بررسی سربارهای سخت‌افزاری و بهره‌وری انرژی در پیاده‌سازی انواع چندی‌سازی ممیزثابت در شتاب‌دهنده شبکه عصبی عمیق

مرضیه مستعلی‌زاده* علی انصارمحمدی نجمه نظری*** مصطفی ارسالی صالحی نسب*****

* دانشجو مقطع ارشد، دانشکده برق و کامپیوتر، دانشگاه تهران، تهران، ایران
** دانشجو مقطع دکتری، دانشکده برق و کامپیوتر، دانشگاه تهران، تهران، ایران
*** دانشجو مقطع دکتری، دانشکده برق و کامپیوتر، دانشگاه تهران، تهران، ایران
**** استادیار، دانشکده مهندسی برق و کامپیوتر، دانشگاه تهران، تهران، ایران

چکیده

یکی از کارآمدترین راه‌کارهای فشرده‌سازی و کاهش انرژی مصرفی شبکه‌های عصبی عمیق در دستگاه‌های نهفته، چندی‌سازی با استفاده از نمایش اعداد ممیز ثابت است. در سال‌های اخیر، روش‌های متنوعی برای بهبود صحت شبکه‌های چندی‌سازی‌شده مطرح شده است که اغلب سربارهای محاسباتی زیادی به شبکه تحمیل می‌کنند، اگرچه این موضوع تاکنون از دید طراحان شبکه‌های عصبی عمیق پنهان مانده‌است.

در این پژوهش، روش‌های مختلف چندی‌سازی ممیزثابت، بر اساس مولفه‌های تاثیرگذار در سربارهای سخت افزاری، طبقه‌بندی و مدل شده است. پس از آن، معماری‌های سخت‌افزاری ارائه‌شده برای هریک از مدل‌ها به صورت عادلانه، با در نظرگرفتن هزینه‌فایده‌ی بین صحت شبکه و بهره‌وری انرژی سخت‌افزار، بررسی و مقایسه می‌شوند. نتایج نشان می‌دهد تکنیک‌هایی که برای کاهش خطای روش‌های چندی‌سازی به کار گرفته می‌شود، اگرچه به افزایش صحت شبکه‌های عصبی منجر می‌شود اما از طرف دیگر بهره‌وری انرژی سخت‌افزار را کاهش می‌دهد. براساس نتایج شبیه‌سازی، افزودن ضریب مقیاس و آفست به چندی‌سازی ممیزثابت LSQ، صحت شبکه را حدود ۰/۱ افزایش می‌دهد اما بهره‌وری انرژی سخت‌افزار حدود ۳ برابر کمتر شده است. این موضوع لزوم توجه به سربارهای سخت‌افزاری را به‌خصوص در سیستم‌های نهفته، بیش از پیش نشان می‌دهد.

واژگان کلیدی: شبکه‌های عصبی عمیق، سیستم‌های نهفته، بهره‌وری انرژی، چندی‌سازی ممیزثابت

۱. مقدمه

امروزه شبکه‌های عصبی عمیق[1] کاربردهای مهم و گسترده‌ای در حوزه‌های مختلف به‌دست آورده‌اند که از آن جمله می‌توان به طبقه‌بندی تصاویر[2]، بینایی ماشین[3] و تشخیص اشیاء[4] اشاره کرد. عملکرد بسیار خوب این شبکه‌ها در حوزه‌های مختلف باعث شده‌است که اخیرا در سیستم‌های نهفته[5] نیز مورد استقبال قرار گیرند.

با این وجود، حجم بسیار زیاد محاسبات شبکه‌های عصبی عمیق و فضای زیادی که برای ذخیره‌سازی داده‌ها نیاز دارند باعث شده که ورود آن‌ها به سیستم‌های نهفته با چالش روبرو شود زیرا این

نویسنده مسئول: مصطفی ارسالی صالحی‌نسب، mersali@ut.ac.ir

1. Deep Neural Networks (DNN)
2. Image Classification
3. Computer Vision
4. Object Recognition
5. Embedded Systems

سیستم‌ها بودجه‌ی محدودی برای انرژی مصرفی و فضای ذخیره‌سازی دارند. به‌عنوان مثال، شبکه‌ی ResNet-50 با ۵۰ لایه‌ی مختلف حدود ۲۵/۵ میلیون داده دارد و باید ۳/۹ میلیارد عملیات ضرب‌وانباشت [1] را انجام دهد [۱]. به همین دلیل، روش‌های فشرده‌سازی شبکه، از جمله چندی‌سازی [2] و هرس کردن [3] شبکه [۲-۴] معرفی شدند تا حجم محاسبات و فضای ذخیره‌سازی را کاهش دهند.

در میان روش‌های مختلف فشرده‌سازی شبکه‌های عصبی عمیق، چندی‌سازی محبوبیت بیشتری پیدا کرده و پژوهش‌های بسیاری در این حوزه انجام شده است [۵-۱۲]. چندی‌سازی به بیان ساده، نگاشت اعداد حقیقی به اعداد گسسته‌ی محدود، به منظور کاهش هزینه‌ی حافظه و محاسبات است. در همین راستا، چندی‌سازی‌های مختلف معرفی شده‌اند که از نمایش اعداد متنوعی چون ممیزثابت، باینری، توان ۲ و ممیزشناور استفاده کرده‌اند و برای رسیدن به صحت قابل قبول عرض بیت‌های ۱۶ تا ۱ بیت را به‌کار گرفته‌اند.

نتایج اولین پژوهش‌ها نشان می‌دهد که چندی‌سازی ممیزثابت، برای حفظ صحت شبکه‌های عصبی به ۱۶ بیت نیاز داشت [۱۳, ۱۴]. اگر چه چندی‌سازی ممیزثابت ۱۶ بیت می‌تواند حجم محاسبات و فضای ذخیره‌سازی موردنیاز را تا ۲ برابر به نسبت ممیزشناور ۳۲ بیتی کاهش دهد، اما به دلیل محدودیت‌هایی که برای کاهش منابع دارد، نمی‌تواند راه‌حل مناسبی برای سیستم‌های نهفته باشد.

مقاله‌های [۱۵, ۱۶] تعداد بیت‌های لازم برای چندی‌سازی ممیزثابت را تا ۸ بیت کاهش داده‌اند. در نتیجه، در مقایسه با محاسبات ۳۲ بیتی تقریبا تا ۴ برابر در حافظه و محاسبات بهبود حاصل شده است. معماری‌های بسیاری [۱۷, ۱۸] برای اجرای شبکه‌های عصبی بر مبنای چندی‌سازی ممیزثابت ارائه شده است که هدف آن‌ها رسیدن به حداکثر کارایی انرژی بوده است. در این معماری‌ها سربارهای روش نمایش وزن‌ها و فعال‌سازها [4] به صورت یک عدد ممیزثابت در نظر گرفته نشده است.

محققان برای دستیابی به حداقل خطای چندی‌سازی از روش‌های دیگری نیز در جهت بهبود دقت استفاده کرده‌اند. مقاله‌های [۱۱, ۱۹] تلاش کرده‌اند تا محدوده‌ی پویای چندی‌سازی را در حین آموزش شبکه بهتر تنظیم کنند و از این طریق توانسته‌اند تعداد بیت مورد نیاز را به ۴ بیت کاهش دهند. نکته‌ای که در این روش‌ها از منظر پیاده‌سازی سخت‌افزاری پنهان مانده، این است که نحوه‌ی نمایش وزن‌ها و فعال‌سازها فراتر از یک عدد ممیزثابت تنها است. بدین معنی که در این روش‌ها برای نمایش وزن‌ها (یا فعال‌سازها) از حاصل‌ضرب یک عدد ممیزثابت و ضریب مقیاس [5] استفاده شده است و مقادیر ضریب مقیاس به صورت اعداد ممیزشناور در نظر گرفته شده‌اند که در عملیات سخت‌افزاری هزینه‌ی زیادی دارند و به این هزینه‌ها توجه نشده است.

[۲۰-۲۲] هر کدام با استفاده از تکنیک‌های مخصوص به خود سعی کرده‌اند که خطای چندی‌سازی و تعداد بیت را کاهش دهند و در نهایت، با استفاده از حدود ۳ یا ۴ بیت، به صحت‌های نزدیک به صحت کامل شبکه برسند. هرچند با استفاده از این روش‌ها، خطای چندی‌سازی با استفاده از نمایش اعداد ممیزثابت به حداقل رسیده اما نیاز است که مقادیر چندی‌سازی‌شده با استفاده از تعدادی پارامتر با دقت کامل مقیاس شوند و همچنین برای انتقال به بازه‌ی خاصی از اعداد، با یک مقدار آفست جمع شوند. در این روش‌ها نیز، به تاثیر پارامترهای ۳۲ بیتی افزوده‌شده (یعنی ضریب مقیاس و آفست) بر روی هزینه‌ی سخت افزار توجه زیادی نشده است. با استفاده از این روش‌ها، شبکه‌های فشرده‌شده برای اجرا روی سیستم‌های نهفته مناسب تر خواهند شد البته هم‌چنان حجم محاسبات ضرب‌وانباشت در آن‌ها زیاد و نیازمند بهینه‌سازی است.

همان‌طور که گفته شد، روش‌های مختلف چندی‌سازی ممیزثابت با استفاده از حدود ۳ یا ۴ بیت به صحت‌های نزدیک به صحت کامل شبکه رسیده‌اند. در نمایش اعداد ممیزثابت هرچند هزینه‌ی سربارهای اضافه‌ی ضریب مقیاس و آفست در نظر گرفته نشده است، اما به دلیل محبوبیت و رواج زیاد، بهینه‌سازی‌های متنوعی بر روی معماری آن صورت گرفته است. براساس تحقیقات ما، تا کنون هیچ پژوهشی به طور خاص این روش‌ها را از نظر ملاحظات پیاده‌سازی سخت‌افزاری مثل توان مصرفی [6] و بهره‌وری انرژی [7] بررسی و مقایسه نکرده است. ما در این مقاله قصد داریم ابتدا چندی‌سازی‌های مختلف را با توجه به مولفه‌های مهم و تاثیرگذار در پیاده‌سازی سخت‌افزاری طبقه‌بندی کنیم و به منظور ساده‌تر شدن طبقه بندی، یک مدل برای هر طبقه ارائه می‌کنیم. سپس برای هر مدل با توجه به پژوهش‌های انجام شده و چندی‌سازی‌های مختلف ارائه شده، بهترین چندی‌سازی را از نظر صحت شبکه انتخاب می‌کنیم تا در ادامه، هزینه‌ی سربارهای سخت افزاری با توجه به آن چندی‌سازی‌ها مقایسه شوند. سپس برای هر مدل، یک معماری طراحی می‌کنیم و در نهایت نتایج توان و انرژی مصرفی مدل‌های مختلف چندی‌سازی به ازای عرض بیت و صحت‌های مختلف را مقایسه می‌کنیم. به بیان دیگر، چالش اصلی مقاله این است که برای مدل‌های مختلف چندی‌سازی‌شده که در مرحله‌ی آموزش استفاده می‌شوند و سربارهای مختلف و تاثیرگذاری در سخت افزار دارند، پارامترهای

1. Multiply Accumulate
2. Quantization
3. Pruning
4. Activation
5. Scaling Factor
6. Power Consumption
7. Energy Efficiency

انرژی و توان سخت‌افزار مقایسه شوند. به طور خلاصه نوآوری‌های این مقاله به شرح زیر است:

- بررسی چندی‌سازی‌های ممیزثابت از نظر خطای چندی‌سازی و صحت شبکه و طبقه‌بندی آن‌ها برمبنای مولفه‌های تاثیرگذار در سخت‌افزار.
- ارائه‌ی یک مدل برای هر طبقه از چندی‌سازی‌ها.
- ارائه‌ی روابط یک‌دست برای عملیات ضرب داخلی دو بردار با توجه به نمایش پارامترها در هر مدل.
- ارائه‌ی یک معماری پایه برای تحقق و ارزیابی سخت‌افزاری هر مدل از چندی‌سازی.
- بررسی ملاحظات پیاده سازی سخت افزاری برای هر مدل از چندی‌سازی.

در ادامه، در بخش دوم مقاله، ابتدا به بررسی مفاهیم پایه‌ای مرتبط با شبکه‌های عصبی عمیق و سپس روش‌های مختلف چندی‌سازی پرداخته‌ایم. در قسمت سوم کارهای مرتبط بررسی‌شده و در قسمت چهارم انگیزه‌ی انجام این مقاله بیان شده است. سپس در بخش پنجم، برای هر کدام از روش‌های چندی‌سازی یک مدل معرفی شده است و معماری مربوط به هر مدل را ارائه کرده‌ایم و در نهایت در بخش ششم نتایج انرژی و توان مصرفی مدل‌های مختلف با درنظر گرفتن تعداد بیت و صحت شبکه ارائه شده است. در پایان، در بخش هفتم نتیجه گیری آورده شده است.

