Fast Multidimensional Partial Fourier Transform with Automatic Hyperparameter Selection

Yong-chan Park
Seoul National University
Seoul, Republic of Korea
wjdakf3948@snu.ac.kr

Jongjin Kim
Seoul National University
Seoul, Republic of Korea
j2kim99@snu.ac.kr

U Kang
Seoul National University
Seoul, Republic of Korea
ukang@snu.ac.kr

Abstract

Given a multidimensional array, how can we optimize the computation process for a part of Fourier coefficients? Discrete Fourier transform plays an overarching role in various data mining tasks. Recent interest has focused on efficiently calculating a small part of Fourier coefficients, exploiting the energy compaction property of real-world data. Current methods for partial Fourier transform frequently encounter efficiency issues, yet the adoption of pre-computation techniques within the PFT algorithm has shown promising performance. However, PFT still faces limitations in handling multidimensional data efficiently and requires manual hyperparameter tuning, leading to additional costs.

In this paper, we propose Auto-MPFT (Automatic Multidimensional Partial Fourier Transform), which efficiently computes a subset of Fourier coefficients in multidimensional data *without* the need for manual hyperparameter search. Auto-MPFT leverages multivariate polynomial approximation for trigonometric functions, generalizing its domain to multidimensional Euclidean space. Moreover, we present a convex optimization-based algorithm for automatically selecting the optimal hyperparameter of Auto-MPFT. We provide a rigorous proof for the explicit reformulation of the original optimization problem of Auto-MPFT, demonstrating the process that converts it into a well-established unconstrained convex optimization problem. Extensive experiments show that Auto-MPFT surpasses existing partial Fourier transform methods and optimized FFT libraries, achieving up to 7.6× increase in speed without sacrificing accuracy. In addition, our optimization algorithm accurately finds the optimal hyperparameter for Auto-MPFT, significantly reducing the cost associated with hyperparameter search.

CCS Concepts

• **Theory of computation** → **Numeric approximation algorithms**.

Keywords

Partial Fourier transform; Fast Fourier transform

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671667

ACM Reference Format:
Yong-chan Park, Jongjin Kim, and U Kang. 2024. Fast Multidimensional Partial Fourier Transform with Automatic Hyperparameter Selection. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671667

1 Introduction

Discrete Fourier transform (DFT) is a central algorithm in many data mining tasks, including anomaly detection [15, 22, 23, 31], latent pattern extraction [21, 25, 33], and image processing [8, 18, 27]. Recently, there has been a growing interest in efficiently calculating only a part of Fourier coefficients by leveraging the energy compaction property of data. A majority of real-world data, such as time series, image, and video, have a very compact representation in the frequency domain. As an illustration, Figure 1 visualizes the Fourier coefficients of natural images from ImageNet, where the coefficients are mostly equal to zero except in a few low-frequency parts. This indicates that one can gain significant computational benefits by focusing only on the part of non-zero coefficients and skipping the computation of unnecessary coefficients.

However, many existing methods for partial Fourier transform, such as Goertzel algorithm [6, 13], Subband DFT [14, 26], and Pruned FFT [2, 16, 17, 28, 30], suffer from low efficiency. Compared to the classic fast Fourier transform (FFT) computing the full coefficients, these methods outperform FFT only when the output has a much smaller size than the input, limiting their usage in practice. A recent work [20] proposes PFT, which leverages pre-computation techniques for partial Fourier transform, and raises the performance to a level that it can replace FFT in practical applications. However, the effectiveness of PFT is constrained by two major limitations. First, PFT is specialized for one-dimensional data, so it does not operate efficiently for multidimensional data. Although it is possible to apply PFT to each axis of multidimensional data, we theoretically and experimentally show that this approach is less efficient than algorithms specifically designed for multidimensional data. Second, PFT requires a manual hyperparameter tuning every time the size of the input or output changes. This leads to an extra cost of the method, especially when the data are multidimensional, because in that case the search space of the hyperparameter is also huge.

In this paper, we propose Auto-MPFT (Automatic Multidimensional Partial Fourier Transform), a fast and accurate algorithm that computes partial Fourier coefficients of multidimensional data *without* necessity for manual hyperparameter search. Auto-MPFT approximates a set of trigonometric factors in DFT using multivariate polynomials. Polynomial approximation enables Auto-MPFT to significantly reduce the computational cost by efficiently processing

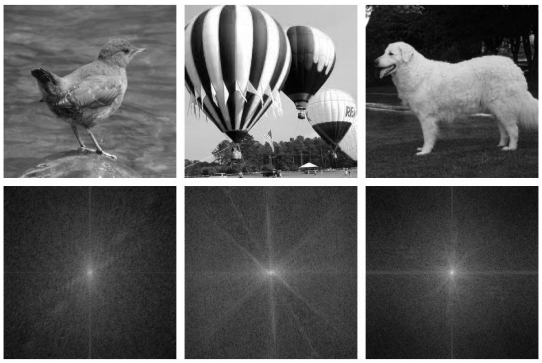


Figure 1: Examples of input images and their Fourier maps, where the Fourier coefficients are shown as log-amplitudes. Except for a few low-frequency parts (around the center), the Fourier coefficients are predominantly close to zero.

multidimensional data with matrix multiplications and multidimensional FFTs. Furthermore, we propose a convex optimization-based algorithm for automatically selecting the optimal hyperparameter of Auto-MPFT. The key of our method is to approximate the complicated constraint function in Auto-MPFT by deriving an explicit reformulation of the constraint function based on Chebyshev approximation [7]. We prove that it leads to an unconstrained convex optimization problem, which is efficiently solved by Newton's method. Extensive experiments show that Auto-MPFT outperforms the state-of-the-art partial Fourier transform methods as well as optimized FFT libraries, Intel MKL and FFTW, achieving a remarkable increase in speed without compromising accuracy. We also show that our convex optimization-based algorithm successfully finds the optimal hyperparameter of Auto-MPFT, significantly reducing the extra cost due to the hyperparameter search.

We summarize our contributions as follows:

- **Algorithm.** We present Auto-MPFT, an efficient method that computes a part of Fourier coefficients of multidimensional data.
- **Automatic hyperparameter selection.** We propose a convex optimization-based algorithm for automatic selection of the optimal hyperparameter of Auto-MPFT.
- **Performance.** We experimentally show that Auto-MPFT outperforms the state-of-the-art baselines without sacrificing accuracy.

We provide the source code and datasets used in our paper at **https://github.com/snudatalab/Auto-MPFT/**.

2 Related Works

We present an overview of different methods for computing partial Fourier coefficients, comparing them to our Auto-MPFT.

Fast Fourier transform. Fast Fourier transform (FFT) [5, 9, 19] is an efficient algorithm for rapidly computing the discrete Fourier transform (DFT), reducing the computational complexity from $O(N^2)$ to $O(N \log N)$, where N is an input size. While there are specialized algorithms for the partial Fourier transform, it is noteworthy that the FFT, designed for computing full coefficients, often proves to be a superior choice. This preference is due to the fact that FFT is a well-established and highly optimized algorithm over the years, making it not only straightforward to implement but also frequently outperforming specialized algorithms for partial Fourier transform. We conduct comprehensive theoretical and experimental comparisons between our proposed Auto-MPFT and FFT. Specifically, we show that Auto-MPFT significantly outperforms the FFT by an order of magnitude of speedup with comparable accuracy.

Goertzel algorithm. Goertzel algorithm [6, 13] is an early method for computing partial Fourier coefficients. It essentially mimics the process of computing individual coefficients one by one, entailing a computational complexity of $O(MN)$ for M coefficients in an input of size N. This indicates that Goertzel algorithm becomes advantageous over FFT only when the number of coefficients M is less than $\log N$. Although a few variants aimed at improving the Goertzel algorithm have been proposed [4], their performance gains are limited to a small constant factor. Consequently, these improvements do not significantly alter the algorithm's overall efficiency. Thus, the Goertzel algorithm is less favorable in scenarios where a considerable number of coefficients is required.

Subband DFT. Subband DFT [14, 26] breaks down the input data into smaller sub-blocks and efficiently approximates a part of Fourier coefficients by eliminating sub-blocks with low energy contributions, resulting in $O(N+M \log N)$ time complexity. Despite the computational advantages, Subband DFT suffers from low accuracy, consistently showing a large relative error of around 10^{-1} [14]. While Subband DFT may offer computational benefits, its accuracy limitation makes it less suitable for applications demanding precise and reliable results. Note that Auto-MPFT enables the evaluation of Fourier coefficients with arbitrary numerical precision.

Pruned FFT. Pruned FFT [2, 16, 17, 28, 30] is a modification of the standard FFT, designed for computing a subset of Fourier coefficients. In this method, operations in a flow graph are pruned by removing those that do not influence the specified range in the frequency domain, achieving $O(N \log M)$ time cost. However, the pruning strategy does not result in significant computational savings because it leads to increased complexity in maintaining the accuracy of the desired frequency range. Moreover, it is noteworthy that the standard FFT is significantly more optimized than Pruned FFT. In comparison to Pruned FFT, our Auto-MPFT is highly efficient as it directly leverages the standard FFT as a subroutine.

PFT. PFT [20] is the current state-of-the-art method for partial Fourier transform. The method uses a polynomial approximation technique for efficient computations of DFT, reducing the time complexity to $O(N+M \log M)$. However, there are two downsides to the method. First, PFT is designed specifically for one-dimensional inputs, making it less effective when applied to multidimensional data. For example, Figure 2 compares the application of PFT and Auto-MPFT to a two-dimensional input of size $S \times S$, where the goal is to efficiently compute $T \times T$ low-frequency coefficients. Because a multidimensional DFT is equivalent to applying multiple one-dimensional DFTs for each dimension, one can use multiple PFTs for each axis as in Figure 2. However, this approach requires

$$S \cdot (S + T \log T) + T \cdot (S + T \log T) \sim S^2 + ST \log T$$

costs, while Auto-MPFT conducts the same computation with only

$$S^2 + T^2 \log T^2 \sim S^2 + T^2 \log T$$

operations (see Section 3.3 for the proof), which is a significant computational gain since $T \ll S$.

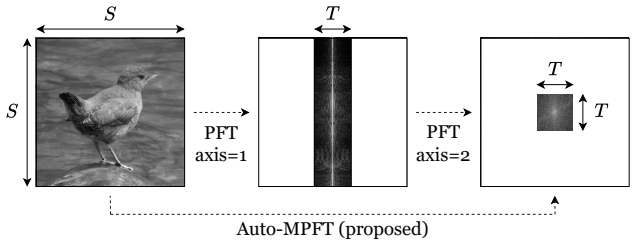


Figure 2: Comparison between PFT and Auto-MPFT for a two-dimensional input. While PFT is applied for each axis, Auto-MPFT is applied only once for the entire input and therefore requires much fewer operations.

