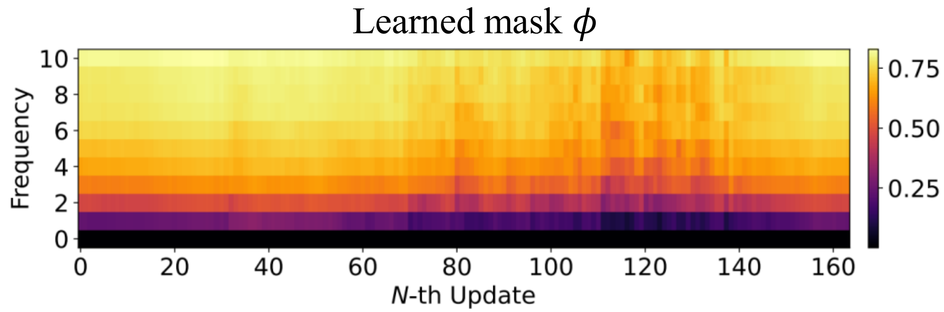

Fourier transform

Frequency filtering

# Proposed Method

- **Idea 2: Frequency Regularization**
  - The mask is learned from streaming data
    - For each new batch:
      - Compute fast Fourier transform along the time mode
      - Accumulate the power spectrum
      - Assign lower penalty to consistently active frequencies
    - $\Phi = [\phi, \dots, \phi]$
    - $\phi_X \leftarrow \lambda \phi_X + \frac{1}{IJ} \sum_{i,j} |F X_{new}| [i, j, :]$
    - $\phi = \exp\left(-\frac{\phi_X}{\|\phi_X\|_2}\right)$

$\Phi \in \mathbb{R}^{K_{new} \times R}$
$\phi_X \in \mathbb{R}^{K_{new}}$
$\phi \in \mathbb{R}^{K_{new}}$



# Proposed Method

- **Idea 3: Incremental Factor Updates**
  - Naïve update:
    - Recompute from all accumulated data
    - Cost grows with $K_{old} + K_{new}$
  - FOCAL update:
    - Reuse old summaries + process only new data
      - Old data are represented by compact auxiliary matrices $\{P_i\}$
      - $P_i \leftarrow \lambda P_i + (new\ contribution)$
    - Cost depends only on $K_{new}$

# Proposed Method

- **FOCAL update workflow**
  - FOCAL update at each streaming step

(1) Receive $\mathcal{X}_{new}$ and optionally $Y_{new}$

(2) Update frequency mask $\Phi$

(3) Update auxiliary matrices $\{P_i\}$

(4) Update factor matrices $A, B, C, D$

(5) Update forgetting factor $\lambda$

# Experiments

- **Q1. Performance.** How accurately and efficiently does FOCAL update factors?
- **Q2. Scalability.** How does FOCAL scale with increasing sizes of newly arrived data?
- **Q3. Ablation study.** How does the adaptive forgetting factor, as well as the frequency regularization, affect the performance?
- **Q4. Visualization.** Which temporal patterns are revealed through visualization of the learned frequency filter?

