Journal of Iranian Association of Electrical and Electronics Engineers
Vol. 22 No. 2, 2025
DOI: 10.61186/jiaeee.22.2.160

Modeling DRAM Access Based on Efficient Tiling in CNN Hardware Accelerators

Sakineh Seydi[1] , Mostafa Ersali Salehi Nasab[2]

[1] PHD Student, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
sakinehseydi@ut.ac.ir
[2] Assistant Professor,School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
mersali@ut.ac.ir

Abstract:

Artificial neural networks are a subset of machine learning inspired by the biological neural networks of the human brain. These networks are applied in various fields, including natural language processing, pattern recognition and image processing. CNNs are an example of networks that have a layered structure. Due to the high volume of computations in CNNs, there is an increased need for bandwidth and memory transfers. researches have shown that the energy consumption and access time of external memory are 200x and 10x greater than internal memory.

One of the main solutions to reduce energy consumption is to increase data reuse and reduce the number of accesses to external memory. Maximizing data usage reduces the number of data movements and memory accesses. One method for data reuse is loop-level scheduling and tiling. This paper models the relationship between the number of accesses to external memory when using tiling. That presented as a mathematical formula that can determine the exact number of DRAM accesses based on network parameters and the tile size. Then optimal parameters are obtained with the goal of minimizing the use of external memory and establishing the relationship between network configuration parameters and tile size.

Keywords: **CNNs, Energy consumption, DRAM, Data reuse, Tiling.**

Article Type: Research

Received: 24. 11. 2024

Revised: 08. 01. 2025

Accepted: 06. 02. 2025

Corresponding author: Mostafa Ersali Salehi Nasab

Corresponding author's address: School of Electrical and Computer Engineering, University College of Engineering, University of Tehran, North Kargar St., Tehran, Iran

1. Motivation of the work

One of the most widely used neural networks is the Convolutional Neural Network (CNN). Due to their high accuracy, these networks are extensively utilized in image classification, object detection, and localization tasks. CNNs are frequently deployed on mobile devices, which require high performance while operating under strict power consumption constraints[1].To achieve higher accuracy, CNNs require deeper layers, a larger number of parameters, and consequently, an increased network size. Additionally, given the large data volume, data movement within the network increases, leading to higher bandwidth requirements and frequent memory transfers. Since memory bandwidth is limited, execution delays occur, resulting in performance degradation. Research indicates that the energy consumption and access time of external DRAM are approximately 200 times and 10 times higher, respectively, than those of internal memory. Therefore, reducing the reliance on external DRAM, minimizing memory transfers, and optimizing data movement can enhance both energy efficiency and overall performance[2][3].

In this paper, we first propose a mathematical formula to calculate the number of DRAM accesses based on network parameters. Then, using this formula, we determine the optimal tile size for each layer of the CNN, considering the structural parameters of the layer. Additionally, we establish the relationship between the optimal tile size and network parameters, which can be beneficial for memory optimization, energy efficiency, and reducing the number of external memory accesses.

2. Contributions

In prior studies, to reduce DRAM accesses, the mathematical formula for calculating external memory accesses has been derived based on loop parameters in CNNs, focusing on optimizing the access pattern for loop tiling[4][5]. However, these studies do not explore the relationship between network structural parameters (such as kernel size and stride), tile size, and DRAM access count. Additionally, some previous works have determined the optimal tile size through simulations rather than providing an exact mathematical model.

In our work, we develop a precise mathematical model that establishes the relationship between the number of DRAM accesses and key parameters such as stride and kernel size when tiling is applied. Leveraging this formula, we derive the optimal tile size to minimize memory accesses.

3. Procedures

In this paper, we address the lack of a precise mathematical relationship between tile size and the number of external memory accesses in CNNs. Previous research has not provided an exact formula for determining the optimal tile size to maximize data reuse and minimize memory accesses. To bridge this gap, we develop a mathematical model that accurately describes the relationship between tile size and memory accesses, ensuring efficient memory usage.

Additionally, we incorporate other critical parameters such as kernel size, stride, and layer-specific structural characteristics into our model. These factors significantly influence tile size and memory access patterns, and understanding their interactions is crucial for optimizing CNN performance. Our primary objective is to reduce the number of DRAM accesses while maximizing data reuse, ultimately leading to lower energy consumption in CNN-based applications. To evaluate the effectiveness of our approach, we analyze the impact of tiling on different CNN architectures, particularly MobileNet, which is widely used in mobile and edge devices. By examining various layers of these networks, we assess how tiling strategies influence memory efficiency and computational performance across diverse architectures.

4. Findings

Based on our proposed mathematical formula for accurately estimating the number of DRAM accesses, we developed an algorithm to determine the optimal tile size for each CNN layer, maximizing data reuse. Additionally, we analyzed the impact of network parameters such as kernel size and stride on memory access patterns. Our results indicate that the optimal tile size does not necessarily increase with the input dimensions (ni). Instead, it remains relatively close to the kernel size (k), with minimal deviation.

To further investigate the effect of input size on optimal tiling, we conducted experiments by keeping kernel size and stride (s) constant while varying input dimensions across different configurations. The findings consistently showed that input size has a negligible impact on optimal tile size, whereas kernel size plays a significant role in determining its value.

Furthermore, we applied our proposed algorithm to MobileNet and determined the optimal tile size for each layer. Using the suggested parameter configurations, the number of external memory accesses was reduced by 40% to 87% compared to the baseline implementation. These results demonstrate the effectiveness of our approach in enhancing memory efficiency and reducing energy consumption in CNN-based applications, particularly for resource-constrained devices such as mobile platforms.

5. Conclusion

Tiling is a key technique for optimizing CNN memory usage and performance. By leveraging spatial and temporal data locality, tiling enables data reuse, minimizing external memory accesses and data movement. In this study, we mathematically model external memory accesses based on network structural parameters and tile size. We then formulate an optimization problem to determine the optimal tile size that minimizes external memory accesses while applying techniques to reduce the search space. Our analysis reveals that optimal tile size is closely related to kernel size, remaining less than four times the kernel size in over 70% of cases. Additionally, we show that increasing stride reduces external memory accesses due to lower

window overlap and causes the optimal tile size to decrease, aligning closely with the kernel size.

نشریه مهندسی برق و الکترونیک ایران

نشریه مهندسی برق و الکترونیک ایران، سال بیست و دوم، شماره دوم، تابستان ۱۴۰۴، صفحات ۱۶۰-۱۷۱

DOI: 10.61186/jiaeee.22.2.160

مدلسازی دسترسی به حافظه براساس کاشی‌بندی موثر در شتابدهنده سخت‌افزاری شبکه‌های عصبی کانولوشنی

سکینه صیدی[۱]، مصطفی ارسالی صالحی نسب[۲]

۱- دانشجوی دکتری- دانشکده مهندسی برق و کامپیوتر- دانشگاه تهران- تهران- ایران

sakinehseydi@ut.ac.ir

۲- استادیار- دانشکده مهندسی برق و کامپیوتر- دانشگاه تهران - تهران- ایران

mersali@ut.ac.ir

چکیده: **شبکه‌های عصبی مصنوعی زیرمجموعه‌ای از یادگیری ماشین هستند که الهام گرفته از شبکه‌های عصبی زیستیِ مغز انسان هستند و قابلیت یادگیری دارند. این شبکه‌ها در حیطه‌های مختلف از جمله پردازش زبان طبیعی، تشخیص الگو، پردازش تصویر، بینایی ماشین و بسیاری از زمینه‌های دیگر کاربرد دارند. شبکه‌های عصبی CNN نمونه‌ای از این شبکه‌ها هستند که ساختار لایه به لایه دارند و عملیات اصلی آن‌ها کانولوشن است. از آنجایی که حجم محسابات و هم‌چنین داده درحال جریان در این شبکه‌ها زیاد است، نیاز به پهنای باند و انتقالات به حافظه بیشتر می‌شود. تحقیقات نشان داده است که انرژی مصرفی حافظه خارجی تقریباً ۲۰۰ برابر و زمان دسترسی به آن ۱۰ برابر حافظه داخلی است، که همین امر باعث افزایش انرژی مصرفی و عدم تعادل در توپولوژی مسیر داده می‌شود. یکی از راهکارهای اصلی برای کاهش انرژی مصرفی، افزایش استفاده مجدد از داده و کاهش تعداد دسترسی‌های حافظه خارجی است. اگر از پارامتر شبکه استفاده حداکثری شود، باعث کاهش در تعداد حرکت‌های داده و دسترسی‌ها به حافظه می‌شود. یکی از روش های استفاده مجدد از داده، زمانبندی سطح حلقه و اعمال تکنیک‌های کاشی‌بندی است. در این مقاله رابطه بین تعداد دسترسی‌های حافظه خارجی را در صورت استفاده از تکنیک کاشی‌بندی، به صورت ریاضی مدل کردیم. این مدل به صورت فرمول ریاضی است که می‌تواند تعداد دقیق دسترسی‌های DRAM را بر اساس پارامترهای شبکه و اندازه کاشی مدل کند. سپس از این مدل ریاضی برای یافتن پارامترهای بهینه در مساله بهینه سازی با هدف کم کردن دسترسی‌های حافظه خارجی، استفاده کردیم.**

کلمات کلیدی: **شبکه‌های عصبی CNN، انرژی مصرفی، حافظه خارجی DRAM، استفاده مجدد از داده، کاشی‌بندی**

نوع مقاله: پژوهشی

دریافت: ۱۴۰۳/۰۹/۰۵

بازنگری: ۱۴۰۳/۱۰/۱۹

پذیرش: ۱۴۰۳/۱۱/۱۸

نام نویسنده‌ی مسئول: دکتر مصطفی ارسالی صالحی نسب

نشانی نویسنده‌ی مسئول: ایران – تهران-کارگرشمالی – دانشکدگان فنی دانشگاه تهران- دانشکده برق و کامپیوتر- اتاق ۲-۷۱۲

