Fast and Accurate Partial Fourier Transform for Time Series Data

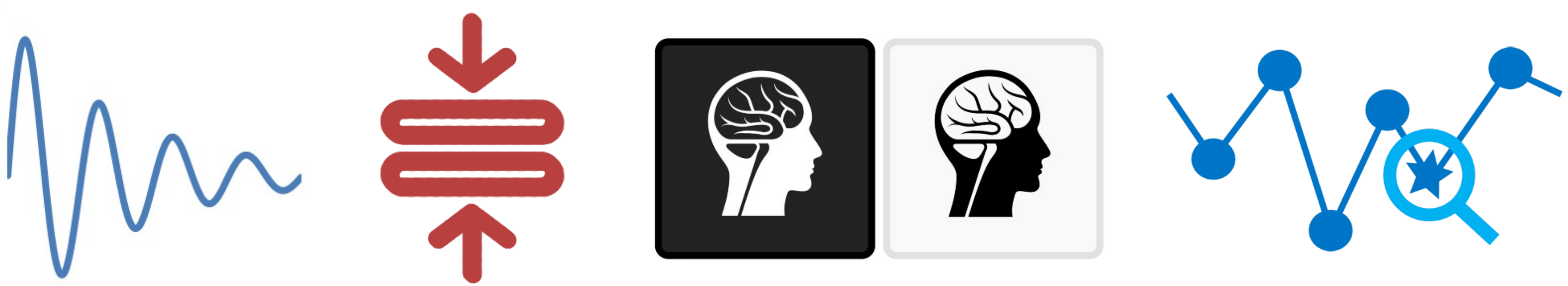
Yong-chan Park, **Jun-Gi Jang**, and **U Kang**
Computer Science & Engineering
Seoul National University

Outline

- **Introduction**
- Existing Works
- Proposed Method
- Experiments
- Conclusion

Fourier Transform

- Fundamental tool for numerous applications
  - Signal / image processing
  - Data compression (e.g., mp3 and jpg)
  - Medical imaging (e.g., MRI)
  - Anomaly detection

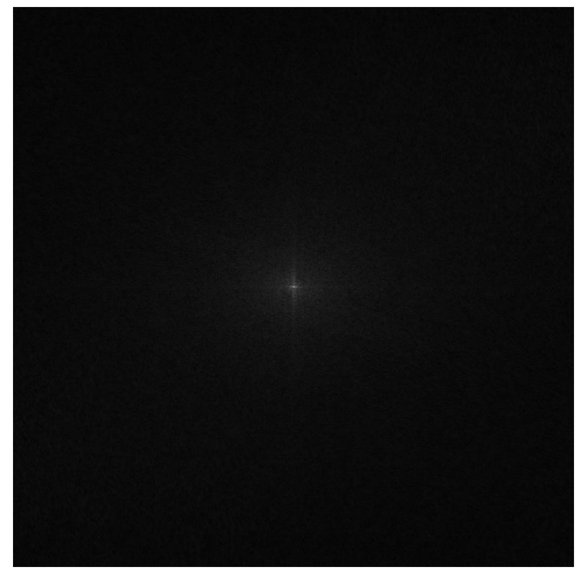


https://www.gettyimages.com
https://www.pinclipart.com
https://www.kissclipart.com
https://commons.wikimedia.org

Fourier Transform

- Temporal or spatial domain → Frequency domain

Spatial domain

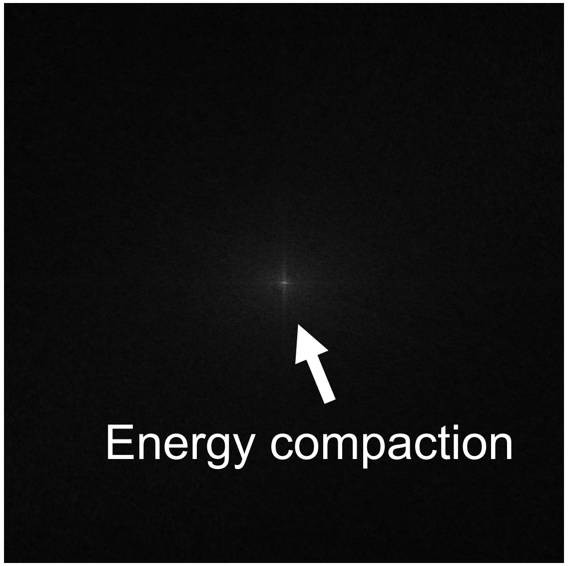


Frequency domain

Fourier Transform

- Strong **energy compaction** or **sparsity**
  - Fourier coefficients are mostly small or equal to zero