۲. مفاهیم اولیه

در این بخش به بررسی مفاهیم اولیه‌ی مورد نیاز در این مقاله می‌پردازیم. این مفاهیم شامل پیش‌زمینه‌هایی از شبکه‌های عصبی عمیق و چندی‌سازی است. سعی کرده‌ایم موضوعاتی را که ارائه می‌کنیم، مفاهیم اصلی این حوزه را در برگیرد تا از ایجاد ابهامات احتمالی جلوگیری شود.

۲.۱ شبکه‌های عصبی عمیق

شبکه‌های عصبی عمیق، به‌عنوان زیرمجموعه‌ای از یادگیری ماشین، دسته‌ای از الگوریتم‌ها هستند که برای شناسایی و تشخیص الگوها استفاده می‌شوند. این شبکه‌ها با استفاده از یک ابزار ادراکی ماشینی[1]، داده‌های خام را دریافت می‌کنند و سعی می‌کنند الگوهای موجود در این داده‌ها را شناسایی کنند. این ویژگی سبب شده است که شبکه‌های عصبی عمیق امروزه کاربردهای گسترده‌ای داشته باشند و با توجه به این کاربردها، امروزه تحقیقات بسیاری در این زمینه و چالش‌های پیاده‌سازی آن صورت می‌گیرد.

ساده‌ترین شبکه‌های عصبی از چند لایه‌ی محدود تشکیل می‌شده‌اند. به عنوان مثال، شبکه‌های پرسپترون چند لایه[2] فقط می‌توانند تصاویر ساده‌ای را پردازش کنند که تعداد اشیاء موجود در آن محدود باشد. با پردازش این تصاویر، شبکه‌ها می‌توانند اشیاء موجود را طبقه‌بندی کنند. برای مثال، مجموعه داده‌ی MNIST را که شامل دست‌نوشته‌ی اعداد ۰ تا ۹ است، می‌توان با این شبکه‌ها دسته‌بندی کرد.

در مسائل واقعی، جزئیات تصاویر و اشیاء از جمله رنگ، موقعیت و نور بسیار متنوع و زیاد است. هم‌چنین ممکن است اشیاء یک تصویر همیشه واضح نباشد، مثلاً در یک تصویر اشیاء بسیار زیاد و نامرتبط وجود داشته باشد یا اینکه فقط بخشی از شیء مورد نظر در تصویر وجود داشته باشد، در این شرایط تشخیص و دسته‌بندی اشیاء به سادگی قبل نیست و شبکه‌های پرسپترون چند لایه دیگر نمی‌توانند اشیاء را دسته‌بندی کنند. به همین دلیل، شبکه‌های عصبی کانولوشنی برای پردازش تصاویر پیچیده‌تر معرفی شدند.

شبکه‌ی عصبی کانولوشنی جزو پرکاربردترین روش‌ها در یادگیری عمیق است. این شبکه‌ها می‌توانند در محیط واقعی و پیچیده که اشیاء بدون الگوی خاصی در کنار هم قرار دارند، ویژگی‌های موثر را استخراج و اطلاعات غیر مرتبط را حذف کنند. سپس بر اساس ویژگی‌های موثر استخراج‌شده، اشیاء را دسته‌بندی کنند.

شبکه‌های عصبی کانولوشنی عموماً از چند لایه‌ی کانولوشن، نمونه‌برداری[3]، نرمال‌سازی[4] و تماماً متصل[5] تشکیل شده اند. از میان این لایه‌ها، تمرکز این پژوهش بر لایه‌های کانولوشنی است. البته از آنجایی‌که عملیات لایه‌ی کانولوشن و تماماً متصل از نظر ماهیت یکسان است و تفاوت‌های جزئی دارد، نتایج این پژوهش را برای حوزه‌ی لایه‌ی تماماً متصل نیز می‌توان در نظر گرفت. در ادامه به توضیح لایه‌ی کانولوشن می‌پردازیم.

یکی از وظایف لایه‌های کانولوشن این است که ویژگی‌های موثر داده‌های ورودی را برجسته سازند و آن‌ها را استخراج کنند. به همین منظور، در این لایه تعدادی فیلتر که به آن کرنل[6] گفته می‌شود، روی تصویر ورودی حرکت می‌کند و عملیات کانولوشن را انجام می‌دهد. این عملیات در رابطه‌ی (۱) نشان داده شده است. سپس یک تابع غیر خطی روی حاصل کانولوشن اعمال می‌شود و نگاشت‌های ویژگی[7] به‌دست می‌آید که در واقع همان ویژگی‌های مهم استخراج شده از تصویر است.

1. Machine Perception
2. Multi-layer Perceptron
3. Pooling
4. Normalization
5. Fully Connected (FC)
6. Kernel
7. Feature Maps

$$\boldsymbol{O}[z][u][x][y] = \boldsymbol{B}[u] + \sum_{k=0}^{C-1}\sum_{i=0}^{S-1}\sum_{j=0}^{R-1} \boldsymbol{I}[z][k][Ux+i][Uy+j] \times \boldsymbol{W}[u][k][i][j]$$

$$0 \le z < N, 0 \le u < M, 0 \le x < F, 0 \le y < E,$$
$$E = \frac{H-R+U}{U}, F = \frac{W-S+U}{U}. \quad (۱)$$

در رابطه‌ی (۱)، مقدار O، I و W و B به ترتیب ماتریس نگاشت ویژگی خروجی، نگاشت ویژگی ورودی، فیلترها و بایاس هستند. U سایز گام است که در واقع نشان می‌دهد به چه میزان روی ورودی حرکت می‌کند.

کانولوشن یک عملیات اصلی در شبکه‌های عصبی است. با توجه به رابطه‌ی (۱) مشخص می‌شود که این عملیات شامل تعداد بسیار زیادی ضرب داخلی[1] بین بردار ورودی (I) و وزن (W) است. به معنای دیگر، وقتی کرنل روی ماتریس نگاشت ورودی حرکت می‌کند، هر بار پنجره‌ای از نگاشت ورودی با اندازه‌ی S×R انتخاب می‌شود. می‌توان کرنل و پنجره‌ی انتخاب‌شده از نگاشت ورودی را به صورت دو بردار در نظر گرفت که اعضای آن‌ها باید نظیر به نظیر در هم ضرب و در نهایت با هم جمع شوند. این، همان تعریف ضرب داخلی است. برای دو بردار دلخواه $\bar{X}, \bar{Y} \in \mathbb{R}^K$ تعریف می‌کنیم:

$$dot\ (\bar{X}, \bar{Y}) = \sum_{i=0}^{K-1} x_i \times y_i \quad (۲)$$

که در آن، تابع dot ضرب داخلی را نشان می‌دهد و $x_i \in \bar{X}$ و $y_i \in \bar{Y}$.

همان‌طور که در بخش‌های قبل گفته شد، شبکه‌های عصبی عمیق کاربردهای فراوانی در حوزه‌های مختلف پیدا کرده‌اند. برای یادگیری داده‌های پیچیده و صحت بالا، به شبکه‌های عصبی عمیقی نیاز داریم که تعداد لایه‌های آن زیاد باشد و از این‌رو محاسبات بسیار زیادی نیاز است، به همین دلیل بار محاسباتی شبکه و حافظه‌ی مورد نیاز برای ذخیره‌ی پارامترهای شبکه بسیار زیاد است [۱].

معمولاً هر چه تعداد لایه‌های شبکه بیشتر باشد، شبکه به صحت بالاتری دست پیدا می‌کند. شبکه‌های عصبی عمیق شامل تعداد بسیار زیادی عملیات ضرب‌وانباشت اعداد ممیزشناور است که برای این عملیات، حجم بسیار زیادی از داده باید از حافظه خوانده شود. هم‌چنین ممکن است در محاسبات شبکه‌های عصبی، به علت تکرار عملیات، یک داده چند بار از حافظه خوانده شود. خواندن این حجم زیاد از داده‌ها نیاز به پهنای باند زیاد دارد و باعث ایجاد گلوگاه در ارتباطات حافظه می‌شود.

در حوزه‌ی سیستم‌های نهفته و کم‌مصرف، مصرف انرژی و حافظه یک چالش اساسی است. به همین دلیل، در طراحی شبکه‌های عصبی عمیق برای این حوزه، علاوه بر کارایی و صحت شبکه، معیارهای دیگری مثل منابع محاسباتی موردنیاز، اندازه‌ی حافظه و مهم‌تر از آن‌ها، مصرف انرژی نیز باید در نظر گرفته شود. برای حل این چالش‌ها، راهکارهای مختلفی معرفی شده که از جمله‌ی آن‌ها می‌توان به کوچک کردن شبکه، کاهش تعدادبیت عملگر و عملوند و کم کردن تعداد عملگر و عملوند اشاره کرد.

۲.۲ چندی‌سازی

راهکارهایی که برای کاهش تعدادبیت عملگر و عملوند به کار می‌روند، عموماً داده‌های ورودی و وزن شبکه را با تعداد بیت کمتر نشان می‌دهند و از این طریق تلاش می‌کنند تا فضای ذخیره‌ی موردنیاز را کاهش دهند و محاسبات بین آن‌ها را ساده‌تر سازند.

یکی از اصلی‌ترین روش‌ها در این حوزه چندی‌سازی است که در آن، داده‌های حقیقی که عموماً ممیز شناور ۳۲ بیتی‌اند، با تعداد بیت کمتر نشان داده می‌شوند. در چندی‌سازی، چندین سطح برای تخمین اعداد وجود دارد که هر کدام از اعداد حقیقی، با روش‌های مختلف به یکی از سطوح تعلق می‌گیرد. تعداد این سطوح، دقت و تعداد بیت مورد نیاز برای نمایش داده را تعیین می‌کند. به طور کلی تعداد بیت مورد نظر را از رابطه‌ی $\log_2 N$ به دست می‌آورند که N تعداد سطوح را مشخص می‌کند. با کاهش تعداد بیت لازم برای نمایش، فضای ذخیره‌سازی مورد نیاز کاهش می‌یابد، هم‌چنین محاسبات ساده‌تر می‌شود.

روش‌های مختلف چندی‌سازی، شیوه‌های متفاوتی برای انجام این کار دارند. در این روش‌ها، پارامترهای هدف (وزن، ورودی یا هردو)، الگوریتم به‌کار رفته، نمادها و نمایش اعداد متفاوت‌اند. برای اینکه در ادامه‌ی این مقاله از مفاهیم یکسان استفاده کنیم، تعریف می‌کنیم؛

$$x^q = Q(x) \quad (۳)$$

که در آن Q() تابع چندی‌سازی است که عدد حقیقیِ x را به عدد چندی‌سازی‌شده‌ی x^q تبدیل می‌کند. عرض بیت x و x^q هر عددی می‌تواند باشد و در روش چندی‌سازی مشخص می‌شود، اما عموماً x یک عدد ممیزشناور ۳۲ بیتی است.

۳. بررسی کارهای پیشین

چندی‌سازی یکی از راه‌کارهای موثر در فشرده‌سازی شبکه‌های عصبی عمیق است که یکی از اهداف ویژه‌ی آن، امکان پیاده‌سازی این شبکه‌ها بر روی دستگاه‌های نهفته است. در چندی‌سازی، تعداد بیت مورد نیاز برای پارامترهای شبکه کاهش داده می‌شود تا از طریق کاهش حافظه و هزینه انجام عملیات محاسباتی، کارایی انرژی شبکه عصبی بر روی دستگاه نهفته افزایش یابد [۲۳-۲۷]. یکی از مولفه‌های مهم در افزایش کارایی انرژی، معماری واحدهای

1. Dot Product

محاسباتی است که ارتباط مستقیم با سیستم نمایش اعداد دارد. متداول‌ترین و مرسوم‌ترین روش چندی‌سازی، نمایش اعداد ممیزثابت بوده است. در این حوزه، [۹, ۱۴, ۲۸] نشان داده‌اند که با ۱۶ بیت چندی‌سازی، شبکه به صحت قابل قبولی می‌رسد. [۹] با توجه به توزیع متفاوت وزن ها در لایه ها، برای هر لایه تعداد بیت متفاوتی را در نظر گرفته است و با این کار توانسته است میانگین تعداد بیت برای لایه‌ها را به ۸ تا ۱۲ بیت برساند. [۱۵, ۱۶] چندی‌سازی ممیزثابت را با یادگیری و تنظیم دقیق بعد از یادگیری شبکه همراه کرده‌اند تا بهترین مقدار برای تعداد بیت چندی‌سازی و همچنین تعداد بیت قسمت اعشار ممیزثابت را پیدا کنند.. در این معماری ها فرض بر استفاده از تنها یک عدد ممیز ثابت برای چندی‌سازی بوده است و هیچ سربار دیگری برای نمایش اعداد لازم نیست. به عنوان مثال معماری‌های [۱۷, ۱۸] با توجه به چندی‌سازی ممیزثابت جهت رسیدن به حداکثر کارایی انرژی ارائه شده است.

