دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی

عنوان پروژه

پیاده‌سازی دو روش ارتباطی میان FPGAها و مقایسه نتایج آنها با یکدیگر با استفاده از شبکه‌های عصبی پیچشی (CNN)

نگارنده

امیر بهنام

استاد راهنما

دکتر حمیدرضا زرندی

شهریور ۱۴۰۳


بسم الله الرحمن الرحيم

صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تایید و تصویب پایان نامه موسوم به فرم کمیته دفاع- موجود در پرونده آموزشی- را قرار دهید.

نکات مهم:

- نگارش پایان نامه/رساله باید به زبان فارسی و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان‌نامه‌های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت پشت و رو(دورو) بلامانع است و انجام آن توصیه می‌شود.

به نام خدا

تعهدنامه اصالت اثر

تاریخ: ۱۴۰۳/۰۴/۱۵

اینجانب امیر بهنام متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

امیر بهنام

امضا

تقدیم به

آنکه جز به فضلش امیدی نیست...

سپاس‌گزاری

بدینوسیله بر خود فرض می‌دانم از زحمات استاد گرانقدر جناب آقای دکتر حمیدرضا زرندی صمیمانه تشکر کنم. بدون تردید انجام این تحقیق و تهیه این گزارش بدون حمایت‌های دلسوزانه و راهنمایی‌های ارزشمند ایشان ممکن نبود. از استاد گرانمایه، جناب آقای دکتر مرتضی صاحب‌الزمانی که زحمت داوری این پایان‌نامه را برعهده داشتند نهایت تشکر را دارم.

امیر بهنام

شهریور ۱۴۰۳

چکیده

در سال‌های اخیر، شبکه‌های عصبی پیچشی[1] به عنوان یکی از ابزارهای قدرتمند و مؤثر در حوزه‌های مختلف یادگیری عمیق[2] معرفی شده‌اند. این شبکه‌ها با بهره‌گیری از ساختارهای لایه‌ای پیچیده و انعطاف‌پذیر خود، توانسته‌اند در مسائلی مانند تشخیص تصویر[3]، تحلیل داده‌های پزشکی، پردازش زبان طبیعی[4]، و حتی تشخیص خودکار الگوها در داده‌های بزرگ، نتایج بسیار دقیق و مؤثری را ارائه دهند. توسعه روزافزون کاربردهای این شبکه‌ها، نیاز به پردازش‌های بسیار سریع و کارآمد را افزایش داده است. از این رو، استفاده از برد FPGA[5] به عنوان یک پلتفرم مناسب برای پیاده‌سازی آنها، به دلیل قابلیت‌های بالای موازی‌سازی و کارایی انرژی، به شدت مورد توجه قرار گرفته است. در این پروژه، دو روش ارتباطی میان FPGAها با استفاده از شبکه‌های عصبی پیچشی پیاده‌سازی و مورد بررسی قرار گرفته است. در پیاده‌سازی ارتباط، از دو صورت باسیم و بی‌سیم استفاده شده است و از پروتکل UART[6] بهره گرفته شده است. نتایج به دست آمده نشان می‌دهد که روش ارتباط سیمی به طور قابل توجهی عملکرد بهتری نسبت به روش بی‌سیم داشته است. این یافته‌ها بر اهمیت انتخاب روش ارتباطی مناسب در افزایش کارایی و سرعت شبکه‌های عصبی پیچشی تأکید دارد و می‌تواند راهگشای توسعه کاربردهای بهینه در حوزه‌های مختلف باشد.

واژه‌های کلیدی:

شبکه‌های عصبی پیچشی، یادگیری عمیق، روش‌های ارتباطی، FPGA، پروتکل UART

[1] Convolutional Neural Network (CNN)
[2] Deep Learning
[3] Image Recognition
[4] Natural Language Processing
[5] Field Programmable Gate Array
[6] Universal Asynchronous Receiver-Transmitter

فهرست مطالب

فهرست اشکال

فهرست جداول

فصل اول: مقدمه

در این بخش، مقدمه پایان‌نامه ارائه می‌گردد. ابتدا مسأله شرح داده می‌شود؛ سپس ضرورت، اهداف، و انگیزه پروژه ذکر می‌شود. در انتها، ساختار گزارش و سرفصل‌های آینده توضیح داده می‌شود.

۱-۱-شرح مسئله

در دنیای امروز، با توسعه روزافزون فناوری و نیاز به پردازش سریعتر و کارآمدتر داده‌ها، استفاده از مدارهای مجتمع دیجیتال مانند FPGA بسیار گسترش یافته است. FPGAها به دلیل انعطاف‌پذیری بالا و قابلیت پیکربندی مجدد در بسیاری از برنامه‌های کاربردی مانند پردازش تصویر[۱]، یادگیری ماشین[۲]، شبکه‌های عصبی مصنوعی[۳] و سیستم‌های تعبیه‌شده[۴] استفاده می‌شوند. با این حال، برای استفاده بهینه از قدرت پردازش FPGAها، اتصال و ارتباط بین چندین FPGA ضروری است.

یکی از مسائل مهم در این زمینه نوع و نحوه اتصال دو FPGA است که می‌تواند تاثیر بسزایی در عملکرد سیستم داشته باشد. انواع مختلفی از اتصال را می‌توان بین FPGA استفاده کرد و انتخاب نوع اتصال مناسب می‌تواند مستقیماً بر عملکرد سیستم تأثیر بگذارد.

در این پایان‌نامه، به پیاده‌سازی دو نوع مختلف اتصال بین دو FPGA با استفاده از شبکه عصبی پیچشی می‌پردازیم. برای این منظور، دو نوع اتصال متفاوت را با استفاده از یک شبکه عصبی پیچشی بین FPGAها پیاده‌سازی کرده و عملکرد آنها را ارزیابی می‌کنیم. نتایج این مطالعه می‌تواند در انتخاب نوع اتصال مناسب برای کاربردهای مشابه مؤثر باشد و به بهبود کارایی سیستم‌های مبتنی بر FPGA کمک کند.

۱-۲-ضرورت و اهداف پروژه

با توجه به اهمیت بالای FPGAها در پیاده‌سازی شبکه‌های عصبی پیچشی و نقش حیاتی ارتباطات بین آنها در عملکرد نهایی سیستم، ضرورت این پروژه روشن می‌شود. این تحقیق با هدف بررسی اثرات دو نوع مختلف اتصال بر توانمندی‌های FPGAها و تحلیل عملکرد این اتصالات از زوایای مختلف انجام می‌شود. ارزیابی دقیق نحوه تأثیر این اتصالات بر سرعت پردازش و سایر جنبه‌های کلیدی عملکرد سیستم که برای کاربردهای بلادرنگ ضروری است، بخش مهمی از این پروژه را تشکیل می‌دهد. انتظار می‌رود نتایج این تحقیق به ارتقاء قابلیت‌های سیستم‌های مبتنی بر FPGA و تسهیل یکپارچه‌سازی آنها با فناوری‌های موجود کمک شایانی کند. همچنین، این یافته‌ها می‌تواند مسیر را برای بهبود طراحی‌های آینده هموار سازد، به‌ویژه در حوزه‌هایی که نیاز به پردازش

[1] Image Processing
[2] Machine Learning
[3] Artificial Neural Network
[4] Embedded Systems

سریع‌تر و کارآمدتر است. علاوه بر این، نتایج این پژوهش می‌تواند به شناخت بهتر از ویژگی‌ها و محدودیت‌های هر نوع اتصال منجر شود و در نهایت به توسعه سیستم‌های هوشمندتر و انعطاف‌پذیرتر برای کاربردهای پیچیده و پیشرفته منتهی گردد.