Another disadvantage of PFT is its reliance on a manual hyperparameter search. Given an input of size N, a user must choose an appropriate divisor p of N to use the PFT algorithm. The overall performance of the algorithm varies greatly depending on the value of p, making it crucial to choose the optimal p. However, PFT does not provide an option to automatically find the optimal value, so in the worst case, one may need to go through trial and error for all divisors of N. The situation becomes even worse when the input is multidimensional because the search space grows exponentially with dimension. Note that Auto-MPFT addresses this problem by automatically finding the optimal value of p using a convex optimization-based algorithm, significantly reducing the extra cost due to the hyperparameter search.

3 Proposed Method

We propose Auto-MPFT, an efficient algorithm for partial Fourier transform of multidimensional data with automatic hyperparameter selection. The main challenges and our approaches are as follows:

(1) **How can we efficiently compute partial Fourier coefficients in multidimensional DFT?** We carefully modify the Cooley-Tukey algorithm to find a set of smooth twiddle factors in multidimensional DFT. We then approximate the twiddle factors using multivariate polynomials. This technique decomposes the computation of partial Fourier coefficients into matrix multiplications and multidimensional FFTs of small sub-blocks of the input, achieving significantly less time cost (Section 3.1).

(2) **How can we automatically find the optimal hyperparameter of Auto-MPFT?** The optimal hyperparameter is the value that minimizes the time complexity of Auto-MPFT. However, we find that the constraint function of such an optimization problem cannot be expressed as an explicit form. We tackle this issue by reformulating the constraint function via Chebyshev approximation and deriving an unconstrained convex optimization problem. This approach allows us to efficiently find the optimal hyperparameter using well-established numerical analysis such as Newton's method (Sections 3.2 and 3.3).

3.1 Multidimensional Partial Fourier Transform

We describe our proposed method in detail. The key ideas of Auto-MPFT for efficient computation of partial Fourier coefficients are as follows: (1) in the configuration phase, Auto-MPFT uses precomputation techniques via multivariate polynomial approximation of trigonometric functions in DFT. Moreover, it automatically finds the optimal hyperparameter and calculates the degree of the approximation polynomial (Algorithm 1). (2) In the computation phase, Auto-MPFT utilizes batch matrix multiplication and FFT algorithms optimized for multidimensional data types, which allows Auto-MPFT to yield theoretically and experimentally superior results compared to existing methods (Algorithm 2).

For a positive integer ν, let $\omega_\nu := e^{-2\pi i/\nu}$ be the ν-th primitive root of unity and $[\nu] := \{0, 1, \cdots, \nu-1\}$. Given a D-dimensional array $\mathscr{A} = (a_{\boldsymbol{n}}) \in \mathbb{C}^{N_1 \times \cdots \times N_D}$, the DFT of it is defined as follows:

$$\hat{a}_{\boldsymbol{m}} = \sum_{\boldsymbol{n} \in \prod_d [N_d]} a_{\boldsymbol{n}} \prod\nolimits_d \omega_{N_d}^{m_d n_d} \quad (1 \le d \le D), \tag{1}$$

where $\boldsymbol{n} = (n_1, \cdots, n_D), \boldsymbol{m} = (m_1, \cdots, m_D) \in \mathbb{Z}^D$ are input and output indices, respectively. Our goal is to compute the Fourier coefficients $\hat{a}_{\boldsymbol{m}}$ for $\boldsymbol{m}$ belonging to the D-dimensional box

$$\mathcal{B}_{\boldsymbol{\mu},\boldsymbol{M}} := \prod\nolimits_d [\mu_d - M_d, \mu_d + M_d],$$

where $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_D), \boldsymbol{M} = (M_1, \cdots, M_D) \in \mathbb{Z}^D$ are the center and radii of the box, respectively. We call the D-dimensional box $\mathcal{B}_{\boldsymbol{\mu},\boldsymbol{M}}$ a "target range." Now assume that for each $d = 1, \cdots, D$, we have $N_d = p_d q_d$, where $p_d, q_d > 1$. Let $\boldsymbol{p} = (p_1, \cdots, p_D) \in \mathbb{Z}^D$ and $\boldsymbol{q} = (q_1, \cdots, q_D) \in \mathbb{Z}^D$. The Cooley-Tukey algorithm [9] decomposes the summation (1) with $\boldsymbol{n} = \boldsymbol{q} \odot \boldsymbol{k} + \boldsymbol{l}$, where $\boldsymbol{k} \in \prod_d [p_d], \boldsymbol{l} \in \prod_d [q_d]$, and $\odot$ is the element-wise product:

$$\begin{aligned} \hat{a}_{\boldsymbol{m}} &= \sum\nolimits_{\boldsymbol{k},\boldsymbol{l}} a_{\boldsymbol{q}\odot\boldsymbol{k}+\boldsymbol{l}} \prod\nolimits_d \omega_{N_d}^{m_d(q_d k_d + l_d)} \\ &= \sum\nolimits_{\boldsymbol{k},\boldsymbol{l}} a_{\boldsymbol{q}\odot\boldsymbol{k}+\boldsymbol{l}} \prod\nolimits_d \omega_{N_d}^{m_d l_d} \omega_{p_d}^{m_d k_d}. \end{aligned} \tag{2}$$

Following the trick $\omega_{N_d}^{m_d l_d} = \omega_{N_d}^{m_d(l_d - q_d/2)} \cdot \omega_{2p_d}^{m_d}$ from [20], we rewrite (2) as follows:

$$\hat{a}_{\boldsymbol{m}} = \sum\nolimits_{\boldsymbol{k},\boldsymbol{l}} a_{\boldsymbol{q}\odot\boldsymbol{k}+\boldsymbol{l}} \prod\nolimits_d \omega_{N_d}^{m_d(l_d - q_d/2)} \omega_{p_d}^{m_d k_d} \omega_{2p_d}^{m_d}. \tag{3}$$

Note that $|l_d| < q_d$ and $|l_d - q_d/2| < q_d/2$, so the twiddle factors $\{\omega_{N_d}^{m_d(l_d - q_d/2)}\}$ is less oscillatory compared to $\{\omega_{N_d}^{m_d l_d}\}$, which allows a more accurate approximation via polynomials. To apply polynomial approximation for the set $\{\omega_{N_d}^{m_d(l_d - q_d/2)}\}$, we provide the following definitions. Let $\|\cdot\|_R$ be the uniform norm restricted to a set $R \subseteq \mathbb{R}$, that is, $\|f\|_R := \sup\{|f(x)| : x \in R\}$ and P_α be the set of polynomials on $\mathbb{R}$ of degree less than α.

Definition 3.1. Given a positive integer α and a non-zero real number ξ, we define $\mathcal{P}_{\alpha,\xi}$ as the best polynomial approximation to $e^{\pi i x}$ of degree less than α with the restriction $|x| \le |\xi|$:

$$\mathcal{P}_{\alpha,\xi} := \operatorname*{arg\,min}_{P \in P_\alpha} \|P(x) - e^{\pi i x}\|_{|x| \le |\xi|},$$

and $\mathcal{P}_{\alpha,\xi} := 1$ when $\xi = 0$. □

The uniqueness and existence of such polynomials are proved in [29]. For the computation of the best polynomial approximation, we use the Chebyshev approximation algorithm [10]. We opt to use the Chebyshev polynomials due to their solid theoretical foundations, including their widespread application in achieving optimal approximations with respect to the uniform norm and their contribution to the derivation of an error bound of Auto-MPFT

as we will describe in Section 3.2. Further investigation into other types of orthogonal polynomials could provide valuable insights and potentially improve the accuracy and efficiency of our proposed method, which we leave as an interesting future work.

Definition 3.2. Given a tolerance $\epsilon > 0$ and a positive integer r, we define $\xi(\epsilon, r)$ to be the scope about the origin such that the exponential function $e^{\pi i x}$ can be approximated by a polynomial of degree less than r with approximation bound ϵ:

$$\xi(\epsilon, r) := \sup\{\xi \geq 0 : \|\mathcal{P}_{r,\xi}(x) - e^{\pi i x}\|_{|x| \leq \xi} \leq \epsilon\}.$$

We express the best polynomial as $\mathcal{P}_{r,\xi(\epsilon,r)}(x) = \sum_{j \in [r]} w_{\epsilon,r,j} x^j$, where $w_{\epsilon,r,j}$ is the j-th coefficient of the polynomial. □

Given a tolerance $\epsilon > 0$, assume that we found a positive integer r_d that satisfies $\xi(\epsilon, r_d) \geq M_d/p_d$ for each d (the algorithm for finding r_d is demonstrated in Section 3.2). Then, for $\mu_d - M_d \leq m_d \leq \mu_d + M_d$, we decompose the twiddle factor $\omega_{N_d}^{m_d(l_d - q_d/2)}$ into $\omega_{N_d}^{\mu_d(l_d - q_d/2)} \omega_{N_d}^{(m_d - \mu_d)(l_d - q_d/2)}$ and approximate the second term by the polynomial $\mathcal{P}_{r_d,\xi(\epsilon,r_d)}(-2(m_d - \mu_d)(l_d - q_d/2)/N_d)$. Because $\xi(\epsilon, r_d) \geq M_d/p_d$, the approximation is valid for all m_d such that $|m_d - \mu_d| \leq |\frac{N_d}{2(l_d - q_d/2)} \cdot \frac{M_d}{p_d}| = |\frac{q_d}{2l_d - q_d} \cdot M_d|$, so in particular, $|m_d - \mu_d| \leq M_d$. Thus, we approximate (3) as follows:

$$\hat{a}_{\boldsymbol{m}} \approx \sum_{\boldsymbol{j},\boldsymbol{k},\boldsymbol{l}} a_{\boldsymbol{l}}^{(\boldsymbol{k})} \prod_d b_{j_d l_d}^{(d)} \omega_{p_d}^{m_d k_d} ((m_d - \mu_d)/p_d)^{j_d} \omega_{2p_d}^{m_d}, \quad (4)$$

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

$$\mathscr{A}^{(\boldsymbol{k})} := (a_{\boldsymbol{l}}^{(\boldsymbol{k})} = a_{\boldsymbol{q} \odot \boldsymbol{k} + \boldsymbol{l}}) \in \mathbb{C}^{q_1 \times \cdots \times q_D},$$

$$B^{(d)} := (b_{j_d l_d}^{(d)} = \omega_{N_d}^{\mu_d(l_d - q_d/2)} w_{\epsilon,r_d,j_d} (1 - 2l_d/q_d)^{j_d}) \in \mathbb{C}^{r_d \times q_d}.$$