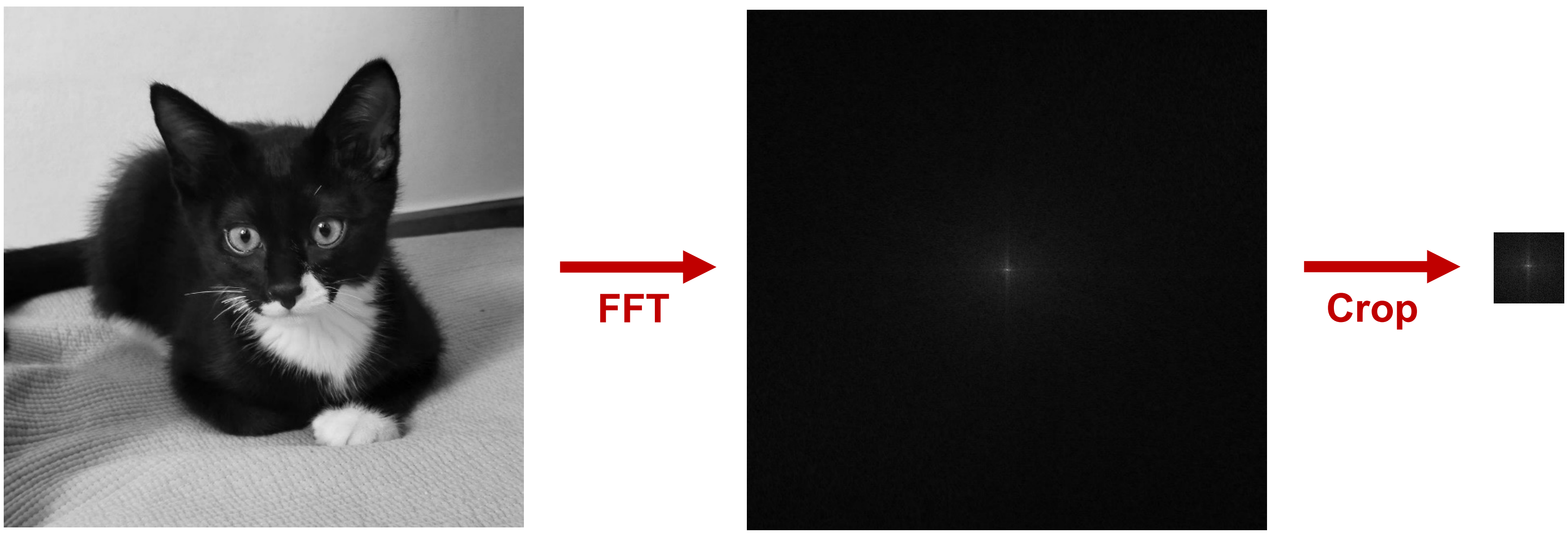
Spatial domain



Frequency domain

Motivation

- Fast Fourier Transform (FFT) is inefficient
  - FFT always computes *all* the coefficients

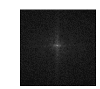


Most of the coefficients
are just discarded!

Motivation

- FFT computes even the unnecessary coefficients
  - Is it possible to efficiently compute only a few of them?

How can we do this directly?



Problem Definition

- Partial Fourier Transform
  - **Given**
    - Complex-valued vector $\boldsymbol{a}$ of size N
    - Non-negative integer $M \ll N$
    - Integer μ
  - **Estimate**
    - Fourier coefficients of $\boldsymbol{a}$ for $[\mu - M, \mu + M]$

Outline

- Introduction
- **Existing Works**
- Proposed Method
- Experiments
- Conclusion

FFT

- **Fast Fourier Transform (FFT)** rapidly computes the full Fourier coefficients
  - FFT has been highly optimized over decades
  - Time complexity: $O(N \log N)$
- **Limitation**
  - No option to efficiently compute only a few coefficients
  - Unnecessary coefficients are just discarded

