مقاله پژوهشی

طراحی شتاب‌دهنده تقریبی کم‌توان بر بستر تراشه‌های FPGA برای کاربردهای هوش مصنوعی

نادیا سهرابی، امیر باوفای طوسی و مهدی صدیقی

چکیده: یکی از روش‌های یادگیری ماشین شبکه‌های عصبی می‌باشند که در کاربردهایی نظیر پردازش تصویر به کار می‌روند. یکی از چالش‌های شبکه‌های عصبی، حجم بالای محاسبات آنهاست. به همین دلیل معماری‌های زیادی برای این گونه کاربردها ارائه شده که راه‌حل‌هایی برای محاسبات پیچیده آنها ارائه می‌دهند. معمولاً برای تسریع الگوریتم‌های شبکه‌های عصبی از شتاب‌دهنده‌های سخت‌افزاری قابل بازپیکربندی مانند تراشه‌های FPGA استفاده می‌شود؛ اما مشکل اصلی این تراشه‌ها توان مصرفی نسبتاً بالای آنهاست. برای کاهش توان مصرفی در تراشه‌های FPGA از تکنیک محاسبات تقریبی می‌توان استفاده کرد. ایده اصلی محاسبات تقریبی این است که با ایجاد تغییراتی در مدار یا کد، بین دقت و مصرف انرژی مصالحه‌ای برقرار شود. در این پژوهش یک شبکه عصبی کانولوشنی برای تشخیص ارقام دست‌نویس به‌صورت دقیق و تقریبی با هدف بهبود توان مصرفی طراحی و پیاده‌سازی شده است. ایده تقریب‌سازی در بخش محاسبات جمع‌کننده شبکه عصبی ارائه شده است. این روش با جلوگیری از انتشار رقم نقلی در بیت‌های پایین جمع‌کننده، توان مصرفی را کاهش می‌دهد. نتایج مقایسه شبکه عصبی به‌صورت دقیق و تقریبی نشان می‌دهد که با تقریب‌سازی ٦ بیت وزن پایین جمع‌کننده، توان مصرفی ۴۳٫۷۵٪ کاهش می‌یابد و هیچ خطایی رخ نمی‌دهد.

***کلیدواژه:* جمع‌کننده تقریبی، شبکه عصبی کانولوشنی، طراحی شبکه عصبی تشخیص ارقام دست‌نویس، محاسبات تقریبی.**

۱- مقدمه

امروزه در زمینه یادگیری ماشین، پیشرفت‌های چشمگیری به وجود آمده است. یکی از پرکاربردترین مباحث در زمینه یادگیری ماشین شبکه‌های عصبی می‌باشد. شبکه‌های عصبی در کاربردهای مختلفی مانند پردازش تصویر، تشخیص صدا و فشرده‌سازی تصویر به کار می‌روند و از اهمیت بالایی برخوردار هستند.

شبکه‌های عصبی در هر لایه دارای حجم زیادی از محاسبات هستند و به‌منظور انجام کارهای پیچیده‌تر و بیشتر، تعداد لایه‌های شبکه‌های عصبی افزایش می‌یابند. این لایه‌ها شامل تعداد زیادی ضرب و جمع هستند که باید با سرعت بالایی انجام شوند. هرچه شبکه‌های عصبی بزرگ‌تر می‌شوند و تعداد لایه‌ها و سرعت پردازش آنها افزایش می‌یابند، مصرف انرژی آنها نیز افزایش می‌یابد. پژوهشگران به دنبال راه‌حلی برای کاهش و مدیریت این مصرف انرژی هستند.

از سوی دیگر، برخی از کاربردها می‌توانند اندکی کاهش دقت در محاسبات خود را تحمل کنند، بدون این که عملکرد کلی آنها دچار آسیب جدی شود. چنین تحمل خطایی در یک الگوی طراحی و پیاده‌سازی جدید به نام محاسبات تقریبی[1] می‌تواند مورد استفاده قرار گیرد. ایده اصلی محاسبات تقریبی این است که با ایجاد تغییراتی در مدار یا کد، بین دقت و مصرف انرژی یا برخی پارامترهای دیگر مصالحه‌ای برقرار شود.

یکی از کاربردهایی که استفاده از محاسبات تقریبی در آن مناسب است، پیاده‌سازی شبکه‌های عصبی است [۱]. مطالعات نشان داده‌اند در شبکه‌های عصبی افزونگی[2] وجود دارد [۲]؛ بنابراین قابلیت تحمل‌پذیری خطا نیز می‌تواند وجود داشته باشد که این امر زمینه را برای به‌کارگیری محاسبات تقریبی مناسب می‌سازد. پس می‌توان نتیجه گرفت که در شبکه‌های عصبی، ایجاد مصالحه‌ای بین دقت و حجم محاسبات یک نیاز مهم است که طراحان را به چالش می‌کشد تا طراحی با کارایی بالاتر و بهره‌وری انرژی بهتر ارائه دهند.

معمولاً برای تسریع در پردازش داده‌ها و الگوریتم‌های شبکه‌های عصبی از پردازنده‌های خاص‌منظوره، پردازنده‌های گرافیکی و یا شتاب‌دهنده‌های سخت‌افزاری استفاده می‌شود. نتایج مقایسه‌ها نشان می‌دهد که طراحی ASIC به دلیل انعطاف‌پذیری بسیار کم و پیچیدگی طراحی، معمولاً در کاربردهای خاص‌منظوره مورد استفاده قرار می‌گیرد. همچنین تراشه‌های FPGA[3] نسبت به پردازنده‌های گرافیکی کارایی بالاتر و توان مصرفی پایین‌تری دارند؛ اما طراحی آنها پیچیده‌تر است [۳].

سیستم‌های قابل بازپیکربندی مانند تراشه‌های FPGA به دلیل قابلیت برنامه‌ریزی مجدد، هزینه کمتر و سرعت توسعه و طراحی سریع‌تر نسبت به تراشه‌های ASIC برای کاربردهای یادگیری عمیق و هوش مصنوعی انتخاب مناسب‌تری هستند. استفاده از تراشه‌های قابل بازپیکربندی در طراحی شتاب‌دهنده‌های یادگیری عمیق مرسوم بوده است؛ اما از نظر مصرف انرژی چالش‌های زیادی و توان مصرفی بالایی دارند. به همین دلیل پژوهش‌هایی برای کاهش توان مصرفی آنها ارائه شده است [۴].

در پژوهش‌های پیشین برخی از رویکردهای رایج برای طراحی تقریبی شبکه‌های عصبی مانند هرس وزن[4] و کاهش عرض بیتی داده را می‌توان به‌عنوان نمونه‌هایی از تقریب‌سازی محاسبات در نظر گرفت [۵]. همچنین

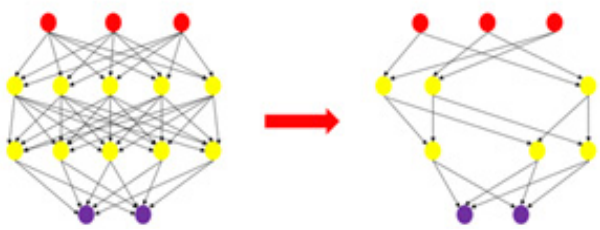
این مقاله در تاریخ ۲۲ اردیبهشت ماه ۱۴۰۳ دریافت و در تاریخ ۱۶ دی ماه ۱۴۰۳ بازنگری شد.

نادیا سهرابی، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، ایران، (email: nadiasohrabi@aut.ac.ir).

امیر باوفای طوسی (نویسنده مسئول)، دانشکده کامپیوتر، دانشگاه سجاد، مشهد، ایران، (email: abavafat@sadjad.ac.ir).

مهدی صدیقی، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، ایران، (email: msedighi@aut.ac.ir).

1. Approximate Computing
2. Redundancy
3. Field-Programmable Gate Array
4. Weight Pruning



شکل ۱: عملیات هرس وزن.

در برخی از پژوهش‌های پیشین مانند [۱] و [۶] بر روی نوع دیگری از روش‌های محاسبات تقریبی برای طراحی شتاب‌دهنده‌های شبکه عصبی، یعنی جایگزینی مدارهای حسابی با نسخه‌های تقریبی آنها تمرکز شده است.

یکی دیگر از تکنیک‌های متداول در مدارهای CMOS برای صرفه‌جویی انرژی، کاهش ولتاژ منبع تغذیه است [۷]؛ اما این تکنیک معایبی دارد: ۱) با کاهش ولتاژ، تأخیر گیت‌ها و مدار افزایش می‌یابد. برای غلبه بر این مشکل ولتاژ آستانه را کاهش می‌دهند. ۲) حاشیه نویز یکی از نگرانی‌های مهم در طراحی مدارها و سیستم‌ها می‌باشد. این تکنیک باعث تخریب حاشیه نویز امن مدار می‌شود. برای حل این گونه مشکل‌ها روش‌های جدیدی در برخی از پژوهش‌ها ارائه شده است [۸] و [۹]. این پژوهش‌ها با استفاده از اصول محاسبات تقریبی، تقریب‌های کاربردی را برای طراحی انواع مختلف ضرب‌کننده‌های تقریبی با عملکرد متفاوت پیشنهاد می‌کنند. این پژوهش‌ها بر روی تراشه‌های ASIC طراحی شده‌اند و به دلیل تفاوت‌های بین معماری تراشه‌های FPGA و ASIC، بیشتر این تکنیک‌ها در صورت ترکیب مستقیم برای سیستم‌های مبتنی بر تراشه‌های FPGA، کارایی محدودی را به همراه دارند [۱۰]. به همین دلیل برای تأکید بیشتر بر نیاز به طراحی واحدهای تقریبی مبتنی بر تراشه‌های FPGA، مطالعاتی نظیر [۱۱] و [۱۲] انجام شده است.