In (4), the innermost summation $\sum_{\boldsymbol{l}} a_{\boldsymbol{l}}^{(\boldsymbol{k})} \prod_d b_{j_d l_d}^{(d)}$ can be written as a sequential d-mode product

$$\mathscr{A}^{(\boldsymbol{k})} \times_1 B^{(1)} \times_2 \cdots \times_D B^{(D)}. \quad (5)$$

Note that there are a total of $D!$ parenthesizations to compute (5). We precompute the optimal parenthesization in the configuration phase when $(\boldsymbol{N}, \boldsymbol{M}, \boldsymbol{\mu}, \epsilon)$ is given, so that we can bypass the parenthesization problem in the computation phase. Let us denote the result of (5) by $\mathscr{C}^{(\boldsymbol{k})} := (c_{\boldsymbol{j}}^{(\boldsymbol{k})}) \in \mathbb{C}^{r_1 \times \cdots \times r_D}$. For each $\boldsymbol{j}$, the operation $\sum_{\boldsymbol{k}} c_{\boldsymbol{j}}^{(\boldsymbol{k})} \prod_d \omega_{p_d}^{m_d k_d}$ is a D-dimensional DFT of size $\prod_d p_d$. Let $\hat{c}_{\boldsymbol{m}}^{(j)}$ be the Fourier coefficients of $c_{\boldsymbol{j}}^{(\boldsymbol{k})}$ with respect to $\boldsymbol{k}$. Then, we obtain the following estimation of $\hat{a}_{\boldsymbol{m}}$:

$$\hat{a}_{\boldsymbol{m}} \approx \sum_{\boldsymbol{j}} \hat{c}_{\boldsymbol{m}}^{(j)} \prod_d ((m_d - \mu_d)/p_d)^{j_d} \omega_{2p_d}^{m_d}. \quad (6)$$

The full computation is outlined in Algorithms 1 and 2.

3.2 Automatic Hyperparameter Selection

We propose a convex optimization-based algorithm for selecting the optimal hyperparameter of Auto-MPFT. The key idea is to convert the original optimization problem of Auto-MPFT (Problem 1) into an unconstrained convex optimization problem (Problem 2) by carefully approximating the constraint function.

Algorithm 1: Configuration phase of Auto-MPFT

input : Input size $\boldsymbol{N}$, output descriptors $\boldsymbol{M}$ and $\boldsymbol{\mu}$, and tolerance ϵ

output : Configuration results $B^{(d)}, p_d, q_d, r_d$ for all d, and optimal parenthesization

```
for d = 1, 2, ··· , D do
    Find the solution r_d of Problem 2 by Newton's method
    Find the nearest divisor p_d of N_d to p(r_d)
    q_d ← N_d / p_d
    r_d ← ⌊r_d⌋
    B^(d)[l, j] ← ω_{N_d}^{μ_d(l − q_d/2)} w_{ε,r_d,j} (1 − 2l/q_d)^j
end
Find the optimal parenthesization of Equation (5).
```

Algorithm 2: Computation phase of Auto-MPFT

input : Array $\boldsymbol{a}$ of size $\prod_d N_d$, output descriptors $\boldsymbol{M}$ and $\boldsymbol{\mu}$, and configuration results in Algorithm 1

output : Array $\hat{\boldsymbol{a}}$ of Fourier coefficients of $\boldsymbol{a}$ for $\mathcal{B}_{\boldsymbol{\mu},\boldsymbol{M}}$

```
A^(k)[l] ← a_{q⊙k+l} for k ∈ ∏_d [p_d] and l ∈ ∏_d [q_d]
C^(k) ← A^(k) ×_1 B^(1) ×_2 ··· ×_D B^(D) for k ∈ ∏_d [p_d]
Ĉ^(j)[·] ← FFT(C^(·)[j]) for j ∈ ∏_d [r_d]
for m ∈ B_{μ,M} do
    â[m] ← Σ_j Ĉ^(j)[m] ∏_d ((m_d − μ_d)/p_d)^{j_d} ω_{2p_d}^{m_d}
end
```

3.2.1 Building an Optimization Problem. Recall that the optimal hyperparameter is given by the minimizer of the time complexity of Auto-MPFT. Thus, we first need to derive the time cost function of our proposed method. Following convention, we consider only the computation phase for a time cost because the configuration phase contains only data-independent processes. For simplicity, we use the following notations: $N = \prod_d N_d$, $M = \prod_d M_d$, $p = \prod_d p_d$, $q = \prod_d q_d$, and $r = \prod_d r_d$, where $d = 1, 2, \cdots, D$.

The estimation (4) involves matrix multiplications (5) for each $\boldsymbol{k} \in \prod_d [p_d]$. Without loss of generality, we assume that the optimal parenthesization is given in the order $\times_1, \times_2, \cdots, \times_D$, which requires $O(r_1 q_1 \cdots q_D + r_1 r_2 q_2 \cdots q_D + \cdots + r_1 \cdots r_D q_D)$ operations. Then, the total cost of computing $\mathscr{C}^{(\boldsymbol{k})}$ for all $\boldsymbol{k}$ is given by $O(pqr) = O(Nr)$ since

$$\begin{aligned} &p(r_1 q_1 \cdots q_D + r_1 r_2 q_2 \cdots q_D + \cdots + r_1 \cdots r_D q_D) \\ &= p\left(\frac{qr}{r_2 \cdots r_D} + \frac{qr}{q_1 r_3 \cdots r_D} + \cdots + \frac{qr}{q_1 \cdots q_{D-1}}\right) \\ &\leq pqr(1 + 1/2 + \cdots + 1/2^{D-1}) < 2pqr, \end{aligned}$$

for $r_d \geq 1$ and $q_d \geq 2$. We next perform r FFTs of size p to calculate $\hat{c}_{\boldsymbol{m}}^{(j)}$, which takes $O(rp \log p)$ time. The remaining computation (6) requires $O(r)$ operations for each $\boldsymbol{m}$, giving an $O(Mr)$ running time. Hence, the time cost of Auto-MPFT can be written as

$$O((N + p \log p + M)r). \quad (7)$$

Now let us consider the objective function (7) for each dimension $d = 1, 2, \cdots, D$. Recall that N_d and M_d are input and output size descriptors, respectively, p_d is a positive divisor of N_d, and r_d is the number of approximating terms depending on given tolerance ϵ. Unfortunately, the variables p_d and r_d take discrete integer values, precluding directly using the continuous optimization methods. Moreover, the constraint that p_d divides N_d results in an irregular domain of p_d depending on the value of N_d. To tackle these problems, we slightly relax the constraints, extending the domain of p_d and r_d to the positive real numbers and discarding the necessity that p_d divides N_d. This leads to the following optimization problem (from now on we omit the subscript d for brevity):

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 challenge of this optimization problem is mainly due to the function ξ, which cannot be expressed in an explicit form. Thus, we propose reformulating the problem into an unconstrained convex optimization problem by approximating the constraint function.

3.2.2 Approximating Error. The main idea of our optimization process is to approximate the constraint function $\xi(\epsilon, r)$ in Problem 1 and derive an explicit reformulation of the optimization problem. Given a tolerance $0 < \epsilon < 1$, denote

$$r^* = \min\{r \in \mathbb{N} : \xi(\epsilon, r) \geq M/p\}, \tag{8}$$

and $c = \xi(\epsilon, r^*) \geq M/p$. Consider the best polynomial approximation to $e^{c\pi ix}$ on $|x| \leq 1$ (this is equivalent to approximating $e^{\pi ix}$ on $|x| \leq c$), and its Taylor series $e^{c\pi ix} = \sum_{n\geq 0}(c\pi ix)^n/n!$. For a non-negative integer n, the n-th power of x can be written as follows [7]: $x^n = \frac{1}{2^{n-1}}\left(T_n(x) + \binom{n}{1}T_{n-2}(x) + \binom{n}{2}T_{n-4}(x) + \cdots\right)$, where $T_n(x)$ is the Chebyshev polynomial of degree n (for even n, the coefficient of $T_0(x)$ is divided by 2). Then, we have

$$e^{c\pi ix} = \sum_{n\geq 0}\frac{(c\pi ix)^n}{n!} = \sum_{n\geq 0}\frac{(c\pi i)^n}{n!}\frac{1}{2^{n-1}}\sum_{k=0}^{\lfloor n/2\rfloor}\binom{n}{k}T_{n-2k}(x).$$

Dropping the T_{n-2k} terms for $n - 2k \geq r$ gives the Chebyshev approximation of degree less than r. Let $\eta(r)$ be the maximum error of the approximation, so that $\xi(\eta(r), r) = c$. Explicitly,

$$\eta(r) \coloneqq \max_{|x|\leq 1}\left|\sum_{n-2k\geq r}\frac{(c\pi i)^n}{n!}\frac{1}{2^{n-1}}\binom{n}{k}T_{n-2k}(x)\right|.$$

Substituting $n \leftarrow n + 2k$, we can rewrite this as

$$\eta(r) = \max_{|x|\leq 1}\left|\sum_{n\geq r}\sum_{k\geq 0}2i^n\left(\frac{c\pi}{2}\right)^{n+2k}\frac{(-1)^k}{k!(n+k)!}T_n(x)\right|.$$

We may express the involved terms using the Bessel function [1], $J_n(w) = \sum_{k\geq 0}\frac{(-1)^k}{k!(n+k)!}\left(\frac{w}{2}\right)^{n+2k}$, where n is a non-negative integer and $w \in \mathbb{R}$, which implies $\eta(r) = \max_{|x|\leq 1}\left|\sum_{n\geq r}2i^nJ_n(c\pi)T_n(x)\right|$. We now assume that the number r of approximating terms has a certain lower bound, namely $r \geq c\pi - 1$. This is a reasonable assumption due to the following lemma:

Lemma 1. *Given $w > 0$, the sequence $n \mapsto J_n(w)$ is strictly decreasing for $n \geq w - 1$ and converges to zero as n tends to ∞.* □

Proof. Let ν be an integer such that $\nu \geq w - 1$. We should show that $J_{\nu+1}(w) < J_\nu(w)$ holds. Since the Bessel function satisfies the recurrence relation

$$J_n(w) = \frac{2(n+1)}{w}J_{n+1}(w) - J_{n+2}(w), \quad \forall n \geq 0, \tag{9}$$

the inequality is equivalent to $J_{\nu+1}(w) < \frac{2(\nu+1)}{w}J_{\nu+1}(w) - J_{\nu+2}(w)$, or

$$J_{\nu+2}(w) < \left(\frac{2(\nu+1)}{w} - 1\right)J_{\nu+1}(w).$$

Replacing $J_{\nu+1}(w)$ using the recurrence relation again yields

$$J_{\nu+3}(w) < \left(\frac{2(\nu+2)}{w} - \frac{1}{\frac{2(\nu+1)}{w} - 1}\right)J_{\nu+2}(w).$$