۱- مقدمه

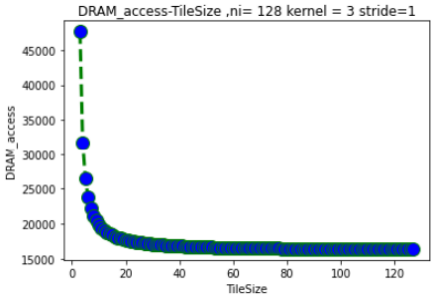
هوش مصنوعی تکنیکی است که به یک کامپیوتر یا ماشین امکان تقلید رفتار، عملکرد یا اعمال انسان را می‌دهد [۱]. حوزه‌های مختلفی مانند یادگیری‌عمیق و یادگیری ماشین، زیرمجموعه هوش مصنوعی هستند [۲]. یادگیری ماشین و یادگیری عمیق به عنوان ابزاری قوی، در تشخیص صدا، پردازش زبان طبیعی، پزشکی و بسیاری از حوزه‌های دیگر کاربرد دارند. در هر دو تکنولوژی، یادگیری از داده با استفاده از الگوریتم‌ها انجام می‌شود. با این تفاوت که یادگیری ماشین از تئوری‌های آماری برای یادگیری از داده استفاده می‌کند ولی یادگیری عمیق از شبکه‌های عصبی برای یادگیری، استفاده می‌کند. شبکه‌های عصبی در حوزه‌های مختلفی مانند پزشکی، تشخیص آریتمی‌های قلبی [۳]، تصحیح خودکار غلط‌های تایپی [۴]، پردازش تصویر و بسیاری از حوزه-های دیگر کاربرد دارد.

نمونه‌ای از شبکه‌های عصبی پرکاربرد، شبکه‌های عصبی CNN هستند. این شبکه‌ها به دلیل دقت بالایی که دارند، در کلاسه‌بندی تصویر[۱]، تشخیص اشیا[۲] و محلی‌سازی[۳]، کاربرد دارند. از این شبکه‌ها در دستگاه-های همراه، به وفور استفاده می‌شوند. این دستگاه‌ها نیاز به کارایی بالا و هم‌چنین توان مصرفی محدودی دارند. بنابراین نیاز است که اجرای شبکه‌های CNN بر روی این دستگاه‌ها از لحاظ مصرف توان و انرژی، بهینه شود [۵].

شبکه‌های CNN برای رسیدن به دقت بالاتر، نیاز به لایه‌های عمیق‌تر، تعداد پارامترهای بیشتر و در نتیجه حجم شبکه بزرگتری دارند. از همین رو، مجموعه داده شبکه‌های CNN بزرگ هستند. از طرفی تاثیر زمان دسترسی[۴] به حافظه‌های SRAM و DRAM بر کارایی و بهره‌وری انرژی[۵]، در مقایسه با تاثیر محاسبات بر آن‌ها، به شدت بیشتر بوده و غالب است [۶].

هم چنین باتوجه به اینکه حجم داده زیاد است، حرکت داده[۶] نیز در شبکه زیاد شده و نیاز به پهنای باند و انتقالات به حافظه نیز بیشتر می-شود. به دلیل اینکه پهنای باند حافظه محدود است، در نتیجه تاخیر در روند اجرای کار اتفاق می‌افتد و کارایی از دست می‌رود. تحقیقات نشان داده است که انرژی مصرفی و زمان دسترسی به حافظه خارجی DRAM به ترتیب تقریباً ۲۰۰ و ۱۰ برابر حافظه داخلی است. بنابراین کاهش استفاده از حافظه خارجیDRAM و انتقالات به آن و حرکت داده، می‌تواند بهبود انرژی مصرفی و کارایی را در پیش داشته باشد [۷] [۸] [۹] [۵].

برای حل چالش‌های مطرح شده و کاهش تعداد دسترسی‌های حافظه خارجی DRAM، تحقیقات قبلی با هدف بیشینه کردن استفاده مجدد از داده، در کلیه سطوح حافظه، انجام شده است. [۱۰] [۱۱] [۵] در یک گروه از تحقیقات انجام شده، برای استفاده مجدد از داده در سطح معماری سخت‌افزار وتوپولوژی مسیر داده، از حافظه‌های کم‌هزینه برای ذخیره داده‌های پرتکرار استفاده می‌شود که با این کار تعداد دسترسی به حافظه‌های پرهزینه، کمتر می‌گردد. در تحقیقات مربوط به سطح شبکه و بین لایه‌ای، با تغییر مسیر و ادغام[۷] پردازش‌های چند لایه باهم، از داده‌های بین لایه‌ها نیز، استفاده مجدد می‌شود [۱۲] [۱۳] [۱۴] [۵]. این امر باعث می‌شود که انتقالات به حافظه خارجی کاهش یابد. در دسته دیگری از تحقیقات قبلی، با طراحی زمانبند[۸]های در سطح حلقه و استفاده از تکنیک‌هایی مانند کاشی[۹]بندی، تعداد دسترسی‌های حافظه خارجی را کاهش می‌دهند [۱۵] [۹]. در [۱۶] برای کاهش دسترسی های حافظه DRAM، ترکیبی از پارتیشن کردن لایه‌ها به صورت تطبیقی و زمابندی محاسبات، پیشنهاد شده است. درواقع با کاشی کردن انواع داده‌ها، پارتیشن کردن لایه‌ها انجام شده است و با انحصاری کردن ترتیب لایه‌بندی شبکه CNN، زمانبندی را مدیریت کرده است. در [۱۳]، یک چارچوب[۱۰] طراحی شده است که برای شبکه ورودی و سخت افزار موردنظر، زمانبندی محاسبات طوری انجام شود که از نظر دسترسی حافظه خارجی بهینه باشد. برای یافتن پیکربندی بهینه سخت افزاری، یک فرمول ریاضی برای تعداد دسترسی‌های DRAM پیشنهاد شده است و سپس از این فرمول برای مساله بهینه‌سازی استفاده شده و در نهایت پارامترهای بهینه، به دست آمده است. اما درCNNFlow صحبتی از سایز کاشی بهینه و ارتباط آن با پارامترهای ساختاری هر لایه از شبکه، مانند اندازه کرنل، اندازه گام و ابعاد داده‌ها، گفته نشده است. درصورتی که باتوجه به نتایج شبیه‌سازی‌های ما که در شکل ۱ نشان داده‌ایم، مشخص است که اندازه کاشی می‌تواند در تعداد دسترسی‌های DRAM تاثیر داشته باشد.



شکل (۱): تاثیر اندازه کاشی در تعداد دسترسی‌های DRAM

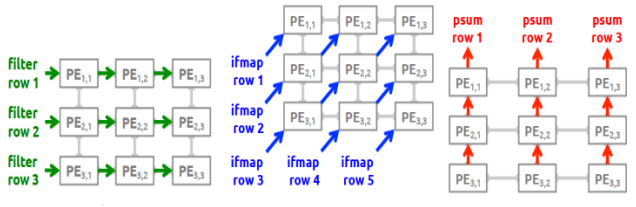
در این مقاله، در ابتدا یک فرمول ریاضی برای محاسبه تعداد دسترسی-های DRAM با توجه به پارامترهای شبکه، پیشنهاد کرده‌ایم. سپس با استفاده از این فرمول ریاضی، سایز بهینه اندازه کاشی برای هر لایه از شبکه CNN باتوجه به پارامترهای ساختاری لایه، به دست می‌آید. همچنین ارتباط بین سایز کاشی بهینه و پارامترهای شبکه را نیز به دست آوردیم که می تواند در بهینه سازی حافظه، انرژی مصرفی و تعداد دسترسی‌های حافظه خارجی، مفید باشد.

۲- کارهای پیشین

در راستای بهبود چالش‌ها، شتابدهنده‌هایی برای پیاده سازی شبکه‌های CNN طراحی شده‌اند. این شتابدهنده‌ها با تمرکز بر واحدهای محاسباتی و اتصالات آن‌ها، بهینه‌سازی حافظه و کنترلر برای کرنل‌های

مختلف، طراحی شده‌اند [۱۷] [۱۸]. بسیاری از این شتابدهنده‌ها از حافظه on-chip کنترل‌شده با نرم‌افزار[۱۱]، استفاده می‌کنند [۱۷] [۱۹]. ولی به دلیل اینکه شبکه‌هایCNN عمیق، معمولا از هزاران لایه با اشکال مختلف و بیش از ۵۰ میلیون پارامتر وزن، تشکیل شده‌اند، پیاده سازی آن‌ها با مشکل کمبود منابع محاسباتی و حافظه مواجه است و به ناچار از حافظه خارجی DRAM برای انتقال اطلاعات استفاده می شود. از طرفی زمان دسترسی و انرژی مصرفی حافظه DRAM بسیار بیشتر از حافظه داخلی است. بسیاری از تحقیقات با تمرکز بر کاهش تعداد دسترسی های DRAM و نقل و انتقالات به آن انجام شده است [۲۰]. در حالت کلی برای کاهش دسترسی‌های شبکه به حافظه خارجی DRAM، استفاده مجدد حداکثری از داده در هر سطح از حافظه، پیشنهاد شده است. در گروهی از تحقیقات با آگاهی از الگوهای استفاده داده، مسیرداده‌های مختلفی از داده ایجاد می کنند که در سطح توپولوژی و سخت افزار، استفاده مجدد از داده داشته باشند. باتوجه انتخاب نوع داده برای ذخیره و استفاده مجدد در سطح سخت‌افزار، چند مسیر داده ایجاد می‌شود. در [۱۱]، شتابدهنده‌ی سخت‌افزاری ASIC ارائه شده است که در مسیر داده آن، برای چهار واحد محاسباتی، فیلتر وزن یکسان و ورودی‌های مختلف اعمال می‌شود. درواقع این شتابدهنده، استفاده مجدد داده از فیلترهای وزن را در سطح واحدهای پردازشی دارد که موازی‌سازی و استفاده مجدد را افزایش داده است. در معماری [۱۵]، آرایه‌های دوبعدی سیستولیک برای محسابات الگوریتم CNN، در نظر گرفته شده است. در این معماری، هرکدام از واحدهای پردازشی امکان انجام یک ضرب و جمع[۱۲] را در هرکلاک دارند. این شتابدهنده تا زمانی که محاسبات یک گره به صورت کامل انجام شود، داده‌های میانی را ذخیره می‌کند. با این کار حرکت دیتای میانی و تعداد دسترسی‌های DRAM را کاهش می‌دهد. در [۲۱]، شتابدهنده‌ای طراحی شده است که در آن هم برای وزن‌ها و ورودی و هم برای خروجی‌های میانی، بافر قرار داده شده است. این بافرها به صورت پینگ پنگی هستند تا تعادل بین زمان انتقال داده و پردازش داده رعایت گردد و کارایی بهبود داشته باشد. در این کار استفاده مجدد از ورودی و وزن وجود دارد. تمرکز اصلی مسیرداده‌های روش‌های گفته شده اخیر، بر استفاده حداکثری از انواع داده و هم‌چنین کاهش حرکت داده‌ی میانی است.

