Fast Multidimensional Partial Fourier Transform with Automatic Hyperparameter Selection

KDD 2024

Yong-chan Park, Jongjin Kim, and U Kang
Data Mining Lab
Dept. of CSE
Seoul National University

Outline

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

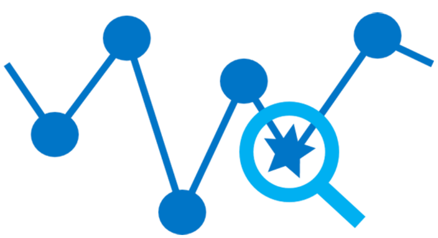
Fourier Transform

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

Signal processing

Data compression

Medical imaging

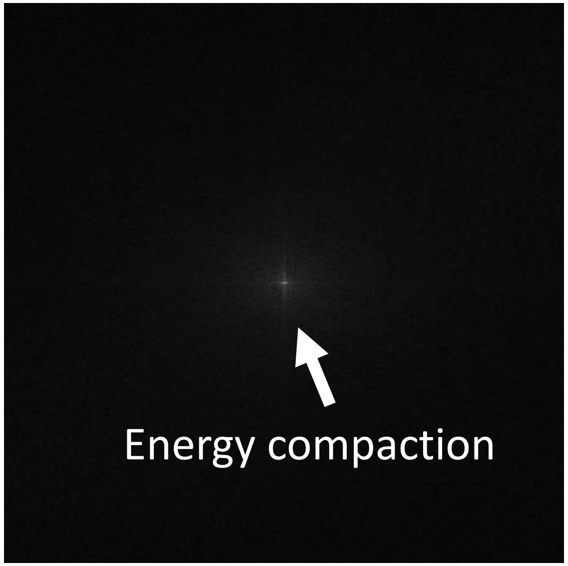


Anomaly detection

Motivation

- **Energy compaction** in the frequency domain
  - Fourier coefficients are mostly small or equal to zero

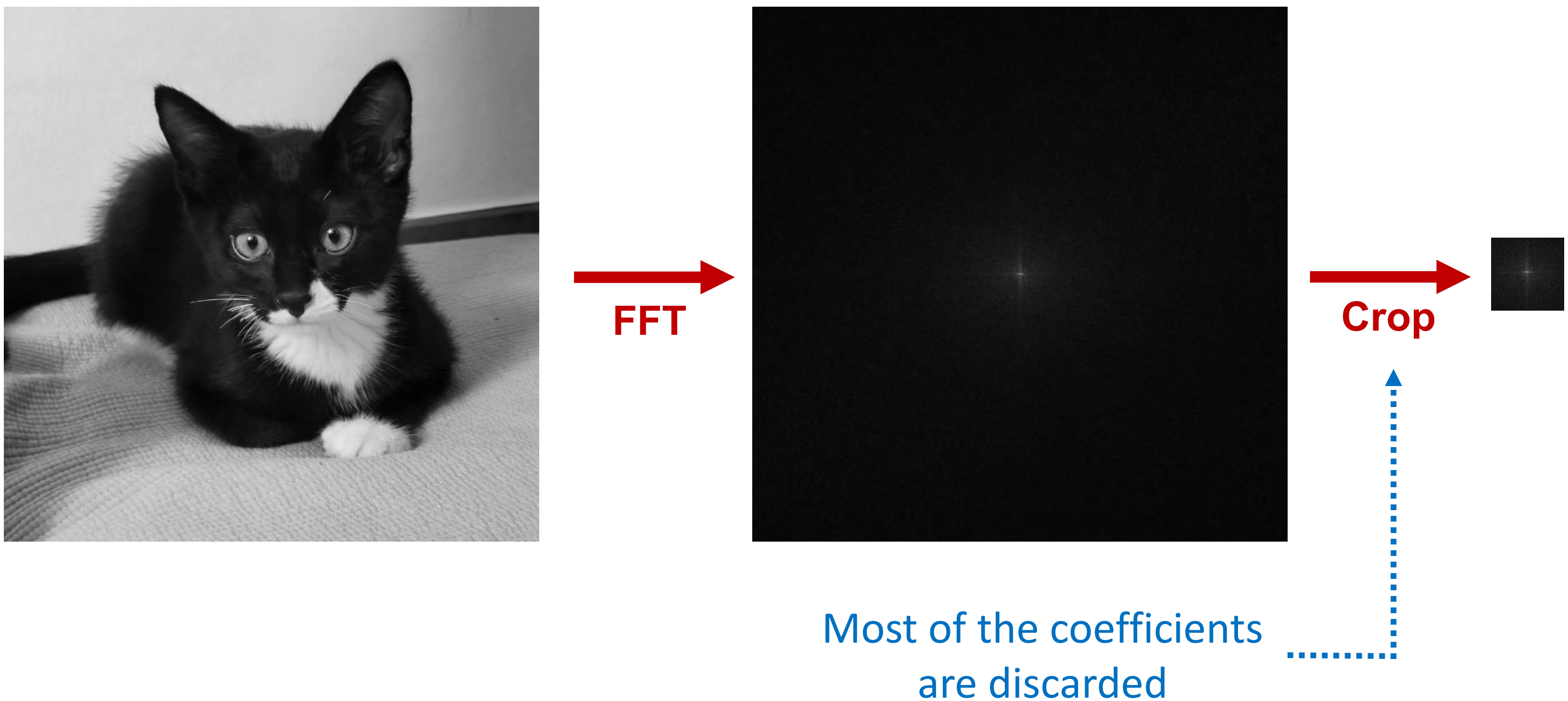
Spatial domain



Frequency domain

Motivation

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



Motivation

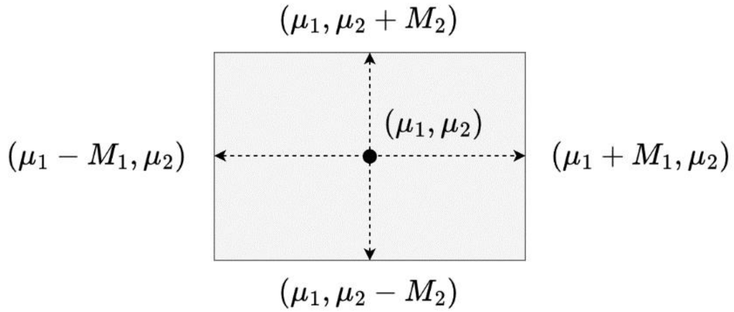
- *How can we optimize the computation process for a part of Fourier coefficients?*