In general, we obtain the following equivalent condition:

$$J_{\nu+s+1}(w) < M_s \cdot J_{\nu+s}(w),$$

where $M_0 = 1$ and $M_s = \frac{2(\nu+s)}{w} - \frac{1}{M_{s-1}}$ for $s \geq 1$. A simple induction shows that $M_s \geq 1$ for all s because

$$M_s = \frac{2(\nu+s)}{w} - \frac{1}{M_{s-1}} \geq \frac{2(\nu+1)}{w} - \frac{1}{M_{s-1}} \geq 2 - \frac{1}{M_{s-1}} \geq 1$$

provided that $M_{s-1} \geq 1$. Thus, it is sufficient to prove that there exists an integer $s \geq 0$ such that $J_{\nu+s+1}(w) < J_{\nu+s}(w)$. Now

$$J_n(w) = \frac{1}{n!}\left(\frac{w}{2}\right)^n\sum_{k\geq 0}\frac{n!}{k!(n+k)!}\left(-\frac{w^2}{4}\right)^k \sim \frac{1}{n!}\left(\frac{w}{2}\right)^n$$

for $n \gg w^2$, hence

$$\frac{J_{\nu+s+1}(w)}{J_{\nu+s}(w)} \sim \frac{1}{\nu+s+1}\left(\frac{w}{2}\right) < 1$$

for sufficiently large s. This completes the proof. □

In other words, the condition $r \geq c\pi - 1$ together with Lemma 1 ensures that the magnitude $|2i^nJ_n(c\pi)|$ of coefficients in the Chebyshev approximation gap strictly decreases. Thus, we can estimate the extreme points of the function $x \mapsto \sum_{n\geq r}2i^nJ_n(c\pi)T_n(x)$ by the extreme points x_k of the dominant term $T_r(x)$:

$$x_k = \cos(k\pi/r), \quad k = 0, 1, \cdots, r.$$

Furthermore, it is easy to check that the magnitude of extrema of $\sum_{n\geq r}2i^nJ_n(c\pi)T_n(x)$ peaks at around $x = 0$, which implies $\eta(r) \approx |\sum_{n\geq r}2i^nJ_n(c\pi)T_n(x_{\lfloor r/2\rfloor})|$, or

$$\eta(r) \approx \begin{cases}\left|\sum_{n\geq r}2i^nJ_n(c\pi)\cos(\pi n/2)\right| & r\text{: even}\\ \left|\sum_{n\geq r}2i^nJ_n(c\pi)\cos(\pi n(r-1)/2r)\right| & r\text{: odd}\end{cases} \tag{10}$$

using $T_n(\cos\theta) = \cos(n\theta)$. Our next goal is to prove that $\eta(r)$ is bounded above as in Lemma 2. We then use the upper bound to derive an approximate relation between the parameters p and r.

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

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

Proof. See Supplement A.1. □

3.2.3 Finding a Relation Between p and r. Assume that an integer $r_0 \geq 2$ satisfies the equation $U(r_0) = \epsilon$. Then

$$\eta(r_0) \leq U(r_0) = \epsilon = \eta(r^*),$$

where the last equality holds since $\xi(\epsilon, r^*) = c = \xi(\eta(r^*), r^*)$. Note that $\eta(r)$ is non-decreasing by definition, so $r^* \leq r_0$. This implies that solving the equation

$$U(r) = \epsilon \tag{11}$$

and rounding down the solution $\hat{r} \in \mathbb{R}$ gives an estimate of $r^* \sim \lfloor \hat{r} \rfloor$. Unfortunately, (11) does not have an algebraic solution due to the presence of factorial term. To address this problem, we employ the fixed-point iteration method to compute an approximate solution of the equation. Specifically, we consider U to be a function of C and find an implicit expression of r with respect to C. By this technique, we can derive an approximate relation $p(r)$ which denotes the value of p depending on r. Letting $\alpha = \frac{4-\sqrt{e}}{2\sqrt{17}}$, we write (11) as follows:

$$\frac{C^r}{\alpha r!} e^{-\frac{C^2}{r+1}} = \epsilon \iff C^r = \alpha\epsilon r! \cdot e^{\frac{C^2}{r+1}} \iff C = (\alpha\epsilon r!)^{\frac{1}{r}} e^{\frac{C^2}{r(r+1)}}.$$

Define $f(x) \coloneqq (\alpha\epsilon r!)^{\frac{1}{r}} e^{\frac{x^2}{r(r+1)}}$ for $x \in \mathbb{R}$. Then, the problem becomes computing a fixed point of f. Since $f'(x) = \frac{(\alpha\epsilon r!)^{1/r}}{r(r+1)} 2x e^{\frac{x^2}{r(r+1)}}$, we have $f'(0) = 0$. Moreover, $f'(C) = \frac{2C}{r(r+1)} f(C) = \frac{2C^2}{r(r+1)}$ provided that C is a fixed point of f. It follows from $C \leq (r+1)/2$ and $r \geq 2$ that $f'(C) = \frac{2C^2}{r(r+1)} \leq \frac{(r+1)^2}{2r(r+1)} = \frac{r+1}{2r} \leq \frac{3}{4}$. Because $f'(x)$ is non-decreasing for $x \geq 0$, there exist $L < 1$ and $\delta > 0$ such that $|f'(x)| \leq L, \forall x \in (-\delta, C+\delta)$. This implies that f is a contraction mapping function on $(-\delta, C+\delta)$, so the fixed-point iteration

$$C_0 \in (-\delta, C+\delta), \quad C_{n+1} = f(C_n), \quad n = 0, 1, 2, \cdots$$

converges to the unique fixed point C by the Banach fixed-point theorem [3]. We set $C_0 = 0$ and estimate C by the result of the second iteration of the algorithm:

$$C \sim C_2 = f(f(0)) = f((\alpha\epsilon r!)^{\frac{1}{r}}) = (\alpha\epsilon r!)^{\frac{1}{r}} e^{\frac{1}{r(r+1)}(\alpha\epsilon r!)^{2/r}}.$$

We now assume that $c \sim M/p$, which is reasonable due to the definition of r^* in (8). This leads to the following approximate relation between the parameters p and r:

$$p \sim \frac{M}{c} = \frac{\pi M}{2} C^{-1} \sim \frac{\pi M}{2} (\alpha\epsilon r!)^{-\frac{1}{r}} e^{-\frac{1}{r(r+1)}(\alpha\epsilon r!)^{2/r}}.$$

3.2.4 Convexity of the Objective Function. We have shown that the parameter p can be expressed in terms of r with the relation,

$$p(r) \coloneqq \frac{\pi M}{2} (\alpha\epsilon r!)^{-\frac{1}{r}} e^{-\frac{1}{r(r+1)}(\alpha\epsilon r!)^{2/r}}, \quad \alpha = \frac{4-\sqrt{e}}{2\sqrt{17}}. \tag{12}$$

By employing this relation, we reduce the objective function of Problem 1 into a functional form dependent on only r, removing the inequality constraint:

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

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

In the following theorem, we prove that the objective function in this problem is convex for $r \geq 1$. Consequently, Problem 2 becomes an unconstrained convex optimization problem.

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

Proof. See Supplement A.2. □

The convexity of the objective function guarantees the convergence of second-order optimization techniques such as Newton's method. After the optimal solution r^* that minimizes the objective function is found, we use the function $p(r)$ to approximate the optimal $p^* = p(r^*)$ and select the nearest divisor of N to p^*, which replaces the manual selection process. The automatic configuration phase of Auto-MPFT is outlined in Algorithm 1.

3.3 Theoretical Analysis

We present theoretical analysis on the time and space complexities of Auto-MPFT and its approximation bound.

3.3.1 Time Complexity. In Section 3.2.1, we have already seen that the time cost of Auto-MPFT can be expressed as $O((N + p \log p + M)r)$. Note that $N = \prod_d N_d$ is an input size, $M = \prod_d M_d$ is an output size, $p = \prod_d p_d$, and $r = \prod_d r_d$, where p_d is a divisor of N_d and r_d is an approximation order given a tolerance ϵ. However, the values of p and r in the above time cost are internally determined by the configuration phase (Algorithm 1). As a result, users cannot directly find out these values, making it inconvenient to use the time cost function. To address this limitation, we transform the time cost into a functional form that depends only on N, M, and ϵ, which are the inputs of Auto-MPFT. We first present the following two lemmas and employ them for the proof of Theorem 6.

Lemma 4. *For each $d = 1, 2, \cdots, D$, we have the asymptotic equation $r_d = O(\log(1/\epsilon))$.* □

Proof. Recall that by Equation (11), we have the following asymptotic equation between ϵ and r_d:

$$\epsilon \sim U(r_d) = \frac{2\sqrt{17}}{4-\sqrt{e}} \frac{C^{r_d}}{r_d!} e^{-\frac{C^2}{r_d+1}}.$$

Then there exist $B_1, B_2 > 0$ such that for sufficiently large r_d,

$$\epsilon < \frac{B_1 \cdot C^{r_d}}{r_d!} e^{-\frac{C^2}{r_d+1}} < \frac{B_1 \cdot C^{r_d}}{r_d!} < \frac{B_2}{e^{r_d}}.$$

It follows that $e^{r_d} < B_2/\epsilon$, and thus $r_d = O(\log(1/\epsilon))$. □

Lemma 5. *If N_d is b_d-smooth for some $b_d \geq 2$ (that is, none of prime factors of N_d is greater than b_d) and $1 \leq M_d \leq N_d$, then there exists a divisor p_d of N_d satisfying $p_d = \Theta(M_d)$.* □

Proof. Following the proof of Theorem 3 of [20], we prove that there exists a divisor p_d of N_d such that $M_d/\sqrt{b_d} \leq p_d < \sqrt{b_d} M_d$. Suppose that none of N_d's divisors belongs to $[M_d/\sqrt{b_d}, \sqrt{b_d} M_d)$. Let $1 = p_1 < p_2 < \cdots < p_n = N_d$ be the enumeration of all positive divisors of N_d in increasing order. It is clear that $p_1 < \sqrt{b_d} M_d$ and $M_d/\sqrt{b_d} < p_n$ since $b_d \geq 2$ and $1 \leq M_d \leq N_d$. Then, there exists an $i \in \{1, 2, \cdots, n-1\}$ so that $p_i < M_d/\sqrt{b_d}$ and $p_{i+1} \geq \sqrt{b_d} M_d$. Since N_d is b_d-smooth and $p_i < N_d$, at least one of $2p_i, 3p_i, \cdots, b_d p_i$ must be a divisor of N_d. However, this is a contradiction because we have $p_{i+1}/p_i > (\sqrt{b_d} M_d)(M_d/\sqrt{b_d})^{-1} = b_d$, so none of $2p_i, 3p_i, \cdots, b_d p_i$ can be a divisor of N_d, which completes the proof. □

Theorem 6. *For $d = 1, 2, \cdots, D$, let N_d be b_d-smooth for some $b_d \geq 2$. Then the time complexity of Auto-MPFT has an asymptotic upper bound $O((N + M \log M)(\log(1/\epsilon))^D)$.* □

Proof. It follows that $r_d = O(\log(1/\epsilon))$ by Lemma 4, which gives an upper bound for $r = \prod_d r_d = O((\log(1/\epsilon))^D)$. Using Lemma 5, we can find a divisor p_d of N_d such that $p_d = \Theta(M_d)$ for all d, resulting in $p = \prod_d p_d = \Theta(\prod_d M_d) = \Theta(M)$. Replacing p and r in the original upper bound of the time complexity yields

$$O((N + p \log p + M)r) = O((N + M \log M)(\log(1/\epsilon))^D),$$

hence the proof. □

3.3.2 Space Complexity. DFT is frequently required to be deployed on devices with limited performance capabilities. Thus, it is essential to consider memory constraints when implementing the algorithm in real-world applications. In Theorem 7, we prove that the space complexity of Auto-MPFT is asymptotically bounded by the sum of the input and output sizes, $N + M$. This indicates that Auto-MPFT is a space-optimal algorithm, requiring only the minimum space necessary for input and output.