همچنین در پژوهش‌های پیشین طرح‌های زیادی برای تقریبی‌کردن جمع‌کننده‌ها ارائه شده است [۱۳] و [۱۴].

در این مقاله، یک شتاب‌دهنده شبکه عصبی عمیق از نوع شبکه عصبی کانولوشنی بر بستر تراشه‌های FPGA پیشنهاد شده که با تقریبی‌سازی بخش محاسبات، توان مصرفی تراشه را کاهش می‌دهد. هدف این پژوهش طراحی و پیاده‌سازی شتاب‌دهنده‌های مبتنی بر تراشه‌های FPGA در کاربردهای هوش مصنوعی با استفاده از محاسبات تقریبی برای کاهش توان مصرفی می‌باشد. با توجه به ساختار شبکه عصبی کانولوشنی و حجم بالای پارامترهای این شبکه، جهت کاهش حجم محاسبات و کاهش توان مصرفی آن از ایده تقریبی‌سازی ۶ بیت از ۱۰ بیت مانتیس اعداد ممیز شناور در بخش محاسبات جمع‌کننده شتاب‌دهنده استفاده شده است. تقریب مورد نظر پیچیدگی محاسباتی را کاهش داده و باعث بهبود توان مصرفی با حفظ دقت شبکه شده است.

جهت پیاده‌سازی شبکه عصبی و آموزش شبکه از ابزار متلب استفاده شده است. سپس با توجه به شبکه آموزش‌دیده در متلب و پارامترهای به‌دست‌آمده از آن، شتاب‌دهنده در سطح انتقال ثبات با استفاده از زبان توصیف سخت‌افزار وریلاگ طراحی و در محیط ویوادو[1] پیاده‌سازی شده و نهایتاً سنتز واحدها در محیط ویوادو انجام شده است. در این پژوهش از تراشه‌ها و ابزارهای شرکت زایلینکس[2] استفاده گردیده است.

در ادامه این مقاله در بخش دوم پژوهش‌های پیشین و مرتبط در زمینه طراحی شتاب‌دهنده‌ها و محاسبات تقریبی بررسی شده است. در بخش سوم روش پیشنهادی شامل نوآوری و جزئیات روش شرح داده شده است. در بخش چهارم روش پیشنهادی مورد ارزیابی قرار گرفته و نتایج آن بررسی شده و نهایتاً بخش پنجم به نتیجه‌گیری اختصاص داده شده است.

۲- پژوهش‌های پیشین

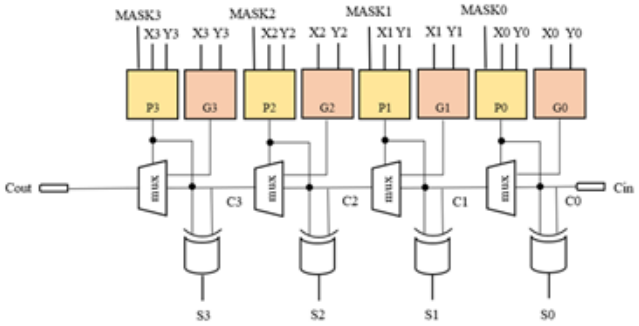
در این بخش به بررسی برخی مطالعات انجام‌شده در زمینه شبکه‌های عصبی و محاسبات تقریبی پرداخته شده است. تکنیک‌های محاسبات تقریبی را می‌توان به دو دسته نرم‌افزاری[3] [۵] و [۱۵] تا [۱۸] و سخت‌افزاری[4] [۱۱]، [۱۲] و [۱۹] تا [۲۲] طبقه‌بندی کرد. هنگامی که تقریب در نرم‌افزار اعمال می‌شود، تغییرات الگوریتمی در کد ایجاد می‌شود تا دقت و توان مصرفی را جبران کند. در روش‌های سخت‌افزاری، تغییرات ممکن است به‌صورت دستی در سطح مدار یا نظیر کارهای اخیر، با تغییر توصیف طراحی در سطوح بالاتر از طریق روش‌های سنتز تا حدودی خودکار انجام شوند. روش هرس وزن، یک تکنیک نرم‌افزاری [۵] برای از بین بردن گره‌های زائد و حفظ اتصالات مفید شبکه‌های عصبی است که می‌تواند منابع محاسباتی و ذخیره‌سازی را کاهش دهد. عملیات هرس وزن سرعت را بهبود می‌بخشد و همچنین انرژی مورد نیاز را برای اجرای شبکه‌های بزرگ ضمن حفظ دقت پیش‌بینی کاهش می‌دهد. این روش ضمن حفظ دقت شبکه، یک شبکه عصبی متراکم را به یک شبکه عصبی تنک تبدیل می‌کند که باعث کاهش محاسبات و میزان حافظه مورد نیاز جهت ذخیره‌سازی وزن‌ها می‌شود. شکل ۱ عملیات هرس وزن را نشان می‌دهد.

روش دیگر تقریب‌سازی، کاهش عرض بیتی داده می‌باشد. عرض بیتی تأثیر بسیاری در ساختار معماری شتاب‌دهنده، واحدهای محاسباتی، دقت نتایج و میزان منابع مصرفی دارد. بررسی‌ها نشان می‌دهد که در لایه‌های مختلف شبکه‌های عصبی، عرض بیتی متفاوتی مورد نیاز است. قالب عددی رایج برای کاربردهای یادگیری عمیق دقت تک‌بیتی (۳۲بیتی) است؛ با این حال قالب‌های با دقت کم، مزایایی مانند کاهش پیچیدگی و افزایش کارایی واحدهای محاسباتی دارند که بتوانند عملیاتی مانند ضرب و جمع ماتریس را تسریع کنند.

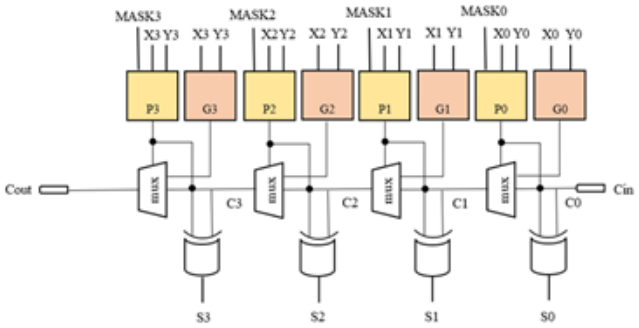
پژوهش [۱۹] روشی برای شبکه‌های عصبی کانولوشنی ارائه می‌دهد که از محاسبات با دقت ثابت[5] ۸بیتی استفاده می‌کند. این روش با کاهش دقت محاسبات از ممیز شناور به دقت ثابت ۸بیتی، حجم داده‌ها و توان مصرفی را کاهش می‌دهد. این کاهش دقت تأثیر قابل توجهی بر دقت کلی مدل ندارد، زیرا شبکه‌های عصبی کانولوشنی نسبت به کاهش دقت تا حدی مقاوم هستند.

در [۲۰] یک جمع‌کننده تقریبی به نام جمع‌کننده قابل پوشاندن رقم نقلی[6] ارائه شده است. دلیل ارائه این جمع‌کننده آن است که می‌تواند برای برای کاربردهای مختلف به‌صورت پویا دقت جمع را تغییر دهد. در [۱۶]، CMA[7] برای تراشه‌های ASIC طراحی شده است. در روش [۲۰] برای یک جمع‌کننده هشت‌بیتی CMA از یک بیت *MASK* برای کنترل انتشار

1. Vivado
2. Xilinx
3. Software
4. Hardware
5. Fixed-Point
6. Maskable Carry Adder
7. Carry-Maskable Adder



شکل ۲: جمع‌کننده تقریبی ۴بیتی [۱۶].



شکل ۳: تمام جمع‌کننده قابل پوشاندن رقم نقلی [۱۵].

رقم نقلی، یک نیم‌جمع‌کننده و ۷ تمام‌جمع‌کننده استفاده شده است. در شکل ۲ مدار یک جمع‌کننده تقریبی ۴بیتی نشان داده شده است. در این جمع‌کننده، علاوه بر دو عدد ورودی X و Y و رقم نقلی Cin، یک ورودی جدید برای کنترل دقت به نام $MASK$ تعبیه شده است.

روابط خروجی هر بلوک در عملیات جمع تقریبی در (۱) و (۲) نشان داده شده است

$$P = (X.Y.MASK) + (X \,???\, Y) \quad (۱)$$

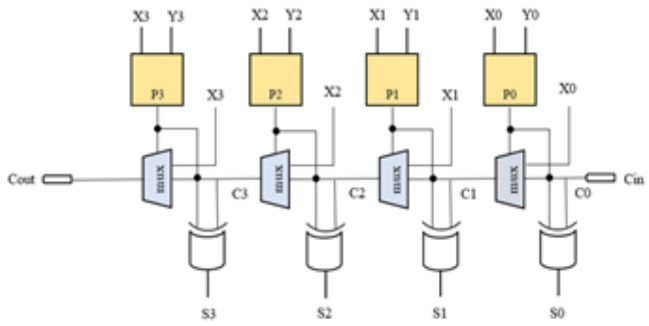
$$G = X.Y \quad (۲)$$

هنگامی که مقدار بیت $MASK$، ۰ باشد جمع‌کننده جمع دقیق را انجام می‌دهد. هنگامی که مقدار بیت $MASK$ ۱ باشد جمع‌کننده با جلوگیری از انتشار رقم نقلی از بیت پایین به بیت بالا، جمع تقریبی را انجام می‌دهد. در حالت تقریبی، نتیجه جمع حاصل‌جمع منطقی (OR) دو ورودی است؛ پس عدم انتشار بیت نقلی باعث کاهش توان مصرفی و تأخیر می‌شود.