مقاله‌های [۱۱, ۱۹] با استفاده از پیدا کردن بازه‌ی پویای مناسب برای چندی‌سازی حین آموزش شبکه توانسته‌اند تعداد بیت مورد نیاز را به کمتر از ۸ بیت کاهش دهند. نحوه‌ی نمایش وزن‌ها و فعال‌سازها در این چندی‌سازی ها فراتر از تنها یک عدد ممیزثابت است. در این روش‌ها برای نمایش وزن‌ها (فعال‌سازها) از حاصل‌ضرب یک عدد ممیزثابت و ضریب مقیاس استفاده شده است. مقادیر ضریب مقیاس به صورت اعداد ممیزشناور در نظر گرفته شده است که انجام عملیات مختلف روی آن‌ها هزینه‌ی زیادی دارد و به این نکته در این روش‌ها از جهت پیاده سازی در سخت افزار توجه نشده است،

مقاله‌ی DSQ [۲۱] با استفاده از چندی‌سازی تفاضلی نرم[1]، توانسته است صحت شبکه‌های چندی‌سازی شده با تعداد بیت پایین را به شبکه‌های صحت‌کامل نزدیک کند. این مقاله می‌تواند به دقیق بودن گرادیان در انتشار به عقب[2] و کاهش خطای چندی‌سازی در فرآیند رو به جلو[3] با یک محدوده برش[4] مناسب کمک کند. اگرچه این روش صحت شبکه‌ی چندی‌سازی‌شده را افزایش داده است ولی نکته‌ای که در پیاده‌سازی سخت‌افزاری این روش به آن توجه نشده، این است که وزن‌ها و فعال‌سازها به مقادیر مقیاس‌شده و همراه باآفست تبدیل می‌شوند. این موضوع باعث می‌شود که نتوان عملیات ضرب یک وزن در یک فعال‌ساز را تنها با یک عملیات ضرب انجام داد.

مقاله‌ی QIL [۱۹] با استفاده از ترکیب یک تابع انتقال و گسسته‌ساز توانسته است مشکلات حاصل از چندی‌سازی را کاهش دهد و صحت شبکه چندی‌سازی‌شده را افزایش دهد. در این مقاله به دو نکته‌ی محدوده‌ی برش و هرس کردن (تبدیل اعداد کوچک به صفر) توجه ویژه شده است. این آستانه‌ها، در حین یادگیری شبکه با استفاده از پارامتری کردن تابع انتقال به صورت خودکار به‌دست می‌آیند. پس از آن، خروجی تابع انتقال را (که بین بازه ی ۱ و ۱- است) به تعداد بیت محدود چندی‌سازی کرده است. نکته‌ی مهم در خروجی چندی‌سازی، نمایش مقدار چندی‌سازی‌شده با یک عدد ممیزثابت است که باعث کاهش سربارهای سخت‌افزاری می‌شود.

مقاله‌ی LSQ [۱۱] در حین آموزش شبکه، از یک چندی‌سازی آموزش‌پذیر استفاده می‌کند تا از این طریق، بهترین اندازه‌ی گام را برای سطوح چندی‌سازی به‌دست آورد. اندازه‌ی گام برای سطوح چندی‌سازی به صورت یک عدد دقت کامل در نظر گرفته شده است و در واقع نقش ضریب مقیاس را در چندی‌سازی دارد. هر چند در این مقاله اشاره شده است که محاسبات مربوط به این ضریب مقیاس می تواند با عملیات نرمال‌سازی لایه‌ی بعد ادغام شود اما هزینه‌ی اعمال شده به سخت‌افزار می‌تواند قابل ملاحظه باشد. در ادامه، نویسندگان این مقاله به منظور کاهش خطای چندی‌سازی، با توجه به اینکه سطوح چندی‌سازی را نامتقارن در نظر گرفته‌اند، از مقدار آفستی استفاده می‌کنند تا مقادیر وزن‌ها را با دقت بیشتری به سطوح چندی‌سازی نگاشت کنند. این کار صحت شبکه را بهبود داده است. البته این مقاله به سربار و هزینه‌ی سخت‌افزاری این آفست اشاره‌ای نمی‌کند (LSQ+ [۶]).

مقاله‌ی uL2Q [۲۰] با استفاده از تجزیه و تحلیل جامع داده‌های کمّی به دنبال پیدا کردن بهترین چندی‌سازی برای داده‌های با توزیع نرمال است. این مقاله، در ابتدا وزن‌های شبکه را با فرض توزیع نرمال دریافت می‌کند. سپس توزیع آن‌ها را به توزیع نرمال استاندارد تبدیل می‌کند. از این طریق، برای هر تعداد بیت چندی‌سازی بهترین ضرایب مورد نیاز برای مقیاس به توزیع نرمال را مشخص می‌کند. در نهایت، برای استفاده از مقدار چندی‌سازی‌شده در شبکه نیاز است که عملیات استانداردسازی به صورت معکوس انجام شود. برای این کار، باید دو پارامتر ضریب مقیاس و آفست، به ازای هر بخش از داده به صورت دقت‌کامل آموزش ببینند که هزینه‌ی آن باید در معماری سخت‌افزار در نظر گرفته شود.

علاوه بر این، مقاله‌ی bitpruning [۲۲] آفست‌هایی را برای انتقال مقادیر وزن و فعال‌ساز تعریف کرده است تا از حداکثر بازه‌ی پویای قابل نمایش با تعداد بیت محدود استفاده کند و از این طریق، خطای چندی‌سازی با تعداد بیت کمتر را کاهش دهد. در این مقاله تعداد بیت مناسب برای هر لایه یا دسته‌ای از اعداد، به صورت خودکار در زمان آموزش به دست می‌آید. هر دو مقدار آفست در این مقاله، به صورت دقت‌کامل در نظر گرفته شده است و البته به هزینه‌های تحمیل شده به سخت افزار توجه نشده است.

در مقاله‌هایی شبیه به bitpruning، وزن‌ها و فعال‌سازها به صورت حاصل‌ضرب یک عدد ممیزثابت در ضریب مقیاس در نظر گرفته

1. Differentiable Soft Quantization
2. Back Propagation
3. Feed Forward
4. Clipping Range

شده‌اند که با یک آفست جمع می‌شوند تا اعداد به یک بازه‌ی مشخص منتقل شوند و از حداکثر بازه‌ی پویای قابل نمایش با اعداد ممیزثابت استفاده شود. در این حالت، هر پارامتر از مجموع دو مولفه‌ی مجزا تشکیل شده است. لذا هنگام ضرب دو پارامتر، خروجی مورد نظر شامل ۴ مولفه‌ی ضرب جزئی[1] است که طبیعتاً هزینه‌ی بسیار بیشتری نسبت به استفاده از تنها یک عدد با نمایش اعداد ممیزثابت خواهد داشت.

مقاله‌ی CSQ [۲۹] با اشاره به اینکه نابرابر بودن تعداد اعداد مثبت و منفی موجود در نمایش اعداد می‌تواند سبب بهینه نبودن خطای چندی‌سازی باشد، یک چندی‌سازی متقارن معرفی می‌کند که علاوه بر یکسان کردن تعداد سطوح مثبت و منفی، میتواند کماکان از سخت افزارهای استاندارد موجود برای انجام عملیات ضرب بهره ببرد.

جدول ۱ چندی‌سازی‌های مختلف ممیزثابت به همراه سربارهای تاثیر گذار بر سخت افزار را نشان می‌دهد. وجود این سربارها در روش‌های مختلف و از طرف دیگر صحت های مختلف این روش ها سبب شد تا ما در این پژوهش به دنبال مقایسه‌ی این چندی‌سازی‌ها از منظر ملاحظات پیاده‌سازی سخت افزاری باشیم.

جدول ۱. مقایسه انواع چندی‌سازی ممیز ثابت

نام مقاله	سال	تعداد بیت	نمایش اعداد	ضریب مقیاس	آفست
Dadiannao [14]	2014	16	FxP	-	-
Limited Numerical Precision [28]	2015	16	FxP	-	-
low precision [9]	2015	10-12-20	FxP	-	-
reduced precision [30]	2015	8-12	FxP	-	-
Hardware-oriented approximation [16]	2016	8	FxP	-	-
Ristretto [15]	2018	8	FxP	-	-
QIL [19]	2019	2-5	FxP	-	-
LSQ [11]	2019	2-4	FxP	√	-
LSQ+ [6]	2020	2-4	FxP	√	√
uL2Q [20]	2019	1-8	FxP	√	√
DSQ [21]	2019	1-4	FxP	√	√
BitPruning [22]	2019	1-4	FxP	√	√
LLSQ [31]	2020	2-4	FxP	√	-
CSQ [29]	2022	2-4	FxP	√	√

۴. انگیزه‌ی مقاله و تعریف مسأله

هدف اصلی ما در انجام این پژوهش مقایسه‌ای منصفانه و عادلانه میان روش‌های مختلف چندی‌سازی ممیزثابت است. در واقع، ما قصد داریم به این سوال پاسخ دهیم که در صحت یکسان، کدام روش چندی‌سازی بهره‌وری انرژی بیشتری دارد؟

صحت روش‌های مختلف چندی‌سازی ممیزثابت در جدول ۲ نشان داده شده است. روش‌های مختلف چندی‌سازی بر روی شبکه‌های عصبی مختلف از جمله *ResNet-18*، *ResNet-34* و *ResNet-50* [۳۲] که به‌وسیله‌ی مجموعه‌داده‌ی ImageNet آموزش داده شده‌اند، اعمال شده است. در این جدول، صحت اصلی شبکه‌ها و صحت شبکه‌های چندی‌سازی‌شده در عرض بیت‌های ۱ تا ۸ گزارش شده است.

جدول ۲. مقایسه‌ی صحت روش‌های مختلف چندی‌سازی ممیزثابت در شبکه‌های مختلف بر روی مجموعه‌داده‌ی *ImageNet*.

ImageNet						
Network	Method	Accuracy				
		Baseline (32 bit)	Bit-width			
			2	3	4	8
ResNet-18	LSQ [11] [2]	70.1	66.7	69.4	70.7	
	LSQ+ [6]		66.8	69.4	70.8	
	FAQ [33]				69.8	70.02
	PACT [8]		64.4	68.1	69.2	
	NICE [5]			67.7	69.8	
	QIL [19]		65.7	69.2	70.1	70
	DSQ [21]		65.17	68.66	69.56	
	uL2Q [20]		65.5		66	65.4
ResNet-34	LSQ	74.1	71.6	73.4	74.1	74.1
	NICE			71.7	73.5	
	FAQ				73.3	73.7
	DSQ		70.02	72.54	72.76	
	QIL		70.6	73.1	73.7	
ResNet-50	LSQ	76.9	73.7	75.8	76.7	76.8
	PACT		72.2	75.3	76.5	
	NICE			75.1	76.5	
	FAQ				76.3	76.5

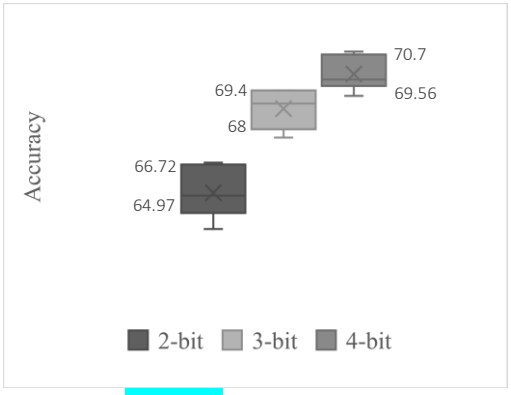
مقایسه‌ی صحت روش‌های مختلف با صحت اصلی شبکه در جدول ۲ نشان می‌دهد که روش‌های مختلف چندی‌سازی ممیزثابت، با حدود ۲ تا ۴ بیت برای وزن و فعال‌سازها، به صحت‌هایی بسیار نزدیک به صحت اصلی رسیده‌اند. برای بررسی دقیق‌تر، صحت روش‌های چندی‌سازی در بازه‌ی ۲ تا ۴ بیت بر روی شبکه‌ی ResNet-18، به‌عنوان نمونه، در شکل ۱ نشان داده شده‌است. این

1. Partial Product

2. برای اینکه بتوانیم نتایج LSQ را با +LSQ مقایسه کنیم، نتایج صحت هر دو مقاله را از +LSQ گزارش کرده‌ایم.