Theorem 7. *Auto-MPFT is space-optimal, that is, the space complexity of it has an asymptotic upper bound $O(N + M)$.* □

Proof. In the configuration phase (Algorithm 1), Auto-MPFT stores matrices of size $r_d \times q_d$ $(d = 1, \cdots, D)$ for multivariate polynomial approximation, which requires $O(\sum_d r_d q_d)$ space cost. In the computation phase (Algorithm 2), we need $O(N)$ space to read an input data array (line 1 in Algorithm 2), $O(pr)$ space for polynomial approximation results (line 2 in Algorithm 2), and $O(M)$ space to save an output (lines 4-6 in Algorithm 2), which sum up to $O(N + pr + M)$. Typically, the number r_d is much smaller than p_d and q_d if $N_d = p_d q_d$ is sufficiently large, so we may assume that (1) $O(\sum_d r_d q_d) = o(\sum_d p_d q_d) = o(\sum_d N_d) = o(N)$ and (2) $O(pr) = o(pq) = o(N)$. These lead to a total of

$$O(\textstyle\sum_d r_d q_d + N + pr + M) = O(N + M)$$

space complexity of Auto-MPFT. □

3.3.3 Approximation Bound. We present a theoretical bound for the approximation of the polynomial $\mathcal{P}$. The estimated Fourier coefficient of $\boldsymbol{a}$ is denoted as $\mathcal{E}(\hat{\boldsymbol{a}})$. According to Theorem 8, the approximation bound within the target range is contingent on the data-specific total weight $\|\boldsymbol{a}\|_1$ of the original array and the given tolerance ϵ, where $\|\cdot\|_1$ represents the ℓ_1 norm. It is important to note that ϵ influences the number r of approximating terms (see Lemma 4), consequently affecting the error bound $\|\hat{\boldsymbol{a}} - \mathcal{E}(\hat{\boldsymbol{a}})\|_{\mathcal{B}_{\boldsymbol{\mu},\boldsymbol{M}}}$. This shows that, by adjusting ϵ accordingly, Fourier coefficients can be computed with arbitrary numerical precision using Auto-MPFT.

Theorem 8. *Given a sufficiently small tolerance $\epsilon > 0$, the estimated Fourier coefficient $\mathcal{E}(\hat{\boldsymbol{a}})$ in (4) satisfies*

$$\|\hat{\boldsymbol{a}} - \mathcal{E}(\hat{\boldsymbol{a}})\|_{\mathcal{B}_{\boldsymbol{\mu},\boldsymbol{M}}} \leq \|\boldsymbol{a}\|_1 \cdot (2D - 1)\epsilon,$$

where $\|\cdot\|_R$ denotes the uniform norm restricted to set $R \subseteq \mathbb{R}^D$. □

Proof. See Supplement A.3. □

Table 1: Summary of datasets.

Dataset	Type	# of Images	Size
$\{S_n\}_{n=8}^{15}$	Synthetic	1K	$2^n \times 2^n$
Cityscapes[1]	Real-world	5K	2048×1024
ADE20K[2]	Real-world	20K	2048×2048
DF2K[3]	Real-world	3K	2040×1536
RiceLeaf[4]	Real-world	3.3K	3120×3120
Bird[5]	Real-world	306	6000×4000

4 Experiments

Through experiments, we answer the following questions:

- **Q1 Running time (Section 4.2).** How rapidly does Auto-MPFT compute a part of Fourier coefficients compared to baselines without compromising accuracy?
- **Q2 Automatic hyperparameter selection (Section 4.3).** How accurately and quickly does the optimization-based algorithm find the optimal hyperparameter of Auto-MPFT?
- **Q3 Impact of varying precision (Section 4.4).** What impact does varying precision settings have on the runtime of Auto-MPFT?

4.1 Experimental Setup

Machine. Our system utilizes an Intel Core i7-10700KF @ 3.80GHz processor paired with 32GB of RAM.

Datasets. Although our proposed method can be applied to any multidimensional data, we focus our experiments on two-dimensional image datasets for clarity of presentation. Table 1 summarizes the datasets used in our experiments. For $n = 8, \cdots, 15$, S_n contains 1,000 matrices of size $2^n \times 2^n$ with elements being random real numbers ranging from 0 to 1. Cityscapes and ADE20K offer a wide range of indoor and outdoor scene images with detailed scene segmentation labels, supporting semantic segmentation research. DF2K is an image dataset for image super-resolution and restoration tasks which is composed of around 3,000 2k resolution images. RiceLeaf contains about 3,300 4k images of rice leaves for rice disease detection. Bird is a dataset for bird species classification task and contains 306 high-resolution images of birds. Note that the images in each dataset were resized to the same resolution.

Baselines. We compare Auto-MPFT with PFT and Pruned FFT as well as optimized FFT libraries, FFTW and Intel Math Kernel Library. All of the methods are implemented using C++.

(1) **FFTW**: FFTW[6] [11, 12] is among the fastest publicly available FFT implementations with hardware-specific optimizations. We employ the optimized version of FFTW 3.3.5, without including the pre-processing for configuration as the run-time cost.
(2) **MKL**: Intel Math Kernel Library[7] (MKL), known for its optimized mathematical functions such as FFT, often outperforms FFTW in terms of running time. We conduct all the experiments using an Intel processor to ensure optimal performance.

[1] https://www.cityscapes-dataset.com/
[2] https://groups.csail.mit.edu/vision/datasets/ADE20K/
[3] https://www.kaggle.com/datasets/thaihoa1476050/df2k-ost
[4] https://www.kaggle.com/datasets/shayanriyaz/riceleafs
[5] https://www.kaggle.com/datasets/akash2907/bird-species-classification
[6] http://www.fftw.org/index.html
[7] https://software.intel.com/mkl

(3) **Pruned FFT**: Pruned FFT[8] [2, 16, 17, 28, 30] is a pruned variant of FFT specialized for rapid computation of specific part of an output, utilizing FFTW as a subroutine.
(4) **PFT**: PFT[9] [20] is the current state-of-the-art algorithm for one-dimensional partial Fourier transform. For multidimensional data, PFT is applied for each axis of the data.

Measure. We opt for single-precision floating-point format for all experiments. The tolerance ϵ is adjusted to maintain a relative ℓ_2 error below 10^{-6}, thereby ensuring that the estimated coefficients possess at least 6 significant figures across all methods. Explicitly,

$$\text{Relative } \ell_2 \text{ Error} = \sqrt{\frac{\sum_{m\in\mathcal{B}} |\hat{a}_m - \mathcal{E}(\hat{a})_m|^2}{\sum_{m\in\mathcal{B}} |\hat{a}_m|^2}} < 10^{-6},$$

where $\hat{\boldsymbol{a}}$ is the actual Fourier coefficient, $\mathcal{E}(\hat{\boldsymbol{a}})$ is the estimation of $\hat{\boldsymbol{a}}$, and $\mathcal{B}$ is the target range.

4.2 Running Time (Q1)

The running time of Auto-MPFT is measured across synthetic and real-world datasets, with variations in input and output sizes.

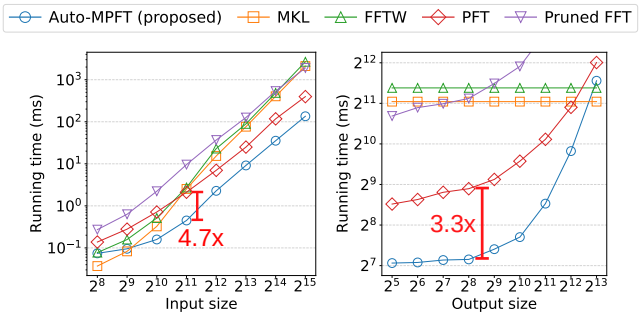
4.2.1 Synthetic Datasets. We generate 1,000 random synthetic matrices of size $2^n \times 2^n$ for each $n = 8, \cdots, 15$, and evaluate the average running time of Auto-MPFT and competitors for all the matrices with different settings.

Running time vs. input size. We fix the target range to be $2^6 \times 2^6$ Fourier coefficients centered at the origin and evaluate the average running time vs. input sizes $2^8 \times 2^8, \cdots, 2^{15} \times 2^{15}$. In Figure 3(a), the running time of the five algorithms is illustrated concerning varying input sizes. We observe that Auto-MPFT outperforms the baselines in most cases where the output has a sufficiently smaller size than the input. As a result, Auto-MPFT achieves a speedup of up to 4.7× compared to the baselines. Note that when M is very close to N, the time cost of Auto-MPFT tends to $O(N + N \log N)$ according to Theorem 6, thus it exhibits a slightly slower speed than FFT that has an $O(N \log N)$ time complexity.

Running time vs. output size. In the next setting, we fix the input size to $N = 2^{15} \times 2^{15}$ and evaluate the average running time vs. output sizes $2^5 \times 2^5, \cdots, 2^{13} \times 2^{13}$. Figure 3(b) shows the results, where Auto-MPFT consistently outperforms PFT and Pruned FFT, achieving up to 3.3× speedup. We also observe that the running times of the full FFT methods (MKL and FFTW) do not benefit from the information of output size.

4.2.2 Real-World Datasets. We evaluate Auto-MPFT on the five real-world datasets with output sizes $2^4 \times 2^4, \cdots, 2^8 \times 2^8$ for Cityscapes, ADE20K, and DF2K, and $2^4 \times 2^4, \cdots, 2^9 \times 2^9$ for Rice-Leaf and Bird. As illustrated in Figure 4, Auto-MPFT outperforms the competitors across all datasets, delivering speedups of up to 7.6×. Notably, PFT shows low efficiency especially when the size of the input is relatively small (Cityscapes, ADE20K, and DF2K), which is not the case for Auto-MPFT. These results clearly show the robustness of Auto-MPFT in diverse real-world settings.