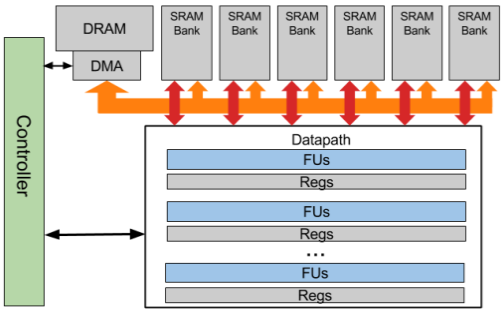
در معماری Eyeriss، [۵]، با هدف بیشینه کردن استفاده مجدد از داده‌های ورودی و وزن‌ها و کاهش حرکت حاصل‌جمع‌های جزئی، مسیر داده شکل ۲، بانام Row Stationary مطرح شده است. در این مسیرداده، نقشه ویژگی‌های ورودی(ifmap row1) به صورت ردیفی تقسیم‌بندی می‌شوند و ردیف‌هایی از وزن‌ها(filter row1) نیز از پیش بارگیری شده و در حافظه‌های واحدهای پردازشی قرار دارند. نقشه ویژگی‌های ورودی نیز به صورت مورب وارد شده‌اند، سپس در هر ستون یک حاصل جمع جزئی، تولید می‌شود.



شکل (۲): مسیرداده Row Stationary [۵]

در دسته دیگری از تحقیقات برای کاهش تعداد دسترسی‌های حافظه از زمانبندی کردن محاسبات بهره می‌برند. زمانبندی کردن محاسبات یک مفهوم مهم در بهینه‌سازی کارایی شبکه‌های CNN است و مشخص می‌کند که کجا و چه زمانی هرکدام از عملیات الگوریتم CNN، قرار است اتفاق بیفتد. زمانبندی کردن می‌تواند با اهداف مختلف از جمله حداکثرکردن استفاده از داده، موازی‌سازی بیشتر، کم‌ترکردن حافظه و بهبود انرژی انجام شود. در اینجا زمانبندهای با هدف کاهش حافظه و تعداد دسترسی‌های آن، مطرح است.

MemFlow، چارچوبی است که با هدف استفاده حداکثری از داده و حداقل شدن دسترسی DRAM، برای محاسبات، زمانبند بهینه ارائه می‌کند [۹]. درMemFlow که معماری آن در شکل ۳ آورده شده است، جریان داده را در سه مرحله با استفاده ازکنترلر،کنترل می‌کنند. در مرحله اول داده را کاشی می‌کنند. در مرحله دوم، ترتیبی ازمحاسبات تحت عنوان Macro Node را ایجاد می‌کنند که این محسابات کل فضای SRAM را گرفته و درنتیجه باید به ترتیب انجام شوند. در مرحله سوم، به هر کدام از کاشی‌های داده، تعدادی از بانک‌های SRAM اختصاص می‌دهند. برحسب پارامترهای استخراجی از این سه مرحله، تعداد دسترسی‌های حافظه خارجی DRAM را فرمول می‌کنند و سپس توسط چارچوب پیشنهادی خود، پارامترهای اندازه کاشی، ترتیب Macro Nodeها و تعداد بانک‌های SRAM را مشخص می‌کنند. کنترلر نیز با در نظر گرفتن این پارامترها، مسیر داده را کنترل و زمانبندی می‌کند. اما مشکلی که MemFlow دارد، زمان طولانی برای مشخص شدن خروجی‌ها و مقادیر بهینه است که در CNNFlow این مشکل حل شده است.



شکل (۳): معماری MemFlow [۹]

CNNFlow یک چارچوب است که به صورت اتوماتیک محاسبات الگوریتم CNN را با هدف داشتن بیشترین استفاده مجدد از داده،

زمانبندی می‌کند [۱۳]. باتوجه به اینکه هرلایه از کانولوشن از چند حلقه با عمق بالا تشکیل شده است، در CNNFlow باتوجه به نقشه محاسباتی آن، در ابتدا داده‌های مربوط به حلقه‌ها پارتیشن شده و در کاشی‌های سطح SRAM گروه می‌شوند. سپس هریک از کاشی‌های داده سطح SRAM را به کاشی‌های سطح واحدهای محاسباتی پارتیشن می‌کند. در ادامه با آگاهی از انواع حلقه‌های لایه کانولوشن و اینکه این حلقه‌ها در کنار هم چه الگوهای استفاده مجدد از داده را خواهند داشت، یک فرمول ریاضی دقیق برای تعداد دسترسی‌های DRAM و SRAM مدل می‌کند. ترتیب حلقه‌ها و پیکربندی SRAM جز پارامترهای این فرمول هستند. در ادامه چارچوبی توسعه داده است که با هدف کم کردن تعداد دسترسی‌های DRAM با استفاده از فرمول ریاضی، مساله بهینه-سازی را حل کرده و مقادیر بهینه را ارائه می‌کند. هم‌چنین به دلیل اینکه فضای حالت مورد جستجو بسیار حجیم و زمانبر است، در CNNFlow با استفاده از تکنیک‌های اتوماتیک کردن پارتیشن فضای حافظه و انتخاب سایز کاشی مناسب فضای حالت را کوچک‌تر و زمان اجرا را نیز کاهش داده است. CNNFlow با داشتن سخت افزار و حافظه ثابت، به عنوان ورودی مساله، ترتیب حلقه ها و پارتیشن بندی حافظه بهینه را در خروجی به دست می دهد و نتیجه ای برای مقدار بهینه اندازه کاشی برای هر لایه، ندارد. از طرفی پارامترهای ساختاری شبکه مانند اندازه کرنل، گام و ابعاد داده برای هر لایه را نیز در مدل ریاضی خود در نظر نگرفته است. در SmartShuttle برای کاهش تعداد دسترسی‌های DRAM یک الگوریتم مبتنی بر تحلیل ارائه می‌کند که برای هر لایه، یک سایز کاشی مناسب و یک روش زمابند مناسب ارائه می کند، اما این تحلیل مبتنی بر روش‌های تجربی است و هیچ پایه و قانون ریاضی ندارد [۱۶]. در [۲۲] شتابدهنده‌ای مبتنی بر FPGA برای شبکه‌های CNN پیشنهاد کرده‌اند که در آن از تکنیک ادغام لایه‌ها و کاشی‌بندی خروجی، برای بهره‌وری حافظه استفاده شده است. در این مقاله، نقشه ویژگی خروجی به شبکه‌هایی تحت عنوان کاشی تقسیم می‌شوند و سپس عناصری از ورودی را که پارامترهای هرکاشی خروجی را تولید کرده‌اند به دست می‌آورند و توسط شتابدهنده پردازش مربوط به تولید کل پارامترهای کاشی خروجی، یکجا انجام شده و خروجی‌ها تولید می‌شوند. باتوجه به اینکه از ادغام لایه‌ها به همراه کاشی‌بندی استفاده شده است، در استفاده از حافظه خارجی بهبود داشته‌اند و هم-چنین در پردازش‌ها موازی‌سازی داشته‌اند. در مقاله مذکور، کاشی‌بندی خروجی انجام شده است و باتوجه به اندازه آن، سایز کاشی ورودی استخراج شده است و صحبتی در رابطه با اندازه کاشی ورودی و هم-چنین یک مدل ریاضی برای بررسی شاخص‌های حافظه، انجام نشده است. در [۲۳] چارچوبی ارائه شده است که از طریق تکنیک کاشی‌بندی حلقه، اجرای شبکه CNN را بهبود می‌دهند. در این چارچوب، هم داده-های ورودی و خروجی کاشی‌بندی شده‌اند و هم در سطح حلقه‌های پردازشی شبکه‌های CNN، کاشی‌بندی انجام شده است. با استفاده از کاشی‌بندی سطح حلقه، محاسبات را به بلوک‌های کوچک‌تری تقسیم می‌کنند تا استفاده بهتری از حلفظه و منابع پردازشی داشته باشند. این چارچوب با کاهش انتقالات حافظه، مصرف انرژی را کاهش می‌دهد و برای شبکه‌های عصبی مختلف قابلیت پیکربندی دارد. در [۲۴]، شتابدهنده‌ای برای شبکه‌های عصبی CNN ارائه کرده‌اند که از روش کاشی‌بندی مسیر داده استفاده کرده‌اند. هدف اصلی این شتابدهنده، افزایش کارایی و جلوگیری از هدر رفتن منابع سخت‌افزاری مانند واحدهای پردازشی است. با استفاده از این شتابدهنده، امکان کاشی‌بندی داده‌های ورودی، خروجی و فیلترهای وزن وجود دارد. ویژگی مهمی که این شتابدهنده دارد، تعداد پردازنده‌های فعال بسته به نیاز می‌توانند متغیر باشند و بهره‌وری را افزایش می‌دهند. هدف اصلی شتابدهنده، استفاده حداکثری از واحدهای پردازشی با استفاده از تکنیک کاشی‌بندی سطح مسیر داده است. در [۲۵]، یک روش بهینه‌سازی ارائه شده است که هدف اصلی آن، پردازش موثر تنسورهای داده است. درواقع در این کار با استفاده از کاشی‌بندی تنسورها و تبدیل کردن آن‌ها به بلوک‌های کوچک‌تر، فقط داده‌های غیرصفر را وارد کاشی و پردازش می‌کنند و بدین ترتیب استفاده از حافظه و حجم محاسبات را مدیریت می‌کنند. در مقاله پیشنهادی، ما در ابتدا یک مدل ریاضی با استفاده از پارامترهای پیکربندی لایه مانند اندازه کرنل، گام، تنسور ورودی و اندازه کاشی، برای تعداد دسترسی‌های DRAM و تعداد استفاده مجدد از داده تعریف کردیم. سپس از این مدل برای مساله بهینه‌سازی استفاده کرده و سایز کاشی بهینه را به دست آوردیم. هم چنین ارتباط سایز کاشی بهینه و پارامترهای ساختاری یک لایه از شبکه، برای کاهش حرکت داده و استفاده از حافظه خارجی را نیز استخراج کردیم. در انتها نیز شبکه عصبی موبایلنت را بر روی شتابدهنده Eyeriss پیاده‌سازی کردیم و با استفاده از مدل ریاضی پیشنهادی‌مان، اندازه کاشی بهینه را به دست آوردیم و برای هر یک از لایه‌ها، تعداد دسترسی‌های DRAM را برای دو حالت استفاده از کاشی‌بندی و حالت پایه به دست آوردیم.