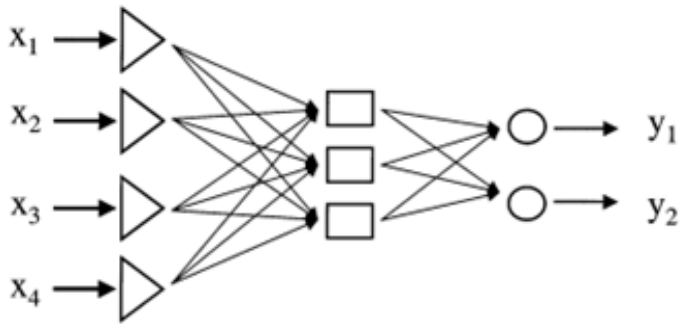
۱-۳- ساختار گزارش

در فصل ابتدایی این گزارش، مقدمه‌ای بر مسئله مورد نظر این پروژه ارائه شد. در فصل دوم، به مفاهیم اولیه نظری از جمله معماری شبکه عصبی پیچشی، پروتکل ارتباطیUART، و ابزارها و بردها پرداخته می‌شود. فصل سوم به پیاده‌سازی ساختار ماژول‌ها و نحوه تعامل و اتصال بین اجزای پروژه اختصاص دارد و توضیحات مربوط به آن ارائه می‌گردد. در فصل چهارم، نتایج پیاده‌سازی هر دو حالت به‌تفصیل بیان شده و مقایسه آنها ارائه می‌شود. در فصل پنجم، که فصل انتهایی است، نتایج حاصل از این پروژه تحلیل شده و با ارائه پیشنهادهایی برای آینده، پایان‌نامه جمع‌بندی می‌شود.

فصل دوم: مفاهیم اولیه

۲-۱- معرفی شبکه عصبی

شبکه عصبی یک مدل محاسباتی است که از روشی که شبکه‌های عصبی بیولوژیکی در مغز انسان اطلاعات را پردازش می‌کنند، الهام گرفته شده است. توسعه شبکه‌های عصبی به اوایل قرن بیستم باز می‌گردد. مدل‌های شبکه‌های عصبی مصنوعی بر اساس عملکرد نورون‌ها[۱] از شبکه‌های عصبی بیولوژیکی الهام گرفته شده‌اند. این شبکه‌ها از لایه‌های به هم پیوسته‌ای از گره‌ها یا نورون‌ها تشکیل شده‌اند که هر یک عملکرد ریاضی خاصی را انجام می‌دهند. ساختار اصلی یک شبکه عصبی شامل یک لایه ورودی[۲]، یک یا چند لایه پنهان[۳] و یک لایه خروجی[۴] است. هر نورون تعدادی سیگنال ورودی از نورون‌های دیگر دریافت می‌کند، این ورودی‌ها را در وزن‌ها ضرب می‌کند تا تعاملات سیناپسی را شبیه‌سازی کند، ورودی‌های وزنی را جمع کرده و آنها را با مقدار بایاس[۵] (معمولاً یک) ترکیب می‌کند. این نتیجه سپس به یک تابع فعال‌سازی غیرخطی[۶] وارد می‌شود که سیگنال خروجی نورون را تولید می‌کند. شبکه با تنظیم وزن این اتصالات از طریق فرآیندی به نام آموزش[۷]، که معمولاً با استفاده از روش پس‌انتشار[۸] انجام می‌شود، یاد می‌گیرد [۱]. در طول آموزش، شبکه در معرض یک مجموعه داده بزرگ قرار می‌گیرد و وزن‌ها به منظور به حداقل رساندن تفاوت بین خروجی‌های پیش‌بینی شده و واقعی تنظیم می‌شوند. در شکل ۲-۱ [۲] نمای کلی از شبکه عصبی مشاهده می‌شود.



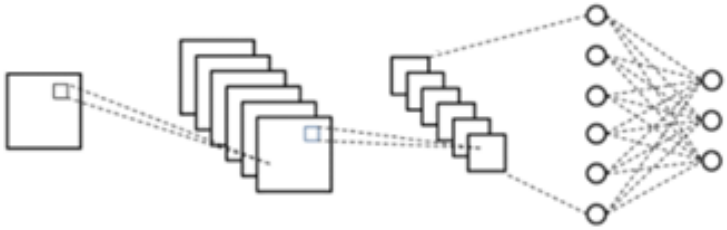
شکل ۲-۱ معماری شبکه عصبی [۲]

[1] Neurons
[2] Input Layer
[3] Hidden Layer
[4] Output Layer
[5] Bias
[6] Non-Linear Activation Function
[7] Learning
[8] Backpropagation

شبکه‌های عصبی ابزارهای قدرتمندی برای طیف گسترده‌ای از کاربردها، از جمله تشخیص تصویر و گفتار، پردازش زبان طبیعی و سیستم‌های مستقل هستند. توانایی آنها در یادگیری و تعمیم از مجموعه داده‌های بزرگ، آنها را به ویژه برای کارهای پیچیده‌ای که برنامه‌ریزی صریح آنها دشوار است، موثر می‌کند. یادگیری عمیق، زیرمجموعه‌ای از یادگیری ماشینی که شامل شبکه‌های عصبی با لایه‌های پنهان بسیار است، به پیشرفت‌های قابل توجهی در هوش مصنوعی منجر شده است. این شبکه‌های عصبی عمیق می‌توانند به‌طور خودکار ویژگی‌های سلسله‌مراتبی را از داده‌های خام استخراج کنند و امکان ایجاد مدل‌هایی را فراهم کنند که عملکرد پیشرفته‌ای در حوزه‌های مختلف به دست آورند [۳]. همانطور که تحقیقات در زمینه شبکه‌های عصبی به تکامل خود ادامه می‌دهد، انتظار می‌رود که این شبکه‌ها نقش حیاتی فزاینده‌ای در پیشرفت فناوری و حل مشکلات دنیای واقعی ایفا کنند.

۲-۲- شبکه عصبی پیچشی

شبکه‌های عصبی پیچشی نوعی شبکه عصبی مصنوعی تخصصی هستند که عمدتاً برای پردازش داده‌های شبکه‌ای ساختاریافته، مانند تصاویر، طراحی شده‌اند. این شبکه‌ها در وظایفی مانند طبقه‌بندی تصاویر، تشخیص اشیاء و تقسیم‌بندی بسیار مؤثر هستند. آنها از چندین لایه تشکیل شده‌اند که شامل لایه‌های پیچشی[1]، لایه‌های ادغام[2] و لایه‌های کاملاً متصل[3] می‌شوند. لایه‌های پیچشی فیلترهایی را به داده‌های ورودی اعمال می‌کنند تا ویژگی‌ها را استخراج کنند، در حالی که لایه‌های ادغام ابعاد داده‌ها را کاهش داده و شبکه را کارآمدتر و کمتر مستعد بیش‌برازش[4] می‌کنند. این ساختار سلسله‌مراتبی به شبکه اجازه می‌دهد تا الگوها را در سطوح مختلف انتزاع، از لبه‌های ساده تا اشیاء پیچیده، یاد بگیرد و تشخیص دهد [۳]. در شکل ۲-۲ [٤] ساختار شبکه عصبی پیچشی مشاهده می‌شود.