Goertzel Algorithm

- **Goertzel algorithm** is one of the first methods for computing partial Fourier coefficients
  - Time complexity: $O(MN)$
- **Limitation**
  - Essentially the same as computing individual coefficients
  - It is limited to rare scenarios where a very few number of coefficients are required

Subband DFT

- **Subband DFT** decomposes the input into a set of subsequences, and removes some of them with small energy contribution
  - Time complexity: $O(N + M \log N)$
- **Limitation**
  - Substantial issue of low accuracy
  - No option to set an error bound

Pruned FFT

- **FFT Pruning** is a modification of the standard split-radix FFT
  - Almost optimized because it uses FFT as a subroutine
  - Time complexity: $O(N \log M)$
- **Limitation**
  - The performance gains are rather modest in practice
  - **PFT (proposed)** significantly outperforms Pruned FFT

Outline

- Introduction
- Existing Works
- **Proposed Method**
- Experiments
- Conclusion

Overview

- **PFT (Partial Fourier Transform)**
  - Efficiently computes a part of Fourier coefficients
  - Time complexity: $O(N + M \log M)$ (state-of-the-art)
  - Provides an option to set an arbitrary numerical precision
- Main ideas
  - Polynomial approximation
  - Base exponential function
  - Reordering operators

Polynomial Approximation

- PFT approximates a set of smooth twiddle factors by **polynomials**
  - Significantly reduces the computational cost due to the mixture of many twiddle factors = trigonometric functions
  - Typically, a smoother function results in a better polynomial approximation

Polynomial Approximation

$\boldsymbol{a}$: complex-valued array of size N

$\hat{\boldsymbol{a}}$: Fourier transform of $\boldsymbol{a}$

$[N] = \{0, 1, \dots, N-1\}$

$N = pq \ (p, q > 1)$

$\omega_N = e^{-2\pi i/N}$

$$\hat{a}_m = \sum_{n \in [N]} a_n \omega_N^{mn} = \omega_{2p}^{m} \sum_{k \in [p]} \sum_{l \in [q]} a_{qk+l} \underline{\omega_N^{m(l-q/2)}} \omega_p^{mk}$$

Definition of F.T. — Modification — **Smooth twiddle factors**

Polynomial Approximation

$\boldsymbol{a}$: complex-valued array of size N

$\hat{\boldsymbol{a}}$: Fourier transform of $\boldsymbol{a}$

$[N] = \{0, 1, \dots, N-1\}$

$N = pq \ (p, q > 1)$

$\omega_N = e^{-2\pi i/N}$

$$\hat{a}_m = \sum_{n \in [N]} a_n \omega_N^{mn} = \omega_{2p}^{m} \sum_{k \in [p]} \sum_{l \in [q]} a_{qk+l} \omega_N^{m(l-q/2)} \omega_p^{mk}$$

Smooth twiddle factors

However, approximating all the factors is time-consuming

Base Exponential Function

- Fix a **base exponential function** and exploit the laws of exponents: $\boldsymbol{e^{ab} = (e^{a})^{b}}$
  - All data-independent factors can be precomputed
  - Bypasses the approximation problem

Base Exponential Function

$||f||_A = \sup\{|f(x)| : x \in A\}$

P_α: set of polynomials of degree at most α

$\xi, u \in \mathbb{R}$

ϵ: tolerance

r: number of approximation terms

$$\mathcal{P}_{\alpha,\xi,u} := \arg\min_{P \in P_\alpha} \|P(x) - e^{uix}\|_{|x| \le |\xi|}$$

$$\xi(\epsilon, r) := \sup\{\xi \ge 0 : \|\mathcal{P}_{r-1,\xi,\pi}(x) - \underline{e^{\pi ix}}\|_{|x| \le \xi} \le \epsilon\}$$

We use this as a base

Base Exponential Function

- If $\xi(\epsilon, r) \geq M/p$, the following holds for $|m| \leq M$:

$$\hat{a}_m = \sum_{n \in [N]} a_n \omega_N^{mn} = \omega_{2p}^{m} \sum_{k \in [p]} \sum_{l \in [q]} a_{qk+l} \underline{\omega_N^{m(l-q/2)}} \omega_p^{mk}$$

Definition of F.T.