نمودار[1] به خوبی پراکندگی صحت روش‌های مختلف چندی‌سازی را نشان می‌دهد.

در شکل ۱، نمودارهای مستطیلی از سمت چپ به راست، به‌ترتیب عرض بیت‌های ۲، ۳ و ۴ را نشان می‌دهند. در هر بیت، پراکندگی صحت‌ها با استفاده از یک مستطیل و دو خط عمودی مشخص شده است. علامت × داخل مستطیل، نشان‌دهنده‌ی میانه‌ی[2] صحت‌هاست و مستطیل رنگی نشان‌دهنده‌ی ۵۰ درصد از صحت‌هاست که نزدیک به میانه قرار دارند. هم‌چنین دو خط کوتاه افقی در بالا و پایین هر نمودار، صحت بیشینه و کمینه[3] را در آن دسته مشخص می‌کند. در نمودار شکل ۱ مشاهده می‌شود که ارتفاع مستطیل، در دسته‌ی ۲، ۳ و ۴ بیت، به‌ترتیب ۱/۷، ۱/۴ و ۱/۱ درصد است. این موضوع نشان می‌دهد که روش‌های چندی‌سازی ممیزثابت نشان‌داده شده در این نمودار با اختلاف ۱/۱ تا ۱/۷ درصد، به صحت نزدیک به هم رسیده‌اند.



شکل ۱. پراکندگی صحت روش‌های مختلف چندی‌سازی ممیز ثابت در بازه‌ی ۲ تا ۴ بیت برروی *ResNet-18* که با مجموعه‌داده‌ی *ImageNet* آموزش داده شده است.

روش‌های مختلف چندی‌سازی ممیزثابت، سیستم نمایش اعداد متفاوتی دارند. به همین دلیل، سخت‌افزار مورد نیاز برای اجرای آن‌ها متفاوت خواهد بود. تفاوت در سخت‌افزاری که هر روش به آن نیاز دارد، این سوال را ایجاد می‌کند که آیا بهره‌وری انرژی روش‌های مختلف چندی‌سازی ممیزثابت متفاوت است؟ اگر بله، کدام روش کمترین توان مصرفی و بیشترین کارایی انرژی را دارد؟ آیا روش‌های چندی‌سازی‌ای که صحت بالایی به دست می‌آورند، مثل نقاط بیشینه در شکل ۱ ([۱۱])، برای دستیابی به این صحت بالا توان مصرفی بسیار زیادی دارند؟ این سوالات و مشابه آن‌ها ما را برآن داشت تا بهره‌وری انرژی روش‌های مختلف را بررسی کنیم. نتیجه‌ی چنین مقایسه‌ای می‌تواند به انتخاب یک چندی‌سازی کم‌مصرف و کارآمد از نظر انرژی منجر شود. البته می‌دانیم که مقایسه‌ی منصفانه‌ی روش‌های مختلف چندی‌سازی ممیزثابت، به علت سخت‌افزار متفاوت آن‌ها، چالش‌برانگیز است اما تلاش می‌کنیم تا حد امکان شرایط عادلانه‌ای را برای مقایسه ایجاد کنیم.

۵. روش پیشنهادی

روش پیشنهادی برای انجام این پژوهش شامل دو بخش کلی است. ابتدا شیوه‌های مختلف چندی‌سازی در مدل‌های مشخص دسته‌بندی می‌شود و پس از آن برای هر مدل یک معماری ارائه می‌شود.

۵.۱ دسته‌بندی شیوه‌های مختلف چندی‌سازی به مدل‌های پایه

چندی‌سازی کردن اعداد ورودی و وزن در شبکه‌های عصبی عمیق، به ناچار منجر به کاهش صحت این شبکه‌ها می‌شود. پژوهشگران مختلف این حوزه تلاش کرده‌اند تا با استفاده از راهکارهای مختلف خطای چندی‌سازی را کاهش دهند و صحت شبکه را به صحت اصلی نزدیک کنند. راهکارهای مختلفی جهت نگاشت بهتر بازه‌ی پارامترهای شبکه به سطوح چندی‌سازی در جهت افزایش دقت چندی‌سازی و به حداقل رساندن اختلاف بین مسیر انتشار به جلو و عقب ارایه شده است که در نتیجه بعضی از این روش ها سبب اضافه شدن ضریب مقیاس [۱۱, ۱۲, ۳۳] و آفست [۲۰, ۲۱] به نمایش ممیزثابت می شود.

با بررسی سیستم نمایش اعداد ممیزثابت در چندی‌سازی‌های مختلف به این نتیجه می‌رسیم که می‌توان این چندی‌سازی‌ها را در چند دسته‌ی کلی قرار داد. دسته‌ی اول که آن را دسته‌ی پایه می‌نامیم، هیچ‌گونه ضریب مقیاس یا آفستی ندارد. دسته‌ی دوم همراه با ضریب مقیاس است و دسته‌ی سوم همراه با ضریب مقیاس و آفست است.

چندی‌سازی ممیزثابت پایه را از نظر اجرای روی سخت‌افزار، شبیه به ممیزثابت با ضریب مقیاس در نظر می‌گیریم. چرا که در چندی‌سازی ممیزثابت با ضریب مقیاس، می‌توان ضریب مقیاس را از رابطه فاکتور گرفت. این موضوع در ادامه به تفصیل بیان خواهد شد. بنابر این توضیحات، در این پژوهش دو مدل معرفی می‌شود؛ چندی‌سازی همراه با ضریبِ مقیاس بدون آفست و چندی‌سازی همراه با ضریب مقیاس و آفست. برای اختصار، دسته‌ی اول را **ممیزثابت بدون آفست** و دسته‌ی دوم را **ممیزثابت با آفست** می‌نامیم.

۵.۱.۱ ممیز ثابت بدون آفست

در چندی‌سازی ممیزثابت بدون آفست، اعداد حقیقی به اعداد ممیزثابت همراه با ضریب مقیاس تبدیل می‌شوند. فرض کنیم $\bar{X}$ و $\bar{W}$ به‌ترتیب بردارهای ورودی و وزن باشند به‌صورتی که

$$\bar{X}, \bar{W} \in \mathbb{R}^K$$

و K مشخص‌کننده‌ی اندازه‌ی ضرب داخلی است. هم چنین فرض کنید اعداد حقیقی $x_k \in \bar{X}$ و $w_k \in \bar{W}$ به ترتیب نشان‌دهنده‌ی

1. Box and Whiskers Charts
2. Median
3. Maximum and Minimum

یک عدد از ورودی و وزن باشد که $k \in \{1,2,\dots,K\}$. در چندی‌سازی ممیزثابت، این اعداد حقیقی به x_k^q و w_k^q تبدیل می‌شوند به‌طوری‌که

$$x_k^q = Q(x_k) = \alpha_x^q d_{x_k} \qquad \alpha_x^q, d_{x_k} \in F$$
$$w_k^q = Q(w_k) = \alpha_w^q d_{w_k} \qquad \alpha_w^q, d_{w_k} \in F \qquad (۴)$$

در این رابطه α_x^q و α_w^q ضرایب مقیاس ۸ بیتی هستند. براساس بررسی‌های انجام‌شده، انتخاب ۸ بیت برای ضریب مقیاس، شبکه را به دقت قابل قبولی می‌رساند، به همین دلیل ما در این پژوهش ۸ بیت را انتخاب کرده‌ایم. α_x^q برای تمام اعداد ورودی و α_w^q برای تمام وزن‌ها در بردار Kتایی یکسان در نظر گرفته شده است. d_{x_k} و d_{w_k} در این رابطه مقادیر چندی‌سازی‌شده را به ترتیب برای ورودی و وزن نشان می‌دهند. مجموعه‌ی F سیستم نمایش اعداد ممیزثابت است. در این سیستم، هر عدد از دو بخش تشکیل شده است، یک بخش عدد صحیح[1] و یک بخش کسر[2]، این دو بخش توسط ممیزثابت از هم جدا می‌شوند.

همانطور که در بخش‌های قبل گفته شد، ضرب داخلی یکی از بخش‌های اصلی و سنگین در شبکه‌های عصبی عمیق است و یکی از راه‌های کاهش حجم و انرژی مصرفی این عملیات، چندی‌سازی است. با استفاده از چندی‌سازی ممیزثابت بدون آفست، عملیات ضرب داخلی به صورت زیر انجام می‌شود؛

$$dot\ (\bar{X}^q.\bar{W}^q) = \sum_{k=1}^{K} (x_k{}^q w_k{}^q) = \sum_{k=1}^{K} \left((\alpha_x^q d_{x_k})(\alpha_w^q d_{w_k})\right) = \alpha_x^q \alpha_w^q \sum_{k=1}^{K} (d_{x_k} d_{w_k}) \qquad (۵)$$

همان‌طور که گفتیم، ضرایب α_x^q و α_w^q برای تمام اعداد ورودی و وزن در ضرب داخلی یکسان‌در نظر گرفته شده است. بنابراین می‌توانیم $\alpha_x^q \alpha_w^q$ را از جمع فاکتور بگیریم. ذکر این نکته ضروری است که برای اینکه عملیات لایه‌ی کانولوشن را ساده‌تر کنیم، $\alpha_x^q \alpha_w^q$ را با بخش نرمال‌سازی ادغام می‌کنیم. در این صورت، بخش باقیمانده‌ی $\sum_{k=1}^{K}(d_{x_k} d_{w_k})$ را می‌توانیم با سخت‌افزاری ساده انجام دهیم که در بخش بعد به تفصیل توضیح داده خواهد شد. هم‌چنین با دقت به بخش $\sum_{k=1}^{K}(d_{x_k} d_{w_k})$ می‌توان دریافت که این رابطه، دقیقاً عملیات ضرب داخلی را برای چندی‌سازی بدون ضریب نشان می‌دهد، یعنی حالتی که اعداد ورودی و وزن فقط به مقادیر d_{x_k} و d_{w_k} چندی‌سازی می‌شوند. در این شرایط، می‌توانیم از معماری طراحی‌شده برای چندی‌سازی بدون آفست، برای انجام عملیات ضرب داخلی در چندی‌سازی بدون ضریب مقیاس نیز استفاده کنیم.

۵.۱.۲ ممیزثابت با آفست

در چندی‌سازی ممیزثابت با آفست، اعداد چندی‌سازی شده علاوه بر ضریب مقیاس یک آفست نیز دارند. اضافه کردن یک آفست با این هدف انجام می‌شود تا بازه‌ی قابل نمایش به پارامترهای شبکه منطبق‌تر شود و خطای چندی‌سازی کاهش یابد.

همانند بخش قبل، x_k و w_k را به ترتیب، یک عدد از ورودی و وزن در نظر می‌گیریم. در چندی‌سازی ممیزثابت با آفست داریم؛

$$x_k^q = Q(x_k) = \alpha_x^q d_{x_k} + \beta_x^q \qquad \alpha_x^q, d_{x_k}, \beta_x^q \in F$$
$$w_k^q = Q(w_k) = \alpha_w^q d_{w_k} + \beta_w^q \qquad \alpha_w^q, d_{w_k}, \beta_w^q \in F \qquad (۶)$$

که در این رابطه، β_x^q و β_w^q آفست‌های چندی‌سازی هستند که ما ۸ بیتی در نظر گرفته‌ایم. در قسمت قبل دلیل انتخاب ۸ بیت برای ضرایب مقیاس و آفست آورده شده است. همانند ضرایب مقیاس، β_x^q و β_w^q برای تمام پارامترهای ضرب داخلی یکسان در نظر گرفته شده است.