How can we do this directly?

Problem Definition

- Multidimensional Partial Fourier Transform
  - **Given**
    - D-dimensional array $\boldsymbol{x} \in \mathbb{C}^{N_1 \times \cdots \times N_D}$
    - Center of box $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_D) \in \mathbb{Z}^D$
    - Radius of box $\boldsymbol{M} = (M_1, \cdots, M_D) \in \mathbb{Z}^D$
  - **Estimate**
    - Fourier coefficients of $\boldsymbol{x}$ for D-dimensional box

$$\Pi_{d=1}^{D}[\mu_d - M_d, \mu_d + M_d]$$



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: $O(N \log N)$ (N: input size)
- **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: $O(MN)$ (N, M: input / output sizes)
- **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: $O(N + M \log N)$ (N, M: input / output sizes)
- **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: $O(N \log M)$ (N, M: input / output sizes)
- **Limitation**
  - The performance gains are rather modest in practice

PFT

- **Partial Fourier Transform (PFT)**
  - Current state-of-the-art algorithm
  - Time complexity: $O(N + M \log M)$ (N, M: input / output sizes)
- **Limitation**
  - Only works for 1-dimensional inputs
  - Requires manual hyperparameter tuning

Outline

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

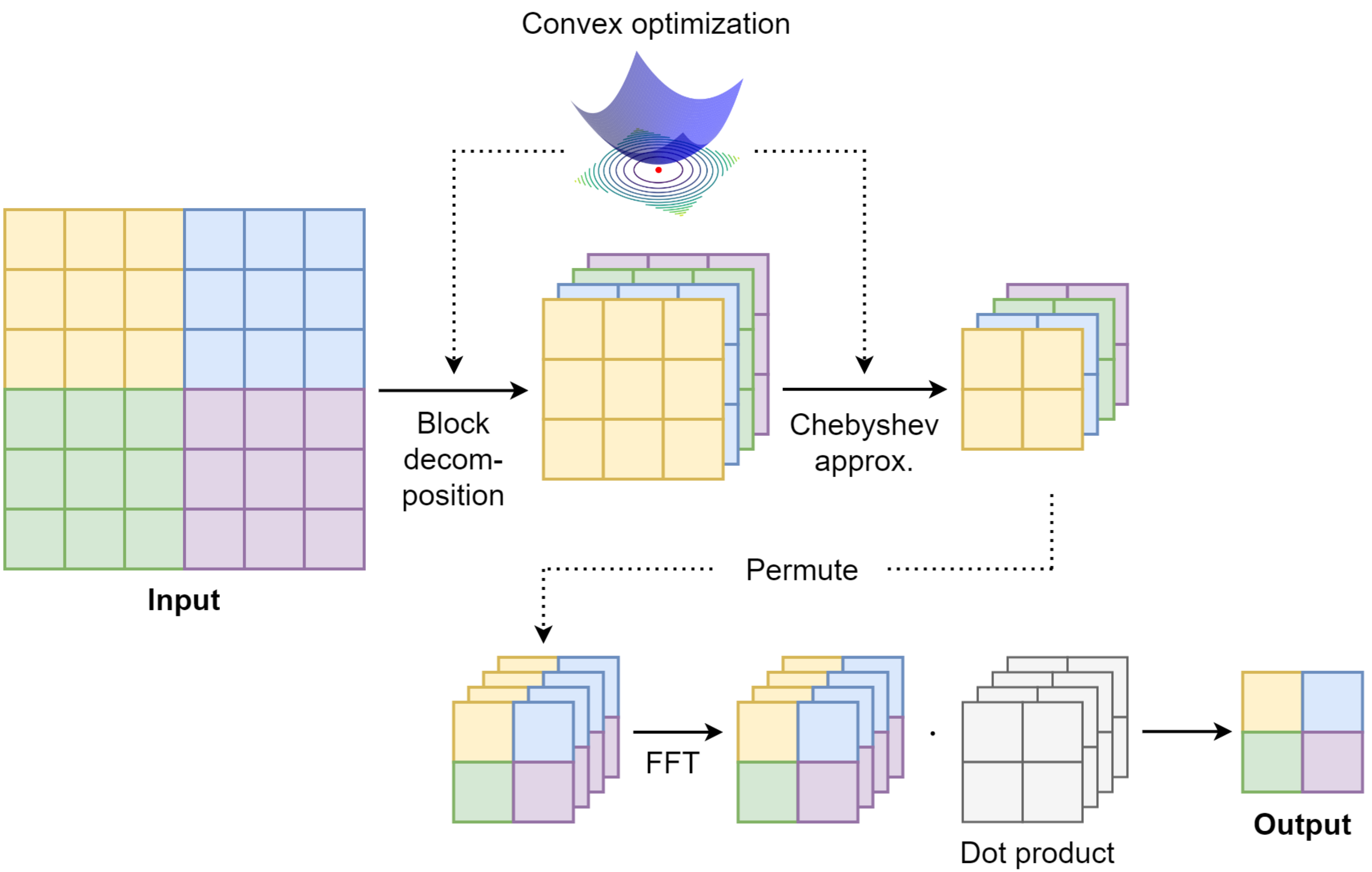
Proposed Method

- **Challenges**
  - **C1**. Multidimensional partial Fourier transform
    - How can we efficiently compute partial Fourier coefficients in multidimensional DFT?
  - **C2**. Automatic hyperparameter selection
    - How can we automatically find the optimal hyperparameter of Auto-MPFT?