Modification

Approximation

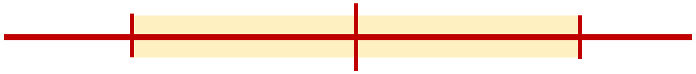
$$\sim \omega_{2p}^{m} \sum_{k \in [p]} \sum_{l \in [q]} a_{qk+l} \underline{\mathcal{P}_{r-1,\xi(\epsilon,r),\pi}(-2m(l-q/2)/N)} \omega_p^{mk}$$

We approximate only the base, and re-scale the polynomial

General Target Range

- A slight modification of the algorithm allows the target range to be arbitrarily centered at $\mu \in \mathbb{Z}$

$m \in [\mu - M, \mu + M]$



$A = (a_{k,l}) = a_{qk+l}$

$B = (b_{l,j}) = \omega_N^{\mu(l-q/2)} w_{\epsilon,r,j} (1 - 2l/q)^j$

$w_{\epsilon,r,j}$: j^{th} coefficient of $\mathcal{P}_{r-1,\xi(\epsilon,r),\pi}$

$$\hat{a}_m \sim \omega_{2p}^m \sum_{j \in [r]} ((m-\mu)/p)^j \sum_{k \in [p]} \omega_p^{mk} \sum_{l \in [q]} a_{k,l} b_{l,j}$$

Reordering operators

- **Reordering operators** results in a significant computational benefit
  - Achieve the lowest time complexity $O(N + M \log M)$

$$\hat{a}_m \sim \omega_{2p}^{m} \sum_{j \in [r]} ((m - \mu)/p)^j \sum_{k \in [p]} \omega_p^{mk} \sum_{l \in [q]} a_{k,l} b_{l,j}$$

Mat. Mul.

FFTs

Dot Products

Outline

- Introduction
- Existing Works
- Proposed Method
- **Experiments**
- Conclusion

Experimental Setup

- We use both synthetic and real-world datasets

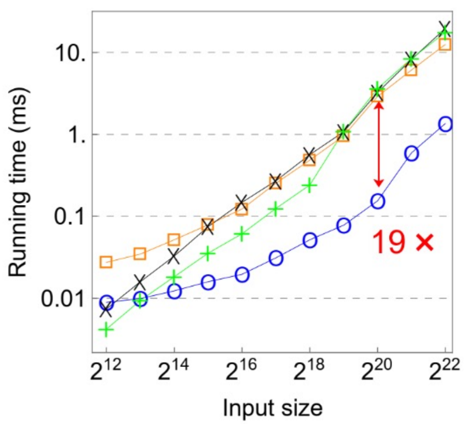
Dataset	Type	# of Time Series	Length
$\{S_n\}_{n=12}^{22}$	Synthetic	1000	2^n
Urban Sound	Real-world	4347	32000
Air Condition	Real-world	29	19735
Stock	Real-world	4	1012

- Measure: relative ℓ_2 error
  - We use single-precision floating-point format, and set the relative ℓ_2 error to be less than 10^{-6}

Speed

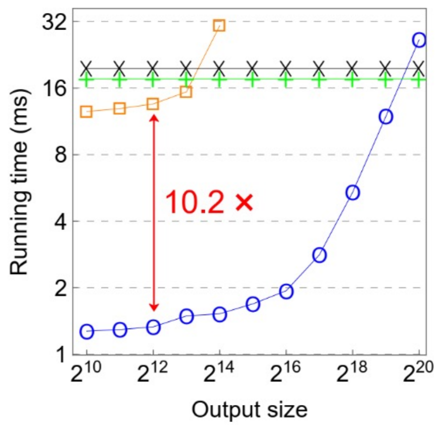
- PFT shows state-of-the-art speed on all the datasets ***without*** sacrificing accuracy
  - Even when N is not a power of 2 or has a large prime