شکل ۲-۲ معماری شبکه عصبی پیچشی [۴]

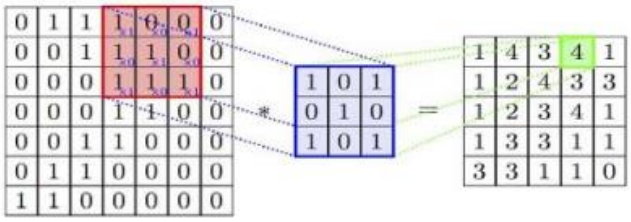
یکی از مزایای کلیدی شبکه‌های عصبی پیچشی، توانایی آنها در یادگیری خودکار سلسله‌مراتب فضایی ویژگی‌ها

[1] Convolutional Layers
[2] Pooling Layers
[3] Fully-Connected Layers
[4] Overfitting

از تصاویر ورودی است که آنها را برای وظایف بینایی کامپیوتری بسیار مؤثر می‌سازد. این شبکه‌ها از سیستم بینایی انسان الهام گرفته‌اند، جایی که نورون‌ها در قشر بینایی به مناطق خاصی از میدان دید پاسخ می‌دهند. این اتصال محلی در شبکه‌های عصبی پیچشی نیز منعکس شده است، به طوری که هر نورون به یک منطقه کوچک از داده‌های ورودی متصل است. علاوه بر این، این شبکه‌ها نسبت به ترجمه مقاوم هستند، به این معنا که می‌توانند اشیاء را بدون توجه به موقعیت آنها در تصویر تشخیص دهند. این ویژگی، همراه با توانایی آنها در پردازش حجم زیادی از داده‌ها، آنها را به ابزاری قدرتمند در کاربردهای مختلف، از جمله تحلیل تصاویر پزشکی، رانندگی خودکار و تشخیص چهره تبدیل کرده است.

۲-۲-۱- لایه‌ی پیچشی

لایه پیچشی در شبکه عصبی پیچشی برای یادگیری خودکار و تطبیقی سلسله‌مراتب فضایی ویژگی‌ها از تصاویر ورودی طراحی شده است. عملیات اصلی در یک لایه پیچشی، عملیات پیچش است، که در آن یک ماتریس کوچک به نام فیلتر[1] روی تصویر ورودی حرکت داده می‌شود و حاصل ضرب نقطه‌ای بین عناصر فیلتر بخش‌های متناظر در تصویر ورودی محاسبه می‌شود. این فرآیند منجر به تولید یک نقشه ویژگی[2] می‌شود که نشان‌دهنده حضور ویژگی‌های آموخته‌شده (مانند لبه‌ها، بافت‌ها یا الگوهای پیچیده‌تر) در نقاط مختلف تصویر ورودی است [۵]. در شکل ۲-۳ [۶] نمای کلی از محاسبات این لایه مشاهده می‌شود.



شکل ۲-۳ عملایت لایه پیچشی [۶]

لایه‌های پیچشی معمولاً از فیلترهای متعدد برای یادگیری ویژگی‌های مختلف استفاده می‌کنند و نقشه‌های ویژگی متعددی را تولید می‌کنند. سپس این نقشه‌ها از طریق توابع فعال‌سازی برای معرفی غیرخطی بودن عبور داده می‌شوند. گنجاندن بایاس به مدل کمک می‌کند تا الگوهای زیربنایی را بهتر تقریب بزند، با اجازه دادن به فعال‌سازی‌ها برای جابجایی، که در نتیجه، توانایی مدل برای یادگیری از داده‌ها را بهبود می‌بخشد. این ترکیب از عملیات پیچش، افزودن بایاس، و استفاده از تابع فعال‌سازی، مکانیسم اصلی را تشکیل می‌دهد که شبکه عصبی پیچشی از طریق آن ویژگی‌های سلسله‌مراتبی را از تصاویر شناسایی و یاد می‌گیرد، که این فرآیند برای

[1] Filter
[2] Feature Map

کارهایی مانند طبقه‌بندی تصویر، تشخیص اشیا، و تقسیم‌بندی حیاتی است [۵].

لایه پیچشی اولین لایه‌ای است که می‌تواند ویژگی‌ها را از تصاویر استخراج کند. عملیات پیچش به ما اجازه می‌دهد تا رابطه بین قسمت‌های مختلف یک تصویر را حفظ کنیم، زیرا پیکسل‌ها به پیکسل‌های مجاور و نزدیک وابسته هستند. به عنوان مثال، استفاده از فیلتری کوچک با گام[1] منظم بر روی یک تصویر منجر به تولید تصویری با ابعاد کوچکتر خواهد شد. این عملیات باعث کاهش اندازه تصویر می‌شود بدون اینکه روابط فضایی بین پیکسل‌ها از بین برود [۷].

۲-۲-۱-۱- گام

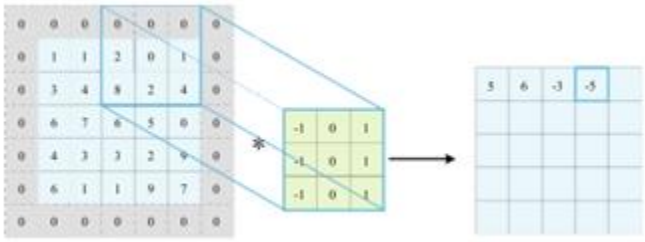
گام در شبکه‌های عصبی پیچشی به اندازه حرکت فیلتر پیچش در تصویر ورودی اشاره دارد. هنگامی که گام برابر با یک است، فیلتر هر بار یک پیکسل حرکت می‌کند و در نتیجه یک نقشه ویژگی با ابعاد مشابه با ورودی اصلی ایجاد می‌شود. گام‌های بزرگ‌تر، مانند دو یا بیشتر، باعث می‌شود که فیلتر چند پیکسل را به‌صورت یکجا طی کند و نقشه ویژگی خروجی کوچک‌تری به دست آید. گام میزان پایین نمونه‌سازی را کنترل می‌کند و بر وضوح و کارایی محاسباتی شبکه تأثیر می‌گذارد. با تنظیم گام، شبکه عصبی پیچشی می‌تواند بین حفظ جزئیات مکانی دقیق و کاهش بار محاسباتی و ابعاد داده‌ها تعادل برقرار کند [۳].

۲-۲-۱-۲- حاشیه‌گذاری

حاشیه‌گذاری در شبکه‌های عصبی پیچشی به تکنیک اضافه کردن پیکسل‌های اضافی (معمولاً صفر) در اطراف مرزهای یک تصویر ورودی قبل از اعمال عملیات پیچش اشاره دارد. هدف اصلی از حاشیه‌گذاری، حفظ ابعاد فضایی حجم ورودی هنگام عبور از لایه‌های پیچشی است [۳]. بدون حاشیه‌گذاری، ابعاد فضایی نقشه‌های ویژگی با هر لایه پیچشی کوچک می‌شود، زیرا هسته نمی‌تواند به طور کامل لبه‌ها و گوشه‌های تصویر ورودی را طی کند. با افزودن حاشیه‌گذاری، هسته می‌تواند روی کل تصویر ورودی بپیچد و اطمینان حاصل کند که نقشه ویژگی خروجی همان ابعاد فضایی ورودی را دارد یا حداقل کاهش اندازه ناشی از عملیات پیچیدگی را کاهش می‌دهد [۵]. انواع متداول حاشیه‌گذاری عبارتند از حاشیه‌گذاری یکسان[2]، که صفرهای کافی برای حفظ اندازه خروجی هنگام استفاده از گام یک را اضافه می‌کند، و حاشیه‌گذاری معتبر[3]، که هیچ حاشیه‌گذاری اعمال نمی‌کند و منجر به اندازه خروجی کوچکتر می‌شود. این عملیات برای حف اطلاعات مکانی، کاهش جلوه‌های لبه، و امکان یادگیری موثر ویژگی‌ها در سراسر تصویر بسیار مهم است. در شکل ۲-۴ [۸] نمای کلی از فرآیند

[1] Stride
[2] Same Padding
[3] Valid Padding

حاشیه‌گذاری مشاهده می‌شود.