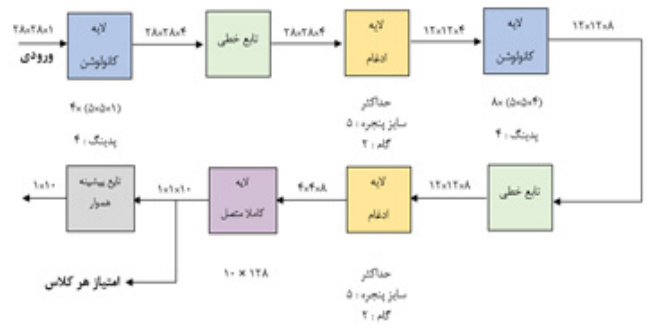
در این جمع‌کننده با ۱ قراردادن $MASK$ بیت‌های وزن پایین بر روی ۱ و $MASK$ بیت‌های وزن بالا بر روی ۰، بیت‌های وزن بالا به‌طور دقیق محاسبه می‌شوند و بیت‌های وزن پایین به‌صورت تقریبی محاسبه می‌شوند؛ بنابراین با کنترل $MASK$ بر اساس کاربرد می‌توان مصالحه‌ای را بین دقت، تأخیر و مصرف انرژی برقرار کرد (شکل ۳).

در [۲۰] CMA برای تراشه‌های ASIC طراحی گردیده است و نحوه پیاده‌سازی CMA بر روی تراشه‌های FPGA بررسی نشده است. اگر عبارات بولی CMA به ابزارهای سنتز تراشه‌های FPGA داده شود، هر تمام‌جمع‌کننده آن به یک یا دو جدول جست‌وجو نگاشت می‌شود. در این حالت، جداول جست‌وجو به‌صورت سری به هم متصل می‌شوند. این پیاده‌سازی از نظر تأخیر و مصرف انرژی بهینه نیست؛ زیرا جداول جست‌وجو تأخیر و توان مصرفی بالایی دارند. به همین دلیل [۲۱] روشی برای پیاده‌سازی CMA بر روی تراشه‌های FPGA ارائه داده است.

تراشه‌های FPGA دارای زنجیره رقم نقلی هستند. شکل ۴ شماتیکی از یک جمع‌کننده دقیق ۴بیتی را با استفاده از یک واحد زنجیره رقم نقلی



شکل ۴: جمع‌کننده دقیق ۴بیتی با زنجیره رقم نقلی [۲۱].



شکل ۵: معماری شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس.

نشان می‌دهد. واحد زنجیره رقم نقلی از ۴ مالتی‌پلکسر، ۴ دروازه XOR و ۴ جدول جست‌وجو تشکیل شده است. در جمع‌کننده دقیق هر جدول جست‌وجو XOR دو ورودی را محاسبه می‌کند؛ در نتیجه، رقم نقلی از طریق جداول جست‌وجو منتشر نمی‌شود. نهایتاً این پژوهش با استفاده از جمع‌کننده تقریبی، جمع‌کننده شکل ۴ را ارائه داده که در این جمع‌کننده از زنجیره رقم نقلی استفاده شده است [۲۱].

پژوهش [۲۲] با تمرکز بر تکرارهای محاسباتی در پردازش داده‌ها و عملیات در شبکه‌های عصبی کانولوشنی، روشی برای کاهش مصرف انرژی ارائه می‌دهد. این روش عملیات مشابه در فیلترها را شناسایی می‌کند و تنها آنها را یک بار اجرا می‌کند. سپس داده‌های پردازش‌شده را برای سایر محاسبات باز استفاده می‌کند. حذف این تکرارها یا اشتراک‌گذاری محاسبات با کاهش بار محاسباتی و کاهش مصرف حافظه، منابع سخت‌افزاری را آزاد می‌کند. روش پیشنهادی توانسته است با حفظ دقت توان پویا شتاب‌دهنده را ۱۰/۷۹٪ تا ۱۲/۱۷٪ کاهش دهد. همچنین زمان اجرای لایه‌های مختلف را کاهش داده و باعث افزایش سرعت پردازش شده است.

۳- پیاده‌سازی شتاب‌دهنده تقریبی کم‌توان

در این بخش، طراحی معماری شتاب‌دهنده شبکه عصبی کانولوشنی در محیط متلب و پیاده‌سازی آن با زبان توصیف سخت‌افزار وریلاگ در محیط ویوادو شرح داده شده است.

۳-۱ طراحی شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس

در این مقاله برای طراحی و آموزش شبکه عصبی کانولوشنی از جعبه ابزار یادگیری ماشین در محیط متلب استفاده شده است. متلب یک ابزار پرکاربرد برای یادگیری ماشین است و چندین توابع داخلی، جعبه ابزار و کتابخانه برای پشتیبانی از یادگیری ماشین ارائه می‌دهد. معماری مدل در شکل ۵ نشان داده شده است. شبکه عصبی کانولوشنی برای طبقه‌بندی

شکل ۶: مجموعه تصاویر ارقام دست‌نویس (MNIST).

اعداد دست‌نویس طراحی و پیاده‌سازی شده و شامل لایه‌های کانولوشن[1]، لایه ادغام[2]، لایه کاملاً متصل[3] و تابع ReLU می‌باشد.

در این پژوهش برای آموزش و آزمودن شبکه از مجموعه تصاویر ارقام دست‌نویس (MNIST) استفاده شده است. ورودی شبکه یک عکس سیاه و سفید از ارقام دست‌نویس با ابعاد ۲۸×۲۸ پیکسل می‌باشد. هر پیکسل یک تصویر خاکستری است؛ به همین دلیل ورودی شبکه یک ماتریس دوبعدی از اعداد است که هر عدد مقداری بین ۰ و ۲۵۵ دارد که توسط ابزار متلب تبدیل به اعداد بین ۰ و ۱ شده‌اند. مقدار صفر معادل با رنگ سیاه و مقدار ۲۵۵ معادل با رنگ سفید است. همان طور که در شکل ۶ نشان داده شده است، هر عکس نشان‌دهنده یکی از اعداد ۰ تا ۹ از دیتاست MNIST می‌باشد و خروجی شبکه شامل ده کلاس (۰ تا ۹) است که هر کلاس امتیاز کسب‌شده توسط آن کلاس را نشان می‌دهد.

۳-۲ آموزش شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس

آموزش شبکه شامل تنظیم وزن نورون‌های شبکه برای حداقل‌کردن تفاوت بین خروجی پیش‌بینی‌شده و خروجی واقعی است. این امر از طریق فرایندی به نام انتشار روبه‌جلو[4] به دست می‌آید که الگوریتمی برای محاسبه گرادیان تابع هزینه با توجه به وزن‌هاست. شکل ۷ نشان‌دهنده فرایند آموزش در محیط متلب می‌باشد. همان طور که در شکل نشان داده شده است، فرایند آموزش برای شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس در ۱۰ دوره تکرار می‌شود و هر دوره شامل یک گذر کامل از مجموعه داده آموزشی است. در هر دوره پس از فرایند انتشار روبه‌جلو، با استفاده از تابع هزینه خطای خروجی محاسبه می‌شود.

پس از محاسبه خطای خروجی، در فرایند پس‌انتشار مشتق تابع هزینه محاسبه می‌شود و نهایتاً با استفاده از یک الگوریتم بهینه‌سازی مانند تابع گرادیان نزولی تصادفی (SGD)[5] وزن‌ها به‌روزرسانی می‌شوند. وزن نورون‌ها پس از هر دسته از نمونه‌های آموزشی به‌روزرسانی می‌شود.

در طول فرایند آموزش، شبکه یاد می‌گیرد که ویژگی‌های مهم ارقام دست‌نویس را شناسایی و آنها را در یکی از ده کلاس طبقه‌بندی کند. در شکل ۷ دو نمودار دقت و خطای شبکه عصبی نشان می‌دهند وزن‌های شبکه به گونه‌ای تنظیم می‌شوند که با کاهش خطا مدل دقیق‌تری در طول زمان ایجاد شود. دقت تست شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس در این مدل ۹۸٪ می‌باشد.

۳-۳ پیاده‌سازی شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس

در این پژوهش ابتدا مدل دقیق شبکه عصبی کانولوشنی و سپس مدل تقریبی آن شرح داده می‌شود. در شکل ۸ ساختار کلی مدل و سلسله‌مراتب آن نشان داده شده است. این شبکه به ترتیب دارای واحدهای کانولوشن، تابع فعال‌ساز (ReLU)، ادغام و کاملاً متصل می‌باشد.

در این مدل، خروجی هر واحد، ورودی واحد بعدی آن است که هر واحد با یک‌شدن سیگنال شروع خود فعال می‌شود.

ورودی‌های شبکه شامل سیگنال تک‌بیتی پالس ساعت، ریست و سیگنال شروع می‌باشد. خروجی شبکه شامل ۱۶ بیت نشان‌دهنده امتیاز هر کلاس و سیگنال اتمام کار شبکه است. در این شبکه ۷۸۴ پیکسل تصویر ورودی (۱۶بیتی) در یک حافظه RAM ذخیره می‌شوند و بعد از یک‌شدن سیگنال فعال‌سازی شبکه، پیکسل‌ها به‌صورت سریالی از حافظه خوانده و به واحد کانولوشن ارسال می‌شوند.