$(N = 2^{22})$



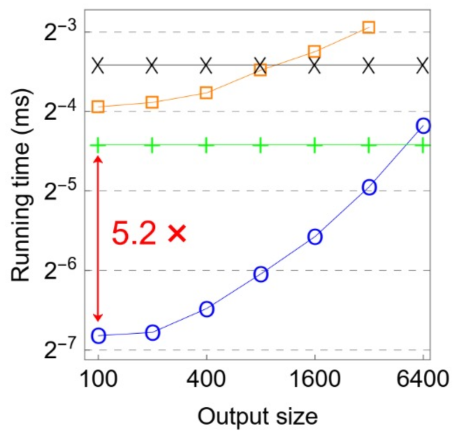
(a) Running time vs. input size

$(N = 2^{22})$



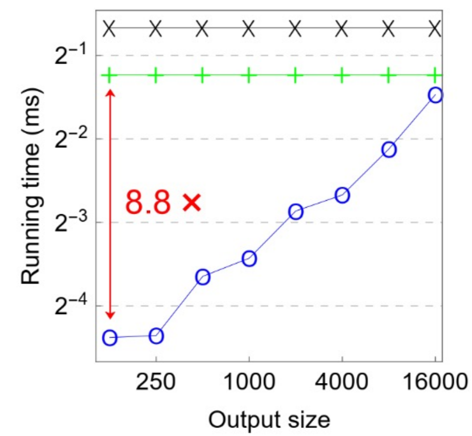
(b) Running time vs. output size

$(N = 2^8 \times 5^3)$



(a) Urban Sound (N = 32000)

$(N = 5 \times 3947)$



(b) Air Condition (N = 19735)

PFT (proposed) | Pruned FFT | FFTW | MKL

Precision vs. Speed

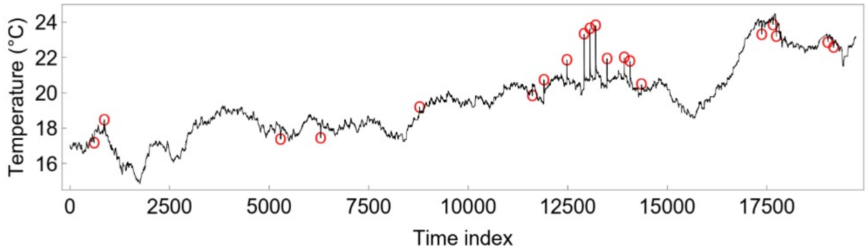
- PFT can set an **arbitrary numerical precision**
  - The trade-off is very useful when the fast evaluation is of utmost importance

$(N = 2^{22})$

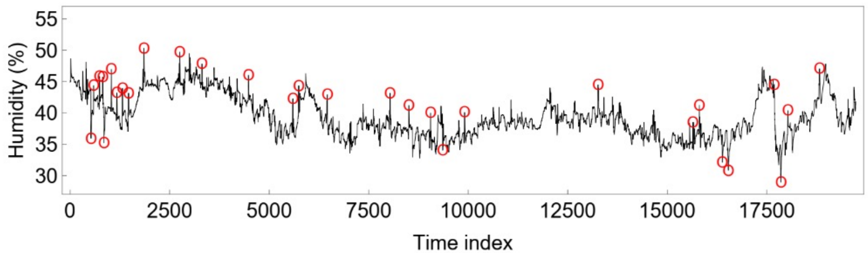
M	Precision		
	10^{-6}	10^{-4}	10^{-2}
2^9	1.273	1.249 (.98)	1.238 (.97)
2^{10}	1.295	1.278 (.99)	1.244 (.96)
2^{11}	1.332	1.293 (.97)	1.251 (.94)
2^{12}	1.491	1.329 (.89)	1.277 (.86)
2^{13}	1.526	1.400 (.92)	1.343 (.88)
2^{14}	1.692	1.607 (.95)	1.512 (.89)
2^{15}	1.940	1.872 (.96)	1.740 (.90)
2^{16}	2.821	2.469 (.88)	2.297 (.81)
2^{17}	5.411	4.590 (.85)	4.058 (.75)
2^{18}	11.924	9.927 (.83)	8.733 (.73)

Anomaly Detection (1)

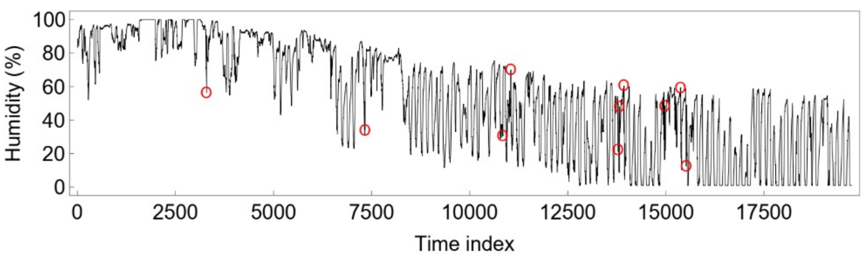
- PFT successfully detects the anomalies, regardless of different patterns of data
  - The outliers found by PFT exactly coincide with those by FFT