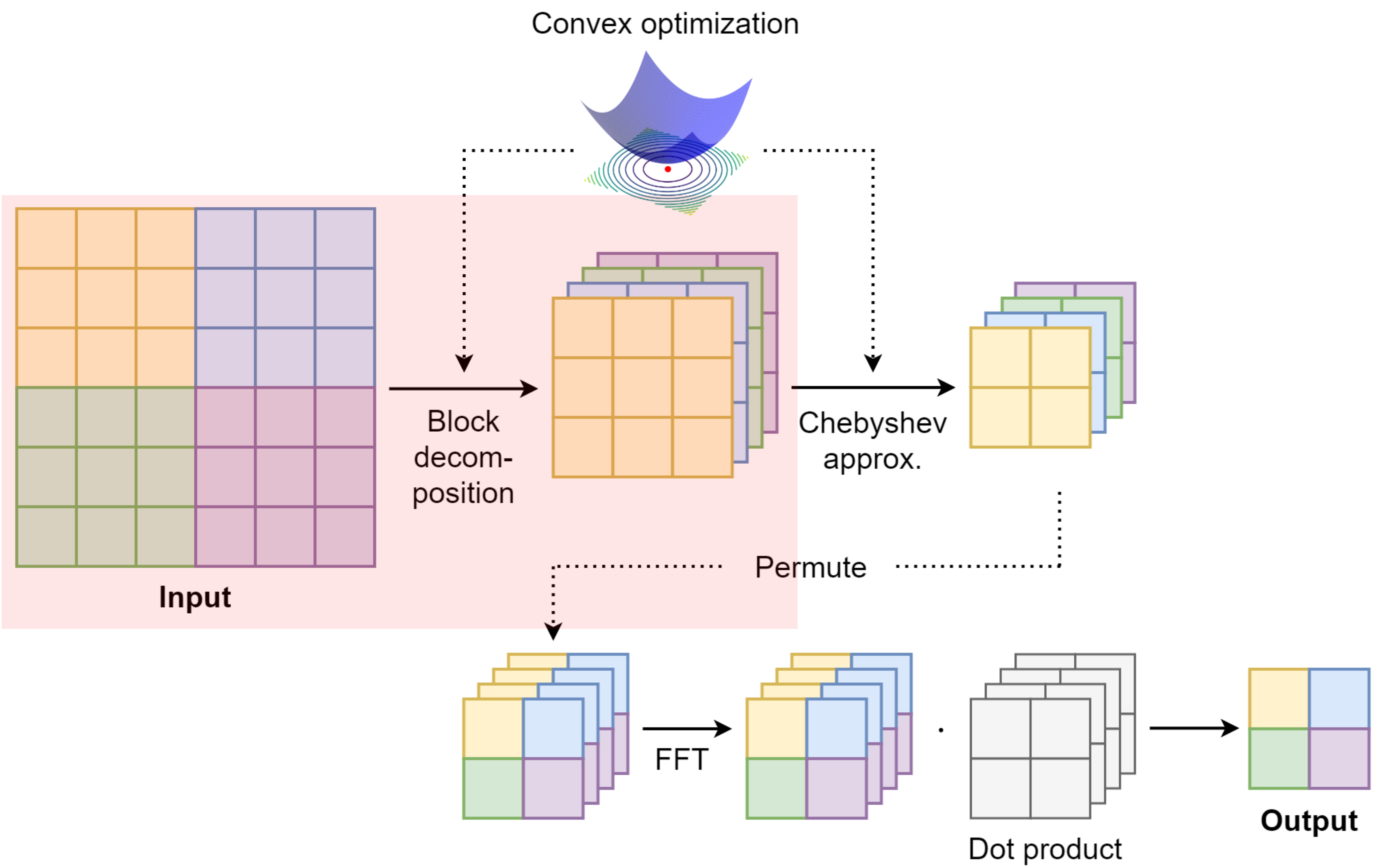
Proposed Method

- **Auto-MPFT**
  - Automatic Multidimensional Partial Fourier Transform
  - **(1) Multidimensional Partial Fourier Transform**
    - (1.1) Block decomposition
    - (1.2) Chebyshev approximation
    - (1.3) Batch FFT & dot product
  - **(2) Automatic Hyperparameter Selection**
    - (2.1) Finding time complexity
    - (2.2) Error approximation
    - (2.3) Fixed-point iteration
    - (2.4) Unconstrained convex optimization