# Experiments

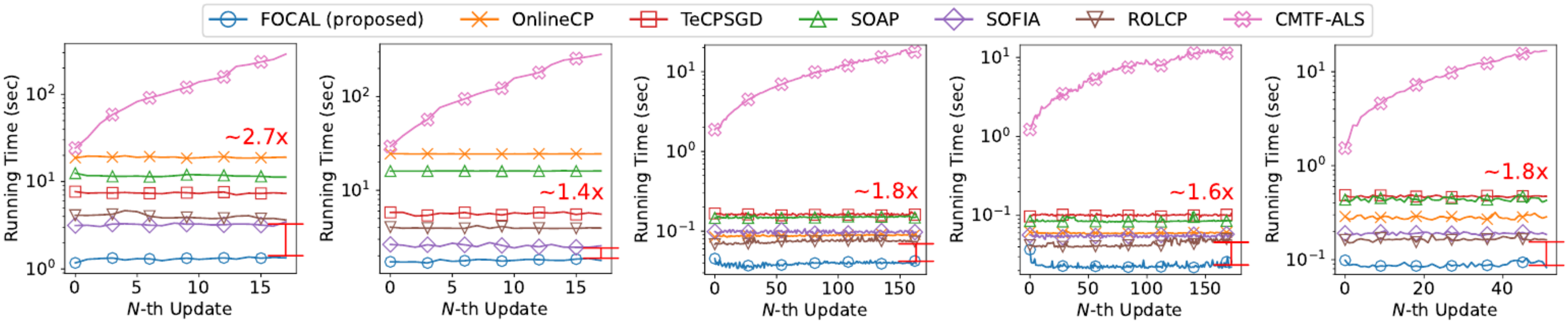
- **Experimental Setup**
  - Datasets

| Dataset | Stream Type | Tensor Size | Matrix Size |
|---|---|---|---|
| **ML-100K**[3] | Single-Tensor | $1682 \times 943 \times 200$ | $1682 \times 19$ |
| **Amazon**[4] | Single-Tensor | $901 \times 1720 \times 200$ | $901 \times 170$ |
| **Stock-US**[1] | Matrix-Tensor | $500 \times 88 \times 3651$ | $3651 \times 10$ |
| **Stock-JPN**[1] | Matrix-Tensor | $300 \times 88 \times 3856$ | $3856 \times 10$ |
| **Stock-CHN**[1] | Matrix-Tensor | $1471 \times 88 \times 1170$ | $1170 \times 10$ |

  - Baselines
    - OnlineCP, TeCPSGD, SOAP, SOFIA, ROLCP, CMTF-ALS
  - Metrics
    - Running time
    - Local reconstruction error
    - Global reconstruction error

# Experiments

- **Speed result**
  - FOCAL is consistently faster




(a) Running time on ML-100K (b) Running time on Amazon (c) Running time on Stock-US (d) Running time on Stock-JPN (e) Running time on Stock-CHN

- Lowest running time across
- Up to 2.7× faster than competitors
- Much faster than rerunning CMTF-ALS

# Experiments

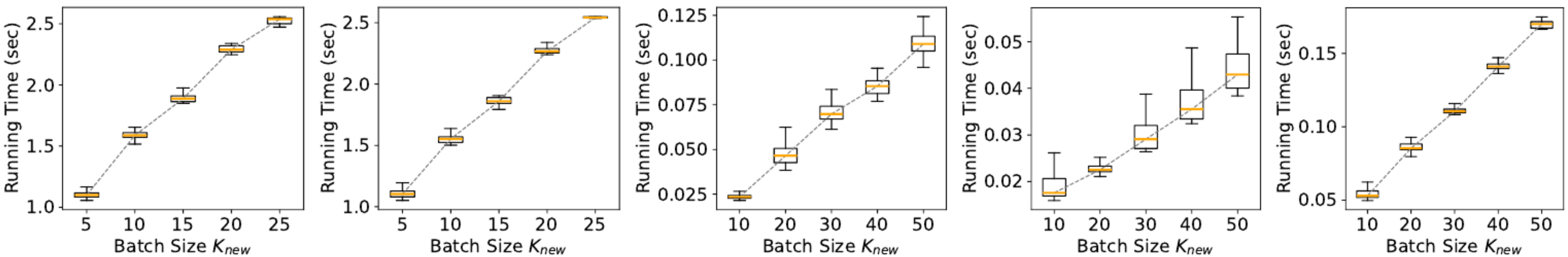
- **Accuracy result**
  - FOCAL also improves accuracy

| Method | Single-Tensor Streaming | | Matrix-Tensor Streaming | | |
|---|---|---|---|---|---|
| | ML-100K | Amazon | Stock-US | Stock-JPN | Stock-CHN |
| OnlineCP [54] | 0.9880 ± 0.0137 | 0.9961 ± 0.0056 | 0.8607 ± 0.0822 | 0.8272 ± 0.0721 | 0.7930 ± 0.0388 |
| TeCPSGD [36] | 0.9793 ± 0.0103 | 0.9908 ± 0.0092 | 0.7719 ± 0.0996 | 0.7394 ± 0.0740 | 0.7027 ± 0.0552 |
| SOAP [38] | 1.2773 ± 0.2316 | 1.6224 ± 0.6292 | 0.9700 ± 01842 | 0.9438 ± 0.1737 | 0.9465 ± 0.1756 |
| SOFIA [29] | 0.9925 ± 0.0165 | 0.9970 ± 0.0216 | 0.7144 ± 0.1712 | 0.6687 ± 0.1631 | 0.7201 ± 0.1836 |
| ROLCP [1] | 0.9938 ± 0.0070 | 0.9980 ± 0.0016 | <u>0.6754 ± 0.0617</u> | <u>0.6563 ± 0.0727</u> | <u>0.6793 ± 0.0845</u> |
| CMTF-ALS [2] | <u>0.9335 ± 0.0506</u> | <u>0.9744 ± 0.0413</u> | 0.9628 ± 0.0807 | 0.9013 ± 0.0903 | 0.8803 ± 0.0427 |
| FOCAL (proposed) | **0.8050 ± 0.0642** | **0.9656 ± 0.0155** | **0.6044 ± 0.0663** | **0.6097 ± 0.0522** | **0.5614 ± 0.0362** |

  - Why?
    - FOCAL uses coupled information, adapts to recent changes, and stabilizes the temporal factor