۳- انگیزه و مفاهیم اولیه

در این قسمت ابتدا توضیحاتی در رابطه با انگیزه و نوآوری شرح داده شده است. سپس در قسمت بعدی مفاهیم پایه‌ای طرح، آورده شده است.

۳-۱- انگیزه و نوآوری

همانطور که در بخش‌های قبلی اشاره شد، شبکه‌های عصبی CNN به دلیل دقت بالایی که دارند، در بسیاری از حوزه‌ها کاربرد دارند. از طرفی برای افزایش دقت و کارایی بیشتر لازم است که حجم داده و تعداد لایه-ها بیشتر شود که در این صورت شبکه عمیق‌تر و حجیم‌تر می‌شود. حجم زیاد شبکه و مجموعه داده زیاد باعث می‌شود که حرکت داده در شبکه زیاد شود و نیاز به حافظه بیشتر از حافظه داخلی on-chip باشد و در نتیجه انتقالات به حافظه خارجی DRAM بیشتر شود. از طرفی زمان دسترسی و انرژی مصرفی حافظه خارجی DRAM بسیار بیشتر از

حافظه داخلی است و راهکاری که بتواند این دسترسی‌ها را کاهش دهد، می‌تواند تاثیر زیادی در بهبود انرژی و زمان اجرا داشته باشد.
در جهت کاهش نقل و انتقالات به حافظه DRAM، در این مقاله، اقدامات زیر انجام شده است:

- ارائه یک مدل ریاضی برای محاسبه تعداد دسترسی‌های حافظه خارجی DRAM براساس پیکربندی ساختار شبکه.
- استفاده از مدل پیشنهادی در بهینه‌سازی پارامترهای اندازه کاشی و ساختار شبکه، برای حداقل کردن تعداد دسترسی‌های DRAM شبکه CNN هدف.
- کاهش زمان جستجوی فضای حالت برای یافتن پارامترهای بهینه.
-

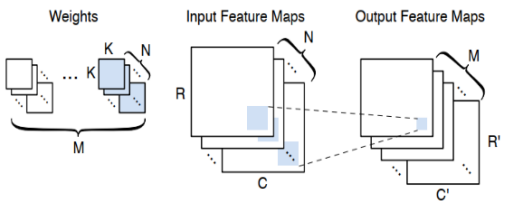
۳-۲- مفاهیم پایه‌ای

- در این قسمت مفاهیم پایه‌ای شرح داده شده است.

۳-۲-۱- شبکه‌های عصبی CNN

شبکه‌های‌عصبی CNN، معمولا از سه لایه جمع‌آوری[۱۳]، تماماً متصل[۱۴]و کانولوشنی تشکیل شده است.

- لایه‌های کانولوشنی: بلوک‌های اصلی سازنده شبکه‌های CNN هستند و بیشتر بار محاسباتی شبکه، مخصوص این لایه‌ها است. کار اصلی این لایه‌ها استخراج سلسه مراتبی از ویژگی‌ها از روی تصویر ورودی است.
  - عملیات کانولوشن: لایه‌های کانولوشنی مجموعه‌ای از فیلترهای وزن را بر داده ورودی اعمال می‌کنند که معمولا ابعاد فیلترهای وزن کوچک است. باتوجه به شکل ۴ ، هر کدام از فیلترها بر روی تصویر ورودی حرکت کرده و ضرب نقطه‌ای بین مقادیر وزن و ورودی انجام شده و خروجی ایجاد می‌شود، که در این فرایند، نقشه ویژگی[۱۵] ایجاد می‌شود.



شکل (۴): عملیات کانولوشن در یک لایه از شبکه CNN [۱۰]

  - نقشه‌های ویژگی: هرکدام از فیلترها ویژگی‌های مختلفی را در ورودی تشخیص می‌دهد. از جمله این ویژگی‌ها می‌توان لبه‌ها، بافت یا الگوی خاصی را نام برد. نقشه‌های به دست آمده نشان‌دهنده‌ی حضور این ویژگی‌ها در مکان‌های فضایی مختلف در تصویر ورودی است.
  - گام[۱۶]: تعداد پیکسل‌هایی که فیلتر بر روی تصویر حرکت می‌کند. اگر گام برابر x باشد، به این معنی است که فیلتر هر بار x پیکسل حرکت می‌کند. گام‌های بزرگتر منجر به نقشه‌های ویژگی کوچکتر می‌شود.
  - حاشیه‌گذاری[۱۷]: اضافه کردن پیکسل‌های اضافی در اطراف حاشیه تصویر ورودی است. درواقع این کار به حفظ ابعاد، پس از عمل کانولوشن، کمک می‌کند.
  - نمایش ریاضی: خروجی لایه کانولوشنی از رابطه زیر به دست می‌آید:

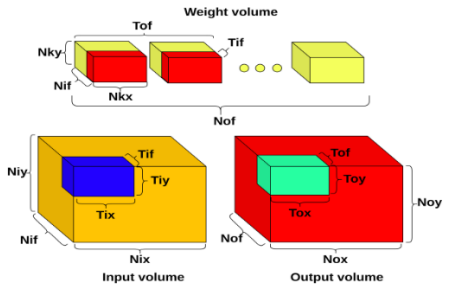
$$(i,j)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(i+m,j+n)\times k(m,n) \qquad (1)$$

  - در معادله ۱ ، O مقدار خروجی، I تصویر ورودی، K فیلتر وزن و M و N ابعاد کرنل وزن هستند.
- لایه‌های جمع‌آوری[۱۸]: یکی از لایه‌های اصلی شبکه CNN است. هدف اصلی آن کاهش ابعاد نقشه‌های ویژگی است. هم‌چنین تعداد پارامترها و محاسبات را نیز در شبکه کاهش می‌دهد و در عین حال اطلاعات مهم را حفظ می‌کنند. هم‌چنین لایه‌های جمع‌آوری، پیچیدگی‌های مدل را کاهش داده و از رخ دادن بیش برازش[۱۹] جلوگیری می‌کند.
- لایه‌های تماماً متصل[۲۰]: لایه‌هایی هستند که در آن‌ها هر نورون از یک لایه به کل نورون‌های لایه قبلی وصل شده است. هرچه عمق لایه‌ها بیشتر باشد، شبکه روابط پیچیده‌تری را می‌تواند آموزش ببیند. معمولا در شبکه‌های CNN پس از اینکه در لایه‌های کانولوشنی، ویژگی‌ها استخراج شدند، توسط لایه‌های تماماً متصل، کلاسه‌بندی می‌شوند.

۳-۲-۲- مفهوم کاشی‌بندی

کاشی‌بندی تکنیکی است که برای بهبود کارایی و بهینه کردن استفاده از حافظه، در شبکه‌های CNN استفاده می‌شود. با توجه به شکل۵، روش کار به این صورت که داده‌های تنسورهای شبکه، به بلوک‌های کوچکتری در قالب کاشی‌های قابل مدیریت، پارتیشن می‌شوند که این بلوک‌ها هم در حافظه بهتر جاگیر می‌شوند و هم از منابع محاسباتی استفاده بهتری دارند. داده ورودی که سه‌بعدی هستند، در کاشی‌های سه بعدی کوچک‌تری پارتیشن شده‌اند و به دلیل اینکه این ورودی‌ها با فیلترهای وزن هم‌عمق پردازش می‌شوند، فیلترهای وزن نیز با عمق برابر عمق کاشی ورودی،Tif، پارتیشن می‌شوند. با استفاده از این تکنیک می‌توان با واکشی داده‌های نزدیک به هم، از اصل محلیت مکانی بهره برد و هم‌چنین با ذخیره داده‌هایی که قرار است چندین بار از آن‌ها استفاده شود، از محلیت زمانی بهره برد. درواقع روشی است که هم استفاده از

حافظه DRAM را کاهش داده و باعث بهبود پهنای باند می‌شود و هم استفاده مجدد داده را افزایش می‌دهد. از طرفی دیگر با نزدیک کردن داده به واحدهای محاسباتی، زمان اجرا و کارایی را نیز بهبود می‌دهد که در ادامه شرح داده شده است.



شکل (۵):کاشی‌بندی داده‌های وزن، ورودی و خروجی [۱۴]

از تکنیک کاشی‌بندی برای بهبود در موارد زیر، استفاده می‌شود.