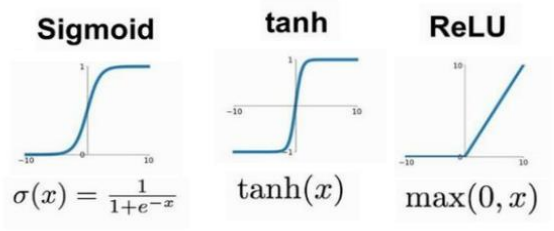


شکل ۲-۴ حاشیه‌گذاری [۸]

۲-۲-۱-۳- تابع فعال‌سازی

تابع فعال‌سازی در شبکه عصبی پیچشی، غیرخطی بودن را به مدل معرفی کرده و به آن اجازه می‌دهد تا الگوهای پیچیده را در داده‌ها یاد بگیرد و نمایش دهد. پس از هر عملیات پیچش، تابع فعال‌سازی بر روی نقشه ویژگی حاصل اعمال شده و سیگنال ورودی را تبدیل می‌کند. واحد خطی اصلاح‌شده[1] یکی از متداول‌ترین توابع فعال‌سازی است که تمام مقادیر منفی را صفر کرده و مقادیر مثبت را بدون تغییر می‌گذارد، در نتیجه آموزش را سریع‌تر و کارآمدتر می‌سازد. این به کاهش مسائلی مانند ناپدید شدن گرادیان‌ها[2] کمک می‌کند، که می‌تواند روند یادگیری در شبکه‌های عمیق را مختل کند. سایر توابع فعال‌سازی مانند سیگموئید[3] و هذلولی[4] زمینه‌های خاص، به‌ویژه برای لایه‌های خروجی که خروجی‌های محدود یا احتمالی مورد نیاز است، استفاده می‌شوند. به عنوان مثال، تابع سیگموئید مقادیر ورودی را در محدوده‌ای بین صفر و یک ترسیم کرده و آن را برای کارهای طبقه‌بندی باینری مناسب می‌کند. با معرفی غیرخطی بودن، توابع فعال‌سازی شبکه‌های عصبی پیچشی را قادر می‌سازند تا طیف وسیعی از توابع را تقریب زده و ساختارهای داده‌ای پیچیده را که برای کارهایی مانند طبقه‌بندی تصویر، تشخیص اشیا و تقسیم‌بندی ضروری هستند، ضبط کنند. این تبدیل غیرخطی به شبکه‌های عصبی پیچشی اجازه می‌دهد تا روابط و تعاملات پیچیده در داده‌ها را مدل‌سازی کرده و انعطاف‌پذیری و قدرت مورد نیاز برای برنامه‌های بینایی کامپیوتری مختلف را برای آن‌ها فراهم کند. در شکل ۲-۵ [۹] انواع توابع فعال‌سازی مشاهده می‌شود.

[1] Rectified Linear Unit (ReLU)
[2] Gradients
[3] Sigmoid
[4] Tanh

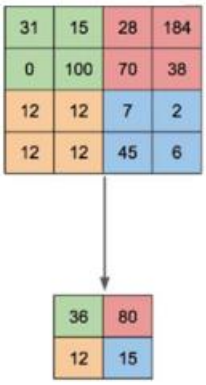


شکل ۲-۵ انواع توابع فعال‌سازی [۹]

۲-۲-۲- لایه‌ی ادغام

در شبکه‌های عصبی پیچشی، لایه‌های ادغام برای کاهش ابعاد فضایی (عرض و ارتفاع) نقشه‌های ویژگی ورودی و در عین حال حفظ عمق آنها ضروری هستند. این کاهش به کاهش بار محاسباتی، کاهش تعداد پارامترها و کنترل بیش از حد برازش کمک می‌کند [۳]. دو نوع متداول ادغام، ادغام میانگین[۱] و ادغام حداکثر[۲] است.

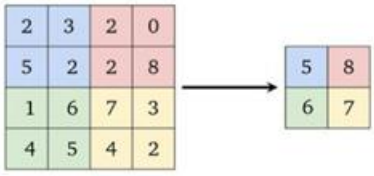
ادغام میانگین شامل گرفتن میانگین همه مقادیر در یک زیربخش مشخص از نقشه ویژگی ورودی است. به عنوان مثال، با استفاده از یک پنجره مشخص و یک گام معین، عملیات ادغام میانگین هر بلوک پیکسل را محاسبه می‌کند و یک پیکسل واحد در نقشه ویژگی خروجی ایجاد می‌کند که این میانگین را نشان می‌دهد. این فرآیند با خلاصه کردن اطلاعات در هر بخش، ورودی را صاف می‌کند و نویز را کاهش می‌دهد. این روش به ویژه در سناریوهایی که مکان دقیق ویژگی‌ها به اندازه حضور عمومی آنها در یک بخش مهم نیست، مفید است. با این حال، ادغام میانگین گاهی اوقات می‌تواند ویژگی‌های مهم را با ترکیب همه مقادیر با هم رقیق کند [۵]. در شکل ۲-۶ [۱۰] فرآیند ادغام میانگین مشاهده می‌شود.



شکل ۲-۶ ادغام میانگین [۱۰]

[1] Average Pooling
[2] Max Pooling

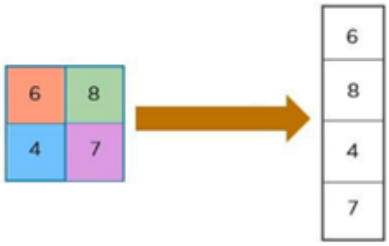
از سوی دیگر، ادغام حداکثر، بیشترین مقدار را از هر زیربخش انتخاب می‌کند. با استفاده از یک پنجره مشابه و گام معین، ادغام حداکثر بالاترین مقدار را از هر بلوک انتخاب می‌کند. این روش به‌طور مؤثر برجسته‌ترین ویژگی‌ها را در هر پنجره ثبت می‌کند و قوی‌ترین فعال‌سازی‌ها را حفظ می‌کند و در عین حال اطلاعات کمتر مهم را حذف می‌کند. این ویژگی می‌تواند برای برجسته کردن ویژگی‌های مهم و حصول اطمینان از حفظ حیاتی‌ترین جنبه‌های داده‌های ورودی توسط شبکه مفید باشد. ادغام حداکثر در عمل به خوبی کار می‌کند، زیرا برجسته‌ترین ویژگی‌ها را حفظ می‌کند، که اغلب برای کارهایی مانند تشخیص و طبقه‌بندی شیء مرتبط هستند [۵]. در شکل ۲-۷ [۱۱] فرآیند ادغام حداکثر مشاهده می‌شود.



شکل ۲-۷ ادغام حداکثر [۱۱]

۲-۲-۳- عملیات تسطیح

عملیات تسطیح به تبدیل یک ماتریس دو بعدی از مقادیر پیکسل به یک بردار یک‌بعدی اشاره دارد. این مرحله بعد از لایه ادغام انجام می‌شود تا داده‌ها برای لایه‌های کاملاً متصل که وظایف طبقه‌بندی یا رگرسیون[1] را انجام می‌دهند، آماده شود. این عملیات خروجی چندبعدی را از لایه‌های پیچشی و ادغام می‌گیرد و آن را به یک بردار طولانی تبدیل می‌کند. این فرآیند ساختار داده را ساده می‌کند تا بتوان آن را به لایه‌های کاملاً متصل که انتظار ورودی یک‌بعدی را دارند، وارد کرد [۵]. در شکل ۲-۸ [۶] فرآیند عملیات تسطیح مشاهده می‌شود.



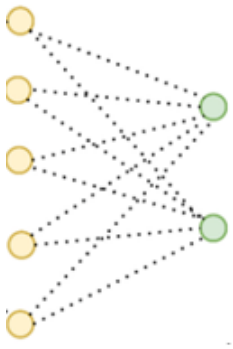
شکل ۲-۸ عملیات تسطیح [۶]

[1] Regression

۲-۲-۴- لایه‌ی کاملاً متصل

لایه کاملاً متصل یکی از اجزای اساسی در شبکه‌های عصبی پیچشی است که معمولاً در مراحل پایانی شبکه به کار می‌رود. این لایه با هدف ادغام و ترکیب ویژگی‌های استخراج‌شده توسط لایه‌های پیچشی و ادغام طراحی شده است تا تصمیم نهایی یا پیش‌بینی را انجام دهد. در این لایه، هر نورون به تمام نورون‌های لایه قبلی متصل است، که این اتصال کامل به مدل اجازه می‌دهد تا تمامی ویژگی‌های آموخته‌شده در لایه‌های قبلی را در نظر بگیرد [۳].