در این چندی‌سازی، ضرب داخلی به صورت زیر انجام می‌شود؛

$$\begin{aligned} dot\ (\overline{X^q}.\overline{W^q}) &= \sum_{k=1}^{K} (x_k{}^q w_k{}^q) = \\ &= \sum_{k=1}^{K} \left((\alpha_x^q d_{x_k} + \beta_x^q)(\alpha_w^q d_{w_k} + \beta_w^q)\right) \\ &= \sum_{k=1}^{K} \Big((\alpha_x^q \alpha_w^q)(d_{x_k} d_{w_k}) + (\alpha_x^q \beta_w^q d_{x_k}) + (\alpha_w^q \beta_x^q d_{w_k}) + (\beta_x^q \beta_w^q)\Big) \\ &= (\alpha_x^q \alpha_w^q) \sum_{k=1}^{K} (d_{x_k} d_{w_k}) + (\alpha_x^q \beta_w^q) \sum_{k=1}^{K} d_{x_k} + (\alpha_w^q \beta_x^q) \sum_{k=1}^{K} d_{w_k} + \beta_x^q \beta_w^q \end{aligned} \qquad (۷)$$

همان‌طور که مشاهده می‌شود، حاصل نهایی ضرب داخلی در چندی‌سازی ممیزثابت با آفست ۴ بخش دارد؛ بخش‌های $(\alpha_x^q \alpha_w^q) \sum_{k=1}^{K} (d_{x_k} d_{w_k})$ ، $(\alpha_x^q \beta_w^q) \sum_{k=1}^{K} d_{x_k}$ و $(\alpha_w^q \beta_x^q) \sum_{k=1}^{K} d_{w_k}$ توسط سخت‌افزار و در زمان اجرا محاسبه می‌شوند. بخش $\beta_x^q \beta_w^q$ یک ثابت است که می‌تواند از قبل محاسبه شود و در نهایت با مقادیر بخش‌های قبل جمع شود.

۵.۲ معماری پیشنهادی مدل‌های پایه‌ی چندی‌سازی

1. Integer
2. Fraction

در بخش قبل دو مدل پایه برای چندی‌سازی‌های ممیزثابت معرفی شد که شامل ممیزثابت بدون آفست و ممیزثابت با آفست بود. همان‌طور که پیش از این بیان شد، هدف از این پژوهش مقایسه‌ی بی‌طرفانه و عادلانه بین مدل‌های مختلف چندی‌سازی از منظر پیاده‌سازی سخت‌افزاری، از جمله توان مصرفی و مساحت است.

به منظور انجام این مقایسه، در این پژوهش برای هر مدل یک معماری پیشنهاد شده است. ممکن است برای هرکدام از این مدل‌ها معماری‌های دیگری در پژوهش‌های مختلف ارائه شده باشد که از لحاظ توان مصرفی، مساحت یا انرژی نسبت به این معماری پیشنهادی کارآمدتر باشند اما از آنجایی‌که معماری‌های ارائه‌شده متنوع و متفاوت‌اند، مقایسه‌ی آن‌ها بسیار سخت است و نمی‌توان معیارهای واحدی برای ارزیابی و مقایسه‌ی آن‌ها در نظر گرفت. از طرفی بیشتر این تکنیک ها و معماری‌ها را می‌توان بر روی هر دو مدل چندی‌سازی ممیزثابت اعمال کرد. به همین دلیل، در این پژوهش پایه‌ترین معماری برای هر مدل در نظر گرفته شده است تا بتوان جزئیات را تا حد امکان شبیه به هم طراحی کرد و معماری‌ها را منصفانه مقایسه کرد.

برای اینکه جایگاه این پژوهش در شبکه‌های عصبی عمیق مشخص شود لازم به یادآوری است که اصلی‌ترین عملیات در این شبکه‌ها، عملیات کانولوشن است و همان‌طور که در بخش‌های قبل بیان شد (روابط (۱) و (۲))، کانولوشن را می‌توان به‌عنوان تعداد بسیار زیادی ضرب داخلی بین بردارهای ورودی و وزن در نظر گرفت. در بخش قبل، برای هر کدام از مدل‌های پایه‌ی چندی‌سازی سیستم نمایش اعداد و ضرب داخلی را نشان دادیم، در این بخش معماری مربوط به ضرب داخلی هر چندی‌سازی بر اساس نمایش اعداد بخش قبل معرفی می‌شود. در ادامه جزئیات هر معماری به تفصیل بیان خواهد شد.

۵.۲.۱ ممیزثابت بدون آفست

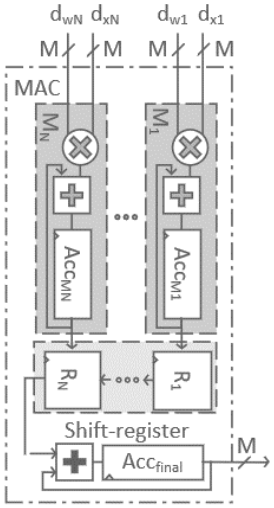
با توجه به رابطه‌ی (۵) مشخص می‌شود ضرب داخلی در چندی‌سازی ممیزثابت بدون آفست، شامل یک عملیات ضرب‌وانباشت و سپس ضرب در یک ضریب است. قبلاً گفته شد که برای ساده‌کردن ضرب داخلی و بخش کانولوشن، ضریب را با لایه‌ی بعد ادغام می‌کنیم بنابراین فقط ضرب‌وانباشت در این لایه باقی می‌ماند.

شکل ۲ معماری هسته‌ی پردازشی[1] ضرب داخلی در چندی‌سازی ممیزثابت بدون آفست را نشان می‌دهد. هسته‌ی اصلی این معماری، واحد ضرب‌وانباشت است که در شکل به‌صورت بخش‌های رنگی با شناسه‌ی M_i نشان‌داده شده که $i \in \{1, \dots, N\}$ است. N بیان‌گر تعداد کل ضرب‌وانباشت‌هایی است که در هر سیکل، به صورت موازی اجرا می‌شوند. در واقع N ضریب موازی‌سازی در سطح واحد پردازشی است.

به طور کلی، این معماری شامل دو بخش است که می‌توانند به صورت موازی با هم اجرا شوند و یک خط لوله[2] را تشکیل دهند. در بخش اول که آن را *ضرب‌وانباشت* می‌نامیم و شامل N واحد MAC موازی است، اعداد ورودی و وزن دو به دو در هم ضرب‌وانباشت می‌شوند. در بخش دوم که بخش *کاهش*[3] است، نتیجه‌ی ضرب‌های بخش اول با هم جمع می‌شود و حاصل نهایی محاسبه می‌شود.

قبلاً گفته شد که تعداد کل ضرب داخلی را K در نظر گرفته‌ایم. در هر سیکل، N عدد ورودی $d_{x_N}, \dots, d_{x_1}$ و N عدد وزن $d_{w_N}, \dots, d_{w_1}$ که هر کدام M بیتی‌اند وارد می‌شوند و دو به دو به M_i می‌روند. هر M_i یک ضرب کننده‌ی M بیتی برای ضرب اعداد ورودی دارد، پس از آن یک جمع‌کننده و ثبات[4] وجود دارد که برای انباشت نتیجه استفاده می‌شود. پس از گذشت $\frac{K}{N}$ سیکل، تمام ضرب‌های بین بردار ورودی و وزن انجام می‌شود و از این پس باید حاصل ضرب‌های جزئی با هم جمع شود. در این مرحله، پس از رسیدن سیگنال کنترلی مبنی بر تمام شدن K ضرب ورودی در وزن، خروجی ثبات Acc_{Mi} هر کدام از M_i ها وارد ثبات R_i مربوط به خود می‌شود.

مجموعه‌ی $R_1, \dots, R_N$ در واقع یک ثبات انتقالی[5] است که به همراه جمع‌کننده و ثبات بعد از خودش، بخش *کاهش* مدار را تشکیل می‌دهد. در هر سیکل، مقدار R_i وارد R_{i+1} می‌شود و مقدار آخرین ثبات، یعنی R_N که به جمع‌کننده متصل است، با مقدار ثبات Acc_{final} جمع و مجدداً انباشته می‌شود.



شکل ۲. معماری هسته‌ی پردازشی ضرب داخلی در چندی‌سازی ممیزثابت بدون آفست.

برای اینکه بتوان دو بخش *ضرب‌وانباشت* و *کاهش* را به‌صورت خط لوله پیش برد، سیگنال‌های کنترلی طوری تنظیم شده‌اند که تا پایان

1. Processing Engine (PE)
2. Pipeline
3. Reduction
4. Register
5. Shift Register

ضرب‌وانباشت تمام ورودی‌ها و وزن‌ها، مانع از ورود مقدار Acc_{Mi} به R_i می‌شوند. پس از آن یکبار تمام مقادیر Acc_{Mi} وارد R_i می‌شود و مجدداً راه ارتباطی بین این دو گروه ثبات مسدود می‌شود.

در این مرحله بخش *کاهش* مدار نتایج ضرب‌وانباشت‌ها را با هم جمع می‌کند و بخش *ضرب‌وانباشت* می‌تواند ورودی‌های دیگری را از یک عملیات ضرب داخلی جدید دریافت و آن‌ها را پردازش کند.

مرحله‌ی *کاهش*، دارای N ثبات است که در هر سیکل مقدار هر یک از آن‌ها به ثبات بعدی منتقل و در نهایت مجموع مقدار ثبات‌ها محاسبه می‌شود. بنابراین این بخش N سیکل طول می‌کشد. گفتیم که بخش *ضرب‌وانباشت* کار خود را در $\frac{K}{N}$ سیکل انجام می‌دهد. برای اینکه توازن این دو بخش حفظ شود و خط لوله بالاترین کارایی را داشته باشد، لازم است N را طوری انتخاب کنیم که

$$\frac{K}{N} = N \rightarrow N_{opt} = \sqrt{K} \qquad (۸)$$

بر اساس روابط تئوری مشاهده می‌شود که برای ضریب موازی‌سازی یک مقدار بهینه به ازای هر K وجود دارد. اگر $N < N_{opt}$ باشد، آنگاه مدار به بالاترین حد موازی‌سازی خودش نمی‌رسد، تعداد سیکل بیشتری برای محاسبه‌ی ضرب داخلی نیاز دارد و در نتیجه، بهره‌وری انرژی از مقدار بیشینه‌ی خود کمتر می‌شود.

اگر $N > N_{opt}$ باشد، یعنی تعداد واحدهای MAC در بخش *ضرب‌وانباشت* بیشتر می‌شود و کل عملیات ضرب‌وانباشت در تعداد سیکل کمتری انجام می‌شود. از طرف دیگر، تعداد ثبات‌های رجیستر انتقالی R_i هم بیشتر می‌شود، لذا به تعداد سیکل بیشتری برای جمع کردن نتایج نیاز دارد. در این شرایط، بخش *ضرب‌وانباشت* مدار سریع‌تر از بخش *کاهش* به اتمام می‌رسد و تا زمان اتمام کار بخش *کاهش* بیکار می‌ماند. ایجاد شدن این حالتهای بیکاری در خط لوله، کارایی آن را پایین می‌آورد و در نتیجه بهره‌وری انرژی را کاهش می‌دهد.

اکنون اگر بخواهیم موازی‌سازی را از N_{opt} بالاتر ببریم و هم‌چنان تعادل خط لوله را حفظ کنیم، نیاز داریم که در بخش *کاهش* مدار، جمع‌کننده‌های جدیدی را اضافه کنیم تا حاصل ثبات‌ها را به‌صورت موازی با هم و سریع‌تر جمع کنند. علاوه بر آن، به جمع‌کننده‌های دیگری نیز نیاز داریم تا در یک ساختار درختی، تمام نتایج را با هم جمع کنند و حاصل نهایی را محاسبه کنند. پس با بالابردن موازی‌سازی از حد بهینه‌ی آن، گرچه می‌توانیم تعداد ضرب داخلی را در زمان مشابه بالاتر ببریم (توان عملیاتی[1] را افزایش دهیم) اما سرباری که جمع‌کننده‌های جدید به مدار تحمیل می‌کنند، توان مصرفی را بیش از توان عملیاتی افزایش می‌دهد و در نتیجه بهره‌وری انرژی کاهش می‌یابد.

برای اینکه موضوع واضح شود، با یک مثال عددی پیش می‌رویم. فرض کنیم K=۱۰۲۴، در این صورت طبق رابطه‌ی (۸)، باید N را ۳۲ انتخاب کنیم. معماری این حالت در شکل ۳(الف) نشان داده شده است. در این شرایط بخش *ضرب‌وانباشت* و *کاهش* هر دو ۳۲ سیکل طول می‌کشد و توازن در خط لوله حفظ می‌شود. اکنون موازی‌سازی را ۲ برابر و N را ۶۴ انتخاب می‌کنیم. معماری این حالت در شکل ۳(ب) مشاهده می‌شود. در این حالت، بخش *ضرب‌وانباشت* ۱۶ سیکل طول می‌کشد و اگر فقط از یک جمع‌کننده در بخش *کاهش* استفاده کنیم، این بخش پس از ۶۴ سیکل تمام خواهد شد.