- بهره‌وری حافظه[۲۱]: زمانی که داده‌ها کاشی‌بندی می‌شوند، ماتریس داده‌های با حجم بالا به ماتریس‌های کوچکتر شکسته می‌شود و درواقع حجم کمتری از داده قرار است پردازش شوند و نیاز به حافظه داشته باشند. از طرفی به دلیل محلیتی که با استفاده از کاشی‌بندی اتفاق می‌افتد، فقط داده‌های ضروری که قرار است به زودی پردازش شوند در کاشی قرار دارند و استقرار داده‌های غیرضرور کاهش می‌یابد.
- محلیت داده[۲۲]: با استفاده از این تکنیک می‌توان با واکشی داده‌های نزدیک به هم، از اصل محلیت مکانی بهره برد و هم‌چنین با ذخیره داده‌هایی که قرار است چندین بار از آن‌ها استفاده شود، از محلیت زمانی بهره برد و درواقع زمان دسترسی به حافظه را تسریع می‌کند.
- استفاده مجدد از داده: کاشی‌بندی این امکان را می‌دهد که داده‌هایی که به صورت یک بلوک هستند در حافظه محلی، حفظ شوند و از داده‌هایی که داخل کاشی هستند، حداکثر استفاده مجدد شود و به این صورت تعداد دسترسی به حافظه کند خارجی را کاهش می‌دهد.

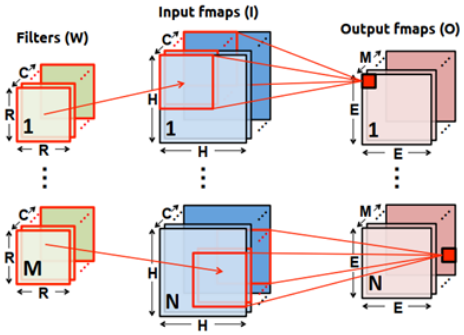
۳-۲-۳- استفاده مجدد داده[۲۳]

ساختار محاسباتی لایه‌های CNN از ۶ لایه تو در تو تشکیل شده است. باتوجه به شکل ۶ و ساختار حلقه‌ای CNNها، فضای حالت بسیار بزرگی برای انتخاب ترتیب حلقه‌ها، ترتیب محاسبات، پارتیشن کردن داده و موازی‌سازی به وجود می‌آید. از طرفی باتوجه به اینکه بیشتر از ۹۰٪ عملیات در شبکه‌های CNN، عمل کانولوشن است و به دلیل ماهیت عمل کانولوشن و ساختار حلقه‌ای لایه‌های CNN، امکان استفاده مجدد از داده عامل تاثیرگزاری برای بهبود کارایی و حافظه، می‌تواند باشد. در همین جهت باتوجه به شکل ۷ و الگوی محسابات، مدل‌های استفاده مجدد از داده آورده شده است:

- کانولوشنی: با در نظر گرفتن R برای ابعاد فیلتر وزن، از هر پیکسل از نقشه ویژگی ورودی، تقریبا، R*R مرتبه استفاده می‌شود، هم‌چنین از هر المان از فیلتر وزن، به اندازه E*E بار(E ابعاد نقشه ویژگی خروجی است) استفاده می‌شود.
- فیلتر: از هر فیلتر وزن، به اندازه تعداد کانال‌های ورودی، دوباره استفاده می‌شود.
- نقشه ویژگی: اگر به تعداد M فیلتر در یک لایه باشد، از هر نقشه ویژگی ورودی، به اندازه M مرتبه دوباره استفاده می‌شود.

```
for (no = 0; no < Nof; no ++)
  for (y = 0; y < Noy; y ++)
    for (x = 0; x < Nox; x ++)
      for (ni = 0; ni < Nif; ni ++)
        for (ky = 0; ky < Nky; ky ++)
          for (kx = 0; kx < Nkx; kx ++)
            pixel_L(no; x, y) += pixel_L-1(ni; S × x + kx, S × y + ky) × weight_L-1(ni, no; kx, ky);
  pixel_L(no; x, y) = pixel_L(no; x, y) + bias(no);
```

شکل (۶): حلقه‌های تو در تو در CNN [۲۰]



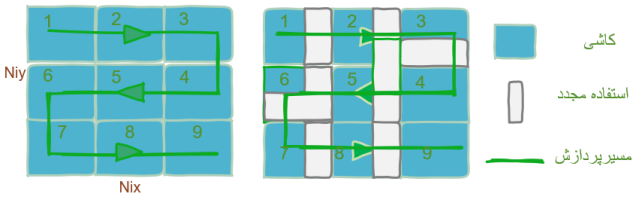
شکل (۷): ساختار بلوکی محسابات یک لایه از CNN [۵]

۴- مدل ریاضی و روش بهینه‌سازی پیشنهادی

در این بخش در ابتدا مدل ریاضی شرح داده شده است. سپس توضیحاتی در رابطه با مساله بهینه سازی و هم‌چنین تکنیک‌های کاهش فضای حالت و شبیه‌سازی، آورده شده است.

۴-۱- مدل ریاضی کاشی‌بندی

یک لایه از شبکه CNN با ابعاد دو بعدی در شکل ۸ آورده شده است.



شکل (۸): نواحی هم‌پوشانی داده حین پردازش داده‌ها و استفاده از کاشی‌بندی

کل تصویر ورودی با ابعاد Nix.Niy در حافظه خارجی DRAM ذخیره شده و قرار است با یک فیلتر وزن با ابعاد K.K کانوالو شود. در صورتی که از کاشی‌بندی برای پردازش داده استفاده شود و مسیر پردازش داده مانند شکل۸، در نظر گرفته شود. در ابتدا مجموعه داده در قالب یک کاشی از حافظه DRAM خوانده شده و در حافظه داخلی نوشته می‌شود. در پردازش بلوک بعدی، یک ناحیه‌ای از حافظه داخلی یا کاشی قبلی، قرار است دوباره استفاده شود، برای همین در حافظه داخلی باقی مانده و مقادیر جدید وارد کاشی می‌شوند. در نتیجه نواحی‌ای که با رنگ سفید در شکل ۸ مشخص شده است، نواحی‌ای هستند که قرارست دوبار از آن‌ها استفاده شود و در حافظه داخلی باقی می‌مانند.

اگر اندازه کاشی t.t در نظر گرفته شود و اندازه کرنل k.k و گام برابر s در نظر گرفته شود، مساحت ناحیه هم‌پوشان که قرار است دوباره استفاده شود از رابطه۲ به دست می‌آید:

$$overlap = t \times (k - s) \qquad (۲)$$

از طرفی تعداد پیکسل خروجی تولید شده با توجه به پیکربندی شبکه، از رابطه3 به دست می‌آید:

$$N_o = \frac{Ni - k}{s} + ۱ \qquad (۳)$$

اگر سایز کاشی t در نظر گرفته شود، تعداد پیکسل‌های خروجی تولید شده از داده‌های ورودی کاشی‌شده، از رابطه ۴ به دست می‌آید:

$$N_t = \frac{t - k}{s} + ۱ \qquad (۴)$$

می‌توان از رابطه ۳و۴ استنتاج کرد که تعداد دفعاتی که نیاز است تا داده‌های ورودی، کاشی شوند، از رابطه 5 به دست می‌آید:

$$\#tile_{use} = \left(\frac{N_o}{N_t}\right)^۲ \qquad (۵)$$

نکته قابل توجه اینست که در زمان انتقال داده از حافظه DRAM، باید تاثیر نواحی هم‌پوشان در نظر گرفته شود.

باتوجه به روابط ۲ تا ۵، می‌توان نتیجه گرفت که تعداد دسترسی‌های حافظه خارجی DRAM برای پردازش یک تصویر ورودی با استفاده از کاشی‌بندی، از رابطه۶ به دست می‌آید:

$$\#DRAM_{access} = t^۲ + t \times (t - (k - s)) \times \left((\frac{Ni - k + s}{t - k + s})^۲ - ۱ \right) \qquad (۶)$$

دررابطه ۶، ni، سایز ورودی، k سایز فیلتر، t سایز کاشی و s اندازه گام است. نکته‌ای که وجود دارد، این است که به دلیل هم‌پوشانی بین ورودی‌هایی که قرار است کرنل وزن در آن‌ها ضرب شود، در کاشی از داده‌های موجود، استفاده مجدد می‌شود. اگر باتوجه به گام، مسیر حرکت کانولوشن افقی باشد، در این صورت هر دفعه به اندازه k-s ستون از داده‌ها در کاشی باقی مانده و به اندازه t-(k-s) ستون داده، از حافظه DRAM خوانده می‌شود. اگر حرکت به صورت عمودی باشد، به همین مقدار ردیف در کاشی باقی می‌ماند.

۴-۲- کاهش فضای حالت

با استفاده از رابطه6 و در نظر گرفتن تعداد دسترسی‌های DRAM به عنوان مساله بهینه‌سازی، می‌توان اندازه کاشی بهینه را به دست آورد. اما اگر سایز ورودی Ni و سایز کرنل وزن k باشد، اندازه کاشی می‌تواند از k تا Ni باشد. که برای ابعاد بزرگ مساله و حجم بزرگ شبکه، ممکن است وقت‌گیر باشد و نیازی نیست که کل اعداد صحیحِ ممکن، ارزیابی شوند.

$$\text{if } [N_o \% N_t \neq ۰] \quad \text{then result padding} \qquad (۷)$$

اگر تعداد خروجی‌های تولیدشده باتوجه به اندازه کاشی، طوری باشد که بر تعداد پیکسل‌های خروجی کل، بخش‌پذیر نباشد، در پردازش داده‌ها باید برای آخرین کاشی افقی، از تکنیک حاشیه‌گذاری استفاده شود. زیرا به علت از دست رفتن صحت داده، نمی‌توان داده واقعی در کاشی جانمایی کرد. از همین رو، کاشی‌هایی در فضای حالت در نظر گرفته شده‌اند که تعداد پیکسل خروجی تولید شده توسط آن‌ها، بخش‌پذیر بر کل پیکسل‌های خروجی باشد که نیازی به حاشیه‌گذاری نباشد.

