Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.777665

FAST FOURIER TRANSFORM FOR ACCELERATING CONVOLUTIONAL NEURAL NETWORKS: TECHNIQUE AND APPLICATIONS

Rohit Bokade
Dept. of Artificial Intelligence, K.J. Somaiya Institute of Technology, Sion
Mumbai, India rohit.bokade@somaiya.edu

Dr. Namrata Gharat
Dept. of Artificial Intelligence, K.J. Somaiya Institute of Technology, Sion, Mumbai, India
ngharat@somaiya.edu

Abstract—Convolutional Neural Networks (CNNs) have demonstrated remarkable success in various fields of computer vision and image processing. However, the computational complexity and resource requirements of CNNs can limit their deployment in real-time applications or on resource-constrained devices. This paper presents a comprehensive study of the Fast Fourier Transform (FFT) as a technique for accelerating CNNs, aiming to reduce computational complexity while maintaining high performance. We explore the fundamentals of FFT-based convolution, its implementation in CNNs, and its implications for network architecture design.

We begin by introducing the theoretical background of the FFT and its application in convolutional operations. Next, we present a comparative analysis of the performance, computational complexity, and memory requirements of traditional spatial-domain CNNs and their FFT-based counterparts. Fur- thermore, we delve into the practical aspects of implementing FFT-accelerated CNNs on different hardware platforms, such as CPUs, GPUs, and specialized accelerators.

Finally, we present various applications of FFT-accelerated CNNs in various domains, highlighting the benefits and chal- lenges of adopting this technique in real-world scenarios. Our analysis demonstrates that FFT-based convolution can lead to significant speedups and resource savings in CNNs, making them more suitable for deployment in time-critical and resource- limited environments. However, certain trade-offs must be con- sidered, such as increased algorithmic complexity and potential loss of accuracy due to numerical approximations.

Index Terms—FFT, CNN, Convolutional Neural Networks, Fast Fourier Transform, Faster CNN

I. INTRODUCTION

In recent years, the field of machine learning has expe- rienced a significant shift with the emergence of Artificial Neural Networks (ANNs). These biologically inspired compu- tational models have surpassed the performance of traditional artificial intelligence methods in many machine learning tasks. One of the most notable ANN architectures is the Convolu- tional Neural Network (CNN).

CNNs share similarities with traditional ANNs, as both are composed of neurons that self-optimize through learning. Each neuron receives input and performs an operation, such as a

scalar product followed by a non-linear function, which is the foundation of numerous ANNs. From the raw input image vectors to the final class score output, the entire network expresses a single perceptive score function (the weight). The last layer contains loss functions associated with the classes, and all the standard techniques developed for traditional ANNs still apply.
However, the computational complexity of CNNs is a sig- nificant challenge, as propagation through convolutional layers can be slow, with each kernel sequentially calculating many dot products for a single forward and backward propagation. Each kernel in each layer must sequentially calculate many dot products for a single forward and backward propagation which equates to O(N 2n2) per kernel per layer where the inputs are N N arrays and the kernels are n n arrays. [1] The FFT is an algorithm for efficiently computing the Discrete Fourier Transform (DFT) of a signal or image. The Convolution Theorem, a fundamental result in mathematics, states that, under suitable conditions, the Fourier transform of a convolution of two functions is the pointwise product of their Fourier transforms. In other words, convolution in one domain (e.g., time domain) is equivalent to pointwise multiplication in the other domain (e.g., frequency domain). This property of the FFT allows for a significant reduction in the computational
complexity of the convolution operation in CNNs. [2]

II. CONVOLUTION THEOREM

The underlying intuition given by the Convolution Theorem which states that for two functions v and u, we have

$$F(v * u) = F(v) \odot F(u) \qquad (1)$$

where denotes the Fourier transform, * denotes convolu- tion and denotes the Hadamard Pointwise Product.his allows for convolution to be calculated more efficiently using Fast Fourier Transforms (FFTs). Since convolution corresponds to the Hadamard product in the Fourier domain and given the efficiency of the Fourier transform, this method involves significantly fewer computational operations than when using the sliding kernel spatial method, and is therefore much faster. [4]

Working in the Fourier domain can be less intuitive, as visualizing the filters learned by Fourier convolution becomes challenging. This issue is common in Convolutional Neural Network (CNN) techniques and is not addressed in this paper. Despite the frequent use of the Fourier domain in image processing and analysis.