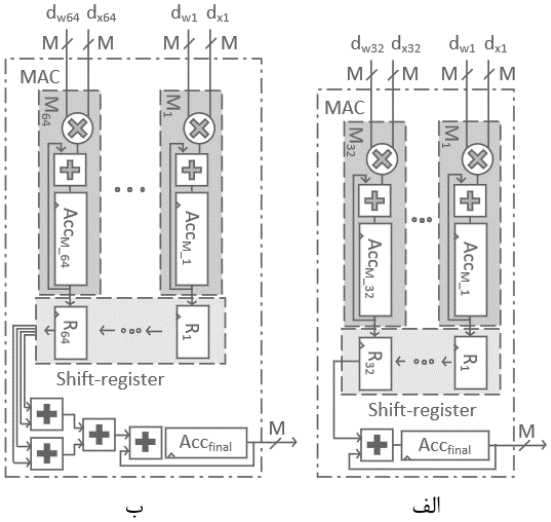
از آنجاییکه بخش *ضرب‌وانباشت* بسیار سریع تمام می‌شود، تا پایان کار بخش *کاهش* باید منتظر بماند، این بیکاری باعث کاهش کارایی معماری می‌شود. برای اینکه توازن خط لوله را حفظ کنیم، لازم است یک جمع‌کننده دیگر به بخش *کاهش* اضافه کنیم تا در هر سیکل ۴ مقدار ثبات با هم جمع شوند و تمام این بخش در ۱۶ سیکل تمام شود. علاوه بر این، به یک جمع‌کننده دیگر نیاز داریم تا در یک ساختار درختی، مقدار دو جمع قبل را محاسبه کند. هم‌چنین یک جمع‌کننده دیگر نیاز است تا به کمک یک ثبات، نتایج را انباشته کند.

در شرایطی که N=۶۴ باشد، نسبت به حالت قبل، دو برابر ضرب داخلی بیشتری در زمان مشابه انجام شده است (توان عملیاتی ۲ برابر شده است)، اما با مقایسه‌ی قسمت (الف) و (ب) در شکل ۳ به این نتیجه می‌رسیم که توان مصرفی از دو برابر بیشتر شده است. (تعداد واحدهای ضرب‌وانباشت و ثبات‌های انتقالی دو برابر شده است اما علاوه بر آن معماری شکل (ب)، نسبت به شکل (الف) سه جمع‌کننده بیشتر دارد و همین موضوع باعث می‌شود توان مصرفی از دو برابر بیشتر شود.)

در این پژوهش بهره‌وری انرژی، توان عملیاتی نسبت به توان مصرفی تعریف شده است و توان عملیاتی تعداد ضرب داخلی در واحد زمان را نشان می‌دهد. در این مثال، توان عملیاتی دقیقاً دو برابر شده است اما توان مصرفی از دوبرابر بیشتر است، بنابراین بهره‌وری انرژی نسبت به حالت قبل کاهش می‌یابد.

این موضوع که از نظر تئوری بررسی شد، در فصل نتایج، با استفاده از نتایج شبیه‌سازی نیز نشان داده خواهد شد.

1. Throughput



شکل ۳. ضرب داخلی در ممیزثابت بدون آفست با فرضِ K=۱۰۲۴. (الف) N=N$_{opt}$=۳۲ و (ب) N=2N$_{opt}$=۶۴.

معماری نشان داده شده در شکل ۲ مقدار عبارت $\sum_{k=1}^{K}(d_{x_k}d_{w_k})$ را محاسبه می‌کند. پس از این مرحله می‌توان ضریب $\alpha_x^q\alpha_w^q$ را در لایه نرمال‌سازی اعمال کرد تا حاصل ضرب داخلی به طور کامل و صحیح محاسبه شود. هم‌چنین می‌توان تنها همین معماری را برای ضرب داخلی چندی‌سازی بدون ضریب استفاده کرد.

۵.۲.۲ ممیزثابت با آفست

شکل ۴ معماری هسته‌ی پردازشی ضرب داخلی را در چندی‌سازی ممیزثابت با آفست نشان می‌دهد. همانند معماری ممیزثابت بدون آفست، این معماری نیز دو بخش کلی دارد؛ بخش *ضرب‌وانباشت* و بخش *کاهش*. این دو بخش می‌توانند موازی با هم اجرا شوند و خط لوله را تشکیل دهند.

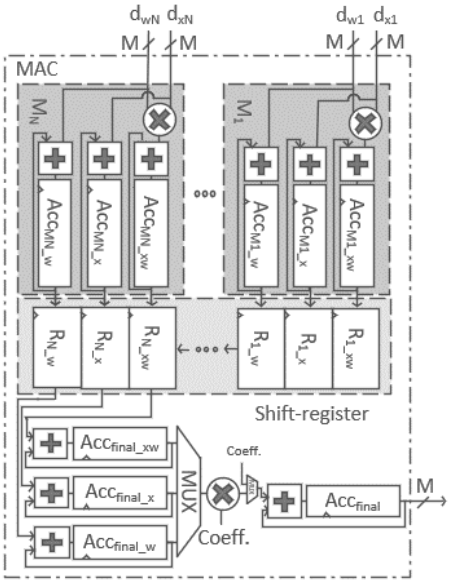
همان‌طور که در رابطه‌ی (۷) دیدیم، حاصل ضرب داخلی در چندی‌سازی ممیزثابت با آفست چهار بخش دارد که سه بخش آن در زمان اجرا و یک بخش آن از قبل محاسبه می‌شود. هر کدام از سه بخشی که باید در زمان اجرا محاسبه شوند، یعنی $(\alpha_x^q\alpha_w^q)\sum_{k=1}^{K}(d_{x_k}d_{w_k})$ ، $(\alpha_x^q\beta_w^q)\sum_{k=1}^{K}d_{x_k}$ و $(\alpha_w^q\beta_x^q)\sum_{k=1}^{K}d_{w_k}$، شامل یک ضرب‌وانباشت و یک ضریب است. بر همین اساس، معماری طوری طراحی شده‌است که ابتدا حاصل ضرب‌وانباشت‌ها محاسبه شود، سپس ضریب مربوط در هر کدام ضرب شود.

قسمت اصلی در این معماری، واحدهای ضرب‌وانباشت است که با اندیس‌های M$_i$ در شکل ۴ مشخص است که $i \in \{1,\dots,N\}$ و همانند معماری ممیزثابت بدون آفست، در اینجا نیز N، ضریب موازی‌سازی و نشان‌دهنده‌ی تعداد واحدهای ضرب‌وانباشتی است که به صورت موازی با هم اجرا می‌شوند.

در هر سیکل، $d_{x_N},\dots,d_{x_1}$ و $d_{w_N},\dots,d_{w_1}$ وارد می‌شوند و دو به دو به M$_i$ می‌روند. در هر M$_i$ یک ضرب‌کننده وجود دارد که حاصل $(d_{x_i}d_{w_i})$ را محاسبه و در ثبات Acc$_{Mi_xw}$ انباشته می‌کند. علاوه بر این، دو ثبات Acc$_{Mi_x}$ و Acc$_{Mi_w}$ وجود دارند که به ترتیب مقدار d_{x_i} و d_{w_i} را انباشته می‌کنند. بخش ضرب‌وانباشت، پس از گذشت $\frac{K}{N}$ سیکل، تمام ضرب‌وانباشت‌های بردار ورودی و وزن را انجام می‌دهد. در این مرحله مقدار این ثبات‌ها وارد ثبات‌های R$_{i_xw}$، R$_{i_x}$ و R$_{i_w}$ می‌شود. مجموعه‌ی این ثبات‌ها یک ثبات انتقالی را تشکیل می‌دهد که به N دسته‌ی سه‌تایی تقسیم شده‌اند. هر کدام از دسته‌ها مربوط به یک M$_i$ است.

پس از اتمام ضرب‌وانباشت و ورود مقادیر M$_i$ به ثبات انتقالی، بخش *کاهش* مدار شروع می‌شود. در این مرحله، در هر سیکل مقدار R$_{i_xw}$ به R$_{i+1_xw}$، مقدار R$_{i_x}$ به R$_{i+1_x}$ و مقدار R$_{i_w}$ به R$_{i+1_w}$ منتقل می‌شود. آخرین مجموعه‌ی سه تایی از ثبات انتقالی به سه انباشت‌گر (شامل جمع‌کننده و ثبات) متصل است که مقادیر ثبات‌های مربوط به خود را انباشته می‌کنند. به عبارت دیگر، پس از گذشت N سیکل، مقدار $\sum_{k=1}^{K}(d_{x_k}d_{w_k})$ ، $\sum_{k=1}^{K}d_{x_k}$ و $\sum_{k=1}^{K}d_{w_k}$ به ترتیب در ثبات‌های Acc$_{final_xw}$، Acc$_{final_x}$ و Acc$_{final_w}$ ذخیره شده است.

در این مرحله باید مقادیر ثبات‌های نهایی در ضریب مربوط ضرب شوند. این کار در چهار سیکل انجام می‌شود، در سه سیکل اول، به ترتیب هر کدام از مقادیر ثبات‌های Acc$_{final_xw}$، Acc$_{final_x}$ و Acc$_{final_w}$ انتخاب شده و در ضریب‌های $(\alpha_x^q\alpha_w^q)$، $(\alpha_x^q\beta_w^q)$ و $(\alpha_w^q\beta_x^q)$ ضرب شده و در ثبات Acc$_{final}$ انباشته می‌شوند. در سیکل آخر نیز، مقدار $\beta_x^q\beta_w^q$ با مقادیر قبل جمع شده و حاصل نهایی محاسبه می‌شود. لازم به ذکر است در صورتی که مقدار K به اندازه‌ی کافی بزرگ انتخاب شود که $\frac{K}{N} \gg 4$ باشد، می‌توان از این چهار سیکل که به بخش *کاهش* اضافه شده است صرف‌نظر کرد و خط لوله را هم‌چنان متوازن در نظر گرفت.



شکل ۴. معماری هسته‌ی پردازشی ضرب داخلی در چندی‌سازی ممیزثابت با آفست

همانند معماری ممیزثابت بدون آفست، در معماری ممیزثابت با آفست نیز دو بخش *ضرب‌وانباشت* و *کاهش* می‌توانند یک خط لوله

را تشکیل دهند. هم‌چنین برای اینکه تعادل این خط لوله حفظ شود، لازم است N طبق رابطه‌ی (۸) محاسبه شود. در این معماری نیز، اگر N کمتر یا بیشتر از N_{opt} قرار داده شود، مشابه، دلایلی که در قسمت قبل گفته شد بهره‌وری انرژی کاهش می‌یابد. این موضوع نیز در بخش نتایج نشان داده خواهد شد.

۵. نتایج

در این بخش، نتایج شبیه‌سازی برای معماری‌های پیشنهادی ارائه و بررسی می‌شوند. پیش از ارائه‌ی نتایج به معرفی ابزارهای استفاده‌شده می‌پردازیم. در این مقاله بهره‌وری انرژی، توان عملیاتی نسبت به توان مصرفی تعریف شده است و توان عملیاتی تعداد ضرب داخلی در واحد زمان را نشان می‌دهد.

۶.۱ ابزارهای استفاده‌شده

در انجام این پژوهش برای مقایسه‌ی سخت‌افزاری مدل‌های مختلف چندی‌سازی، معماری‌های مناسب هر مدل طراحی، پیاده‌سازی و شبیه‌سازی شده‌اند. پیاده‌سازی معماری‌ها با استفاده از زبان سخت‌افزاری Verilog بوده و با کمک ابزار Modelsim شبیه‌سازی شده‌اند تا از صحت عملکرد آن‌ها مطمئن شویم. پس از آن توان مصرفی و مساحت معماری‌ها با استفاده از ابزار Design Compiler و با تکنولوژی 45nm محاسبه شده است.

۶.۲ مقدار انتخاب‌شده برای پارامترها

همان‌طور که در بخش روش پیشنهادی توضیح داده‌شد، در معماری‌های طراحی‌شده‌ی این مقاله، سه پارامتر K، N و M نقش مهمی دارد که به‌ترتیب نشان‌دهنده‌ی اندازه‌ی ضرب داخلی، ضریب موازی‌سازی در معماری و عرض بیت چندی‌سازی است. در شبیه‌سازی‌هایی که در ادامه نتایج آن‌ها را بررسی می‌کنیم، برای M، عموماً بازه‌ی ۱ تا ۱۶ بیت را در نظر گرفته‌ایم چرا که اغلب چندی‌سازی‌های ممیز ثابت در این عرض بیتی ارائه شده‌اند. هم‌چنین از آن‌جایی‌که روش‌های اخیر چندی‌سازی ممیز ثابت در ۲ تا ۴ بیت به صحت‌هایی نزدیک به صحت شبکه‌ی ۳۲ بیتی رسیده‌اند (جدول ۲) در شبیه‌سازی‌ها، ۳ بیت را به‌عنوان نمونه انتخاب کرده‌ایم.