۴-۳- صحت مدل ریاضی

در قسمت‌های قبلی تعداد دسترسی‌های حافظه خارجی DRAM، بر حسب اندازه کاشی و پارامترهای ساختاری شبکه، مدل شد. از مدل‌های ریاضی برای به دست آوردن اندازه کاشی بهینه استفاده شد با شرط اینکه اندازه کاشی طوری انتخاب شود که نیاز به حاشیه‌گذاری نباشد و هم‌چنین تعداد دسترسی‌های DRAM حداقل باشد.

ابتدا از روی ساختار شبکه اندازه کاشی‌های مجاز را استخراج می‌کنیم. سپس با توجه به این نکته که هر چه اندازه کاشی بزرگ‌تر باشد، تعداد دسترسی‌های DRAM کم‌تر می‌شود، از اندازه کاشی کوچک شروع کرده و کاشی‌ای را به عنوان بهینه انتخاب می‌کنیم که بعد از آن، با بزرگ‌تر شدن کاشی، تعداد دسترسی‌ها DRAM کم‌تر از ۱۰٪ کاهش داشته باشد. قابل توجه است که همه این کارها به صورت اتوماتیک انجام می‌شود.

ما صحت فرمول دسترسی به حافظه DRAM را با استفاده از دو روش ارزیابی کردیم.

- روش اول

در این روش، یک شبیه‌ساز طراحی کردیم که یک لایه کانولوشنی را به

صورت مرحله‌به‌مرحله با استفاده از کتابخانه NumPy در پایتون پیاده‌سازی می‌کند. درواقع یک ماژول "Conv" توسعه داده شده است که عملیات کانولوشن را به صورت دقیق انجام می‌دهد. این ماژول با دریافت یک تصویر ورودی و یک فیلتر، خروجی‌های صحیح تولید می‌کند. همچنین، یک ماژول دیگر به نام "TileConvModule" طراحی کردیم که از ماژول کانولوشن برای پردازش داده‌های موجود در یک کاشی (Tile) و هسته وزن مرتبط با آن استفاده می‌کند.

برای بررسی صحت فرمول دسترسی به حافظه، مطابق با جریان پردازش نشان‌داده‌شده در شکل ۸، داده‌ها به صورت کاشی به کاشی به ماژول "TileConvModule" منتقل می‌شوند تا عملیات محاسباتی انجام شود. از یک شمارنده که در ابتدا روی صفر تنظیم شده است، برای ردیابی تعداد دفعات دسترسی به حافظه استفاده کردیم. هر بار که داده‌های جدید از تصویر ورودی به کاشی بارگذاری می‌شود، مقدار شمارنده یک واحد افزایش می‌یابد. مقدار نهایی شمارنده دقیقاً با نتیجه محاسبه‌شده از فرمول مطابقت دارد، که صحت فرمول را تأیید می‌کند. علاوه بر این، صحت عملکرد عملیات کانولوشن نیز به طور کامل بررسی و تأیید شد.

- روش دوم

در این روش، پیاده‌سازی مرجع[26] را که عملیات یک لایه کانولوشنی را به صورت مرحله‌به‌مرحله شبیه‌سازی می‌کرد، سفارشی‌سازی کردیم. این کد سفارشی‌شده را برای انجام محاسبات روی داده‌های کاشی‌شده تنظیم کردیم. همچنین، یک شمارنده به پیاده‌سازی اضافه کردیم تا تعداد دسترسی‌های حافظه را ردیابی کنیم. نتایج این روش نیز با فرمول ما مطابقت داشت و صحت آن را تأیید کرد.

در این بخش، نتایج آورده شده است.

۵- رابطه نتایج

۵-۱- تعداد دسترسی‌های DRAM بر حسب اندازه کاشی

در شکل ۹ نمودار تعداد دسترسی‌های حافظه DRAM را برای یک لایه از شبکه با پیکربندی‌های متفاوت، رسم کرده‌ایم. برای مثال در حالت (الف)، ورودی 60*60، اندازه کرنل وزن ۵*۵ و اندازه گام برابر ۱ می‌باشد. هم‌چنین در سه حالت دیگر نیز پیکربندی‌های دیگری از لایه درنظر گرفته شده است. باتوجه به این نمودارها، مشخص است که با افزایش اندازه کاشی، تعداد دسترسی‌های DRAM کاهش می یابد. می‌توان استنتاج کرد که وجه مشترک همه این نمودارها نزولی بودن تعداد دسترسی‌های DRAM با افزایش اندازه کاشی است. هم چنین در همه این نمودارها، بعد از یک حدی، با افزایش اندازه کاشی، تعداد دسترسی‌ها تقریبا کاهشی نداشته و به صورت خط افقی است. درواقع بعد از یک مقداری، افزایش اندازه کاشی تاثیری در تعداد دسترسی‌ها ندارد و فقط اندازه حافظه بیشتر می‌شود. با استناد به همین موضوع، اندازه کاشی بهینه از بین کاشی‌های مجاز، طوری انتخاب می‌شود که بعد از آن و با بزرگتر شدن اندازه کاشی، تعداد دسترسی‌ها کمتر از ۱۰٪ کاهش داشته باشد.

۵-۲- الگوریتم پیشنهادی برای انتخاب کاشی بهینه

با توجه به پیکربندی یک لایه، از الگوریتم زیر برای انتخاب کاشی بهینه، به صورت اتوماتیک استفاده می‌شود. قابل توجه است که فضای حالت اندازه کاشی در محدوده $k \le tile \le ni$ در نظر گرفته می شود.

Function Optimum_Tile (Ni, K, S):

- Find a valid tile set with no padding with (Ni, k, s):
  - $Tiles \in \{(No \,\%\, Nt) != 0\}$
  - $valid_{tiles} = \{t_1, t_2, \dots t_m\}$
- Find total number of DRAM accesses for valid tiles with equation 6:
  - $\#DRAM_{acc} = \{D_{t1}, D_{t2}, .., D_{tm}\}$
- Choose optimum tile when:
  - $t_{opt} = t_n \;\; when \;\; D_{t(n)} - D_{t(n+1)} \le 10\% * D_{tn}$

در الگوریتم در ابتدا با استفاده از رابطه ۷، فضای حالت کاهش یافته و کاشی‌های مجاز برای نداشتن حاشیه‌گذاری، استخراج می‌شود. سپس با استفاده از فرمول ریاضی معادله (۶)، تعداد دقیق دسترسی‌های DRAM برای هر یک از کاشی‌ها به دست می‌آید و در انتها با استفاده از معادله۸، اندازه کاشی بهینه به دست می‌آید.

$$t_{opt} = t_n \text{ when } D_{t(n)} - D_{t(n+1)} \le \%10 \times D_{t(n)} \qquad (8)$$

برای پیکربندی لایه‌های مختلف، اندازه کاشی بهینه را به دست آوردیم که در جدول ۱، گزارش کردیم.

جدول (۱): اندازه کاشی بهینه برای پیکربندی‌های مختلف

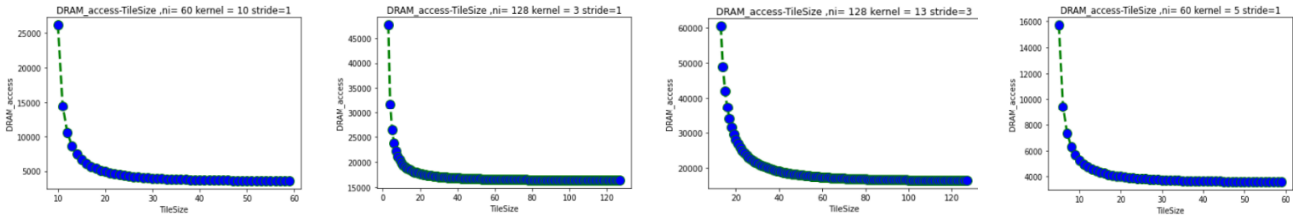
پیکربندی شبکه			کاشی های مجاز	کاشی بهینه
S	K	Ni		
7	12	256	{12,40,54,250}	40
1	9	128	{9,10,11,12,13,14,16,18,20,23,28,32,38,48,68,128}	16
2	9	128	{9,11,13,15,17,19,27,31,37,47,67,127}	15
4	12	128	{12,16,20,28,32,48,68,128}	28
1	29	128	{29,30,32,33,38,48,53,78,128}	78
1	29	256	{29,30,31,32,34,40,47,66,85,104,142,256}	85
2	3	49	{3,5,7,9,13,17,25,49}	5
1	3	62	{3,4,5,6,7,8,12,14,22,32}	5

باتوجه به جدول ۱، مشخص است که اندازه کاشی بهینه از مقدار ابعاد ورودی(ni) فاصله دارد و اینطور نیست که با بزرگتر شدن ابعاد ورودی، حتماً مقدار آن بزرگتر شود. بلکه مقدار آن، نزدیک به اندازه کرنل(k) است و فاصله زیادی با آن ندارد. در همین جهت برای بررسی تاثیر اندازه ورودی بر کاشی بهینه، با در نظر گرفتن مقدار ثابت برای اندازه کرنل و گام(s)، برای پیکربندی‌های مختلف با ابعاد ورودی متفاوت، مقدار کاشی بهینه را به دست آوردیم. وجه مشترک در همه آن‌ها این بود که اندازه

ورودی تاثیر چندانی در مقدار کاشی بهینه نداشته و در عوض اندازه کرنل در مقدار بهینه کاشی تاثیر دارد.

در شکل ۱۲، نمودارهای مربوط به اندازه کاشی بهینه را نسبت به اندازه سایز ورودی، رسم کرده‌ایم. در حالت (الف)، برای زمانی که سایز پنجره ورودی کمتر از۲۰۰ و اندازه کرنل ۲۰ و گام نیز برابر یک است، اندازه کاشی بهینه رسم شده است. خط قرمز رنگ افقی نیز، محلی است که اندازه کاشی بهینه برابر با ۴ برابر اندازه کرنل باشد. در این نمودارها تعداد کاشی‌های بهینه‌ای که اندازه آن‌ها کمتر از ۴ برابر اندازه کرنل باشد، به صورت درصدی محاسبه شده است که در این مثال ٪۷۱ می‌باشد. در حالت‌های (ب) و (ج) نیز، ٪۷۸ و ٪۷۳ اندازه کاشی بهینه کمتر از ۴ برابر اندازه کرنل است.