برای پیاده‌سازی عملیات حسابی این شبکه مانند جمع و ضرب، یک واحد جمع‌کننده و ضرب‌کننده ممیز شناور پیاده‌سازی شده است. در این دو واحد ورودی‌ها (۲ ورودی) اعداد ۱۶بیتی شامل ۱ بیت (بیت علامت)، ۵ بیت نما و ۱۰ بیت مانتیس است. همان طور که در شکل ۹ نشان داده شده است، بیت علامت خروجی با انجام عملیات XOR بر روی بیت علامت ورودی‌ها به دست می‌آید. سپس برای انجام عملیات ضرب یا جمع بعد از یکسان‌سازی نماها، ضرب یا جمع انجام می‌شود. نهایتاً پس از انجام نرمال‌سازی خروجی نهایی تولید می‌شود.

خروجی اصلی شبکه، شامل ۱۰ عدد ۱۶بیتی می‌باشد که خروجی هر نورون امتیاز آن کلاس را به ترتیب نشان می‌دهد. کلاسی که دارای بالاترین امتیاز است نشان‌دهنده کلاس صحیح می‌باشد.

۳-۴ طراحی جمع‌کننده تقریبی

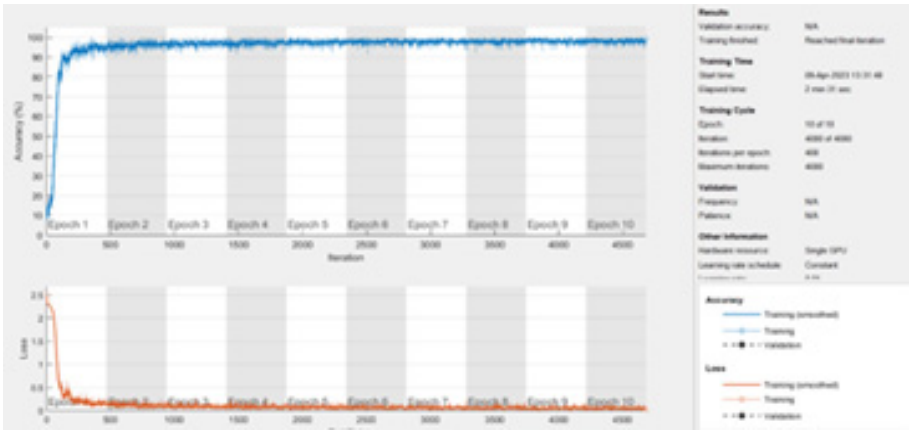
در مطالعات اخیر پژوهش‌های زیادی در سطوح مختلف طراحی از ترانزیستور تا معماری بر روی مدارهای محاسباتی تقریبی انجام شده است. در این مقاله برای تقریب‌سازی شبکه عصبی کانولوشن تشخیص ارقام دست‌نویس، با هدف کاهش توان مصرفی و حفظ دقت مطلوب از جایگذاری جمع‌کننده تقریبی به‌جای جمع‌کننده دقیق استفاده شده است. دلیل انتخاب این جمع‌کننده قابلیت کنترل دقت مورد نیاز در تقریب‌سازی بر اساس کاربرد است. در ادامه به بررسی روش ارائه‌شده و مدل تقریبی شبکه عصبی پرداخته شده است.

۳-۵ جمع‌کننده و تفریق‌کننده تقریبی

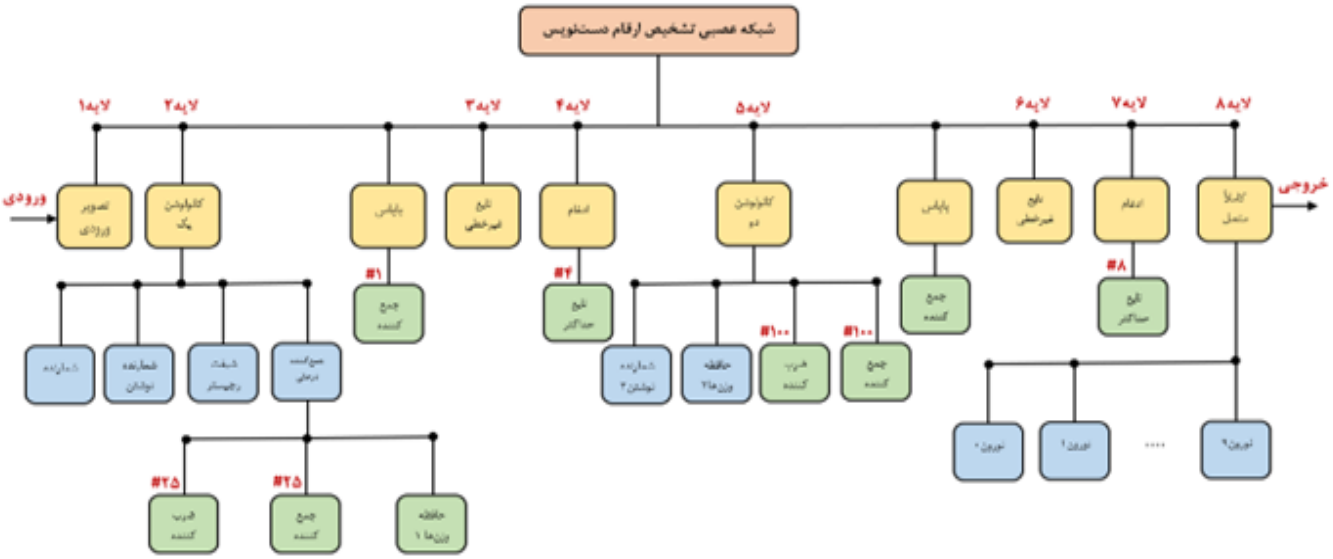
در شکل ۱۰ روابط جمع‌کننده دقیق و تقریبی نشان داده شده است. جمع‌کننده تقریبی استفاده‌شده در این مقاله بر اساس روابط جدول ۱ روش تقریب‌سازی ارائه شده است. در این روش تقریب‌سازی در جمع و تفریق مانتیس اعداد ورودی، در جمع‌کننده ممیز شناور انجام می‌شود.

برای تقریب‌سازی در این پژوهش و رسیدن به دقت لازم شبکه و بهبود توان مصرفی از ایده تقریب‌سازی نیمی از بیت‌های مانتیس اعداد ممیز شناور استفاده شده است. دلیل استفاده از این ایده در تقریب‌سازی در ادامه شرح داده شده است. پیکسل‌های تصویر ورودی ۷۸۴ عدد بین ۰ تا ۲۵۵ هستند که در ابزار متلب به اعدادی بین ۰ و ۱ تبدیل شده‌اند. در مجموعه تصاویر ارقام دست‌نویس به‌طور حدودی می‌توان گفت نیمی از پیکسل‌های هر تصویر، کمتر از حد وسط بازه [۰,۲۵۵] و نیمی دیگر بیشتر از حد وسط بازه می‌باشند. به همین دلیل پیش‌بینی می‌شود که با

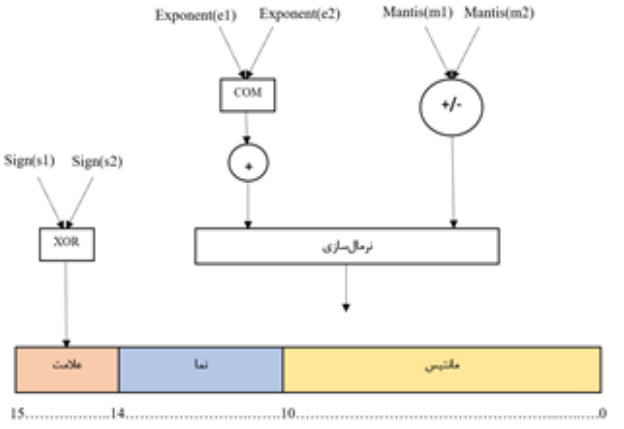
1. Convolution Layer
2. Pooling Layer
3. Fully Connected Layer
4. Feed-Forward
5. Stochastic Gradient Descent



شکل ۷: فرایند آموزش شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس.



شکل ۸: معماری شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس.



شکل ۹: ساختار واحد جمع‌کننده ممیز شناور.

تقریبی‌کردن نیمی از ۱۱ بیت مانتیس جمع‌کننده (۶ بیت وزن پایین)، حداقل ۵۰٪ از اعداد دچار خطا نشوند. منظور از خطا تبدیل‌شدن اعداد مثبت به منفی و بالعکس می‌باشد.

در این شبکه واحد ReLU نقش بسیار مهمی در عملیات تقریب‌سازی شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس دارد؛ زیرا حد آستانه در این ماژول ۰ می‌باشد. اگر تعداد اعداد دارای خطا زیاد باشد، واحد ReLU اعدادی که با خطا تبدیل به منفی شده‌اند را صفر و اعدادی که مثبت شده‌اند را بدون تغییر برمی‌گرداند. نهایتاً با انتشار خطا، دقت شبکه کاهش می‌یابد و ممکن است منجر به تشخیص اشتباه در برخی کلاس‌ها

جدول ۱: روابط جمع و تفریق تقریبی.