پارامتر K، بسیار به مشخصات شبکه‌های عصبی عمیق مختلف و ابعاد لایه‌های کانولوشن آن‌ها بستگی دارد. به‌منظور انتخاب مقدارهای مناسب، شبکه‌های عصبی عمیق مختلفی از جمله ResNet-18 بررسی شده‌اند و در نهایت برای K اعدادی درنظر گرفته شده است که به ابعاد واقعی شبکه‌های عصبی بررسی‌شده نزدیک باشد.

۶.۳ نتایج شبیه‌سازی و سنتز

همان‌طور که از ابتدای این مقاله گفته شد، هدف از انجام این پژوهش مقایسه‌ی روش‌های مختلف چندی‌سازی ممیزثابت شبکه‌های عصبی از لحاظ پارامترهای سخت‌افزاری از جمله توان مصرفی و بهره‌وری انرژی بوده است. در این بخش با استفاده از نتایج شبیه‌سازی‌ها و سنتز سخت افزار به این مقایسه می‌پردازیم.

این بخش شامل ۴ گروه نتایج است؛ یافتن بهترین ضریب موازی‌سازی، بهره‌وری انرژی معماری‌های پیشنهادی، بهره‌وری انرژی روش‌های چندی‌سازی مدرن[1] و تأثیر افزودن ضریب و آفست بر سخت‌افزار.

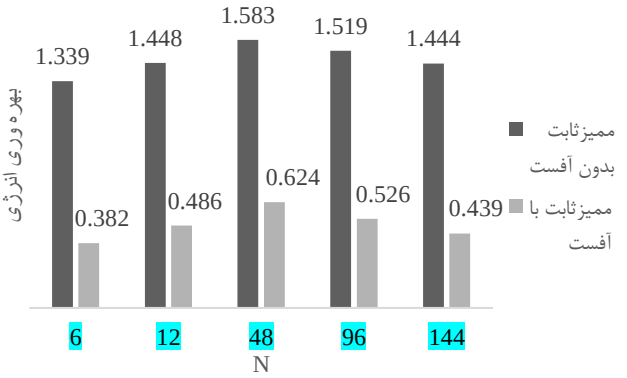
۶.۳.۱ یافتن بهترین ضریب موازی‌سازی

همان‌طور که در بخش معماری پیشنهادی مدل‌های پایه‌ی چندی‌سازی گفته شد، در معماری‌های ممیزثابت بدون آفست و با آفست، N ضریب موازی‌سازی در سطح واحد پردازشی است. هم‌چنین در رابطه‌ی (۸) دیدیم که با توجه به سایر پارامترها، از جمله K، برای ضرایب موازی‌سازی یک مقدار بهینه وجود دارد. از نظر تئوری بررسی شد که اگر ضرایب موازی‌سازی بیشتر یا کمتر از مقدار بهینه انتخاب شوند، بهره‌وری انرژی کاهش می‌یابد. در این بخش این موضوع را با استفاده از نتایج سنتز معماری بررسی خواهیم کرد.

شکل ۵ بهره‌وری انرژی معماری‌های ممیزثابت بدون آفست و با آفست را در ضرایب موازی‌سازی مختلف نشان می‌دهد. محور عمودی در این نمودار بهره‌وری انرژی است. در این شبیه‌سازی K=۹×۲۵۶ و M=۳ در نظر گرفته شده‌است. بنابراین، طبق رابطه‌ی (۸)، مقدار ضریب موازی‌سازی بهینه ۴۸ به‌دست می‌آید. همان‌طور که در شکل مشخص است، هر دو معماری ممیزثابت بدون آفست و با آفست بیشترین بهره‌وری خود را در N=۴۸ به دست آورده‌اند. به‌طور کلی بهره‌وری انرژی در چندی‌سازی ممیزثابت بدون آفست بیشتر از ممیزثابت با آفست است. دلیل این موضوع این است که معماری ممیزثابت با آفست نسبت به ممیزثابت بدون آفست پیچیده‌تر است. این موضوع از مقایسه‌ی شکل ۲ و شکل ۴ نیز به‌دست می‌آید. لذا برای انجام یک ضرب داخلی مشابه (وقتی K یکسان باشد) ممیزثابت با آفست به توان مصرفی بیشتری نیاز دارد.

---

1. State of the Art



شکل ۵. بهره‌وری انرژی معماری ممیزثابت بدون آفست و با آفست در ضرایب موازی‌سازی مختلف. $K=9\times256$ و $N_{opt}=48$

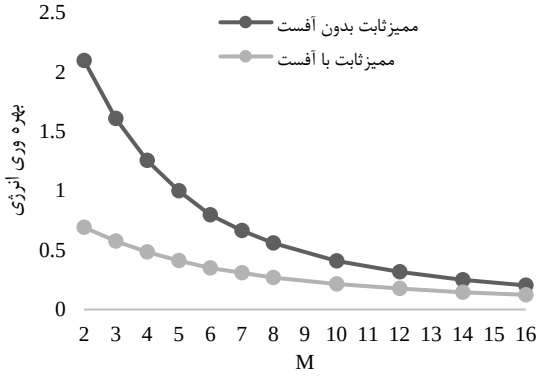
همان‌طور که در شکل ۵ مشخص است، اگر مقدار N را از نقطه‌ی بهینه کمتر انتخاب کنیم (ما در این شبیه‌سازی مقادیر ۶ و ۱۲ را برای مثال در نظر گرفتیم) موازی‌سازی به حداکثر خود نمی‌رسد، بنابراین مدت زمان محاسبه‌ی ضرب داخلی بیشتر شده (توان عملیاتی کم می‌شود) و بهره‌وری انرژی کاهش می‌یابد. اگر N را بیشتر از نقطه‌ی بهینه قرار دهیم (برای مثال ما ۹۶ و ۱۴۴ را در نظر گرفتیم)، بنابر دلایلی که در فصل قبل گفته شد، بهره‌وری انرژی از مقدار بیشنیه‌ی خود کمتر می‌شود.

۶.۳.۲ بهره‌وری انرژی معماری‌های پیشنهادی به ازای عرض بیت‌های مختلف

شکل ۶ بهره‌وری انرژی را برای دو معماری ممیزثابت بدون آفست و ممیزثابت با آفست نشان می‌دهد. نتایج این نمودار برای $K=9\times512$ و $M=\{2,\dots,16\}$ به دست آمده است.

براساس نتایج شکل ۶ مشخص می‌شود که معماری ممیزثابت بدون آفست، بهره‌وری انرژی بیشتری نسبت به ممیزثابت با آفست دارد. این موضوع را قبلاً در شکل ۵ نیز دیده بودیم. دلیل این موضوع نیز این است که معماری ممیزثابت با آفست نسبت به ممیزثابت بدون آفست پیچیده‌تر است و در توان عملیاتی یکسان، توان مصرفی بیشتری دارد لذا بهره‌وری انرژی آن در تمام عرض بیت‌ها کمتر است. نکته‌ی دیگری که در شکل ۶ می‌توان مشاهده کرد، این است که در هر دو معماری بدون آفست و با آفست، با افزایش تعداد بیت چندی‌سازی از ۲ تا ۱۶ بیت، به دلیل زیاد شدن توان مصرفی معماری، بهره‌وری انرژی کاهش می‌یابد چرا که توان عملیاتی تغییر نکرده است. این موضوع می‌تواند در انتخاب عرض بیت چندی‌سازی نقش مهمی ایفا کند.

با توجه به شکل ۶ مشخص می‌شود بهره‌وری انرژی در معماری بدون آفست، با شیب بسیار بیشتری نسبت به معماری با آفست کاهش می‌یابد. به‌طوری که در ۲ بیت، بهره‌وری انرژی معماری بدون آفست، حدوداً ۳ برابر بهره‌وری انرژی معماری با آفست است اما در ۱۶ بیت، تقریباً هر دو معماری به یک مقدار رسیده‌اند. دلیل این موضوع این است که با افزایش تعداد بیت، مدار ضرب‌کننده به صورت توان ۲ و جمع کننده به صورت خطی پیچیده می‌شود. در معماری بدون آفست، بین یک ضرب‌کننده و یک جمع‌کننده هزینه‌فایده[1] وجود دارد اما در معماری با آفست هزینه‌فایده بین یک ضرب‌کننده و ۳ جمع‌کننده است. لذا معماری بدون آفست شیب بیشتری دارد.



شکل ۶. بهره‌وری انرژی برای معماری‌های پیشنهادی، $K=9\times512$ بر حسب تعداد بیت‌های مختلف برای چندی‌سازی.

۶.۳.۳ بهره‌وری انرژی روش‌های چندی‌سازی مدرن

پیش از این گفته شد که هدف از انجام این پژوهش، مقایسه‌ی بی‌طرفانه‌ی مدل‌های مختلف چندی‌سازی ممیز ثابت از نظر ویژگی‌های سخت‌افزاری به‌خصوص بهره‌وری انرژی است. در بخش قبل بهره‌وری انرژی را برای معماری‌های پیشنهادی به‌صورت مستقل بررسی کردیم. در این بخش این موضوع را بررسی می‌کنیم که روش‌های چندی‌سازی موجود چه بهره‌وری انرژی خواهند داشت. برای این کار، از میان روش‌های چندی‌سازی مطرح که در بخش کارهای پیشین مرور شدند، در هر کدام از مدل‌های ممیزثابت بدون آفست و ممیزثابت با آفست، مدلی را انتخاب کردیم که به بالاترین صحت روی شبکه عصبی ResNet-18 رسیده است. با این توضیحات، در گروه ممیزثابت بدون آفست و با آفست، به ترتیب روش‌های LSQ و CSQ انتخاب شده‌اند.

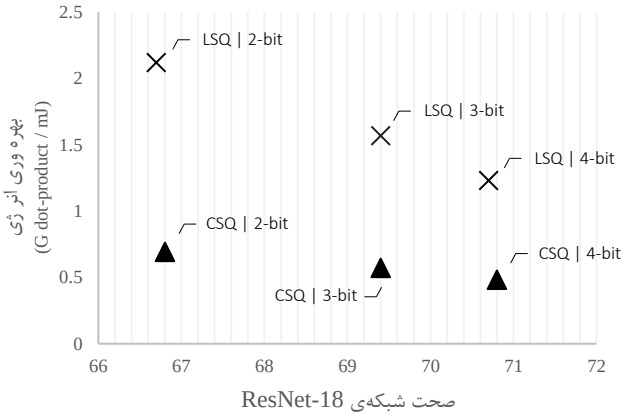
پس از آن بهره‌وری انرژی را برای هر روش چندی‌سازی، با توجه به مدل (بدون آفست یا با آفست) و عرض بیت آن (۲، ۳ یا ۴) محاسبه کردیم. ذکر این نکته ضروری است که ممکن است برای هر کدام از روش‌های چندی‌سازی گفته شده، شتاب‌دهنده‌های سخت‌افزاری دیگری وجود داشته باشد که از لحاظ انرژی مصرفی نسبت به مدل پیشنهادی این پژوهش کارآمدتر بوده و نتایج بهتری به‌دست آورده باشند اما از آنجا که شتاب‌دهنده‌های موجود بسیار با هم متفاوت‌اند و عملاً امکان مقایسه‌ای منصفانه میان آن‌ها وجود ندارد و همچنین بیشتر تکنیک‌ها و معماری‌های ارائه شده را می‌توان به هر دو مدل چندی‌سازی اعمال کرد، بهره‌وری انرژی روش‌های مختلف را با

1. Trade-off

استفاده از معماری‌های پیشنهادی این پژوهش محاسبه کرده‌ایم تا بستری فراهم کنیم که تمام روش‌های چندی‌سازی ممیزثابت در شرایط یکسان ارزیابی شوند.

شکل ۷ نتایج بهره‌وری انرژی را برای روش‌های چندی‌سازی مختلف نشان می‌دهد. محور افقی در این نمودار صحت شبکه‌ی عصبی ResNet-18 است که با مجموعه‌داده‌ی ImageNet آموزش دیده و روش‌های مختلف چندی‌سازی ممیزثابت روی آن اعمال شده است. این نتایج برای عرض بیت ۲، ۳ و ۴ محاسبه شده‌اند.

همان‌طور که در نتایج بخش بهره‌وری انرژی معماری‌های پیشنهادی دیدیم، هرچه عرض بیت در چندی‌سازی بیشتر شود، از یک طرف صحت شبکه بالاتر می‌رود و از طرف دیگر بهره‌وری انرژی کاهش می‌یابد. این موضوع را در شکل ۷ نیز می‌توان مشاهده کرد. برای مثال روش LSQ در ۲ بیت به دقت ۶۶/۷ می‌رسد و بهره‌وری انرژی آن ۲/۱۲ است. با افزایش تعداد بیت به ۴ بیت، صحت شبکه تا ۷۰/۷ افزایش می‌یابد (۳ درصد افزایش) اما از طرف دیگر، بهره‌وری انرژی به ۱/۲۳ کاهش می‌یابد (۱/۷ برابر کمتر).