Proposed Method



Proposed Method



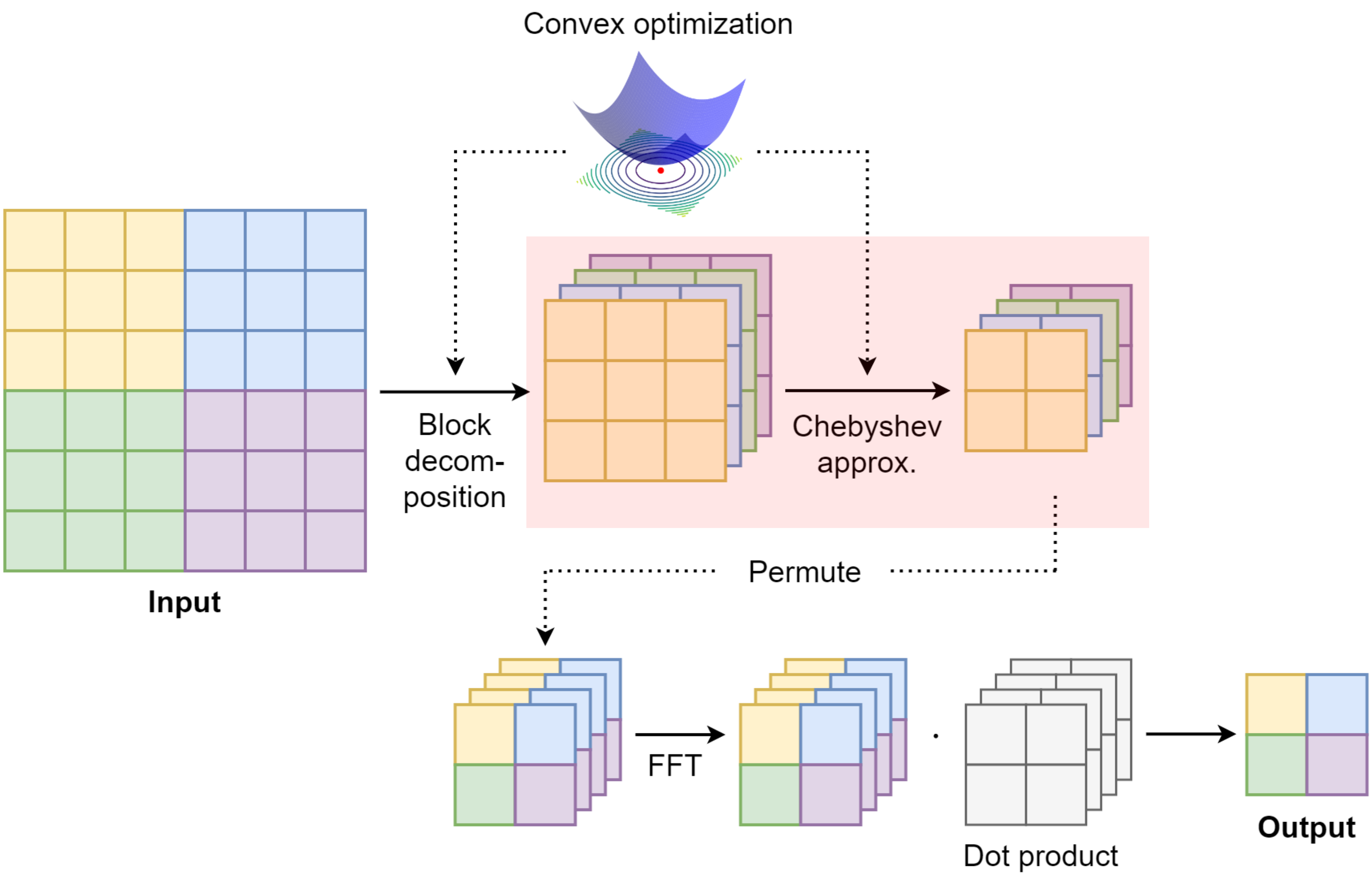
Proposed Method

- **(1) Multidimensional Partial Fourier Transform**
  - **(1.1) Block decomposition**
    - Goal: Find a set of smooth trigonometric factors

$$
\begin{aligned}
\hat{x}_{\boldsymbol{m}} &= \textstyle\sum_{\boldsymbol{n}\in\prod_d[N_d]} x_{\boldsymbol{n}} \prod_d \omega_{N_d}^{m_d n_d} \\
&= \textstyle\sum_{\boldsymbol{k}\in\prod_d[p_d],\boldsymbol{l}\in\prod_d[q_d]} x_{\boldsymbol{q}\odot\boldsymbol{k}+\boldsymbol{l}} \prod_d \omega_{N_d}^{m_d(l_d-q_d/2)} \omega_{p_d}^{m_d k_d} \omega_{2p_d}^{m_d}
\end{aligned}
$$

$d = 1, 2, \cdots, D$: dimension
$(x_{\boldsymbol{n}}) \in \mathbb{C}^{N_1\times\cdots\times N_D}$: D-dimensional array
$(\hat{x}_{\boldsymbol{m}}) \in \mathbb{C}^{N_1\times\cdots\times N_D}$: Fourier coefficient of x
$\omega_\nu \coloneqq e^{-2\pi i/\nu}$: ν-th primitive root of unity
$[\nu] \coloneqq \{0, 1, \cdots, \nu-1\}$
$\boldsymbol{p} = (p_1, \cdots, p_D), \boldsymbol{q} = (q_1, \cdots, q_D) \in \mathbb{Z}^D$ such that $N_d = p_d q_d$
$\odot$: element-wise product

Proposed Method



Proposed Method

- **(1) Multidimensional Partial Fourier Transform**
  - **(1.2) Chebyshev approximation**
    - Goal: Enable pre-computation of the trigonometric factors

$$\hat{x}_{\boldsymbol{m}} = \sum_{\boldsymbol{k},\boldsymbol{l}} x_{\boldsymbol{q}\odot\boldsymbol{k}+\boldsymbol{l}} \prod_d \omega_{N_d}^{m_d(l_d - q_d/2)} \omega_{p_d}^{m_d k_d} \omega_{2p_d}^{m_d}$$
$$\approx \sum_{\boldsymbol{j},\boldsymbol{k},\boldsymbol{l}} x_{\boldsymbol{q}\odot\boldsymbol{k}+\boldsymbol{l}} \prod_d b_{j_d l_d}^{(d)} \left((m_d - \mu_d)/p_d\right)^{j_d} \omega_{p_d}^{m_d k_d} \omega_{2p_d}^{m_d}$$