# Experiments

- **Scalability**
  - FOCAL scales near-linearly with new batch size




(a) Scalability on ML-100K (b) Scalability on Amazon (c) Scalability on Stock-US (d) Scalability on Stock-JPN (e) Scalability on Stock-CHN

  - As $K_{new}$ increases, runtime grows smoothly without sudden computational blow-up

# Experiments

- **Ablation study**
  - Adaptive forgetting reduces local error

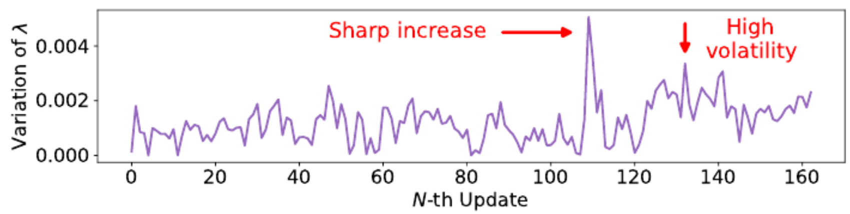
| Dataset | Local Error | | Global Error | |
|---|---|---|---|---|
| | Manual | Adaptive | Manual | Adaptive |
| ML-100K | 0.8218 | **0.8050** | 1.6462 | 1.6681 |
| Amazon | 0.9808 | **0.9656** | 2.3655 | 2.4309 |
| Stock-US | 0.6293 | **0.6044** | 5.5264 | 6.3022 |
| Stock-JPN | 0.6184 | **0.6097** | 3.8301 | 4.2670 |
| Stock-CHN | 0.5733 | **0.5614** | 1.8545 | 2.0864 |

  - Frequency regularization improves accuracy

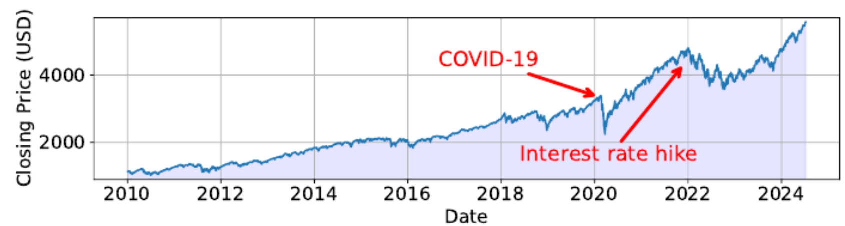
| Dataset | Local Error | | Update Time (sec) | |
|---|---|---|---|---|
| | w/o FR | w/ FR | w/o FR | w/ FR |
| ML-100K | 0.8163 | **0.8050** | 1.1246 | 1.1374 |
| Amazon | 0.9727 | **0.9656** | 1.7502 | 1.7763 |
| Stock-US | 0.6195 | **0.6044** | 0.0389 | 0.0396 |
| Stock-JPN | 0.6212 | **0.6097** | 0.0225 | 0.0230 |
| Stock-CHN | 0.5706 | **0.5614** | 0.0871 | 0.0889 |

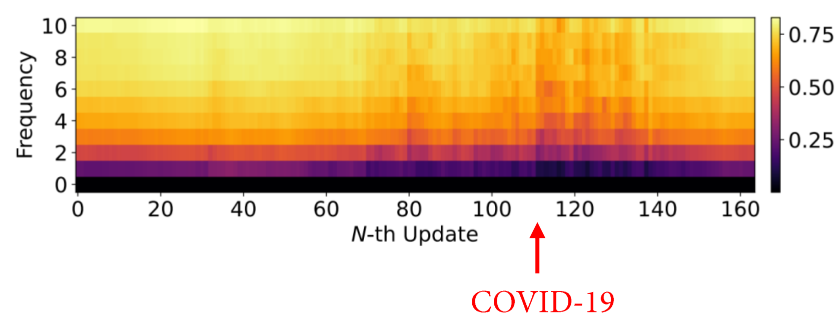
# Experiments

- **Visualization**
  - Forgetting-factor variation reveals abrupt distribution shifts
  - Learned frequency mask shows evolving temporal patterns




(a) Variation of forgetting factor $\lambda$



# Conclusion

- FOCAL enables fast and accurate online CMTF
  - Adaptive
    - Dynamically adjusts its memory using an adaptive forgetting factor
  - Accurate
    - Captures diverse temporal patterns through learned frequency regularization
  - Efficient
    - Reuses previous computations instead of retraining from scratch

- **Key takeaway**
  - Adapt to what changes, and reuse what remains stable

# THANK YOU!

*https://github.com/snudatalab/FOCAL*

*https://yongchanpark.github.io/*

*ycpark6767@gmail.com*