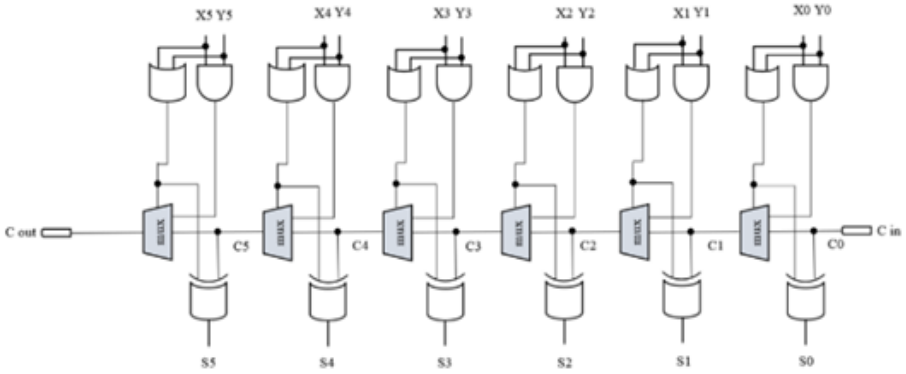
	عملیات جمع	عملیات تفریق
P	$P = x + y$	$P = x + y$
G	$G = x.y$	$G = \sim x.y$
S	$؟؟؟\ Cin\ ؟\ P$	$؟؟؟\ Bin\ ؟\ P$
$Cout\ /\ Bout$	$P = ۰ \rightarrow G$ $P = ۱ \rightarrow Cin$	$P = ۰ \rightarrow Bin$ $P = ۱ \rightarrow G$

شود. در شکل‌های ۱۱و ۱۲، تقریب‌سازی جمع‌کننده و تفریق‌کننده در شبکه تشخیص ارقام دست‌نویس در ۶ بیت نشان داده شده است. این مدل برای تراشه‌های FPGA طراحی شده و برای انتشار رقم نقلی از واحد زنجیره رقم نقلی استفاده می‌کند.

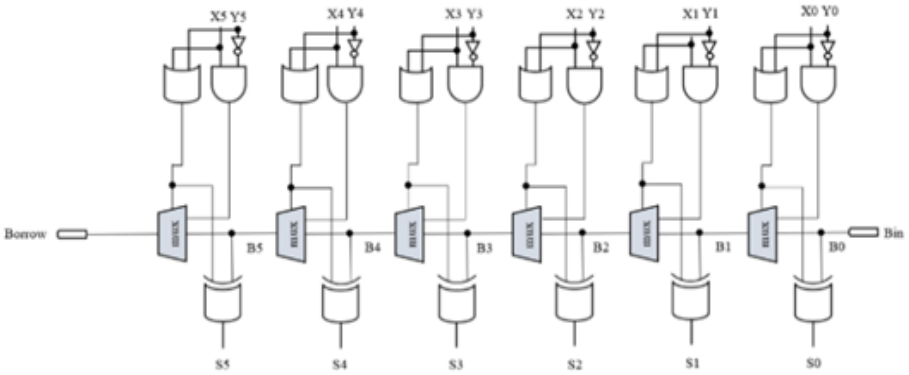
از ایده جمع‌کننده تقریبی در انجام عملیات جمع و تفریق در واحد حسابی جمع ممیز شناور استفاده شده است. همان طور که گفته شد، در جدول ۱ روابط استفاده‌شده برای هر بیت در جمع و تفریق دو ورودی در جمع‌کننده تقریبی آمده است. این جمع‌کننده در واحد جمع‌کننده ممیز شناور بعد از یکسان‌سازی نماها برای جمع و تفریق مانتیس‌ها استفاده می‌شود. در واقع در واحد جمع و تفریق ممیز شناور با هدف کاهش توان مصرفی به‌جای استفاده از عملگر + و −، از این واحدهای تقریبی استفاده شده است.

در جدول ۲، جدول درستی برای مقایسه نتایج بیتی جمع‌کننده تقریبی پیشنهادی و جمع‌کننده دقیق نشان داده شده است. نتایج نشان می‌دهند

شکل ۱۰: روابط جمع‌کننده تقریبی و دقیق.



شکل ۱۱: جمع‌کننده تقریبی ۶بیتی.



شکل ۱۲: تفریق‌کننده تقریبی ۶بیتی.

جدول ۲: مقایسه جدول درستی جمع‌کننده حالت تقریبی و حالت دقیق.

ورودی‌ها			دقیق				تقریبی			
x	*y*	*Cin*	*P*	*G*	*S*	*Cout*	*P*	*G*	*S*	*Cout*
۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰
۰	۰	۱	۰	۰	۱	۰	۰	۰	۱	۰
۰	۰	۰	۱	۰	۱	۰	۱	۰	۱	۰
۰	۱	۱	۱	۰	۰	۱	۱	۰	۰	۱
۱	۰	۰	۱	۰	۱	۰	۱	۰	۱	۰
۱	۰	۱	۱	۰	۰	۱	۱	۰	۰	۱
۱	۱	۰	۰	۱	۰	۱	۱	۱	۱	۰
۱	۱	۱	۰	۱	۱	۱	۱	۱	۰	۱

که جمع‌کننده تقریبی در مقایسه با جمع‌کننده دقیق، زمانی که دو بیت ورودی ۱ هستند، دارای خطاست.

٤- ارزیابی

٤-۱ نتایج شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس بدون تقریب

در این بخش، نتایج امتیاز کلاس‌های حاصل از فرایند آموزش و آزمودن شبکه عصبی از ۱۰ تصویر تصادفی (هر عکس از یک کلاس) در دو محیط متلب و ویوادو به دست آمده و با هم مقایسه شده‌اند. دیتاست تشخیص ارقام دست‌نویس شامل ۶۰۰۰۰ تصویر برای فرایند آموزش و ۱۰۰۰ تصویر برای فرایند آزمودن شبکه می‌باشد. همان طور که در بخش ۳ شکل ۷ نشان داده شده است، نتایج به‌دست‌آمده دقت شبکه را ۹۸/۰۴٪ نشان می‌دهد. همچنین نتایج نشان می‌دهند که امتیازهای هر کلاس در هر دو محیط مشابه هستند. جدول ۳ نتایج امتیاز کلاس‌ها را در محیط ویوادو نشان می‌دهد. در جدول ۳ هر ستون، امتیازات مربوط به یک کلاس را نشان می‌دهد و بزرگ‌ترین عدد هر ستون نشان‌دهنده کلاس خروجی است. به‌طور مثال در ستون چهارم، تصویر ورودی شبکه عصبی تشخیص ارقام دست‌نویس عدد ۴ می‌باشد. بزرگ‌ترین عدد ستون چهارم عدد ۱۶/۸۸ است که متناظر با سطر چهارم می‌باشد. در نتیجه می‌توان گفت خروجی متناظر با تصویر ورودی ستون چهارم، سطر چهارم (کلاس ۴) است که نشان‌دهنده کلاس صحیح می‌باشد.

برای انجام عملیات شبیه‌سازی و سنتز از برد Vertex-۷ و تراشه FPGA از همان خانواده استفاده شده است. جدول ۴ درصد منابع مصرفی از این تراشه را نمایش می‌دهد. جدول ۵ توان مصرفی شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس در حالت بدون تقریب را با فرکانس ۵۰ مگاهرتز توسط ابزار ویوادو نشان می‌دهد.

جدول ۳: نتایج شبیه‌سازی شبکه تشخیص ارقام دست‌نویس در محیط ویوادو.

شماره کلاس	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
۰	۱۶٫۷۶	−۲٫۸۷	۵٫۴۱	۲٫۵۴	−۹٫۲۲	−۳٫۲۳	۷٫۸۲	۱٫۰۹	۰٫۷۰۵	−۲٫۸۲
۱	−۷٫۵۴	۱۴٫۳۳	۷٫۷۹	−۴٫۰۸	۷٫۰۹	−۴٫۸۵	−۶٫۰۲	۰٫۵۴	−۷٫۳۷	−۳٫۹۱
۲	۴٫۶۰	۰٫۴۷	۱۷٫۲۲	۸٫۸۰۷	−۱٫۶۹	۰٫۲۷	۲٫۱۳	۷٫۲۱	−۰٫۳۱	−۳٫۳۰۹
۳	۰٫۰۹	−۲٫۶۲	۰٫۹۵	۱۵٫۸۲	−۸٫۰۰۴	−۲٫۷۹	۲٫۳۸	−۰٫۶۴	۴٫۴۱	−۰٫۹۸
۴	−۵٫۲۷	۵٫۶۸	−۴٫۰۱	−۹٫۰۳	۱۶٫۸۸	−۴٫۳۲	−۷٫۳۲	۰٫۰۷	−۸٫۴۳	۰٫۵۶
۵	۴٫۲۸	۱٫۰۳	−۲٫۱۸	۷٫۶۰۱	−۰٫۵۱	۱۱٫۷۳	۲٫۴۲	۲٫۱۷	۵٫۳۱	۰٫۰۳
۶	۱۰٫۰۴	۰٫۴۸	۶٫۷۶	−۲٫۳۴	−۲٫۹۵	۰٫۶۱	۱۸٫۳۱	−۱۰٫۹۲	−۰٫۹۹	−۶٫۶۸
۷	−۷٫۹۳	۵٫۴۳	−۴٫۰۴	−۲٫۶۵	۸٫۱۶	−۱٫۹۸	−۹٫۴۸	۲۱٫۱۹	−۶٫۳۶	۸٫۷۴
۸	۳٫۵۳	−۱٫۶۰	۰٫۰۷	۲٫۹۳	۶٫۷۶	۴٫۹۳	۵٫۵۴	−۰٫۸۰۹	۱۰٫۹۸	۵٫۳۱
۹	۱٫۲۰۵	۱٫۷۶	۱٫۲۷	۱٫۳۲	۱۱٫۷۰۲	۰٫۴۷	−۰٫۱۱	۱۲٫۲۴	۵٫۴۳	۱۴٫۸۲

جدول ۴: درصد منابع مصرفی شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس در حالت بدون تقریب.