$\boldsymbol{j} \in \prod_d [r_d]$, $\boldsymbol{k} \in \prod_d [p_d]$, $\boldsymbol{l} \in \prod_d [q_d]$

$b_{j_d l_d}^{(d)} \coloneqq \omega_{N_d}^{\mu_d\left(l_d - \frac{q_d}{2}\right)} w_{\epsilon, r_d, j_d} \left(1 - \frac{2 l_d}{q_d}\right)^{j_d} \in \mathbb{C}^{r_d \times q_d}$

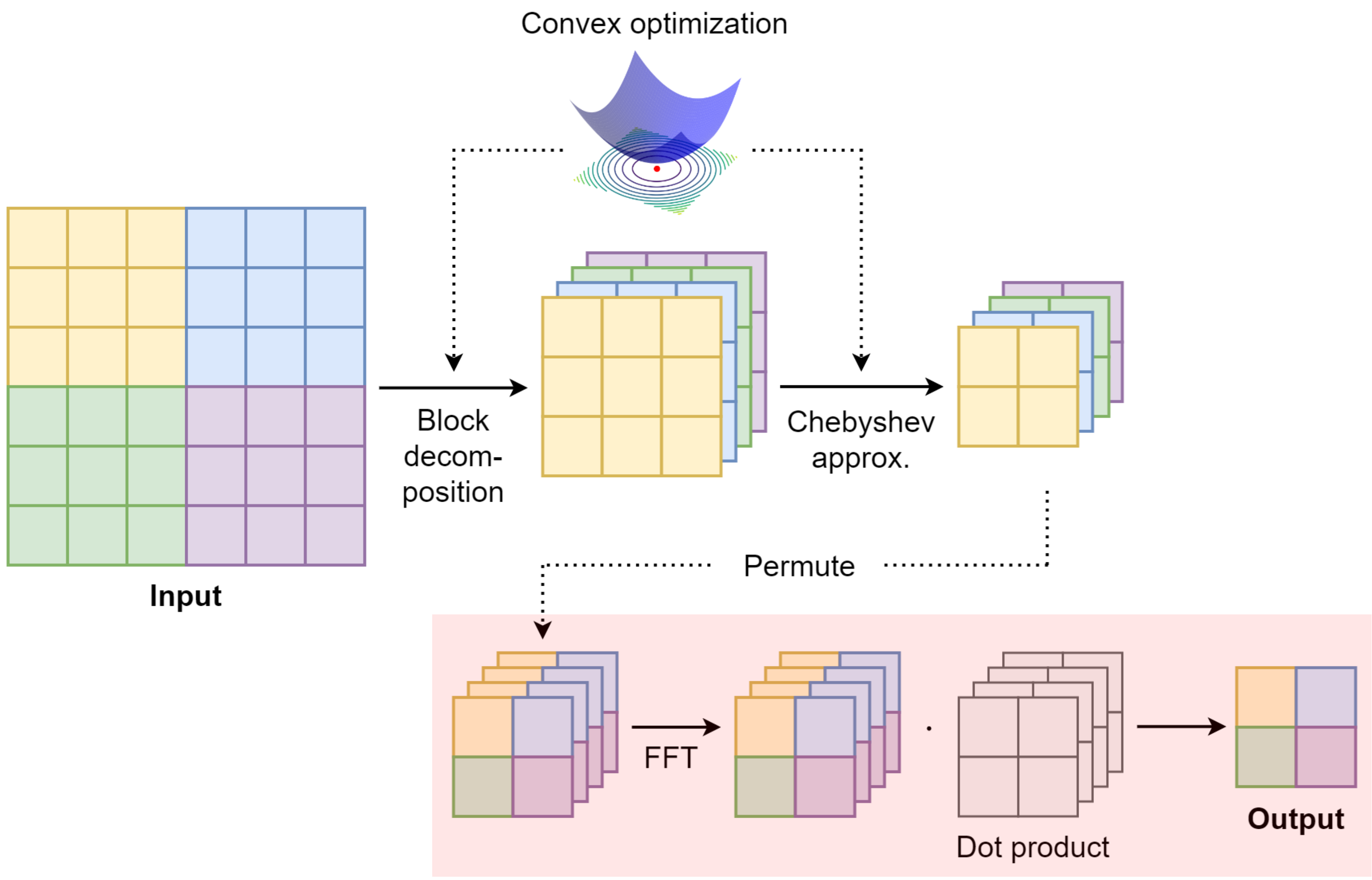
$\mathcal{P}_{\alpha,\xi}$: Chebyshev approximation to $e^{\pi i x}$ of degree less than α with restriction $|x| \leq |\xi|$

$\xi(\epsilon, r) \coloneqq \sup\{\xi \geq 0 : \left\|\mathcal{P}_{r,\xi}(x) - e^{\pi i x}\right\|_{|x| \leq |\xi|} \leq \epsilon\}$ (ϵ: tolerance)

r_d: number of approximating terms for d-th axis such that $\xi(\epsilon, r_d) \geq M_d / p_d$

w_{ϵ, r_d, j_d}: j_d-th coefficient of Chebyshev polynomial $\mathcal{P}_{r_d, \xi(\epsilon, r_d)}$

Proposed Method



Proposed Method

- **(1) Multidimensional Partial Fourier Transform**
  - **(1.3) Batch FFT & dot product**
    - Goal: Efficient computation