A. Proof of convolution theorem

Consider two continuous signals, f(t) and g(t), and their convolution defined as:

$$h(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \qquad (2)$$

The Fourier Transform of h(t) can be found by applying the definition of the Fourier Transform:

$$H(\omega) = F\{h(t)\} = \int_{-\infty}^{\infty} h(t)e^{-j\omega t}dt \qquad (3)$$

Thus, the Fourier Transform of the convolution of two signals is equal to the element-wise multiplication of their individual Fourier Transforms.

B. Implications of convolution theorem on CNNs

The time complexity of the general convolution operation for two signals of size n is O(n2). This complexity arises from the fact that for each element in the output signal, a product and a sum need to be computed for all elements in the input signals. As the size of the input signals increases, the number of operations required for the convolution operation grows quadratically. The Fast Fourier Transform (FFT) algorithm has a time complexity of O(n log n) [5] for a signal of size n. This

$$H(\omega) = F\{h(t)\} = \int_{-\infty}^{\infty} h(t) e^{-j\omega t} dt \quad (3)$$

algorithm significantly reduces the computational cost of calculating the Discrete Fourier Transform (DFT) by exploiting the symmetry and periodicity properties of the complex exponential functions.

Substitute the definition of the convolution from Eq. (2) into Eq. (3):

$$H(\omega) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau \, e^{-j\omega t} dt \quad (4)$$

To perform convolution using FFT, the following steps are involved:

Now, we can exchange the order of integration:

$$H(\omega) = \int_{-\infty}^{\infty} f(\tau) \int_{-\infty}^{\infty} g(t-\tau) e^{-j\omega t} dt \, d\tau \quad (5)$$

signal.

2) Perform element-wise multiplication in the frequency domain: O(n).

Let $u = t - \tau$. Then, $dt = du$ and $t = u + \tau$. Substituting these expressions into Eq. (5), we get:

O(n log n).

The overall time complexity for performing convolution using

$$H(\omega) = \int_{-\infty}^{\infty}$$

$$f(\tau)$$
$$-\infty$$

$$\infty$$
$$g(u)e$$
$$-\infty$$

$$-j\omega(u+\tau)$$

$$du$$

$$d\tau \qquad (6)$$

FFT can be calculated by summing the complexities of these steps:
Convolution using FFT = 2 × O(n log n) + O(n)

$$H(\omega) =$$

$$\infty$$
$$f(\tau)$$
$$-\infty$$

$$\infty$$
$$g(u)e$$
$$-\infty$$

$$-j\omega u$$

$$e-j\omega\tau$$

$$du$$

$$d\tau \qquad (7)$$

$$+ O(n \log n)$$
$$= O(3n \log n) + O(n)$$

(12)

Since $e^{-j\omega\tau}$ does not depend on u, it can be moved outside the inner integral:
$H(\omega) = \int \infty \ f(\tau)e^{-j\omega\tau} \int \infty \ g(u)e^{-j\omega u}du \ d\tau$ (8)

Since O(n log n) dominates O(n) in terms of growth, the overall time complexity for performing convolution using FFT can be simplified to O(n log n):

Observe that the inner integral is the Fourier Transform of g(t),
denoted as G(ω):
$G(\omega) = F\{g(t)\} = \infty \ g(u)e^{-j\omega u}du$ (9)
$-\infty$
Similarly, the outer integral in Eq. (8) represents the Fourier Transform of f(t), denoted as F(ω):
$F(\omega) = F\{f(t)\} = \infty \ f(\tau)e^{-j\omega\tau}\,d\tau$ (10)
$-\infty$
Substituting Eq. (9) and Eq. (10) into Eq. (8), we obtain the Convolution Theorem:
$H(\omega) = F(\omega) \cdot G(\omega)$ (11)
From Eq. (2) H(ω) is Fourier transforms of convolution of signal f(t) and g(t). From Eq. (9) and Eq. (10), the F (ω) and G(ω) are Fourier transforms of f(t) and g(t) respectively.