[8]http://www.fftw.org/pruned.html
[9]https://github.com/snudatalab/PFT



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

Figure 3: (a) Running time vs. input size for output size $2^6 \times 2^6$, and (b) running time vs. output size for input size $2^{15} \times 2^{15}$. The x-axis is the length of one axis of data. To ensure the same precision across all methods, we have standardized the relative error to be strictly below 10^{-6}. Auto-MPFT consistently outperforms the partial Fourier transform methods, PFT and Pruned FFT. In addition, the smaller the output size is, Auto-MPFT becomes more efficient than MKL and FFTW.

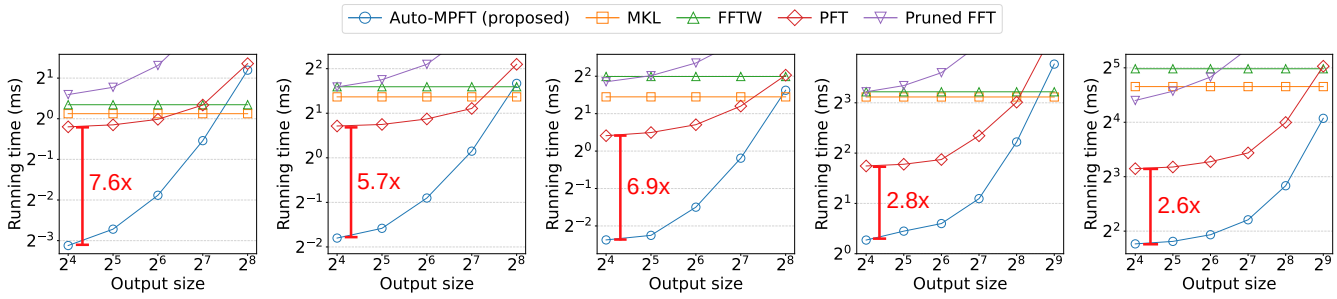
4.3 Automatic Hyperparameter Selection (Q2)

4.3.1 Accuracy of Optimization Algorithm. To evaluate the accuracy of our optimization algorithm for automatic hyperparameter selection, we find the ground-truth optimal value of p for the synthetic datasets $\{S_n\}_{n=8}^{15}$ with varying M, and compare it with the estimated $\hat{p}$ by our method. Table 2 shows that our algorithm accurately finds the optimal value of p in the majority of cases, and in instances where it deviates, the margin of error remains minimal. It is evident from these results that our optimization algorithm can effectively replace manual processes with little sacrifice in accuracy.

4.3.2 Running Time of Optimization Algorithm. We further validate the efficacy of our optimization algorithm by comparing its time cost for selecting the optimal hyperparameter to that of the manual search process. To this end, we fix the output size to $2^8 \times 2^8$ and vary the input size from $2^9 \times 2^9$ to $2^{15} \times 2^{15}$. Table 3 demonstrates that the automatic algorithm by Auto-MPFT achieves remarkably faster processing times compared to the manual process. This discrepancy arises because the manual process necessitates executing the entire algorithm for each p to determine its optimal value, whereas Auto-MPFT simplifies the process by efficiently solving a convex optimization problem. It is also worth mentioning that the discrepancy becomes more significant with larger input sizes.

4.4 Impact of Varying Precision (Q3)

Recall that Auto-MPFT offers the flexibility to set any numerical precision (Theorem 8). We investigate the trade-off between precision and running time of Auto-MPFT by adjusting the tolerance ϵ. For a fixed input size $2^{15} \times 2^{15}$, we vary the precision target from 10^{-6} to 10^{-4} or 10^{-2} across various output sizes. Table 4 shows the results, with the improvement of running times for each setting enclosed in parentheses. The reduction reaches up to 21.2% or 49.8% when the precision is relaxed to 10^{-4} or 10^{-2}, respectively. This indicates that one could gain advantages from the compromise, particularly when speed is crucial despite a slight trade-off in precision.



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

Figure 4: Running time vs. output size for the real-world datasets, where the x-axis is the length of one axis of the target range. We have standardized the relative error of all methods to be strictly below 10^{-6}. Our proposed Auto-MPFT exhibits superior performance across all datasets, especially in cases where the output has a sufficiently smaller size than the input.

Table 2: Validation of our optimization algorithm for automatic hyperparameter search. Entries without parentheses denote that our estimate $\hat{p}$ equals to the ground-truth p, and those with parentheses show both values in the form $\hat{p}(p)$. Auto-MPFT successfully detects the optimal value in most scenarios, with minor errors occurring infrequently.

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

Table 3: Comparison of running time (μs) for finding the optimal hyperparameter p using our optimization algorithm (Auto-MPFT) vs. manual search. *Manual-best* denotes finding the optimal value in a single attempt, whereas *Manual-worst* involves testing all divisors of 2^n except 1 and 2^n. Auto-MPFT significantly outperforms the manual search process.

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

Table 4: Average running time (ms) of Auto-MPFT with input size $2^{15} \times 2^{15}$ and different precision settings. Notably, we observe up to 49.8% improvement in running time when precision requirements are relaxed, offering a beneficial trade-off, particularly when prioritizing fast evaluations.

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%)

5 Conclusions

We propose Auto-MPFT (Automatic Multidimensional Partial Fourier Transform), an efficient and accurate method for computing a part of Fourier coefficients with automatic hyperparameter selection. Auto-MPFT decomposes the original DFT into small sub-blocks and approximates some of trigonometric functions by Chebyshev polynomials, reducing the arithmetic cost. Furthermore, we present an efficient optimization algorithm for finding the optimal hyperparameter of Auto-MPFT. Experiments demonstrate that Auto-MPFT outperforms the state-of-the-art baseline models, delivering up to 7.6× speedup without compromising accuracy. We also illustrate the efficacy of our convex optimization-based algorithm in selecting the optimal hyperparameter of Auto-MPFT, which leads to a significant reduction in the additional cost attributed to hyperparameter search. Future tasks involve enhancing the execution of Auto-MPFT through additional optimization.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) funded by MSIT(2022R1A2C3007921), Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [No.2020-0-00894, Flexible and Efficient Model Compression Method for Various Applications and Environments], [No.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], and [No.RS-2021-II212068, Artificial Intelligence Innovation Hub (Artificial Intelligence Institute, Seoul National University)]. U Kang is the corresponding author.

References

[1] Milton Abramowitz, Irene A Stegun, et al. 1972. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Vol. 55. National bureau of standards Washington, DC.

[2] Nir Ailon and Edo Liberty. 2009. Fast dimension reduction using Rademacher series on dual BCH codes. *Discrete & Computational Geometry* 42, 4 (2009), 615.

[3] Stefan Banach. 1922. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math* 3, 1 (1922), 133–181.

[4] C Boncelet. 1986. A rearranged DFT algorithm requiring N 2/6 multiplications. *IEEE Trans. Acoust. Speech Signal Process.* 34, 6 (1986).

[5] E Oran Brigham. 1988. *The fast Fourier transform and its applications*. Prentice-Hall, Inc.

[6] C Sidney Burrus and TW Parks. 1985. *DFT/FFT and Convolution Algorithms*. Citeseer.

[7] Neal L Carothers. 1998. A short course on approximation theory. (1998).

[8] Wen-Hsiung Chen, CH Smith, and Sam Fralick. 1977. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on communications* 25, 9 (1977), 1004–1009.

[9] James W Cooley and John W Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation* 19, 90 (1965).

[10] W. Fraser. 1965. A Survey of Methods of Computing Minimax and Near-Minimax Polynomial Approximations for Functions of a Single Independent Variable. *J. ACM* 12, 3 (1965), 295–314.

[11] Matteo Frigo and Steven G Johnson. 1998. FFTW: An adaptive software architecture for the FFT. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, Vol. 3. IEEE, 1381–1384.

[12] M. Frigo and S. G. Johnson. 2005. The Design and Implementation of FFTW3. *Proc. IEEE* 93, 2 (2005), 216–231.

[13] Gerald Goertzel. 1958. An algorithm for the evaluation of finite trigonometric series. *The American Mathematical Monthly* 65, 1 (1958), 34–35.

[14] AN Hossen, Ulrich Heute, OV Shentov, and SK Mitra. 1995. Subband DFT Part II: accuracy, complexity and applications. *Signal Processing* 41, 3 (1995), 279–294.

[15] Xiaodi Hou and Liqing Zhang. 2007. Saliency Detection: A Spectral Residual Approach. In *CVPR*. IEEE Computer Society.

[16] J Markel. 1971. FFT pruning. *IEEE transactions on Audio and Electroacoustics* 19, 4 (1971), 305–311.

[17] K Nagai. 1986. Pruning the decimation-in-time FFT algorithm with frequency shift. *IEEE Trans. Acoust. Speech Signal Process.* 34, 4 (1986), 1008–1010.

[18] Humberto Ochoa-Dominguez and Kamisetty Ramamohan Rao. 2019. *Discrete Cosine Transform*. CRC Press.

[19] Alan V Oppenheim. 1999. *Discrete-time signal processing*. Pearson Education India.

[20] Yong-chan Park, Jun-Gi Jang, and U Kang. 2021. Fast and accurate partial fourier transform for time series data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1309–1318.

[21] Peng Qi, Juan Cao, Tianyun Yang, Junbo Guo, and Jintao Li. 2019. Exploiting multi-domain visual information for fake news detection. In *ICDM*. IEEE.

[22] Faraz Rasheed, Peter Peng, Reda Alhajj, and Jon G. Rokne. 2009. Fourier Transform Based Spatial Outlier Mining. In *IDEAL*, Vol. 5788. Springer, 317–324.

[23] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-Series Anomaly Detection Service at Microsoft. In *KDD*. ACM, 3009–3017.

[24] Herbert Robbins. 1955. A remark on Stirling's formula. *The American mathematical monthly* 62, 1 (1955), 26–29.

[25] Tim Schlüter and Stefan Conrad. 2010. An Approach for Automatic Sleep Stage Scoring and Apnea-Hypopnea Detection. In *ICDM*. IEEE, 1007–1012.

[26] OV Shentov, SK Mitra, Ulrich Heute, and AN Hossen. 1995. Subband DFT Part I: Definition, interpretation and extensions. *Signal Processing* 41, 3 (1995), 261–277.

[27] Sheng Shi, Runkai Yang, and Haihang You. 2017. A new two-dimensional Fourier transform algorithm based on image sparsity. In *ICASSP*. IEEE, 1373–1377.