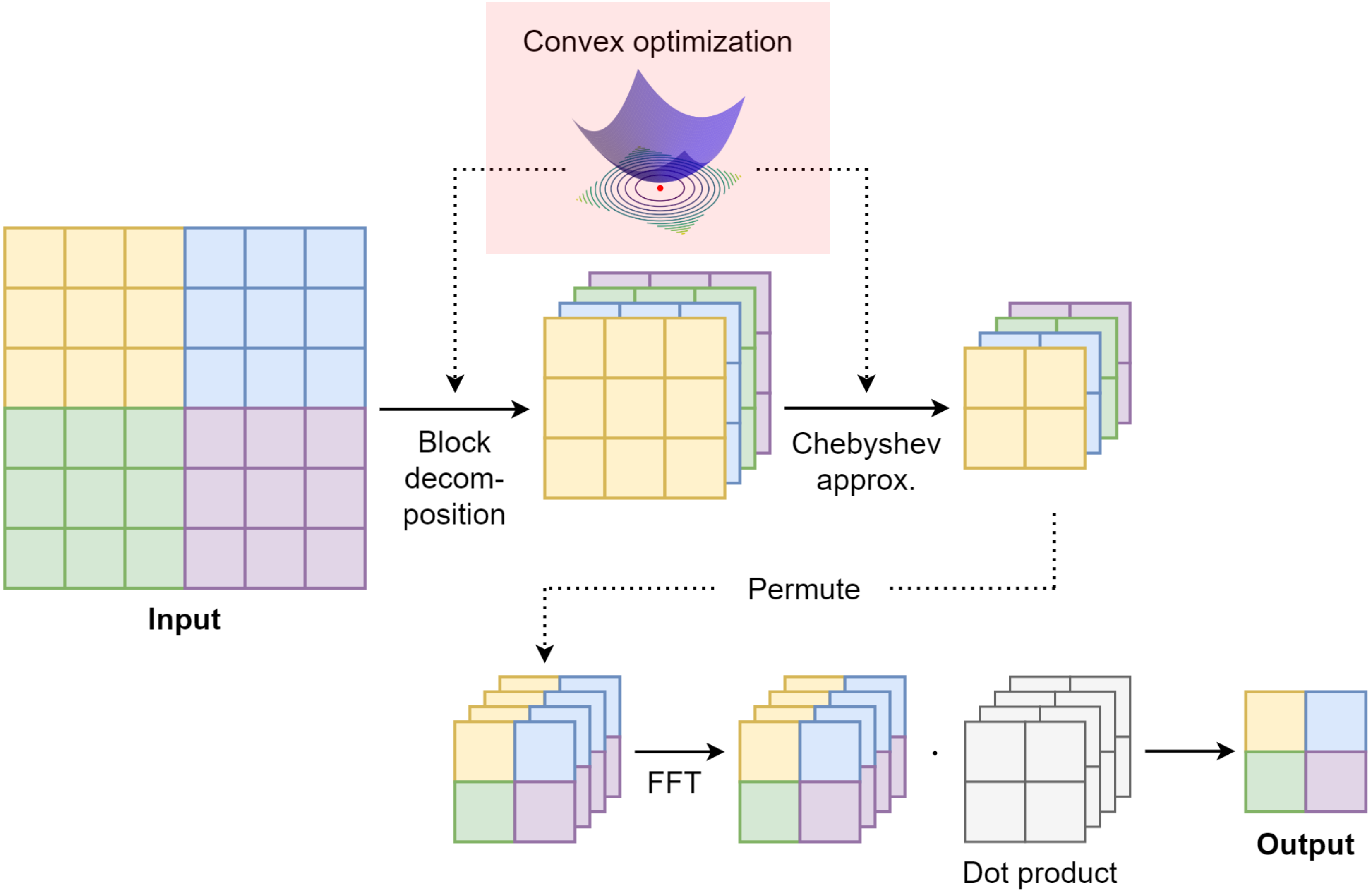
$$
\begin{aligned}
\hat{x}_{\boldsymbol{m}} &\approx \textstyle\sum_{\boldsymbol{j},\boldsymbol{k},\boldsymbol{l}} x_{\boldsymbol{q}\odot\boldsymbol{k}+\boldsymbol{l}} \prod_d b_{j_d l_d}^{(d)} \left((m_d-\mu_d)/p_d\right)^{j_d} \omega_{p_d}^{m_d k_d} \omega_{2p_d}^{m_d} \\
&= \textstyle\sum_{\boldsymbol{j},\boldsymbol{k}} c_{\boldsymbol{j}}^{(\boldsymbol{k})} \prod_d \left((m_d-\mu_d)/p_d\right)^{j_d} \omega_{p_d}^{m_d k_d} \omega_{2p_d}^{m_d} \\
&= \textstyle\sum_{\boldsymbol{j}} \hat{c}_{\boldsymbol{m}}^{(\boldsymbol{j})} \prod_d \left((m_d-\mu_d)/p_d\right)^{j_d} \omega_{2p_d}^{m_d}
\end{aligned}
$$

$\boldsymbol{j} \in \prod_d [r_d]$, $\boldsymbol{k} \in \prod_d [p_d]$, $\boldsymbol{l} \in \prod_d [q_d]$

$c_{\boldsymbol{j}}^{(\boldsymbol{k})} \coloneqq \sum_{\boldsymbol{l}} x_{\boldsymbol{q}\odot\boldsymbol{k}+\boldsymbol{l}} \prod_d b_{j_d l_d}^{(d)}$

$\hat{c}_{\boldsymbol{m}}^{(\boldsymbol{j})}$: D-dimensional discrete Fourier coefficient of $c_{\boldsymbol{j}}^{(\boldsymbol{k})}$ with respect to $\boldsymbol{k}$