عملیات اصلی در لایه کاملاً متصل ضرب ماتریسی بین بردار ورودی (که می‌تواند به‌عنوان نگاشتی از ویژگی‌های خروجی لایه‌های قبلی در نظر گرفته شود) و وزن‌های مربوط به این لایه است. پس از این ضرب، بایاس به هر نورون اضافه می‌شود، و نتیجه از طریق یک تابع فعال‌سازی (مانند تابع واحد خطی اصلاح‌شده) عبور داده می‌شود [۵]. خروجی نهایی این لایه، یک بردار است که مقادیر آن نشان‌دهنده احتمالات کلاس‌های مختلف (در مسائل طبقه‌بندی) یا مقادیر پیش‌بینی‌شده (در مسائل رگرسیون) است. در شکل ۲-۹ [۱۲] فرآیند عملیات لایه‌ی کاملاً متصل مشاهده می‌شود.

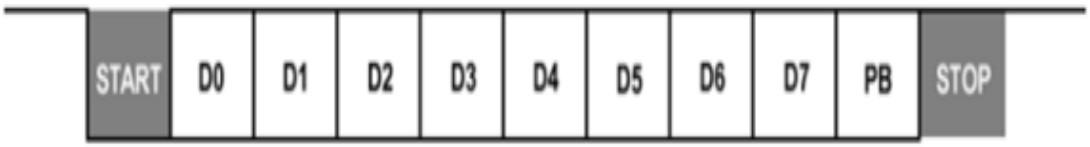


شکل ۲-۹ عملیات لایه کاملاً متصل [۱۲]

لایه‌های کاملاً متصل به دلیل تعداد زیاد پارامترها و اتصالات، اغلب به بخش قابل توجهی از پیچیدگی محاسباتی و ظرفیت یادگیری شبکه منجر می‌شوند. این لایه‌ها نقش مهمی در نقشه‌برداری ویژگی‌های سطح پایین و میانی استخراج‌شده توسط لایه‌های پیچشی به فضای خروجی با ابعاد بالاتر ایفا می‌کنند، و بنابراین برای کارهایی مانند طبقه‌بندی تصویر و تشخیص اشیا حیاتی هستند. با این حال، به دلیل تعداد زیاد پارامترها، این لایه‌ها نیاز به تنظیم دقیق دارند تا از بیش‌برازش مدل جلوگیری شود.

۲-۳- پروتکل UART

UART یک پروتکل ارتباطی سخت‌افزاری است که برای ارتباط سریال ناهمزمان بین دستگاه‌ها استفاده می‌شود. برخلاف ارتباطات همزمان، نیازی به سیگنال ساعت مشترک بین دستگاه‌های فرستنده و گیرنده ندارد. در عوض، برای اطمینان از انتقال دقیق داده‌ها، بر انتقال بسته‌های داده که شامل بیت‌های شروع[۱]، داده، بیت‌های توازن[۲]، و بیت‌های توقف[۳] است، متکی است. بیت شروع، آغاز یک بسته داده را نشان می‌دهد، سپس بیت‌های داده (معمولاً ۷، ۸ یا ۹ بیت)، یک بیت توازن اختیاری برای بررسی خطا، و یک یا چند بیت توقف برای نشان دادن پایان بسته ارسال می‌شود [۱۳]. این ساختار به طور مؤثر انتقال داده‌ها را حتی در غیاب ساعت همگام مدیریت می‌کند و آن را برای کاربردهای مختلف از جمله میکروکنترلرها، ماژول‌های بلوتوث[۴] و پورت‌های سریال مناسب می‌سازد. در شکل ۲-۱۰ [۱۳] پیکربندی ساختار داده‌ی UART مشاهده می‌شود.

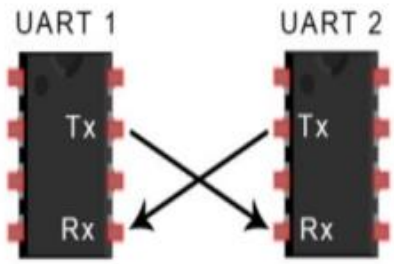


شکل ۲-۱۰ ساختار داده‌ی ارسالی UART [۱۳]

فرآیند ارتباط با ارسال بیت شروع توسط فرستنده[۵] (TX) برای هشدار به گیرنده[۶] (RX) درباره بسته داده ورودی آغاز می‌شود. بیت شروع، خط انتقال (که معمولاً بالا نگه داشته می‌شود) را برای مدت یک بیت پایین می‌کشد و به دستگاه RX نشان می‌دهد که یک فریم داده جدید شروع می‌شود. سپس بیت‌های داده به صورت متوالی ارسال می‌شوند و ابتدا کمترین بیت مهم ارسال می‌شود. یک بیت توازن را می‌توان برای تشخیص خطاها در داده‌های ارسالی، با اطمینان از زوج یا فرد بودن تعداد کل یک‌ها، بسته به حالت توازنی انتخابی، استفاده کرد. در نهایت، یک یا چند بیت توقف پایان بسته را نشان می‌دهد و به گیرنده اجازه می‌دهد تا برای ارسال بعدی آماده شود [۱۴]. گیرنده بیت‌های ورودی را با نرخ بادی[۷] از پیش تنظیم‌شده می‌خواند که برای برقراری ارتباط

[1] Start Bit
[2] Parity Bit
[3] Stop Bit
[4] Bluetooth
[5] Transmitter
[6] Receiver
[7] Baud Rate

موفق باید با نرخ فرستنده مطابقت داشته باشد. عدم وجود یک ساعت مشترک، الزامات سیم‌کشی را ساده می‌کند و امکان سرعت انتقال داده‌های انعطاف‌پذیر را فراهم می‌آورد، اما زمان‌بندی دقیق را نیز برای جلوگیری از دست رفتن داده یا خرابی ضروری می‌سازد. در شکل ۲-۱۱ [۱۵] نحوه‌ی تعامل میان فرستنده و دریافت‌کننده در پروتکل UART مشاهده می‌شود.



شکل ۲-۱۱ نحوه‌ی تعامل میان فرستنده و دریافت کننده در UART [۱۵]

در کاربردهای عملی، این رابط ارتباطی معمولاً در سیستم‌های تعبیه‌شده یافت می‌شود، جایی که سادگی و سهولت پیاده‌سازی حیاتی است. میکروکنترلرها اغلب شامل ماژول‌های داخلی برای تسهیل ارتباط با وسایل جانبی مانند سنسورها، نمایشگرها و ماژول‌های ارتباطی هستند. علاوه بر این، به طور گسترده در پورت‌های سریال کامپیوتر استفاده می‌شود و روشی ساده برای اتصال دستگاه‌های خارجی مانند مودم‌ها و کنسول‌های سریال ارائه می‌دهد. علیرغم سرعت انتقال داده نسبتاً پایین آن در مقایسه با سایر پروتکل‌های ارتباطی، استحکام و سهولت استفاده از این پروتکل، آن را به ابزاری ارزشمند برای نیازهای مختلف ارتباط سریال در الکترونیک مصرفی و صنعتی تبدیل کرده است.

۲-۴- محیط انجام آزمایش و بردها

برای پیاده‌سازی پروژه، از بردها و ابزارهایی استفاده شده است که در این بخش به طور خلاصه هر کدام توضیح داده می‌شوند.