As a result, the convolution operation using FFT has a significantly lower time complexity compared to the general convolution operation, making it more efficient for large input signals and convolutional neural networks

III. EXAMPLE APPLICATIONS OF THE TECHNIQUE

For a 256x256 image convolved with a 3x3 kernel using a stride of 1, the direct convolution method would mean

output size = (input size − filter size + 2 · padding) + 1
stride (14)
which leads to,
output size = (256 − 3 + 2 · 0) + 1 (15)

1
output size = 254 (16)

This means the output will be a 254x254 feature map. To calculate the total number of steps, we need to consider the number of elements in the output feature map:

total steps = output size output size = 254 254 = 64516
(17)
Each of these steps will involve further operations on the elements of the window in each step. To calculate the number of steps required for taking the Fast Fourier Transform (FFT) of the same 256x256 pixel image, we need to consider the complexity of the FFT algorithm. The most common FFT algorithm is the Cooley-Tukey algorithm, which has a complexity of O(n log n), where n is the number of data points.

In our case, the image has 256x256 pixels, which gives us a total of 65,536 data points. We will perform a 2D FFT, which consists of taking the 1D FFT for each row and then for each column. Let's calculate the number of steps for each dimension:
For rows:
n = 256 steps = n × log2(n) steps = 256 ×
log2(256) steps = 256 8 steps = 2, 048
For columns:
n = 256 steps = n × log2(n) steps = 256 ×
log2(256) steps = 256 8 steps = 2, 048
Since we need to perform the FFT for both rows and columns, the total number of steps required for the 2D FFT of the 256x256 image is:
totalsteps = stepsrows + stepscolumns totalsteps = 2, 048 + 2, 048 totalsteps = 4, 096
So, it would take 4,096 steps to perform the FFT on the 256x256 pixel image. similar number of steps will be required for inverse Fourier transform and element wise multiplications. Thus total steps and thus operations will be better by an order of magnitude. Thus reducing the time and computing power required.
As the number of pixels of an image increase the difference between the number of operations could become significantly larger.

IV. CHALLENGES IN ADOPTION

Using FFT-based convolution in real-world scenarios comes with its own set of challenges. One of the primary concerns is the need for zero-padding the input signals and the kernel to ensure that they are of the same size and to avoid circular convolution or wrap-around effects This additional padding can lead to increased memory usage and processing time, especially for large input signals.
Another challenge is the optimal choice of FFT size. Most FFT packages perform optimally for sizes that do not have large prime factors. In practice, rounding up the signal and kernel size to the next power of two is a common approach to achieve better performance.However, this can further increase memory usage and computational requirements.

For small kernel sizes, direct convolution may be faster than FFT-based convolution due to its lower constant factors, de- spite the higher time complexity of O(n2). In such cases, using FFT-based convolution may not yield significant performance improvements.
Moreover, FFT-based convolution is inherently a block- wise algorithm, which can lead to latency issues in real-time applications. This latency may not be acceptable for certain use cases, such as real-time audio processing or live data analysis.