Proposed Method



Proposed Method

- **(2) Automatic Hyperparameter Selection**
  - **(2.1) Finding time complexity**
    - Goal: Build an optimization problem for selecting the hyperparameter

Problem 1. Given $N, M \in \mathbb{N}$, and $\epsilon > 0$,

$$\begin{aligned} \underset{p,r>0}{\operatorname{argmin}} \quad & (N + p \log p + M)r \\ s.t. \quad & \xi(\epsilon, r) \geq M/p \end{aligned}$$

    - The objective function is the time complexity of Auto-MPFT
  - However, there is no closed form for the ξ function, making it difficult to apply optimization techniques

Proposed Method

- **(2) Automatic Hyperparameter Selection**
  - **(2.2) Error approximation**
    - Goal: Approximate the constraint function to closed form

LEMMA 2. *If $r \geq 2$, the approximation error function $\eta(r)$ satisfies*

$$\eta(r) \leq U(r) := \frac{2\sqrt{17}}{4-\sqrt{e}} \frac{C^r}{r!} e^{-\frac{C^2}{r+1}} \quad (C = c\pi/2).$$

$\eta(r)$: maximum error of the Chebyshev approximation to $e^{\pi i x}$ on $|x| \leq c := \xi(\epsilon, r^*)$ of degree less than r

$r^* := \min\{r \in \mathbb{N}: \xi(\epsilon, r) \geq M/p\}$

- If we solve $U(r) = \epsilon$, then $\eta(r) \leq U(r) = \epsilon = \eta(r^*)$, thus $r^* \sim \lfloor r \rfloor$.
- Unfortunately, $U(r) = \epsilon$ does not have an algebraic solution

Proposed Method

- **(2) Automatic Hyperparameter Selection**
  - **(2.3) Fixed-point iteration**
    - Goal: Estimate a numerical solution of $U(r) = \epsilon$
    - Define $f(x) \coloneqq (\alpha\epsilon r!)^{\frac{1}{r}} e^{\frac{x^2}{r(r+1)}}$, where $\alpha = \frac{4-\sqrt{e}}{2\sqrt{17}}$
    - Then, the equation $U(r) = \epsilon$ becomes computing a fixed point of f
    - Because f is a contraction mapping, the fixed-point iteration converges to the unique fixed point by the Banach fixed-point theorem
    - This leads to an approximate relation between the parameters p and r

$$p(r) \coloneqq \frac{\pi M}{2}(\alpha\epsilon r!)^{-\frac{1}{r}} e^{-\frac{1}{r(r+1)}(\alpha\epsilon r!)^{2/r}}, \quad \alpha = \frac{4-\sqrt{e}}{2\sqrt{17}}.$$

Proposed Method

- **(2) Automatic Hyperparameter Selection**
  - **(2.4) Unconstrained convex optimization**
    - Goal: Convert the original problem into a well-established task

Problem 2. Given $N, M \in \mathbb{N}$, and $\epsilon > 0$,

$$\operatorname*{argmin}_{r \geq 1} \, (N + p(r) \log p(r) + M)r$$

THEOREM 3. *The objective function $r \mapsto (N + p(r) \log p(r) + M)r$ of Problem 2 is convex for $r \geq 1$.*

  - The convexity of the objective function guarantees the convergence of second-order optimization techniques such as Newton's method

Outline

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

Experiments

- **Q1. Running time**
  - How rapidly does Auto-MPFT compute a part of Fourier coefficients compared to baselines without compromising accuracy?
- **Q2. Automatic hyperparameter selection**
  - How accurately and quickly does the optimization-based algorithm find the optimal hyperparameter of Auto-MPFT?
- **Q3. Impact of varying precision**
  - What impact does varying precision settings have on the running time of Auto-MPFT?

Experiments

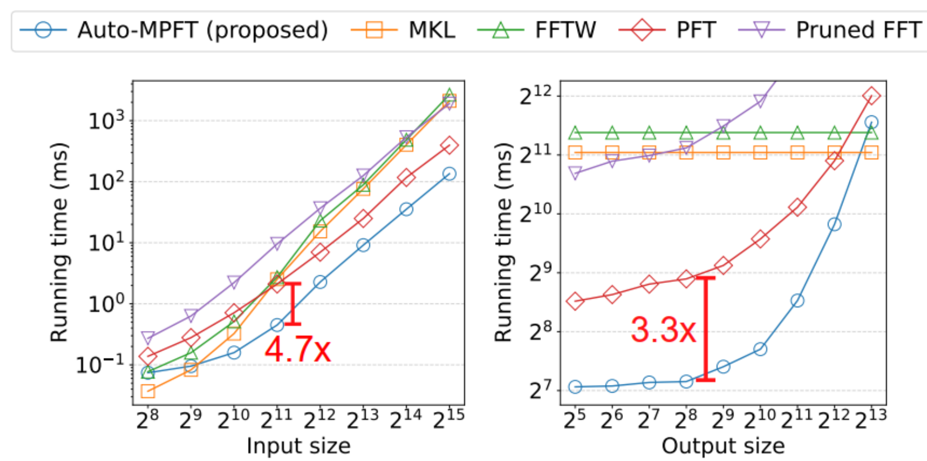
- **Dataset**
  - We use both synthetic and real-world datasets

Dataset	Type	# of Images	Size
$\{S_n\}_{n=8}^{15}$	Synthetic	1K	$2^n \times 2^n$
Cityscapes	Real-world	5K	2048×1024
ADE20K	Real-world	20K	2048×2048
DF2K	Real-world	3K	2040×1536
RiceLeaf	Real-world	3.3K	3120×3120
Bird	Real-world	306	6000×4000