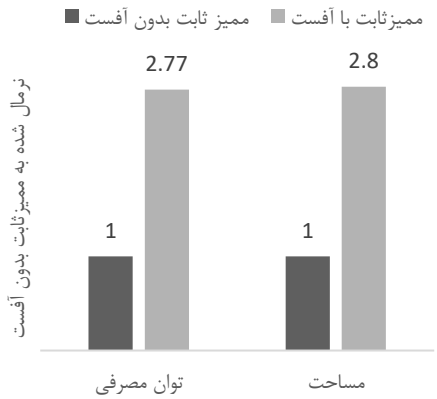
شکل ۷. بهره‌وری انرژی روش‌های مدرن چندی‌سازی در تعداد بیت مختلف. شکل‌های × و Δ به ترتیب مدل‌های ممیزثابت بدون آفست و ممیزثابت با آفست را را نشان می‌دهند. عرض بیت هر مدل به همراه نام آن روی نمودار مشخص است.

نکته‌ی دیگر نمودار شکل ۷ این است که در تمام عرض بیت‌های ۲، ۳ و ۴، مدل ممیزثابت با آفست، به صحت برابر یا بیشتری از ممیزثابت بدون آفست رسیده است. اما، بهره‌وری انرژی آن بسیار کمتر است. به بیان دیگر، بین صحت شبکه و بهره‌وری انرژی در روش‌های مختلف چندی‌سازی ممیزثابت یک هزینه‌فایده‌ی مهم وجود دارد. در مثال شکل ۷، حداکثر افزایش صحت شبکه ۰/۲ درصد است (در ۴ بیت، CSQ به ۰/۲ صحت بیشتری دست یافته است) اما بهره‌وری انرژی روش ممیزثابت با آفست، بسیار کمتر از روش بدون آفست است (به ترتیب ۲/۵، ۲/۷ و ۳/۱ برابر کمتر در ۴، ۳ و ۲ بیت). هزینه‌فایده‌ی میان صحت شبکه و معیارهای سخت‌افزاری مثل بهره‌وری انرژی، در حوزه‌ی چندی‌سازی شبکه‌های عصبی، به خصوص برای سیستم‌های نهفته موضوع بسیار مهمی است که تا کنون کسی به آن توجه نکرده است.

۶.۳.۴ تأثیر افزودن ضریب و آفست برتوان مصرفی و مساحت

در بخش انگیزه مقاله عنوان کردیم که یکی از سوالاتی که منجر به انجام این پژوهش شده است، این است که راهکارهای کاهش خطای چندی‌سازی و افزایش صحت شبکه، نظیر افزودن ضریب و آفست، چه تأثیری بر سخت‌افزار دارد. در این بخش قصد داریم این موضوع را بررسی کنیم.

شکل ۸ توان مصرفی و مساحت را برای معماری‌های ممیزثابت بدون آفست و ممیزثابت با آفست نشان می‌دهد که به نتایج ممیزثابت بدون آفست نرمال شده‌اند. نتایج برای حالتی محاسبه شده‌اند که K=۹×۶۴، N=۲۴ و M=۳ است.



شکل ۸. تأثیر افزودن ضریب و آفست بر توان مصرفی و مساحت سخت‌افزار. نتایج به ممیزثابت بدون آفست نرمال شده‌اند.

در نتایج بخش‌های قبل دیدیم وقتی که روش‌های چندی‌سازی از آفست استفاده می‌کنند، با کاهش خطای چندی‌سازی، صحت شبکه را افزایش می‌دهند (شکل ۷). از طرف دیگر، بررسی نتایج مختلف (شکل ۵ و شکل ۶) نشان داد که بهره‌وری انرژی برای معماری ممیزثابت با آفست، کمتر از ممیزثابت بدون آفست است. در توضیح دلیل این اتفاق، گفتیم که معماری ممیزثابت با آفست، نسبت به بدون آفست، پیچیدگی بیشتری دارد و در توان عملیاتی یکسان، توان مصرفی بیشتری دارد. این موضوع را که پیش از این به‌صورت تئوری بیان کردیم، می‌توان در شکل ۸، براساس نتایج شبیه‌سازی مشاهده کرد. براساس این نتایج، مشخص است که افزودن ضریب و آفست، توان مصرفی و مساحت را به ترتیب ۲/۷۷ و ۲/۸ برابر افزایش می‌دهد.

۶. نتیجه‌گیری

روش‌های مختلف چندی‌سازی سیستم نمایش اعداد متفاوتی دارند و عملیات مختلف را به شیوه‌های مختلف انجام می‌دهند، به همین دلیل به نظر می‌رسد اجرای هر کدام از روش‌ها به سخت‌افزار متفاوتی نیاز داشته باشد. به همین منظور، در این مقاله، روش‌های مختلف چندی‌سازی ممیزثابت در مدل‌های پایه دسته‌بندی شدند که شامل ممیزثابت بدون آفست و ممیزثابت با آفست بود و برای هر مدل یک معماری پیشنهاد داده شد تا بتوان به صورت عادلانه آن‌ها را مقایسه کرد و در نهایت معماری‌های پیشنهادی ارزیابی شدند.

براساس نتایج ارزیابی‌ها، مشخص شد که هر کدام از معماری‌های پیشنهادی برای چندی‌سازی ممیزثابت بدون آفست و با آفست، یک ضریب موازی‌سازی بهینه دارند که اگر کمتر یا بیشتر از آن مقدار باشد، بهره‌وری انرژی کاهش می‌یابد. پس از آن با مقایسه‌ی بهره‌وری انرژی معماری‌های ممیزثابت به این نتیجه رسیدیم که معماری ممیزثابت با آفست، همواره بهره‌وری انرژی کمتری نسبت به ممیزثابت بدون آفست در بیت یکسان دارد که این موضوع ناشی از معماری پیچیده‌تر آن است. در آخر، تأثیر افزودن ضریب و آفست را بر سخت‌افزار بررسی کردیم که مشخص شد افزودن ضریب و آفست به چندی‌سازی ممیزثابت، باعث افزایش توان مصرفی و مساحت تا حدود ۳ برابر می‌شود.

یکی از نتایج مهمی که در بررسی‌ها به‌دست آمده و از ابتدا انگیزه‌ی انجام این پژوهش بوده است، هزینه‌فایده‌ی میان صحت شبکه‌های عصبی چندی‌سازی‌شده و سربارهای سخت‌افزاری هم‌چون بهره‌وری انرژی است. براساس نتایج شبیه‌سازی، مشخص شد که روش‌های ممیزثابت با آفست، می‌توانند نسبت به روش‌های بدون آفست، به صحت‌های بالاتری (در حدود ۰/۱) در شبکه‌های عصبی دست پیدا کنند اما از طرف دیگر، بهره‌وری انرژی آن‌ها بسیار کاهش پیدا می‌کند (به‌طور میانگین ۲/۸ برابر کمتر) این موضوع در حوزه‌ی سیستم‌های نهفته که با محدودیت توان مصرفی و انرژی روبرو هستند، نقش بسیار مهمی دارد.

مراجع

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295-2329, 2017.

[2] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4340-4349.

[3] Z. Zhuang *et al.*, "Discrimination-aware channel pruning for deep neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[4] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[5] C. Baskin *et al.*, "Nice: Noise injection and clamping estimation for neural network quantization," *Mathematics*, vol. 9, no. 17, p. 2144, 2021.

[6] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, and N. Kwak, "Lsq+: Improving low-bit quantization through learnable offsets and better initialization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 696-697.

[7] S.-E. Chang *et al.*, "RMSMP: A Novel Deep Neural Network Quantization Framework with Row-wise Mixed Schemes and Multiple Precisions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5251-5260.

[8] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[9] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," *arXiv preprint arXiv:1412.7024*, 2014.

[10] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Hawq: Hessian aware quantization of neural networks with mixed-precision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 293-302.

[11] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153*, 2019.

[12] M. Ghasemzadeh, M. Samragh, and F. Koushanfar, "ReBNet: Residual binarized neural network," in *2018 IEEE 26th annual international symposium on field-programmable custom computing machines (FCCM)*, 2018: IEEE, pp. 57-64.

[13] T. Chen *et al.*, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 269-284, 2014.

[14] Y. Chen *et al.*, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014: IEEE, pp. 609-622.

[15] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5784-5789, 2018.

[16] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," *arXiv preprint arXiv:1604.03168*, 2016.

[17] S. Sharify *et al.*, "Laconic deep learning inference acceleration," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019: IEEE, pp. 304-317.

architecture," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021: IEEE, pp. 1-5.
[26] M. E. Salehi, "Binary neural networks," 2020.
[27] N. Nazari, S. A. Mirsalari, S. Sinaei, M. E. Salehi, and M. Daneshtalab, "Multi-level binarized lstm in eeg classification for wearable devices," in *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2020: IEEE, pp. 175-181.
[28] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International conference on machine learning*, 2015: PMLR, pp. 1737-1746.
[29] F. Asim, J. Park, A. Azamat, and J. Lee, "Centered Symmetric Quantization for Hardware-Efficient Low-Bit Neural Networks," 2022: British Machine Vision Association (BMVA).
[30] P. Judd *et al.*, "Reduced-precision strategies for bounded memory in deep neural nets," *arXiv preprint arXiv:1511.05236*, 2015.
[31] X. Zhao, Y. Wang, X. Cai, C. Liu, and L. Zhang, "Linear symmetric quantization of neural networks for low-precision integer hardware," 2020.
[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.
[33] J. L. McKinstry *et al.*, "Discovering low-precision networks close to full-precision networks for efficient inference," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, 2019: IEEE, pp. 6-9.
[18] S. Ghodrati, H. Sharma, C. Young, N. S. Kim, and H. Esmaeilzadeh, "Bit-parallel vector composability for neural acceleration," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020: IEEE, pp. 1-6.
[19] S. Jung *et al.*, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4350-4359.
[20] C. Gong *et al.*, "µl2q: An ultra-low loss quantization method for DNN compression," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019: IEEE, pp. 1-8.
[21] R. Gong *et al.*, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4852-4861.
[22] M. Nikolić *et al.*, "Bitpruning: Learning bitlengths for aggressive and accurate quantization," *arXiv preprint arXiv:2002.03090*, 2020.
[23] N. Nazari, M. Loni, M. E. Salehi, M. Daneshtalab, and M. Sjodin, "Tot-net: An endeavor toward optimizing ternary neural networks," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, 2019: IEEE, pp. 305-312.
[24] S. A. Mirsalari, N. Nazari, S. A. Ansarmohammadi, M. E. Salehi, and S. Ghiasi, "E2BNet: MAC-free yet accurate 2-level binarized neural network accelerator for embedded systems," *Journal of Real-Time Image Processing*, vol. 18, pp. 1285-1299, 2021.
[25] S. A. Mirsalari, N. Nazari, S. A. Ansarmohammadi, S. Sinaei, M. E. Salehi, and M. Daneshtalab, "ELC-ECG: Efficient LSTM Cell for ECG classification based on quantized

Energy-Efficient Fixed-Point Hardware Accelerator for Embedded DNNs

***Abstract*:**
Deep Neural Networks (DNNs) have demonstrated remarkable performance in various application domains, such as computer vision, pattern recognition, and natural language processing. However, deploying these models on edge-computing devices poses a challenge due to their extensive memory requirements and computational complexity. These factors make it difficult to deploy DNNs on low-power and limited-resource devices.

One promising technique to address this challenge is quantization, particularly fixed-point quantization. Previous studies have shown that reducing the bit-width of weights and activations, such as to 3 or 4 bits, through fixed-point quantization can preserve the classification accuracy of full-precision neural networks.

Despite extensive research on the compression efficiency of fixed-point quantization techniques, their energy efficiency, as a critical metric in evaluating embedded systems, has not been thoroughly explored. Therefore, this research aims to assess the energy efficiency of fixed-point quantization techniques while maintaining accuracy.

To accomplish this, we present a model and design an architecture for each quantization method. Subsequently, we compare their area and energy efficiency at the same accuracy level. Our experimental results indicate that incorporating scaling factors and offsets into LSQ, a well-known quantization method, improves DNN accuracy by 0.1%. However, this improvement comes at the cost of a 3× decrease in hardware energy efficiency.

This research highlights the significance of evaluating fixed-point quantization techniques not only in terms of compression efficiency but also in terms of energy efficiency when applied to edge-computing devices.

***Keywords*:** Deep Neural Network, Embedded Systems, Energy-Efficiency, Fixed-point Quantization