- دو عدد برد AVA3S400 از خانواده Spartan3

برد AVA3S400 با استفاده از تراشه XS3C400 شرکت Xilinx ساخته شده است. هدف اصلی این برد، بهره‌برداری از ویژگی‌های این تراشه برای پیاده‌سازی الگوریتم‌های دیجیتال و پردازشی از طریق زبان‌های برنامه‌نویسی HDL[1] است.

[1] Hardware Description Language

در جدول (۲-۱) ویژگی‌ها و منابع Spartan3 AVA3S400 توضیح داده شده است.

جدول ۲-۱ ویژگی‌های برد Spartan3 AVA3S400

توضیحات	تعداد/ مقدار
حافظه	۴Mb
تعداد زنگ هشدار[۱]	۱
بیت‌های RAM توزیع شده	۵۶۰۰۰
تعداد کلید چند حالته	۱۶
ضرب‌کننده های تخصیص داده شده	۱۶
مدیر ساعت دیجیتال[۲]	۴
گیت‌های سیستم[۳]	۴۰۰۰۰۰
حداکثر ورودی/خروجی کاربر	۲۶۴
تعداد دیود نوری[۴]	۱۶
تعداد بلوک منطقی قابل تنظیم	۸۹۶
حداکثر سرعت	۴۸۰Mb/s
تعداد بیت‌های بلوک RAM	۲۸۸۰۰۰
کلید فشاری جهت اعمال داده	۴
تعداد 7Segment	۴

شکل ۲-۱۲ برد Spartan3 AVA3S400

[1] Buzzer
[2] Digital Clock Manager
[3] System Gates
[4] Light Emitting Diode

- دو عدد ماژول ESP32-WROOM-32

این ماژول قدرتمند وای‌فای و بلوتوث[1] که توسط Espressif Systems توسعه یافته است، به دلیل قابلیت‌هایش در پروژه‌های اینترنت اشیا[2] و ارتباطات بی‌سیم به طور گسترده‌ای شناخته شده است. در زیر نگاهی عمیق به ویژگی‌های آن آورده شده است:

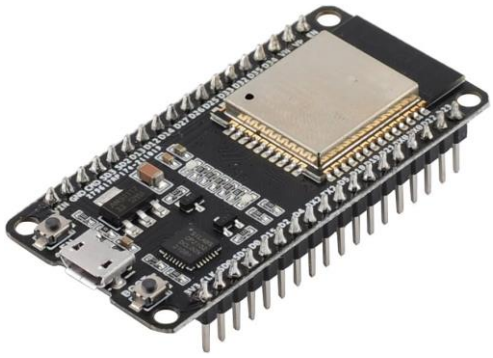
۱) ارتباطات بی‌سیم: این ماژول از نظر اتصال قدرتمند است و از وای‌فای و بلوتوث پشتیبانی می‌کند. این استاندارد از استانداردهای Wi-Fi 802.11 b/g/n پیروی می‌کند و اتصال آسان به شبکه‌ها را امکان‌پذیر می‌کند و همچنین از بلوتوث ۴.۲ و بلوتوث کم انرژی[3] برای ارتباط بدون درز با دستگاه‌های مختلف پشتیبانی می‌کند.
۲) مدیریت انرژی: با در نظر گرفتن بهره‌وری انرژی طراحی شده است و ویژگی‌های پیشرفته مدیریت انرژی را ارائه می‌دهد. تنظیم‌کننده‌های ولتاژ داخلی ماژول، که در محدوده ولتاژ ۲.۷ ولت تا ۳.۶ ولت کار می‌کنند، عملکرد پایدار را تضمین می‌کنند و از حالت‌های مختلف کم مصرف پشتیبانی می‌کنند، که این ماژول را برای پروژه‌هایی که با باتری کار می‌کنند، ایده آل می‌سازد.
۳) حافظه: این ماژول مجهز به ۴ مگابایت حافظه فلش است که می‌تواند سیستم عامل، کد برنامه و سایر داده‌ها را ذخیره کند. همچنین شامل ۴۴۸ کیلوبایت ROM و ۱۶ کیلوبایت RTC SRAM است که فضای ذخیره‌سازی کافی را برای برنامه‌های پیچیده فراهم می‌کند.
۴) ویژگی‌های امنیتی: با اولویت‌بندی امنیت، این ماژول دارای ویژگی‌هایی مانند راه‌اندازی امن، رمزگذاری فلش، و شتاب سخت‌افزار رمزنگاری است که آن را برای برنامه‌هایی که حفاظت از داده‌ها حیاتی است، مناسب می‌کند.
۵) میکروکنترلر: در هسته آن یک پردازنده قدرتمند دو هسته‌ای قرار دارد که می‌تواند با سرعت ۲۴۰ مگاهرتز کار کند. با ۵۲۰ کیلوبایت SRAM و طیف وسیعی از تجهیزات جانبی، میکروکنترلر برای کارهای سخت و برنامه‌های بلادرنگ مناسب است.
۶) پورت‌ها و ورودی/خروجی: این ماژول با ارائه قابلیت‌های I/O گسترده، شامل ۳۶ پین GPIO است که به عنوان ورودی یا خروجی دیجیتال قابل تنظیم است. این پین‌ها از عملکردهای مختلفی مانند PWM، ADC، DAC و سنسورهای لمسی به همراه ADCهای ۱۲ بیتی و DACهای ۸ بیتی برای ورودی و خروجی دقیق آنالوگ پشتیبانی می‌کنند.
۷) کاربردها: این ماژول که به طور گسترده در پروژه‌های اینترنت اشیا، اتوماسیون خانگی، لوازم الکترونیکی پوشیدنی و شبکه‌های حسگر استفاده می‌شود، قدرت پردازشی و ویژگی‌های ارتباطی قوی این ماژول را

[1] Bluetooth
[2] Internet of Things (IoT)
[3] Bluetooth Low Energy (BLE)

به یک راه‌حل ایده‌آل برای توسعه‌دهندگانی تبدیل می‌کند که قصد ایجاد دستگاه‌های هوشمند و متصل را دارند.

۸) پروتکل‌های ارتباطی: این ماژول از پروتکل های ارتباطی مختلف با تمرکز قابل توجه بر روی UART پشتیبانی می‌کند. این تطبیق‌پذیری امکان ادغام مستقیم با حسگرهای خارجی، لوازم جانبی و سایر دستگاه‌ها را فراهم می‌کند و آن را با طیف گسترده‌ای از برنامه ها سازگار می‌کند.

۹) برنامه‌نویسی و پشتیبانی نرم افزار: این ماژول با قابلیت برنامه‌ریزی بالا با محیط‌های توسعه مانند Arduino IDE، MicroPython و Espressif's ESP-IDF سازگار است. می‌توان آن را به زبان‌هایی مانند C/C++، Python و Lua، با کتابخانه‌ها و APIهای گسترده برای کنترل ورودی/خروجی دیجیتال و آنالوگ، وای‌فای، بلوتوث و سایر لوازم جانبی برنامه‌ریزی کرد.



شکل ۲-۱۳ ماژول ESP32-WROOM-32

۲-۵- ابزارها

- ISE مجموعه نرم‌افزاری است که توسط Xilinx برای طراحی و برنامه‌نویسی دستگاه‌های FPGA توسعه‌یافته است. مجموعه‌ای جامع از ابزارها و ویژگی‌ها را برای تسهیل کل جریان طراحی FPGA، از ورود طراحی اولیه تا تولید جریان بیت، فراهم می‌کند.

ISE شامل ماژول‌ها و اجزای مختلفی است که برخی از آنها عبارت‌اند از:

۱) Project Navigator: این ابزار اصلی برای مدیریت و سازماندهی پروژه‌های FPGA است و کاربران را قادر می‌سازد تا فایل‌های طراحی خود را ایجاد و مدیریت کنند، شبیه‌سازی‌ها را اجرا کنند و سنتز و پیاده‌سازی را انجام دهند.