V. OPPORTUNITIES FOR UTILIZATION

Despite the challenges, proposed technique could be used in tandem with hardware based acceleration for FFT calcula- tions inherently provided by different off-the-shelf GPUs for applications involving large target and kernel sizes. This could include accelerating models processing high definition video frames having very large pixel dimensions and hyper-spectral satellite imagery which has both large spatial and spectral resolution.
This technique could further help in scenarios where the speed of predictions is of top-most priority e.g. CNN models used in self driving cars would benefit from boosted speed of

inference especially while making critical decisions on busy roads. It could also be used with more sophisticated forms of CNN like U-Net to enable faster inferences on tasks like segmentation allowing for near-real-time filtering or classification tasks to be performed.
Apart from this, there have been several proposals to use the FFT base convolutional technique and extending it to other operations like spectral-pooling and using FFT-CNNs to train completely in spectral domain [6], thus reducing number of back-and-forth between spectral and spatial domains, resulting in less number of inaccuracies introduced by FFT while taking approximations of DFT.

VI. CONCLUSION AND FUTURE SCOPE

In conclusion, this paper presents a comprehensive study of the Fast Fourier Transform (FFT) as a technique for accelerat- ing Convolutional Neural Networks (CNNs) to reduce compu- tational complexity while maintaining high performance. The Convolution Theorem and the FFT algorithm are explored in depth, demonstrating how they can be applied to signifi- cantly reduce the time complexity of convolution operations in CNNs. A comparative analysis of the performance, compu- tational complexity, and memory requirements of traditional spatial-domain CNNs and their FFT-based counterparts is provided.
Various example applications and challenges in adopting FFT-accelerated CNNs are discussed, highlighting the poten- tial benefits and trade-offs in real-world scenarios. The analy- sis demonstrates that FFT-based convolution can lead to signif- icant speedups and resource savings in CNNs, making them more suitable for deployment in time-critical and resource-limited environments. However, certain trade-offs must be considered, such as increased algorithmic complexity and potential loss of accuracy due to numerical approximations.

Additionally, FFT-based convolution may not always yield significant performance improvements for small kernel sizes or in real-time applications with latency constraints.
Opportunities for utilizing FFT-accelerated CNNs include hardware-based acceleration, large-scale image and video pro- cessing, self-driving cars, and advanced CNN architectures like U-Net for faster inference. Further research can explore techniques to minimize the inaccuracies introduced by FFT approximations and extend the FFT-based approach to other operations, such as spectral-pooling.
Overall, FFT-based convolution offers a promising approach to accelerate CNNs and enable their deployment in a wider range of applications and environments where computational resources and time are critical factors.

ACKNOWLEDGMENT

We would like to express our gratitude to Hon. Principal of K.J.Somaiya Institute of Technology, Mr. Suresh Ukrande for his support and guidance during the research. We would also like to thank Mr. Hariram Chavan, HoD, Department of Artificial Intelligence for guiding us to conduct reseach in this area.

REFERENCES

[1] Highlander, T. & Rodriguez, A. Very Efficient Training of Convolutional Neural Networks using Fast Fourier Transform and Overlap-and-Add. (2016)
[2] Hsiao, V., Nau, D. & Dechter, R. Fast Fourier Transform Reductions for Bayesian Network Inference . Proceedings Of The 25th International Conference On Artificial

Intelligence And Statistics. 151 pp. 6445-6458 (2022,3,28), https://proceedings.mlr.press/v151/hsiao22a.html
[3] Pratt, H., Williams, B., Coenen, F. & Zheng, Y. FCNN: Fourier Convo- lutional Neural Networks. Machine Learning And Knowledge Discovery In Databases. pp. 786-798 (2017)
[4] Vasilache, N., Johnson, J., Mathieu, M., Chintala, S., Piantino, S. & LeCun, Y. Fast convolutional nets with fbfft: A GPU performance evaluation. ArXiv Preprint ArXiv:1412.7580. (2014)
[5] Cooley, J. & Tukey, J. An algorithm for the machine calculation of complex Fourier series. Mathematics Of Computation. 19, 297-301 (1965)
[6] Wang, Z., Lan, Q., Huang, D. & Wen, M. Combining FFT and spectral-pooling for efficient convolution neural network model. 2016 2nd International Conference On Artificial Intelligence And Industrial Engineering (AIIE 2016). pp. 203-206 (2016)