[28] D Skinner. 1976. Pruning the decimation in-time FFT algorithm. *IEEE Trans. Acoust. Speech Signal Process.* 24, 2 (1976), 193–194.

[29] Georgey S Smirnov and Roman G Smirnov. 1999. Best uniform approximation of complex-valued functions by generalized polynomials having restricted ranges. *Journal of approximation theory* 100, 2 (1999), 284–303.

[30] Henrik V Sorensen and C Sidney Burrus. 1993. Efficient computation of the DFT with only a subset of input or output points. *IEEE transactions on signal processing* 41, 3 (1993), 1184–1200.

[31] Ting Hei Wan, Chi Wai Tsang, King Hui, and Edward Chung. 2023. Anomaly detection of train wheels utilizing short-time Fourier transform and unsupervised learning algorithms. *Engineering Applications of Artificial Intelligence* 122 (2023), 106037.

[32] George Neville Watson. 1922. *A treatise on the theory of Bessel functions*. Vol. 3. The University Press.

[33] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. 2017. Stock price prediction via discovering multi-frequency trading patterns. In *KDD*. ACM, 2141–2149.

A Supplement

A.1 Proof of Lemma 2

Proof. We first show that for $r \geq 2$

$$\eta(r) \leq \frac{\sqrt{17}}{2} \sum_{n\geq 0} J_{r+2n}(c\pi). \tag{13}$$

For even r, it is straightforward from (10) that

$$\begin{aligned}\eta(r) &\leq \sum_{n\geq 0} |2i^{r+n} J_{r+n}(c\pi)\cos(\pi(r+n)/2)| \\ &= 2\sum_{n\geq 0} J_{r+2n}(c\pi) \leq \frac{\sqrt{17}}{2}\sum_{n\geq 0} J_{r+2n}(c\pi).\end{aligned}$$

For odd r, let $\theta_r = \pi(r-1)/2r$. Using the recurrence relation (9), we have

$$\begin{aligned}\frac{\eta(r)}{2} &= \left|\sum_{n\geq r} i^n J_n(c\pi)\cos n\theta_r\right| \\ &= \left|\sum_{n\geq 0} i^n J_{r+n}(c\pi)\cos(r+n)\theta_r\right| \\ &= \Bigg|\sum_{n\geq 0} i^{2n} J_{r+2n}(c\pi)\cos(r+2n)\theta_r \\ &\quad + i^{2n+1} J_{r+2n+1}(c\pi)\cos(r+2n+1)\theta_r\Bigg| \\ &= \Bigg|\sum_{n\geq 0} i^{2n} J_{r+2n}(c\pi)\cos(r+2n)\theta_r \\ &\quad + i^{2n+1} C\,\frac{J_{r+2n}(c\pi)+J_{r+2n+2}(c\pi)}{r+2n+1}\cos(r+2n+1)\theta_r\Bigg|.\end{aligned}$$

We separate the $J_r(c\pi)$ term from the summand as follows:

$$\begin{aligned}\frac{\eta(r)}{2} &= \Bigg|\left(\cos r\theta_r + iC\frac{\cos(r+1)\theta_r}{r+1}\right)J_r(c\pi) \\ &\quad + \sum_{n\geq 1} i^{2n}\Bigg(\cos(r+2n)\theta_r + iC\Bigg(\frac{\cos(r+2n+1)\theta_r}{r+2n+1} \\ &\quad - \frac{\cos(r+2n-1)\theta_r}{r+2n-1}\Bigg)\Bigg)J_{r+2n}(c\pi)\Bigg|.\end{aligned}$$

Because $r \geq 2$ is odd and $C/(r+1) \leq 1/2$, the magnitude of the first coefficient satisfies

$$\begin{aligned}\left|\cos r\theta_r + iC\frac{\cos(r+1)\theta_r}{r+1}\right| &= \left|1 + \frac{iC}{r+1}\sin\frac{\pi}{2r}\right| \\ &\leq \left|1+\frac{i}{2}\sin\frac{\pi}{6}\right| = \left|1+\frac{i}{4}\right| = \frac{\sqrt{17}}{4}.\end{aligned}$$

For the magnitude of remainder coefficients, we use the trigonometric identity

$$\cos(r+2n\pm 1)\theta_r = \cos(r+2n)\theta_r\cos\theta_r \mp \sin(r+2n)\theta_r\sin\theta_r,$$

which yields

$$\begin{aligned}\Bigg|\cos(r+2n)\theta_r + \frac{iC}{r+1}\Bigg(&h_-(n,r)\cos(r+2n)\theta_r\cos\theta_r \\ &- h_+(n,r)\sin(r+2n)\theta_r\sin\theta_r\Bigg)\Bigg|,\end{aligned} \tag{14}$$

where $h_\pm(n,r) := \frac{r+1}{r+2n+1} \pm \frac{r+1}{r+2n-1}$. Since $|a\cos\theta_r + b\sin\theta_r| \le \sqrt{a^2+b^2}$ for $a, b \in \mathbb{R}$, (14) becomes

$$\left(\cos^2(r+2n)\theta_r + \frac{C^2}{(r+1)^2}\left(h_-(n,r)\cos(r+2n)\theta_r\cos\theta_r - h_+(n,r)\sin(r+2n)\theta_r\sin\theta_r\right)^2\right)^{1/2}$$
$$\le \max_{n,r}\left(\cos^2(r+2n)\theta_r + \frac{C^2}{(r+1)^2}\left(h_-^2(n,r)\cos^2(r+2n)\theta_r + h_+^2(n,r)\sin^2(r+2n)\theta_r\right)\right)^{1/2}.$$

For $n \ge 1$,

$$|h_+(n,r)| \le 1+1 = 2,\ |h_-(n,r)| = \frac{2(r+1)}{r^2+4nr+4n^2-1} \le \frac{2}{r+3} \le \frac{1}{2}.$$

Thus, the magnitude of remainder coefficients is bounded by

$$\max_{n,r}\sqrt{\left(1+\frac{1}{4^2}\right)\cos^2(r+2n)\theta_r + \sin^2(r+2n)\theta_r} = \max_{n,r}\sqrt{1+\frac{1}{4^2}\cos^2(r+2n)\theta_r} \le \frac{\sqrt{17}}{4}.$$

This implies that

$$\frac{\eta(r)}{2} \le \frac{\sqrt{17}}{4}J_r(c\pi) + \sum_{n\ge1}\frac{\sqrt{17}}{4}J_{r+2n}(c\pi),$$

so the ineqaulity (13). We now use the following inequality [32] to complete the proof:

$$J_n(2w) \le \frac{w^n}{n!}e^{-\frac{w^2}{n+1}}$$

for a non-negative integer n and $w > 0$. From (13), we obtain

$$\begin{aligned}
\eta(r) &\le \frac{\sqrt{17}}{2}\sum_{n\ge0}J_{r+2n}(c\pi)\\
&\le \frac{\sqrt{17}}{2}\sum_{n\ge0}\frac{C^{r+2n}}{(r+2n)!}e^{-\frac{C^2}{r+2n+1}}\\
&= \frac{\sqrt{17}}{2}\frac{C^r}{r!}e^{-\frac{C^2}{r+1}}\left(1+\frac{C^2}{(r+1)(r+2)}e^{\frac{2C^2}{(r+1)(r+2)}} + \frac{C^4}{(r+1)(r+2)(r+3)(r+4)}e^{\frac{4C^2}{(r+1)(r+5)}}+\cdots\right)\\
&\le \frac{\sqrt{17}}{2}\frac{C^r}{r!}e^{-\frac{C^2}{r+1}}\left(1+\frac{C^2}{(r+1)^2}e^{\frac{2C^2}{(r+1)^2}} + \frac{C^4}{(r+1)^4}e^{\frac{4C^2}{(r+1)^2}}+\cdots\right)\\
&= \frac{\sqrt{17}}{2}\frac{C^r}{r!}e^{-\frac{C^2}{r+1}}\left(1-\frac{C^2}{(r+1)^2}e^{\frac{2C^2}{(r+1)^2}}\right)^{-1}.
\end{aligned}$$

Thus, it follows from $C^2/(r+1)^2 \le 1/4$ that

$$\eta(r) \le \frac{\sqrt{17}}{2}\frac{C^r}{r!}e^{-\frac{C^2}{r+1}}\left(1-\frac{1}{4}e^{\frac{2}{4}}\right)^{-1} = \frac{2\sqrt{17}}{4-\sqrt{e}}\frac{C^r}{r!}e^{-\frac{C^2}{r+1}},$$

hence the proof. □

A.2 Proof of Theorem 3

Proof. Note that $(N+p\log p+M)r$ is convex and non-decreasing for each $p, r \ge 1$. Therefore, it is sufficient to show that $p(r)$ is a convex function with respect to r. Indeed, we prove that $p(r)$ is logarithmically convex for $r \ge 1$, from which the convexity follows. We first show that

$$r \mapsto \log(\alpha\epsilon r!)^{-\frac{1}{r}}$$

is convex. It is easy to check that the function $\log(\alpha\epsilon)^{-1/r} = -(\log\alpha\epsilon)/r$ is convex for $r > 0$ since $0 < \alpha, \epsilon < 1$. We can also show that $\log r!^{-1/r} = -(\log r!)/r$ is a convex function considering its second derivative

$$\frac{d^2}{dr^2}\left(-\frac{\log r!}{r}\right) = \frac{-2\log r! + 2r\psi(r+1) - r^2\psi'(r+1)}{r^3}, \tag{15}$$

where $\psi(x) = \frac{d}{dx}\log\Gamma(x)$ is the digamma function [1]. The following property is often useful:

$$\log x \le \psi(x+1) = -\gamma + \sum_{k\ge1}\left(\frac{1}{k} - \frac{1}{x+k}\right) \le \log(x+1), \quad \forall x > 0,$$

where γ is the Euler-Mascheroni constant. We observe that the numerator of (15) is non-negative on $r \ge 0$ because it attains its minimum value 0 at $r = 0$ due to the following inequality:

$$\frac{d}{dr}(-2\log r! + 2r\psi(r+1) - r^2\psi'(r+1)) = -r^2\psi''(r+1) = \sum_{k\ge1}\frac{2r^2}{(r+k)^3} \ge 0$$

for all $r \ge 0$. Since the product of two logarithmically convex functions is also logarithmically convex, we conclude that $(\alpha\epsilon r!)^{-1/r} = (\alpha\epsilon)^{-1/r} \times r!^{-1/r}$ is logarithmically convex for $r > 0$.