۲) یک محیط برای طراحی‌های اولیه فراهم می‌کند که کاربران می‌توانند طرح‌های خود را در مراحل ابتدایی توسعه دهند و از شبیه‌سازی و تحلیل‌های اولیه برای بهبود کیفیت طراحی استفاده کنند.

۳) شامل مجموعه‌ای از ابزارهای پیشرفته برای مرحله پیاده‌سازی است که شامل نگاشت طرح بر روی دستگاه FPGA، انجام بهینه‌سازی مکان‌یابی و مسیریابی، و تولید بیت‌استریم نهایی می‌شود.

۴) یک شبیه‌ساز داخلی است که امکان شبیه‌سازی عملکرد طراحی‌های دیجیتال نوشته‌شده به زبان‌های سخت‌افزاری را فراهم می‌کند. این ابزار قابلیت مشاهده شکل موج، اشکال‌زدایی تعاملی و ایجاد تست‌بنچ‌ها را ارائه می‌دهد.

۵) امکان دیباگ و تحلیل در زمان واقعی را فراهم می‌کند، این امکان را می‌دهد تا عملکرد مدارهای دیجیتال را در حین اجرا بررسی کرده و مشکلات را در لحظه شناسایی و رفع کرد. این ویژگی به بهبود سرعت توسعه و کاهش خطاها کمک می‌کند.

۶) این ابزار این امکان را می‌دهد که مصرف انرژی طراح FPGA را بررسی کرده و بر اساس آن بهینه‌سازی کرد. این ویژگی برای پروژه‌هایی که نیاز به مصرف انرژی زیادی دارند، بسیار کاربردی است.

۷) قادر به تولید گزارش‌های جامع از مراحل مختلف طراحی و پیاده‌سازی است. این گزارش‌ها شامل اطلاعاتی درباره استفاده از منابع، زمان‌بندی، و بهینه‌سازی‌ها می‌شود که به شما در ارزیابی و بهبود طراحی کمک می‌کند.

۸) ابزاری برای بهینه‌سازی استفاده از منابع FPGA مانند LUTها، رجیسترها و بلوک‌های RAM ارائه می‌دهد، که این ابزار به بهبود کارایی و عملکرد طراحی کمک می‌کند.

۹) زبان طراحی XDL: XDL یک قالب داخلی است که برای نمایش سلسله‌مراتب طراحی و اتصال مدارهای دیجیتال استفاده می‌شود. این زبان نقش مهمی در مراحل پیاده‌سازی و تولید بیت‌استریم ایفا می‌کند.

۱۰) ISE از طراحی مدولار پشتیبانی می‌کند، به این معنی که کاربران می‌توانند طراحی‌های پیچیده خود را به ماژول‌های کوچکتر تقسیم کرده و هر ماژول را به صورت جداگانه توسعه و تست کنند. این قابلیت به بهبود ساختار و نگهداری طراحی کمک می‌کند.

- Arduino IDE یک پلتفرم نرم‌افزاری متن‌باز است که به منظور برنامه‌نویسی بردهای آردوینو ایجاد شده است. این پلتفرم با ارائه یک رابط کاربرپسند و ابزارهای متنوع، فرآیند نوشتن، کامپایل و بارگذاری کد بر

روی میکروکنترلرهای آردوینو را تسهیل می‌کند. در اینجا برخی از ویژگی‌های کلیدی IDE Arduino آورده شده است:

۱) پشتیبانی از چندین سیستم‌عامل: IDE برای سیستم‌عامل‌های مختلف از جمله ویندوز، macOS و لینوکس در دسترس است، که به این معنی است که کاربران با انواع مختلف سیستم‌عامل‌ها می‌توانند از این ابزار بهره‌مند شوند.

۲) شامل ویرایشگری است که به شما اجازه می‌دهد کدهای آردوینو را به زبان برنامه‌نویسی C++ بنویسید. این ویرایشگر با امکاناتی مانند برجسته‌سازی نحو و تکمیل خودکار، به ساده‌سازی و سرعت بخشیدن به فرآیند کدنویسی کمک می‌کند.

۳) دارای یک ابزار مدیریت کتابخانه است که به شما امکان می‌دهد به سادگی کتابخانه‌ها را جستجو، نصب و مدیریت کنید. کتابخانه‌ها شامل مجموعه‌های کد آماده‌ای هستند که قابلیت‌های اضافی را به پروژه‌های شما اضافه کرده و کار با حسگرها، عملگرها، و پروتکل‌های ارتباطی را ساده‌تر می‌کنند.

۴) فرآیند تبدیل کد به دستورالعمل‌های قابل فهم برای ماشین را به صورت خودکار انجام می‌دهد و با فشردن یک دکمه می‌توانید کد کامپایل‌شده را به برد آردوینو متصل بارگذاری کنید.

۵) پایشگر سریال یکی از ابزارهای داخلی IDE است که اجازه می‌دهد با برد آردوینو از طریق رابط سریال ارتباط برقرار شود. این ابزار خروجی‌ها و داده‌های ارسال‌شده از آردوینو را نمایش می‌دهد و برای اشکال‌زدایی و نظارت بر عملکرد کد بسیار مفید است.

۶) یکی از ویژگی‌های Arduino IDE دیباگر داخلی است که امکان اشکال‌زدایی مستقیم از کد را فراهم می‌کند. این ابزار به کاربران اجازه می‌دهد تا به صورت گام‌به‌گام کد خود را بررسی کرده و مشکلات موجود را شناسایی و رفع کنند.

۷) ابزار مدیریت برد در IDE این امکان را می‌دهد که پکیج‌های پشتیبانی از انواع بردها را نصب و پیکربندی کنید. این قابلیت کمک می‌کند تا مدل مناسب برد آردوینو را برای پروژه‌ها انتخاب کرده و درایورهای ضروری برای آن را نصب کرد.

۸) از پلاگین‌های مختلف پشتیبانی می‌کند که کاربران می‌توانند برای افزایش قابلیت‌های IDE از آنها استفاده کنند. این پلاگین‌ها می‌توانند ابزارهای اضافی، تم‌های جدید، یا قابلیت‌های پیشرفته‌ای را به محیط توسعه اضافه کنند.

۲-۶- خلاصه

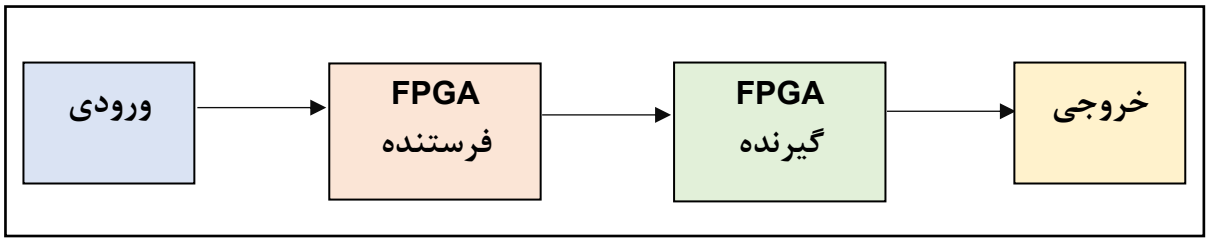
در این بخش، ابتدا معماری شبکه عصبی پیچشی و روش‌های محاسباتی آن به‌طور جامع بررسی شد. سپس به تشریح معماری پروتکل UART و اجزای آن پرداخته شد. در پایان، ابزارها و محیط اجرای این پژوهش به‌طور کامل مورد ارزیابی قرار گرفت.