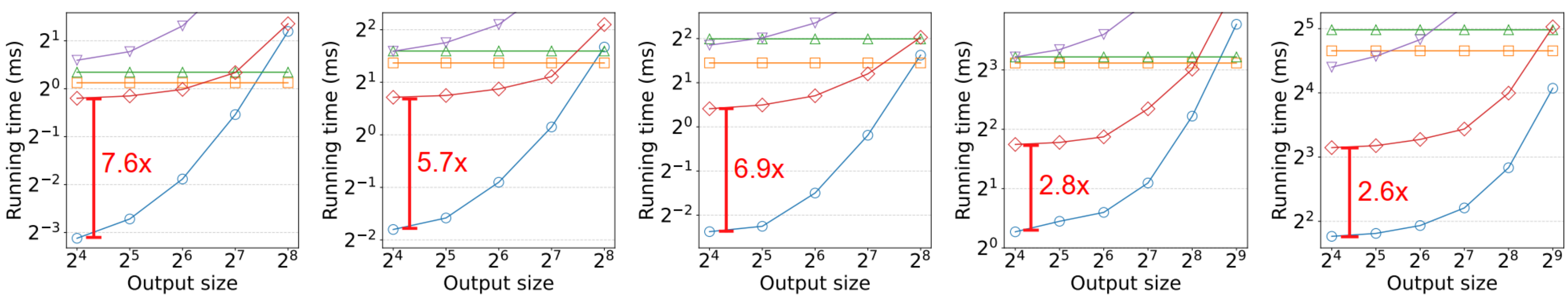
- **Measure**
  - We use single-precision floating-point format, and set the relative ℓ_2 error to be less than 10^{-6}

Experiments

- **Q1. Running time**
  - Auto-MPFT exhibits superior performance across all datasets without compromising accuracy



(a) Running time vs. input size (b) Running time vs. output size



(a) Cityscapes (b) ADE20K (c) DF2K (d) RiceLeaf (e) Bird

Experiments

- **Q2. Automatic hyperparameter selection**

- **Accuracy**
  - Auto-MPFT successfully detects the optimal value in most scenarios, with minor errors occurring infrequently

Output size	Input size							
	$2^{8\times2}$	$2^{9\times2}$	$2^{10\times2}$	$2^{11\times2}$	$2^{12\times2}$	$2^{13\times2}$	$2^{14\times2}$	$2^{15\times2}$
$2^6\times2^6$	2^4	2^5	2^5	2^6	2^6	2^7	$2^8(2^7)$	$2^9(2^8)$
$2^7\times2^7$	2^5	2^5	2^6	2^6	2^7	2^7	$2^8(2^7)$	2^9
$2^8\times2^8$	-	2^6	2^6	2^6	2^7	2^8	2^8	2^9
$2^9\times2^9$	-	-	2^7	2^7	2^7	2^8	2^9	2^9
$2^{10}\times2^{10}$	-	-	-	$2^7(2^8)$	2^8	2^8	2^9	2^9
$2^{11}\times2^{11}$	-	-	-	-	$2^8(2^9)$	2^9	2^9	2^{10}
$2^{12}\times2^{12}$	-	-	-	-	-	$2^9(2^{10})$	2^{10}	2^{10}
$2^{13}\times2^{13}$	-	-	-	-	-	-	$2^{10}(2^{11})$	2^{11}

- **Speed**
  - Auto-MPFT significantly outperforms the manual search process

$2^n\times2^n$	Auto-MPFT	Manual-best	Manual-worst
$2^9\times2^9$	**3.596**	63.30	860.9
$2^{10}\times2^{10}$	**3.431**	70.39	1273.3
$2^{11}\times2^{11}$	**2.829**	85.94	2083.7
$2^{12}\times2^{12}$	**2.225**	60.13	3638.6
$2^{13}\times2^{13}$	**1.653**	79.34	6707.4
$2^{14}\times2^{14}$	**1.626**	116.27	12508.7
$2^{15}\times2^{15}$	**2.971**	124.26	24113.0

Experiments

- **Q3. Impact of varying precision**
  - Auto-MPFT can set an arbitrary numerical precision, offering a beneficial trade-off, particularly when prioritizing fast evaluations

$(N = 2^{15} \times 2^{15})$

Output size	Precision		
	10^{-6}	10^{-4}	10^{-2}
$2^5 \times 2^5$	133.7	131.0 (2.0%)	126.7 (5.2%)
$2^6 \times 2^6$	135.1	132.3 (2.0%)	127.6 (5.6%)
$2^7 \times 2^7$	140.8	138.3 (1.8%)	133.5 (5.2%)
$2^8 \times 2^8$	142.3	139.6 (1.9%)	135.1 (5.1%)
$2^9 \times 2^9$	169.5	161.3 (4.9%)	148.1 (12.6%)
$2^{10} \times 2^{10}$	208.3	188.1 (9.7%)	158.8 (23.8%)
$2^{11} \times 2^{11}$	369.1	290.9 (21.2%)	201.1 (45.5%)
$2^{12} \times 2^{12}$	905.9	732.5 (19.1%)	463.0 (48.9%)
$2^{13} \times 2^{13}$	3012.5	2465.8 (18.1%)	1510.9 (49.8%)

Outline

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

Conclusion

- **Auto-MPFT**
  - Efficiently computes a part of Fourier coefficients
- **Main ideas**
  - Use block decomposition and Chebyshev approximation, significantly reducing the computational cost
  - Propose a convex optimization-based algorithm for automatically selecting the optimal hyperparameter
- **Experimental result**
  - Auto-MPFT shows the state-of-the-art performance

Thank you!

https://github.com/snudatalab/Auto-MPFT