منبع	منابع در دسترس	منابع مصرفی	درصد منابع مصرفی
واحد LUT	۳۴۶۴۰۰	۱۸۶۵۰۴	۵۳٫۸۴
واحد LUTRAM	۱۷۴۲۰۰	۶۴	۰٫۰۴
فلیپ‌فلاپ	۶۹۲۸۰۰	۷۸۰۸۱	۱۱٫۲۷
واحد DSP	۲۸۸۰	۱۳۵	۴٫۶۹

جدول ۵: درصد توان مصرفی شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس در حالت بدون تقریب.

توان مصرفی: ۱٫۶۴۱ W				
پویا			ایستا	
۱٫۳۰۶ W			۰٫۳۳۵ W	
سیگنال	logic	BRAM	DSP	I/O
۰٫۶۰۴	۰٫۵۸۰	۰٫۰۰۱	۰٫۰۵۱	۰٫۰۰۱

جدول ۶: نتایج حاصل از تقریب‌سازی ۶ بیت.

شماره کلاس	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
۰	۱۴٫۹۹	−۲٫۰۳	۶٫۲۴	۵٫۲۴	−۴٫۹۱	−۱٫۶۵	۸٫۴۹	۱٫۷۴	۰٫۴۳	۰٫۷۳
۱	−۲٫۵۳	۱۲٫۹۹	۷٫۷۴	−۰٫۸۱	۸٫۲۴	−۰٫۹۸	−۱٫۸۶	۳٫۸۷	−۲٫۱۷	۱۶٫۹۸
۲	۴٫۲۴	۰٫۴۶	۱۳٫۴۹	۷٫۲۴	−۱٫۶۱	−۰٫۴۲	۳٫۳۵	۸٫۴۹	−۰٫۲۸	۷٫۷۴
۳	۱٫۷۴	−۰٫۰۵	۱٫۵۶	۱۵٫۴۹	−۵٫۴۶	−۰٫۰۶	۴٫۱۲	−۰٫۱۰۸	۴٫۲۴	−۱٫۷۱
۴	−۱٫۰۲	۴٫۹۹	−۱٫۵۴	−۱٫۳۹	۱۴٫۹۹	۰٫۲۴	−۳٫۹۰۲	۳٫۳۷	−۳٫۳۵	۹٫۹۹
۵	۳٫۸۷	۱٫۸۱	−۰٫۵۳	۸٫۹۹	۰٫۰۰۹	۹٫۹۹	۲٫۹۹	۱٫۹۹	۵٫۴۲	۲۹٫۹۸
۶	۳٫۷۸	۰٫۳۴	۷٫۴۹	۰٫۰۰۵	−۰٫۸۴	۱٫۰۶	۱۴٫۹۹	−۵٫۶۶	۰٫۶۴	−۲٫۳۷
۷	−۶٫۰۷	۵٫۳۵	−۱٫۸۲	−۰٫۵۸	۶٫۸۷	−۱٫۰۵	−۵٫۷	۱۹٫۹۸	−۴٫۳۲	۷٫۷۴
۸	۴٫۲۴	۰٫۳۳	۷٫۱۲	۵٫۹۹	۸٫۹۹	۸٫۹۹	۵٫۴۹	۰٫۸۸	۱۰٫۴۶	−۰٫۴۱
۹	۳٫۶۲	۳٫۱۷	۲٫۶۲	۲٫۰۶	۱۲٫۴۹	−۳٫۲۴	۱٫۱۲	۱۳٫۹۹	۶٫۲۳	۳۳٫۹۷

جدول ۷: درصد منابع مصرفی کل شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس در حالت تقریبی.

منبع	منابع در دسترس	منابع مصرفی	درصد منابع مصرفی
واحد LUT	۳۴۶۴۰۰	۱۷۴۳۴	۵۰٫۳۳
واحد LUTRAM	۱۷۴۲۰۰	۶۴	۰٫۰۴
فلیپ‌فلاپ	۶۹۲۸۰۰	۷۸۰۶۸	۱۱٫۲۸
واحد DSP	۲۸۸۰	۱۳۵	۴٫۶۹

۴-۲ نتایج شبکه عصبی تشخیص ارقام دست‌نویس تقریبی

همان طور که در بخش سوم ذکر شد، برای پیاده‌سازی عملیات حسابی مانند جمع و ضرب، یک واحد جمع‌کننده و ضرب‌کننده ممیز شناور پیاده‌سازی شده است. در این روش برای تقریب‌سازی شبکه عصبی از جایگذاری جمع‌کننده تقریبی به‌جای جمع‌کننده دقیق استفاده شده است. همان طور که در بخش ۳ شرح داده شد، پیش‌بینی می‌شود با تقریبی‌کردن نیمی از ۱۱ بیت مانتیس جمع‌کننده (۶ بیت وزن پایین)، حداقل ۵۰٪ از اعداد دچار خطا نشوند. در نتیجه برای تقریب‌سازی شبکه تشخیص ارقام دست‌نویس و دستیابی به دقت لازم و تشخیص کلاس صحیح، تقریب‌سازی ۶ بیت از ۱۱ بیت مانتیس آزموده شده است.

جدول ۶ نتایج حاصل از تقریب‌سازی ۶ بیت بر روی کلاس‌های ۰ تا ۹ را نشان می‌دهد و هر ستون نشان‌دهنده امتیازهای مربوط به آن کلاس است.

نتایج نشان می‌دهد که با تقریب‌سازی ۶ بیت از ۱۱ بیت، دقت شبکه تشخیص ارقام دست‌نویس حفظ می‌شود. در جدول ۷ درصد منابع مصرفی شبکه عصبی تشخیص ارقام دست‌نویس نشان داده شده است. همچنین در جدول ۸ مقایسه‌ای از توان مصرفی کل شبکه عصبی تشخیص ارقام دست‌نویس در حالت تقریبی و حالت دقیق (بدون در نظر گرفتن بیت‌های تقریبی) نشان داده شده است.

مقایسه نتایج دو حالت بدون تقریب و با تقریب نشان می‌دهد که با تقریبی‌سازی ۶ بیت در شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس به دلیل کاهش تعداد دروازه‌ها، توان مصرفی پویا در بخش منطق و سیگنال‌ها بهبود چشمگیری دارد و نهایتاً توان مصرفی کل ۴۳٫۷۵٪ کاهش می‌یابد. حال برای بهبود بیشتر توان مصرفی،

جدول ۸: مقایسه توان مصرفی کل شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس حالت بدون تقریب و حالت تقریبی.

توان مصرفی شبکه عصبی کانولوشنی تشخیص ارقام دست‌نویس										
تقریبی					دقیق					
۰٫۹۲۳					۱٫۶۴۱					
پویا			ایستا		پویا			ایستا		
۰٫۵۹۴			۰٫۳۲۹		۱٫۳۰۶			۰٫۳۳۵		
سیگنال	logic	BRAM	DSP	I/O	سیگنال	logic	BRAM	DSP	I/O	
۰٫۲۴۷	۰٫۲۴۸	۰٫۰۰۱	۰٫۰۲۷	۰٫۰۰۱	۰٫۶۰۴	۰٫۵۸۰	۰٫۰۰۱	۰٫۰۵۱	۰٫۰۰۱	

جدول ۹: نتایج حاصل از تقریب‌سازی ۷ بیت.

شماره کلاس	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
۰	۱۱٫۹۹	-۰٫۹۰۹	۵٫۹۹	۴٫۴۹	-۲٫۶۵	-۰٫۳۴	۷٫۹۹	۱٫۱۲	۰٫۸۱	۰٫۵۳
۱	-۰٫۶۷	۱۰٫۹۹	۷٫۹۹	۰٫۶۸	۶٫۹۹	۰٫۳۲	-۰٫۳۴	۵٫۴۹	-۰٫۵۳	۸٫۹۹
۲	۳٫۲۴	۰٫۹۹	۱۲٫۹۹	۶٫۹۹	-۰٫۳۸	-۰٫۹۹	-۰٫۸۷	۷٫۹۹	۱٫۰۶	۳٫۷۴
۳	۱٫۹۹	۰٫۶۸	۳٫۹۹	۱۹٫۹۸	-۲٫۹۶	۱٫۸۷	۵٫۹۹	-۰٫۲۳	۶٫۹۹	۰٫۶۲
۴	۰٫۶۲	۴٫۴۹	۰٫۴۰۵	۰٫۷۴	۱۲٫۹۹	۲٫۳۷	-۰٫۳۶	۴٫۷۴	-۰٫۵۲	۸٫۴۹
۵	۵٫۴۹	۲٫۴۹	-۰٫۰۰۷	۷٫۴۸	۱٫۱۲	۱۰٫۹۹	۳٫۷۴	۲٫۹۹	۴٫۴۹	۱۷٫۹۸
۶	۷٫۴۹	۰٫۲۸	۵٫۴۹	-۰٫۰۰۲	-۰٫۶۸	۲٫۴۹	۱۱٫۹۹	-۳٫۲۹	-۰٫۴۲	-۱٫۴۳
۷	-۲٫۳۷	۴٫۹۹	-۰٫۰۰۰۰۶	۴٫۴۹	۹٫۹۹	-۰٫۲۴	-۴٫۱۲	۱۷٫۹۸	-۲٫۷۴	۳٫۴۹
۸	۵٫۹۹	۱٫۰۹	۴٫۴۹	۵٫۴۹۶	۸٫۹۹	۸٫۴۹	۵٫۹۹	-۰٫۷۴	۱۰٫۴۹	۱۷٫۹۸
۹	۳٫۷۴	۴٫۲۴	۲٫۹۹	۳٫۷۴۸	۱۵٫۹۹	۴٫۴۹	۲٫۹۹	۱۱٫۹۹	۸٫۹۹	۱۷٫۹۸

جدول ۱۰: نتایج تقریب‌سازی جمع‌کننده روی بیت‌های مختلف.