We use the above result to prove that $r \mapsto (\alpha\epsilon r!)^{-\frac{1}{r}}e^{-\frac{1}{r(r+1)}(\alpha\epsilon r!)^{2/r}}$ is a convex function. Let $u(r) := (\alpha\epsilon r!)^{-1/r}$. Our goal is to show that the following function is convex:

$$r \mapsto \log u(r) - \frac{1}{r(r+1)u(r)^2}. \tag{16}$$

A simple calculus shows that

$$\begin{aligned}
\frac{d^2}{dr^2}\log u(r) &= \frac{2v(r) - r\psi'(r+1)}{r^2},\\
\frac{d^2}{dr^2}\frac{1}{r(r+1)u(r)^2} &= \left(\frac{2(3r^2+3r+1)}{r^2(r+1)^2} + \frac{2\psi'(r+1)}{r} - \frac{4(3r+2)}{r^2(r+1)}v(r) + \frac{4}{r^2}v(r)^2\right)\frac{1}{r(r+1)u(r)^2},
\end{aligned}$$

where $v(r) := \log u(r) + \psi(r+1)$. Since we have already shown that $\frac{d^2}{dr^2}\log u(r) \ge 0$, it is sufficient to consider only when r satisfies $\frac{d^2}{dr^2}\frac{1}{r(r+1)u(r)^2} \ge 0$, otherwise the function (16) trivially has a nonnagative second derivative. For $r \ge 1$, we have

$$\begin{aligned}
r(r+1)u(r)^2 &\ge 1 + \log(r(r+1)u(r)^2)\\
&\ge \log((r+1)^2u(r)^2)\\
&= 2(\log(r+1) + \log u(r))\\
&\ge 2(\psi(r+1) + \log u(r))\\
&= 2v(r),
\end{aligned}$$

so $-\frac{1}{r(r+1)u(r)^2} \geq -\frac{1}{2v(r)}$. Also,

$$\psi'(r+1) = \sum_{k\geq 1} \frac{1}{(r+k)^2} \leq \int_r^\infty \frac{dx}{x^2} = \frac{1}{r}.$$

These inequalities imply that

$$\begin{aligned}
&\frac{d^2}{dr^2}\left(\log u(r) - \frac{1}{r(r+1)u(r)^2}\right) \\
&\geq \frac{2v(r) - r\psi'(r+1)}{r^2} - \left(\frac{2(3r^2+3r+1)}{r^2(r+1)^2} + \frac{2\psi'(r+1)}{r} \right. \\
&\qquad \left. - \frac{4(3r+2)}{r^2(r+1)}v(r) + \frac{4}{r^2}v(r)^2\right)\frac{1}{2v(r)} \\
&\geq \frac{2v(r)-1}{r^2} - \left(\frac{2(3r^2+3r+1)}{r^2(r+1)^2} + \frac{2}{r^2} \right. \\
&\qquad \left. - \frac{4(3r+2)}{r^2(r+1)}v(r) + \frac{4}{r^2}v(r)^2\right)\frac{1}{2v(r)} \\
&= \frac{-(4r^2+5r+2) + v(r)(5r^2+8r+3)}{v(r)r^2(r+1)^2}.
\end{aligned}$$

Thus, if we show that $v(r) \geq 4/5$, then $v(r)r^2(r+1)^2 > 0$, and

$$\begin{aligned}
&-(4r^2+5r+2) + v(r)(5r^2+8r+3) \\
&\geq -(4r^2+5r+2) + \frac{4}{5}(5r^2+8r+3) = \frac{7r+2}{5} \geq 0,
\end{aligned}$$

which proves that (16) is convex. Now, Stirling's formula [24] implies that

$$r! \leq \sqrt{2\pi r}\left(\frac{r}{e}\right)^r e^{\frac{1}{12r}}$$

for all $r \geq 1$. Thus, it follows that

$$\begin{aligned}
r\left(v(r) - \frac{4}{5}\right) &= r\left(-\frac{1}{r}\log(\alpha\epsilon r!) + \psi(r+1) - \frac{4}{5}\right) \\
&\geq r\left(-\frac{1}{r}\log(\alpha r!) + \log r - \frac{4}{5}\right) \\
&= -\log(\alpha r!) + r\log r - \frac{4}{5}r \\
&\geq -\log\left(\alpha\sqrt{2\pi r}\left(\frac{r}{e}\right)^r e^{\frac{1}{12r}}\right) + r\log r - \frac{4}{5}r \\
&= -\log\alpha\sqrt{2\pi r} - r\log r + r - \frac{1}{12r} + r\log r - \frac{4}{5}r \\
&= -\log\alpha\sqrt{2\pi r} + \frac{1}{5}r - \frac{1}{12r} \\
&\geq -\log\alpha\sqrt{2\pi r} + \frac{1}{5}r - \frac{1}{12}.
\end{aligned}$$

Since

$$\frac{d}{dr}\left(-\log\alpha\sqrt{2\pi r} + \frac{1}{5}r - \frac{1}{12}\right) = -\frac{1}{2r} + \frac{1}{5},$$

the function attains its minimum at $r = 5/2$, so

$$r\left(v(r) - \frac{4}{5}\right) \geq -\log\alpha\sqrt{5\pi} + \frac{5}{12} = 0.294377\cdots \geq 0,$$

or $v(r) \geq 4/5$. This completes the proof. □

A.3 Proof of Theorem 8

Proof. Let $v_d = l_d - q_d/2$ and $\mathcal{P}_d = \mathcal{P}_{r_d,\xi(\epsilon,r_d)}$ for $d = 1, 2, \cdots, D$, and $\mathcal{B} = \mathcal{B}_{\boldsymbol{\mu},\boldsymbol{M}}$. Then, it follows that

$$\begin{aligned}
&\|\hat{\boldsymbol{a}} - \mathcal{E}(\hat{\boldsymbol{a}})\|_{\mathcal{B}} \\
&\leq \sum \left\| a_{\boldsymbol{l}}^{(\boldsymbol{k})} \left(\prod\nolimits_d \omega_{N_d}^{v_d m_d} - \prod\nolimits_d \omega_{N_d}^{v_d \mu_d} \mathcal{P}_d(-2v_d(m_d - \mu_d)/N_d)\right)\right\|_{\mathcal{B}} \\
&= \sum |a_{\boldsymbol{l}}^{(\boldsymbol{k})}| \left\| \prod\nolimits_d \omega_{N_d}^{v_d(m_d-\mu_d)} - \prod\nolimits_d \mathcal{P}_d(-2v_d(m_d-\mu_d)/N_d)\right\|_{\mathcal{B}},
\end{aligned}$$

where the summations are over indices $\boldsymbol{k} \in \prod_d [p_d]$ and $\boldsymbol{l} \in \prod_d [r_d]$. Since l_d ranges from 0 to $q_d - 1$, we have $|2v_d/N_d| \leq 2(q_d/2)/N_d = 1/p_d$, and therefore $M_d|2v_d/N_d| \leq M_d/p_d \leq \xi(\epsilon, r_d)$. We replace $-2v_d(x_d - \mu_d)/N_d$ with x'_d:

$$\begin{aligned}
&\|\hat{\boldsymbol{a}} - \mathcal{E}(\hat{\boldsymbol{a}})\|_{\mathcal{B}} \\
&\leq \sum |a_{\boldsymbol{l}}^{(\boldsymbol{k})}| \left\| \prod\nolimits_d e^{\pi i x'_d} - \prod\nolimits_d \mathcal{P}_d(x'_d)\right\|_{|x'_d| \leq M_d|2v_d/N_d|,\ \forall d} \\
&\leq \sum |a_{\boldsymbol{l}}^{(\boldsymbol{k})}| \left\| \prod\nolimits_d e^{\pi i x'_d} - \prod\nolimits_d \mathcal{P}_d(x'_d)\right\|_{|x'_d| \leq \xi(\epsilon,r_d),\ \forall d}.
\end{aligned}$$

Now

$$\begin{aligned}
\prod_d e^{\pi i x'_d} - \prod_d \mathcal{P}_d(x'_d) &= \left(\prod_{d<D} e^{\pi i x'_d}\right)(e^{\pi i x'_D} - \mathcal{P}_D(x'_D)) \\
&\quad + \left(\prod_{d<D} e^{\pi i x'_d} - \prod_{d<D} \mathcal{P}_d(x'_d)\right)\mathcal{P}_D(x'_D).
\end{aligned}$$

Using the above equation recursively, we obtain

$$\begin{aligned}
&\prod_d e^{\pi i x'_d} - \prod_d \mathcal{P}_d(x'_d) \\
&= \sum_{s=1}^{D} \left(\prod_{d<s} e^{\pi i x'_d}\right)(e^{\pi i x'_s} - \mathcal{P}_s(x'_s))\left(\prod_{s<d} \mathcal{P}_d(x'_d)\right).
\end{aligned}$$

Because $|x'_d| \leq \xi(\epsilon, r_d)$ for all d, we have the following inequality:

$$\begin{aligned}
&\left|\prod_d e^{\pi i x'_d} - \prod_d \mathcal{P}_d(x'_d)\right| \\
&\leq \sum_{s=1}^{D} \left|\prod_{d<s} e^{\pi i x'_d}\right| \cdot |e^{\pi i x'_s} - \mathcal{P}_s(x'_s)| \cdot \left|\prod_{s<d} \mathcal{P}_d(x'_d)\right| \\
&\leq \sum_{s=1}^{D} 1^{s-1} \cdot \epsilon \cdot (\epsilon+1)^{D-s} \;=\; (\epsilon+1)^D - 1,
\end{aligned}$$

where the second inequality holds since $|e^{\pi i x'_d}| = 1$ and $|\mathcal{P}_d(x'_d)| \leq |\mathcal{P}_d(x'_d) - e^{\pi i x'_d}| + |e^{\pi i x'_d}| \leq \epsilon + 1$. We may assume that the tolerance is sufficiently small so that $\epsilon < 2/D^2$. Then it is easy to see that

$$\begin{aligned}
(\epsilon+1)^D - 1 &= \sum_{d=0}^{D}\binom{D}{d}\epsilon^d - 1 \;=\; D\epsilon + \sum_{d=2}^{D}\binom{D}{d}\epsilon^d \\
&\leq D\epsilon + \sum_{d=2}^{D}\left(\frac{D^2}{2}\right)^{d-1}\epsilon^d \\
&\leq D\epsilon + \sum_{d=2}^{D}\left(\frac{1}{\epsilon}\right)^{d-1}\epsilon^d \;=\; D\epsilon + \sum_{d=2}^{D}\epsilon \;=\; (2D-1)\epsilon.
\end{aligned}$$

Thus, we obtain the desired approximation bound of Auto-MPFT: $\|\hat{\boldsymbol{a}} - \mathcal{E}(\hat{\boldsymbol{a}})\|_{\mathcal{B}} \leq \sum |a_{\boldsymbol{l}}^{(\boldsymbol{k})}| \cdot (2D-1)\epsilon = \|\boldsymbol{a}\|_1 \cdot (2D-1)\epsilon$. □