(a) Pattern 1: relatively smooth time series



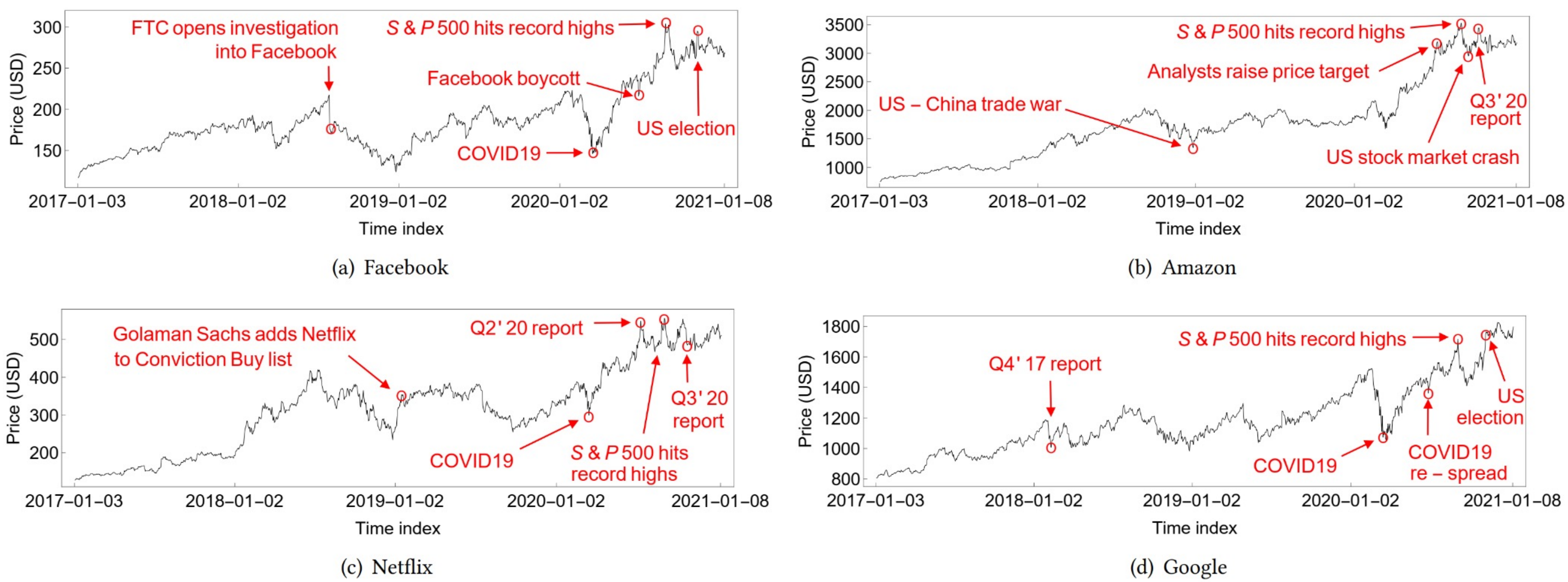
(b) Pattern 2: moderately oscillatory time series



(c) Pattern 3: highly oscillatory time series

Anomaly Detection (2)

- PFT results in interpretable anomaly detections
  - Stock prices of Facebook, Amazon, Netflix, and Google
  - Each outlier is closely related to a real-world event



(a) Facebook

(b) Amazon

(c) Netflix

(d) Google

Outline

- Introduction
- Existing Works
- Proposed Method
- Experiments
- **Conclusion**

Conclusion

- **PFT (Partial Fourier Transform)**
  - Efficiently computes a part of Fourier coefficients
- Main ideas of PFT
  - Polynomial approximation of smooth twiddle factors
  - Base exponential function for precomputation
  - Reordering operators
- Experimental results
  - PFT shows state-of-the-art speed without accuracy loss

Thank you!

https://github.com/snudatalab/PFT