کلاس	دقیق	۰۰۰۰۰۰۰۰۰۰	۰۰۰۰۰۰۰۰۰۱	۰۰۰۰۰۰۰۰۱۰	۰۰۰۰۰۰۰۰۱۱	۰۰۰۰۰۰۰۱۰۰	۰۰۰۰۰۰۰۱۰۱
۰	۱٫۰۳	۱٫۰۳	۱٫۰۶	۱٫۰۵	۱٫۰۸	۱٫۰۸	۱٫۰۴
۱	۰٫۴۴	۰٫۴۴	۰٫۵۶	۰٫۵۹	۰٫۶۷	۰٫۶۳	۰٫۶۹
۲	۷٫۱۱	۷٫۱۱	۷٫۱۳	۷٫۰۹	۷٫۰۱	۶٫۹۴	۶٫۹۴
۳	-۰٫۶۹	-۰٫۷۱	-۰٫۶۹	-۰٫۶۸	-۰٫۷۳	-۰٫۷۰۱	-۰٫۸۳۴
۴	۰٫۰۸	۰٫۰۸	۰٫۱۹	۰٫۲۸	۰٫۳۸	۰٫۴۱	۰٫۶۱
۵	۲٫۱۳	۲٫۱۳	۲٫۱۳	۲٫۱۱	۲٫۱۵	۲٫۱۱	۲٫۱۶
۶	-۱۰٫۸۹	-۱۰٫۸۹	-۱۰٫۶۸	-۱۰٫۵۲	-۱۰٫۱۹	-۱۰٫۳۷	-۱۰٫۲۳
۷	۲۱٫۱۱	۲۱٫۱۱	۲۱٫۱۳	۲۱٫۱۱	۲۰٫۷۳	۲۱٫۳۶	۲۰٫۹۸
۸	-۰٫۷	-۰٫۷	-۰٫۶۶	-۰٫۵۶	-۰٫۵۰۹	-۰٫۶۱۲	-۰٫۵۰۳
۹	۱۲٫۳۶	۱۲٫۳۶	۱۲٫۳۷	۱۲٫۴۶	۱۲٫۳۳	۱۲٫۵۱	۱۲٫۵۷

تقریب‌سازی ۷ بیت از ۱۰ بیت مانتیس آزموده شده است. نتایج حاصل از جدول ۹ نشان می‌دهد تقریب‌سازی ۷ بیت در برخی از کلاس‌ها دچار تشخیص اشتباه می‌شود. این مشکل ناشی از آن است که بیش از نیمی از بیت‌های مانتیس تقریبی شده و در زمان انجام عملیات جمع، خطا در واحد کانولوشن انتشار می‌یابد. سپس در خروجی کانولوشن برخی از اعداد مثبت را تبدیل به منفی و برخی از اعداد منفی را مثبت می‌کند.

در نهایت انتشار این خطا باعث تشخیص کلاس اشتباه می‌شود و نتایج نشان می‌دهد با افزایش تعداد بیت‌های تقریبی‌سازی‌شده از ۶ بیت به ۷ بیت، با ایجاد ۲۰٪ خطا توان مصرفی ۴۴٫۹۱٪ کاهش می‌یابد.

جداول ۱۰ و ۱۱ نتایج خروجی شبکه کانولوشنی حاصل از تقریبی‌سازی بیت‌های مختلف را بر روی ورودی تصویر رقم دست‌نویس ۷ نشان می‌دهد. همان طور که پیش‌بینی می‌شد با توجه به حفظ دقت شبکه در تقریبی‌سازی ۶ بیت، در تقریب‌سازی با سایر بیت‌های کمتر از ۶ کلاس درست شناسایی شده است.

به‌منظور بررسی عملکرد روش پیشنهادی، نتایج به‌دست‌آمده در این مقاله با نتایج [۱۹] و [۲۲] مقایسه شده است. هدف از این مقایسه، ارزیابی میزان بهبود دقت و کاهش توان مصرفی در شرایط مشابه می‌باشد. همان طور که در جدول ۱۲ نشان داده شده است، روش پیشنهادی با ۴۳٫۷۵٪ بهبود در توان مصرفی و ۵۰٫۱۱٪ بهبود در تأخیر نسبت به روش ارائه‌شده در [۱۹] که دارای ۳۲٫۷۷٪ بهبود توان مصرفی و ۲۱٫۱۵٪ بهبود در تأخیر است و همچنین روش [۲۲] که ۱۰٫۷۹٪ تا ۱۲٫۱۷٪ بهبود توان مصرفی دارد، عملکرد بهتری را ارائه می‌دهد. این نتایج نشان‌دهنده اثربخشی قابل توجه روش پیشنهادی در بهینه‌سازی مصرف انرژی است. همچنین لازم به ذکر است روش پیشنهادی در مقایسه با دو روش ذکرشده، هیچ گونه کاهش دقتی نداشته که نشان‌دهنده حفظ کیفیت نتایج به‌دست‌آمده است.

۵- نتیجه‌گیری

یکی از پرکاربردترین مباحث در زمینه یادگیری ماشین، شبکه‌های عصبی هستند. در این شبکه‌ها برای رسیدن به دقت بالا در خروجی، تعداد پارامترهای شبکه (وزن‌ها) افزایش می‌یابند و باعث افزایش حجم پردازش داده‌های شبکه می‌شوند؛ پس نیاز به تسریع پردازش داده‌های شبکه است.

جدول ۱۱: نتایج تقریب‌سازی جمع‌کننده روی بیت‌های مختلف.

کلاس	دقیق	۰۰۰۰۰۰۰۰۱۱۰	۰۰۰۰۰۰۰۰۱۱۱	۰۰۰۰۰۰۰۱۰۰۰	۰۰۰۰۰۰۱۱۱۱	۰۰۰۰۰۱۱۱۱۱۱	۰۰۰۰۱۱۱۱۱۱۱	۰۰۰۱۱۱۱۱۱۱۱
۰	۱٫۰۳	۱٫۱۰	۱٫۱۴	۱٫۰۱	۰٫۹۷	۰٫۷۹	۱٫۷۴	۱٫۱۲
۱	۰٫۴۴	۰٫۸۳	۰٫۸۲	۰٫۸۱	۱٫۶۴	۲٫۳۷	۳٫۸۷	۵٫۴۹
۲	۷٫۱۱	۷٫۱۲	۷٫۰۲	۶٫۹۳	۶٫۹۹	۷٫۱۲	۸٫۴۹	۷٫۹۹
۳	−۰٫۷۱	−۰٫۷۳	−۰٫۶۰۱	−۰٫۷۴	−۰٫۵۵	−۰٫۷۱	−۰٫۱۰۸	−۰٫۲۳
۴	۰٫۰۸	۰٫۶۶	۰٫۷۹	۰٫۷۱	۱٫۳۷	۲٫۱۸	۳٫۳۷	۴٫۷۴
۵	۲٫۱۳	۲٫۱۶	۲٫۱۹	۲٫۱۳	۲٫۲۰۹	۲٫۱۴	۱٫۹۹	۲٫۹۹
۶	−۱۰٫۸۹	−۱۰٫۰۶	−۹٫۶۸	−۱۰٫۱۳	−۸٫۹۳	−۷٫۶۶	−۵٫۶۶	−۳٫۲۹
۷	۲۱٫۱	۲۱٫۱۱	۲۰٫۸۶	۲۰٫۸۶	۲۰٫۹۸	۲۰٫۹۸	۱۹٫۹۸	۱۷٫۹۸
۸	−۰٫۶۹	−۰٫۴۷	−۰٫۲۴	−۰٫۳۸	−۰٫۰۹	۰٫۵۰۳	۰٫۸۸	−۰٫۷۴
۹	۱۲٫۳۶	۱۲٫۵۸	۱۲٫۴۳	۱۲٫۶۲	۱۲٫۴۹	۱۳٫۲۴	۱۳٫۹۹	۱۱٫۹۹

جدول ۱۲: بررسی عملکرد روش پیشنهادی.

پژوهش	دیتاست	دقت	بهبود تأخیر	بهبود توان مصرفی
Zhisheng, et al. [۱۹]	MNIST	٪۹۸٫۱۹	٪۲۱٫۱۵	٪۳۲٫۷۷
Piyasena, et al. [۲۲]	MNIST	٪۹۹٫۹	–	٪۱۰٫۷۹ تا ٪۱۲٫۱۷
روش پیشنهادی	MNIST	٪۱۰۰	٪۵۰٫۱۱	٪۴۳٫۷۵

یکی از روش‌های تسریع الگوریتم‌های شبکه‌های عصبی، استفاده از شتاب‌دهنده‌های سخت‌افزاری است. معمولاً از یک بستر همه‌منظوره مانند طراحی ASIC به دلیل انعطاف‌پذیری بسیار کم و پیچیدگی طراحی در کاربردهای خاص‌منظوره استفاده می‌شود. به همین جهت در این مقاله برای پیاده‌سازی شبکه عصبی کانولوشنی از بستر تراشه‌های FPGA (به نظر مصرف انرژی چالش‌هایی دارند. یکی از روش‌های کاهش توان مصرفی استفاده از مفهوم محاسبات تقریبی در تراشه‌های FPGA است. همچنین مطالعات نشان داده‌اند در شبکه‌های عصبی قابلیت تحمل‌پذیری خطا می‌تواند وجود داشته باشد که این امر زمینه را برای به‌کارگیری محاسبات تقریبی مناسب می‌سازد.