برای برخی از پیکربندی ها، به دلیل اینکه اندازه کاشی های مجاز که استفاده از آن‌ها، منجر به حاشیه‌گذاری نشوند، بسیار کم است. در نتیجه اندازه کاشی بهینه بسیار نزدیک به سایز ورودی و درواقع عدد بزرگی است که این موارد استثنا محسوب می شوند. باتوجه به نکات گفته شده، می‌توان نتیجه گرفت که اندازه کاشی اگر بیشتر از ۴برابر کرنل باشد، تاثیری در تعداد دسترسی ها ندارد و فقط سربار حافظه ایجاد میکند.



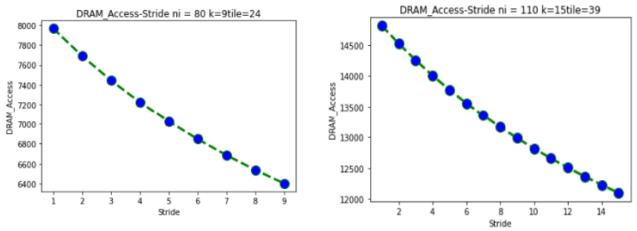
(د) (ج) (ب) **(الف)**

شکل (۹): نمودار تعداد دسترسی‌های DRAM برحسب اندازه کاشی (ساختار هر یک از شبکه‌ها در عنوان نمودارها آورده شده است

۵-۳- بررسی تاثیر گام بر تعداد دسترسیDRAM

برای بررسی تاثیر اندازه گام بر تعداد دسترسی‌های DRAM، ابتدا در ساختار شبکه، مقدار ورودی و اندازه کرنل را ثابت در نظر گرفتیم و به ازای همه گام‌های ممکن، اندازه کاشی بهینه را به دست آوردیم. سپس با قراردادن میانگین اندازه کاشی بهینه، تعداد دسترسی‌های DRAM را برای حالت‌های مختلف به دست آوردیم.

در شکل۱۰، تعداد دسترسی‌های DRAM را بر حسب اندازه گام، رسم کرده‌ایم. باتوجه به اینکه، هر چه اندازه گام افزایش یابد، هم پوشانی پنجره‌های کانولوشن با هم کم شده و نیازی نیست که یک داده را بیشتر از یک بار خواند. در نتیجه تعداد مراجعات به DRAM کاهش می یابد. اگر اندازه گام برابر پهنای کرنل باشد، هیچ همپوشانی وجود ندارد و تعداد دسترسی‌ها در کم‌ترین حالت و برابر با سایز پنجره ورودی می‌شود. در نمودارها نیز مشخص است که با افزایش اندازه گام، تعداد دسترسی-های DRAM کاهش می‌یابد.



(ب) **(الف)**

شکل (۱۰): تعداد دسترسی های DRAM برحسب اندازه گام

۵-۴- پیاده‌سازی موبایلنت

شبکه عصبی موبایلنت[۲۴]، یکی از شبکه‌های CNN محبوب و پرکاربرد است که برای استفاده در دستگاه‌های منابع محدود، طراحی شده است [۲۷] [۲۸] [۲۹] [۳۰]. در شکل (۱۱، معماری موبایلنت به تفکیک لایه آورده شده است که معماری آن بر اساس کانولوشن جداشونده عمقی[۲۵] است که محاسبات را به میزان قابل توجهی نسبت به کانولوشن معمولی کاهش می‌دهد [۳۱] [۳۲]. در این مقاله شبکه موبایلنت را انتخاب و سپس با تکنیک کاشی‌بندی برروی مسیر داده Row Stationary، تعداد دسترسی‌های DRAM را به دست آوردیم که در جدول۲، نتایج ثبت شده است.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

شکل (۱۱): معماری شبکه عصبی موبایلنت

در جدول۲، خروجی‌ها به تفکیک لایه‌های مختلف موبایلنت گزارش شده است. در این جدول، اندازه کاشی بهینه برای هر لایه، تعداد دسترسی‌های حافظه خارجی DRAM برای حالت پایه Eyeriss و استفاده از روش پیشنهادی و در ستون آخر نیز، درصد بهبود تعداد دسترسی‌های DRAM ثبت شده است. باتوجه به جدول۲، کاهش تعداد دسترسی‌های حافظه از ۴۰٪ تا ۸۷٪ است که بیشترین بهبودها در لایه هایی مشاهده شده است که اندازه گام آن‌ها ۱ است. در این لایه‌ها به دلیل همپوشانی بیشتر پنجره‌های کانولوشنی، استفاده مجدد از داده-ها افزایش می‌یابد و کاشی‌بندی را موثرتر می‌کند و دسترسی به حافظه را به میزان قابل توجهی کاهش می‌دهد. با این حال، در لایه‌های با کانولوشن نقطه ای که اندازه هسته ۱ است، هیچ همپوشانی بین پنجره-های محاسباتی وجود ندارد و بنابراین، استفاده مجدد از داده‌ها حداقل است .در نتیجه کاشی‌بندی تاثیری بر کاهش دسترسی به حافظه در این لایه‌ها، ندارد.

جدول (۱): نتایج تعداد دسترسی‌های DRAM به تفکیک لایه برای شبکه عصبی موبایلنت

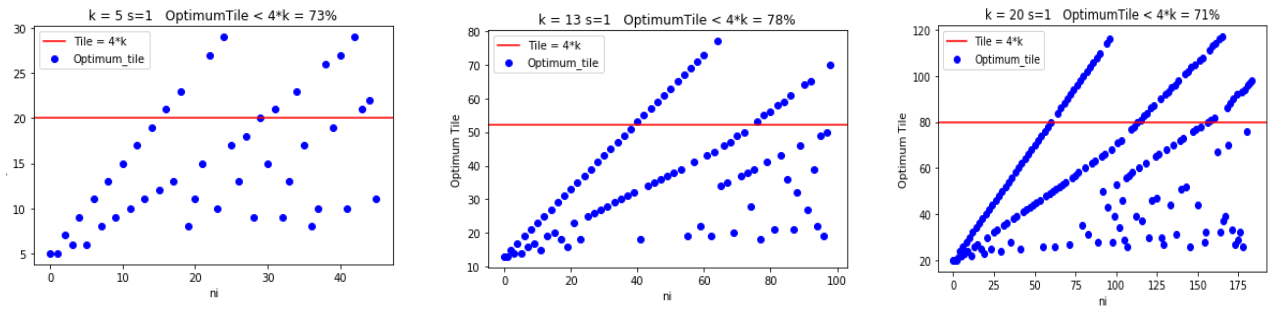
لایه	کاشی بهینه	#DRAM(Eyeriss)	#DRAM(کاشی-بندی)	بهبود(٪)
1	75	1.07415e+07	4.8457e+06	55%
2	12	3.4848e+06	465408	86.7%
3	1	2.56901e+07	2.56901e+07	0
4	11	1.77422e+06	868102	51.1%
5	1	2.56901e+07	2.56901e+07	0
6	8	3.35923e+06	499712	85.3%
7	1	5.13802e+07	5.13802e+07	0
8	7	871200	452629	48.4%
9	1	2.56901e+07	2.56901e+07	0
10	15	1.5575e+06	207360	86.7%
11	1	5.13802e+07	5.13802e+07	0
12	27	419904	200714	52.2%
13	1	2.56901e+07	2.56901e+07	0
14	5	663552	128000	82.1%
15	1	5.13802e+07	5.13802e+07	0
16	5	663552	128000	82.1%
17	1	5.13802e+07	5.13802e+07	0
18	5	663552	128000	82.1%
19	1	5.13802e+07	5.13802e+07	0
20	5	663552	128000	82.1%
21	1	5.13802e+07	5.13802e+07	0
22	5	663552	128000	82.1%
23	1	5.13802e+07	5.13802e+07	0
24	5	194688	110720	44.9%
25	1	2.56901e+07	2.56901e+07	0
26	3	82944	58368	37%
27	1	5.13802e+07	5.13802e+07	0

۶- نتیجه‌گیری

برای استفاده از شبکه‌های عصبی CNN در کاربردهای مختلف نیاز است که دقت آن‌ها بیشتر شود. برای رسیدن به دقت بالا، باید تعداد لایه‌ها و پارامترهای شبکه بیشتر و به عبارتی دیگر شبکه عمیق و حجیم شود. از طرفی زیاد شدن پارامترها وتعداد لایه‌ها، حجم و حرکت داده بین سطوح حافظه را افزایش می‌دهد و نیاز به حافظه بیشتر می‌شود. هم‌چنین به دلیل اینکه حجم داده زیاد می شود، حافظه داخلی برای این حجم داده کافی نیست و نیاز به حافظه خارجی بیشتر می‌شود. از طرفی انرژی مصرفی و زمان دسترسی به حافظه خارجی خیلی بیشتر از حافظه داخلی است و کاهش استفاده از آن، می‌تواند بهبود زیادی در کارایی شبکه داشته باشد.

کاشی‌بندی تکنیکی است که برای بهبود کارایی و بهینه کردن استفاده از حافظه، در شبکه‌های CNN استفاده می‌شود. با استفاده از این روش، می‌توان با بهره‌گیری از اصل محلیت مکانی و زمانی از انواع داده‌ها استفاده مجدد کرد و تعداد مراجعات به حافظه خارجی و همچنین حرکت داده را کاهش داد.