در این مقاله از شبکه عصبی کانولوشنی برای تشخیص ارقام دست‌نویس استفاده شده است. برای تسریع محاسبات، این شبکه تشخیص ارقام دست‌نویس بر روی یک تراشه FPGA پیاده‌سازی شده است. ایده تقریب‌سازی برای کاهش توان مصرفی در بخش عملیات جمع پیاده‌سازی شده است. با استفاده از روش پیشنهادی و تقریب‌سازی ۶ بیت از ۱۰ بیت جمع مانتیس اعداد ممیز شناور به علت ساده‌سازی مدار در سطح گیت با حفظ دقت لازم پیچیدگی محاسبات شبکه کاهش می‌یابد و همچنین توان مصرفی ٪۴۳٫۷۵ بهبود می‌یابد. با افزایش تعداد بیت تقریب‌سازی از ۶ بیت به ۷ بیت با ایجاد ٪۲۰ خطا، توان مصرفی ٪۴۴٫۹۱ کاهش می‌یابد که این میزان خطا با توجه به توان مصرفی مناسب نیست و تقریب‌سازی روی ۶ بیت مناسب‌تر است.

۶- کارهای آینده

ضرب‌کننده‌ها در واحدهای محاسباتی شبکه‌های عصبی، کاربرد و مصرف انرژی بالایی دارند؛ به همین دلیل به نظر می‌رسد که با پژوهش بر روی تقریب‌سازی ساختار ضرب‌کننده‌های شبکه عصبی کانولوشنی می‌توان توان مصرفی را باز هم بیشتر بهبود داد. همچنین می‌توان استفاده از مدارهای محاسباتی تقریبی را برای سایر انواع شبکه‌های یادگیری عمیق با هدف بهبود مصرف انرژی، مساحت یا کارایی بررسی کرد.

مراجع

[1] Y. Qian, *et al.*, "Approximate logic synthesis in the loop for designing low-power neural network accelerator," in *Proc. IEEE Int. Symp. on Circuits and Systems*, 5 pp., Daegu, Korea, 22-28 May 2021.

[2] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy efficient neural computing," *IEEE Trans. on Computers*, vol. 70, no. 4, pp. 614-625, Apr. 2020.

[3] www.altera.com

[4] M. Hamdan, "VHDL auto-generation tool for optimized hardware acceleration of convolutional neural networks on FPGA (VGT)," A thesis submitted to the graduate faculty, Iowa State University, 2018.

[5] C. L. Giles and C. W. Omlin, "Pruning recurrent neural networks for improved generalization performance," *IEEE Trans. on Neural Networks*, vol. 5, no. 5, pp. 848-851, Sept. 1994.

[6] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," *IEEE Trans. on Computers*, vol. 70, no. 4, pp. 614-625, Apr. 2021.

[7] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-Vdd," in *Proc. of the 41st Annual Design Automation Conf.*, pp. 735-740, San Diego, CA, USA, 7-11 Jun. 2004.

[8] K. Yin Kyaw, W. Ling Goh, and K. Seng Yeo, "Low-power high-speed multiplier for error-tolerant application," in *Proc. IEEE Int. Conf. of Electron Devices and Solid-State Circuits*, 4 pp., Hong Kong, China, 15-17 Dec. 2010.

[9] S. S. P. Goswami, B. Paul, S. Dutt, and G. Trivedi, "Comparative review of approximate multipliers," in *Proc. 30th Int. Conf. Radioelektronika*, 6 pp., Bratislava, Slovakia, 15-16 Apr. 2020.

[10] M. Vasudevan and C. Chakrabarti, "In image processing using approximate datapath units," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 1544-1547, Melbourne, Australia, 1-5 Jun. 2014.

[11] S. Ullah, *et al.*, "Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators," in *Proc. 55th ACM/ESDA/IEEE Design Automation Conf.*, 6 pp., San Francisco, CA, USA 24-28 Jun. 2018.

[12] S. Ullah, S. Rehman, M. Shafique, and A. Kumar, "High-performance accurate and approximate multipliers for FPGA-based hardware accelerators," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 2, pp. 211-224, Feb. 2021.

[13] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "Automated high-level synthesis of low power/area approximate computing circuits," *First Workshop on Approximate Computing Across the System Stack,* 6 pp., Salt Lake City, UT, USA, 2-2 Mar. 2014.

[14] Y. Kim, Y. Zhang, and P. Li, "An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 130-137, San Jose, CA, USA, 18-21 Nov. 2013.

[15] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: modeling and analysis of circuits for approximate computing," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 667-673, 7-10 Nov. 2011.

[16] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*, Cambridge University Press, vol. 1, pp. 14-15, 2011.

[17] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *Proc. Intl. Conf. Architectural Support for Programming Languages and Operating Systems*, pp. 301-312, London, UK, 3-7 Mar. 2012.

[18] K. Lengwehasatit and A. Ortega, "Scalable variable complexity approximate forward DCT," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 14, no. 11, pp. 1236-1248, Nov. 2004.
[19] Z. Li, et al., "Laius: an 8-bit fixed-point CNN hardware inference engine," in Proc. IEEE Int. Symp. on Parallel and Distributed Processing with Applications and IEEE Int. Conf. on Ubiquitous Computing and Communications, pp. 143-150, Guangzhou, China, 12-15 Dec. 2017.
[20] T. Yang, T. Sato, and T. Ukezono, "An accuracy-configurable adder for low-power applications," *IEICE Trans. on Electronics*, vol. E103-C, no. 3, pp. 68-76, 2020.
[21] M. Sano, *et al.*, "An accuracy-controllable approximate adder for FPGAs," in *Proc. 4th Int. Symp. on Advanced Technologies and Applications in the Internet of Things*, pp. 60-66, Ibaraki, Japan 24-26 Aug. 2022.
[22] D. Piyasena, R. Wickramasinghe, D. Paul, S. Lam, and M. Wu, "Reducing dynamic power in streaming CNN hardware accelerators by exploiting computational redundancies," in *Proc. 29th Int. Conf. on Field Programmable Logic and Applications,* pp. 354-359, Barcelona, Spain, 8-12 Sept. 2019.

نادیا سهرابی تحصیلات خود را در مقطع کارشناسی مهندسی کامپیوتر در سال ۱۳۹۹ از دانشگاه صنعتی شاهرود به پایان رسانده و در سال ۱۴۰۲ موفق به اخذ مدرک کارشناسی ارشد مهندسی کامپیوتر با گرایش معماری کامپیوتر از دانشگاه صنعتی امیرکبیر گردیده است. وی در طول دوران تحصیل خود بر حوزه‌های مرتبط با طراحی دستگاه‌های دیجیتال و سخت‌افزارهای هوش مصنوعی تمرکز داشته است. زمینه‌های تحقیقاتی مورد علاقه وی شامل طراحی سخت‌افزارهای کم‌مصرف، شتاب‌دهنده‌های هوش مصنوعی، معماری دستگاه‌های یادگیری عمیق، و پیاده‌سازی الگوریتم‌های یادگیری ماشین بر بستر FPGA می‌باشد.

امیر باوفای طوسی با هدف شکستن مرز دانش‌های عملی و تجربی در مهندسی کامپیوتر به‌منظور حل مشکلات صنعتی کشور و پرورش دانشجویان با مهارت از سال ۱۳۷۹ همکاری خود را با دانشکده کامپیوتر دانشگاه سجاد آغاز نمود. وی از سال ۱۴۰۱ مدیر گروه دانشکده کامپیوتر دانشگاه سجاد و فارغ‌التحصیل دکتری مهندسی کامپیوتر از دانشگاه صنعتی امیرکبیر در سال ۱۳۹۹ می‌باشد. نام‌برده با تدریس دروس آزمون المپیاد کامپیوتر همچون معماری کامپیوتر، مدارهای منطقی و سیستم‌عامل با همکاری سایر اساتید سجاد دانشجویانی پرورش داده است که از سال ۱۳۹۰ تاکنون هرسال در آزمون نهایی المپیاد دانشجویی رتبه‌های تک‌رقمی و کمتر از ۵۰ کشور و در آزمون کارشناسی ارشد کامپیوتر از سال ۱۳۸۱ تاکنون رتبه‌های دورقمی و برتر کشور را کسب می‌کنند. ایشان مقام دوم پژوهش‌های توسعه‌ای در بیستمین جشنواره بین‌المللی خوارزمی و دارای USPatent می‌باشد. ایشان با عقد قرارداد صنعتی در سال ۱۳۹۰ آزمایشگاه پژوهشی ESRL را تأسیس نمود. زمینه‌های تحقیقاتی ایشان عبارات‌اند از: FPGA و سنتز دیجیتال، شتاب‌دهنده‌های هوش مصنوعی، کاربردهای پردازش تصویر، سیستم‌های هوشمند، پردازش‌های سریع و برنامه‌نویسی موازی، اینترنت اشیاء، سیستم‌های توزیع‌شده و محاسبات ابری می‌باشد.

مهدی صدیقی در سال ۱۳۶۹ مدرک کارشناسی مهندسی برق گرایش سخت‌افزار کامپیوتر خود را از دانشگاه صنعتی شریف، و در سال‌های ۱۳۷۳ و ۱۳۷۷ مدارک کارشناسی ارشد و دکترای خود را در رشته مهندسی برق و کامپیوتر از دانشگاه کلرادو در بولدر آمریکا دریافت نمود. در فاصله سال‌های ۱۳۷۵ تا ۱۳۸۰ ایشان به‌عنوان طراح ارشد دستگاه‌های دیجیتال در شرکت‌های مختلفی در دره سیلیکون به کار مشغول بود. از سال ۱۳۸۰ تاکنون دکتر صدیقی عضو هیأت‌علمی دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر می‌باشد. زمینه‌های علمی موردعلاقه ایشان شامل محاسبات کوانتومی، سنتز دستگاه‌های حسابی، و دستگاه‌های نهفته می‌باشد.