در این مقاله، تعداد دسترسی‌های حافظه خارجی بر حسب پارامترهای ساختاری شبکه و اندازه کاشی، به صورت ریاضی مدل شده است. سپس در یک مساله بهینه‌سازی، اندازه کاشی بهینه برای داشتن حداقل تعداد دسترسی حافظه خارجی، استخراج می شود و تکنیک‌هایی برای کاهش فضای حالت کاشی‌ها، اعمال می‌شود. در ادامه، تاثیر هریک از پارامترهای شبکه بر اندازه کاشی سنجیده شد. مشخص گردید که اندازه کاشی بهینه با اندازه کرنل رابطه دارد و در بیشتر از ۷۰٪ مواقع، کمتر از ۴ برابر اندازه کرنل است. همچنین تاثیر اندازه گام نیز بر تعداد دسترسی‌ها و اندازه کاشی بهینه نشان داد که با افزایش اندازه گام، تعداد دسترسی‌های حافظه خارجی به دلیل کمتر شدن هم‌پوشانی پنجره‌ها، کاهش می‌یابد. همچنین اندازه کاشی بهینه نیز با افزایش گام، کاهش می‌یابد و نزدیک اندازه کرنل می‌شود.



(ج) (ب) (الف)

شکل (۱۲): اندازه کاشی بهینه برحسب اندازه پنجره ورودی مختلف

مراجع

[1] S. Genovese, "Artificial Intelligence: A Guide for Thinking Humans", ORDO, vol. 71, no. 1, pp. 444–449, 2020, doi: 10.1515/ordo-2021-0028.

[2] O. Campesato, "Artificial Intelligence, Machine Learning, and Deep Learning", Artif. Intell. Mach. Learn. Deep Learn., Feb. 2020, doi: 10.1515/9781683924654/HTML.

[3] Pourahangarian F, Kiani A, Karami A, Zanj B. ECG Arrhythmias Detection Using a New Intelligent System Based on Neural Networks and Wavelet Transform. Journal of Iranian Association of Electrical and Electronics Engineers 2012; 9 (1) :33-39

[4] Shahmiri A, Safabakhsh R, Dezhkam R. Automatic Farsi Typo Correction Using a Hybrid Neural Network. Journal of Iranian Association of Electrical and Electronics Engineers 2008; 5 (1) :16-29

[5] ChenYu-Hsin, EmerJoel, and SzeVivienne, "Eyeriss", ACM SIGARCH Comput. Archit. News, vol. 44, no. 3, pp. 367–379, Jun. 2016, doi: 10.1145/3007787.3001177.

[6] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration", Artif. Intell. Rev., vol. 53, no. 7, pp. 5113–5155, Oct. 2020, doi: 10.1007/S10462-020-09816-7/METRICS.

[7] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)", Dig. Tech. Pap. - IEEE Int. Solid-State Circuits Conf., vol. 57, pp. 10–14, 2014, doi: 10.1109/ISSCC.2014.6757323.

[8] S. Zheng et al., "Efficient Scheduling of Irregular Network Structures on CNN Accelerators", IEEE Trans. Comput. Des. Integr. Circuits Syst., vol. 39, no. 11, pp. 3408–3419, Nov. 2020, doi: 10.1109/TCAD.2020.3012215.

[9] Q. Nie and S. Malik, "MemFlow: Memory-Driven Data Scheduling with Datapath Co-Design in Accelerators for Large-Scale Inference Applications", IEEE Trans. Comput. Des. Integr. Circuits Syst., vol. 39, no. 9, pp. 1875–1888, Sep. 2020, doi: 10.1109/TCAD.2019.2925377.

[10] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer CNN accelerators", in Proceedings of the Annual International Symposium on Microarchitecture, MICRO, IEEE Computer Society, Dec. 2016. doi: 10.1109/MICRO.2016.7783725.

[11] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini, "Origami: A convolutional network accelerator", Proc. ACM Gt. Lakes Symp. VLSI, GLSVLSI, vol. 20-22-May-, pp. 199–204, May 2015, doi: 10.1145/2742060.2743766.

[12] I. Dadras, S. Seydi, M. H. Ahmadilivani, J. Raik, and M. E. Salehi, "Fully-Fusible Convolutional Neural Networks for End-to-End Fused Architecture with FPGA Implementation", 2023 30th IEEE Int. Conf. Electron. Circuits Syst., pp. 1–5, Dec. 2023, doi: 10.1109/ICECS58634.2023.10382831.

[13] Q. Nie and S. Malik, "CNNFlow: Memory-driven Data Flow Optimization for Convolutional Neural Networks", ACM Trans. Des. Autom. Electron. Syst., vol. 28, no. 3, Feb. 2022, doi: 10.1145/3577017/ASSET/73AC8D40-245E-445B-B998-83087708C500/ASSETS/GRAPHIC/TODAES-2022-P-2217-F14.JPG.

[14] E. Valpreda et al., "HW-Flow-Fusion: Inter-Layer Scheduling for Convolutional Neural Network Accelerators with Dataflow Architectures", Electron. 2022, Vol. 11, Page 2933, vol. 11, no. 18, p. 2933, Sep. 2022, doi: 10.3390/ELECTRONICS11182933.

[15] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep Learning with Limited Numerical Precision", PMLR, pp. 1737–1746, Jun. 01, 2015. Accessed: Jul. 07, 2024. [Online]. Available: https://proceedings.mlr.press/v37/gupta15.html

[16] J. Li et al., "SmartShuttle: Optimizing off-chip memory accesses for deep learning accelerators", Proc. 2018 Des. Autom. Test Eur. Conf. Exhib. DATE 2018, vol. 2018-January, pp. 343–348, Apr. 2018, doi: 10.23919/DATE.2018.8342033.

[17] J. Qiu et al., "Going deeper with embedded FPGA platform for convolutional neural network", FPGA 2016 - Proc. 2016 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays, pp. 26–35, Feb. 2016, doi: 10.1145/2847263.2847265.

[18] X. Wei et al., "Automated Systolic Array Architecture Synthesis for High Throughput CNN Inference on FPGAs", Proc. - Des. Autom. Conf., vol. Part 128280, Jun. 2017, doi: 10.1145/3061639.3062207.

[19] Z. Du et al., "ShiDianNao: Shifting vision processing closer to the sensor", Proc. - Int. Symp. Comput. Archit., vol. 13-17-June-2015, pp. 92–104, Jun. 2015, doi: 10.1145/2749469.2750389.

[20] Y. Ma, Y. Cao, S. Vrudhula, and J. S. Seo,

"Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA", IEEE Trans. Very Large Scale Integr. Syst., vol. 26, no. 7, pp. 1354–1367, Jul. 2018, doi: 10.1109/TVLSI.2018.2815603.

[21] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks", FPGA 2015 - 2015 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays, pp. 161–170, Feb. 2015, doi: 10.1145/2684746.2689060.

[22] F. Indirli, A. Erdem, and C. Silvano, "A Tile-based Fused-layer CNN Accelerator for FPGAs", ICECS 2020 - 27th IEEE Int. Conf. Electron. Circuits Syst. Proc., Nov. 2020, doi: 10.1109/ICECS49266.2020.9294981.

[23] H. Huang, X. Hu, X. Li, and X. Xiong, "An efficient loop tiling framework for convolutional neural network inference accelerators", IET Circuits, Devices Syst., vol. 16, no. 1, pp. 116–123, Jan. 2022, doi: 10.1049/CDS2.12091.

[24] Y. Li, S. Ma, Y. Guo, R. Xu, and G. Chen, "Configurable CNN Accelerator Based on Tiling Dataflow", Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS, vol. 2018-Novem, pp. 309–313, Jul. 2018, doi: 10.1109/ICSESS.2018.8663795.

[25] Y. S. Lin, H. C. Lu, Y. Bin Tsao, Y. M. Chih, W. C. Chen, and S. Y. Chien, "GrateTile: Efficient Sparse Tensor Tiling for CNN Processing", IEEE Work. Signal Process. Syst. SiPS Des. Implement., vol. 2020-October, Oct. 2020, doi: 10.1109/SIPS50750.2020.9195243.

[26] "DLcoursera/Convolutional Neural Networks/week01/Convolution+model+-+Step+by+Step+-+v2.ipynb at master · csaybar/DLcoursera · GitHub", Accessed: Dec. 15, 2024. [Online]. Available: https://github.com/csaybar/DLcoursera/blob/master/Convolutional Neural Networks/week01/Convolution%2Bmodel%2B-%2BStep%2Bby%2BStep%2B-%2Bv2.ipynb

[27] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", Apr. 2017, Accessed: Dec. 02, 2024. [Online]. Available: https://arxiv.org/abs/1704.04861v1

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks", pp. 4510–4520, 2018.

[29] B. Koonce, "MobileNetV3", Convolutional Neural Networks with Swift Tensorflow, pp. 125–144, 2021, doi: 10.1007/978-1-4842-6168-2_11.

[30] Asadi Amiri S, Andi M. Classification of Pistachio Varieties Using MobileNet Deep Learning Model. Journal of Iranian Association of Electrical and Electronics Engineers 2025; 22 (1) :133-140

[31] A. Ghorbani and M. Amon, "Implementation of convolutional neural network accelerator on FPGA using high-level synthesis method," *National Conference on Electrical and Electronics Industry*, Dec. 3, 2020. Accessed: Jan. 15, 2025.

[32] Aghaei N, Akbarizadeh G, Kosarian A. Using ShuffleNet to design a deep semantic segmentation model for oil spill detection in synthetic aperture radar images. Journal of Iranian Association of Electrical and Electronics Engineers 2022; 19 (3) :131-144

زیرنویس‌ها

[1] Image Classification
[2] Object Detection
[3] Localization
[4] Memory Access Time
[5] Energy Efficiency
[6] Data Movement
[7] Fusion
[8] Scheduler
[9] Tiling
[10] Framework
[11] Software-controlled
[12] Multiply Accumulator
[13] Pooling
[14] Fully Connected
[15] Feature Map
[16] Stride
[17] Padding
[18] Pooling
[19] Overfitting
[20] Fully Connected
[21] Memory Efficiency
[22] Data Locality
[23] Data Reuse
[24] MobileNet
[25] Depthwise Separable Convolution