فصل سوم: طراحی و پیاده‌سازی

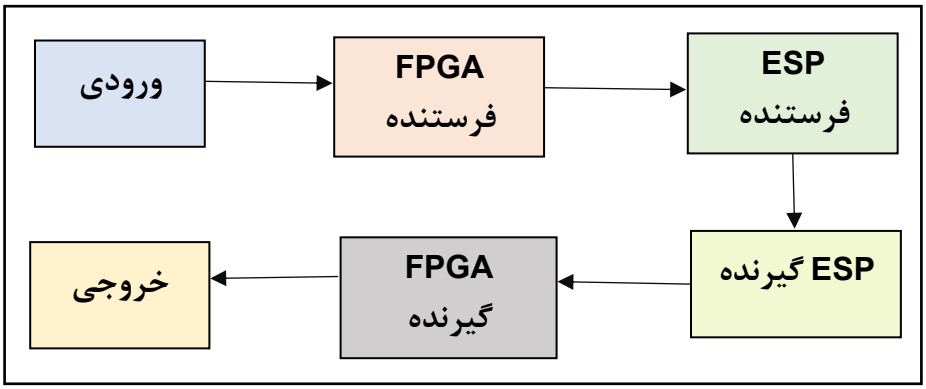
در این قسمت نحوه‌ی پیاده‌سازی, ساختار ماژول‌ها و اجرای پروژه بطور کامل شرح داده خواهد شد.

۳-۱- معماری کلی سیستم

فرآیند کلی پروژه به این صورت است که در ابتدا، FPGA اول داده‌های ورودی را دریافت کرده و محاسباتی را انجام می‌دهد و نتایج میانی تولید می‌کند. برای ادامه محاسبات، این نتایج میانی به FPGA دوم ارسال می‌شود. این انتقال، FPGA دوم را قادر می‌سازد تا محاسبات نهایی را بر اساس پیشرفت FPGA اول انجام دهد. با تقسیم بار کاری بین دو FPGA، سیستم به پردازش موازی دست می‌یابد و در نتیجه، اجرای کارآمدتری حاصل می‌شود. در نهایت، با تکمیل محاسبات، FPGA دوم خروجی نهایی را تولید می‌کند. این خروجی نشان‌دهنده اوج کل محاسبات شبکه عصبی پیچشی است و به عنوان نتیجه مورد انتظار عمل می‌کند. برای درک بهتر این پروژه، نمای کلی سیستم برای حالت باسیم و بی‌سیم به ترتیب در شکل (۳-۱) و (۳-۲) نشان داده شده است.



شکل ۳-۱ نمای کلی سیستم برای حالت باسیم

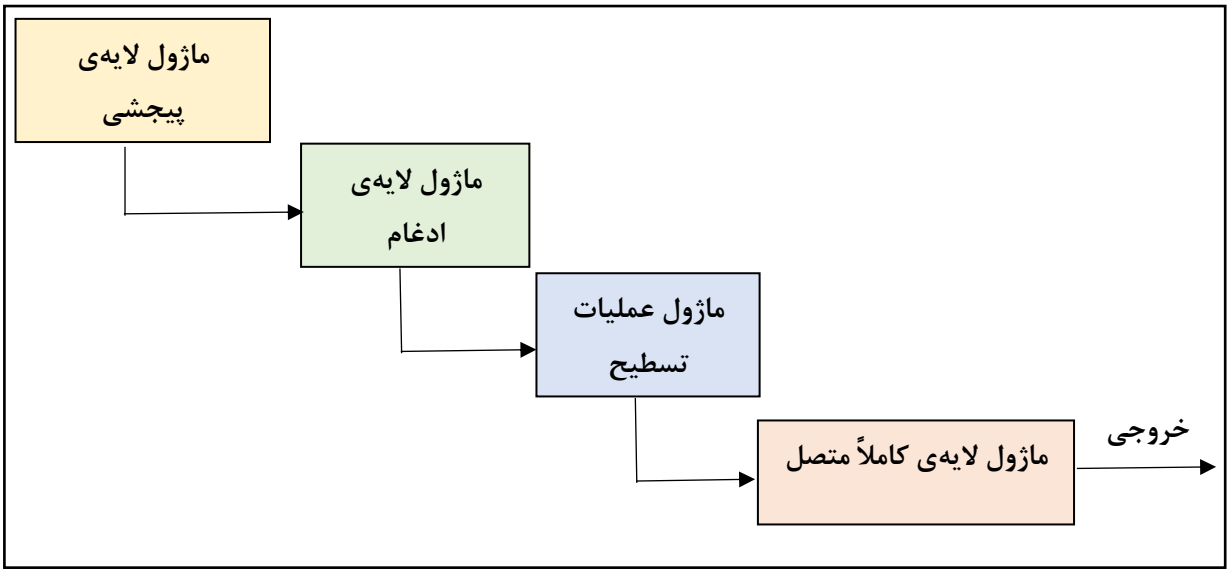


شکل ۳-۲ نمای کلی سیستم برای حالت بی‌سیم

با توجه به توضیحات ذکر شده در ادامه به بررسی ساختار ماژول‌ها و سپس به شرح دقیق معماری سیستم برای دو حالت باسیم و بی‌سیم می‌پردازیم.

۳-۲- مدل معماری پیاده‌سازی شده برای شبکه عصبی پیچشی

در این پژوهش، به منظور پیاده‌سازی شبکه عصبی پیچشی، از چهار ماژول شامل لایه‌ی پیچشی، لایه‌ی ادغام، عملیات تسطیح، و لایه‌ی کاملاً متصل استفاده شده است. این ماژول‌ها به‌صورت سلسله‌مراتبی و متوالی عمل کرده و عملیات را به‌طور کارآمد انجام می‌دهند. در شکل (۳-۳)، ساختار و ترتیب اجرای مدل پیاده‌سازی شده به‌وضوح مشاهده می‌شود و نمایی کلی از نحوه عملکرد این شبکه ارائه می‌گردد.

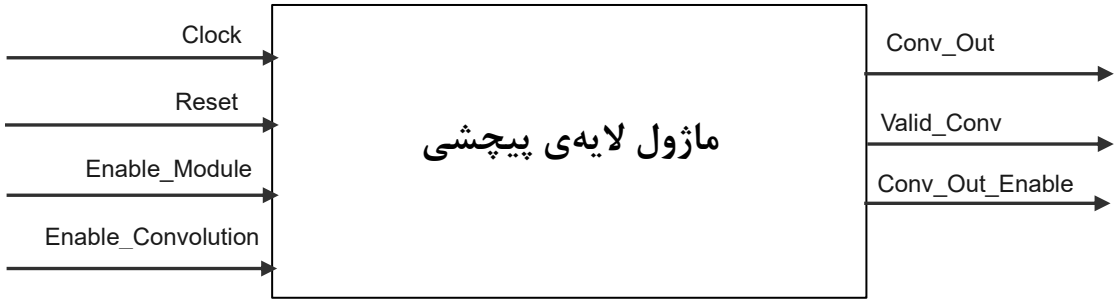


شکل ۳-۳ نمای کلی مدل پیاده‌سازی شده برای شبکه عصبی پیچشی

در ادامه به بررسی ساختار ماژول‌های شبکه عصبی پیچشی می‌پردازیم.

۳-۲-۱- ماژول لایه‌ی پیچشی

شکل (۳-۴) نشان‌دهنده ماژول لایه‌ی پیچشی است. این ماژول برای انجام عملیات ریاضی طراحی شده است که در آن هر فیلتر بر روی ورودی حرکت می‌کند، مقادیر فیلتر را با مقادیر متناظر ورودی ضرب می‌کند و سپس آنها را جمع می‌کند تا یک مقدار واحد در نقشه ویژگی خروجی تولید شود. جزئیات پایه‌های ماژول در جدول (۳-۱) ذکر شده است.



شکل ۳-۴ ماژول لایه‌ی پیچشی

جدول ۳-۱ توضیحات سیگنال‌های ماژول لایه‌ی پیچشی

سیگنال	نوع سیگنال	طول سیگنال	توضیحات
Clock	ورودی	۱	سیگنال ساعت ورودی
Reset	ورودی	۱	راه‌اندازی مجدد ماژول
Enable_Convolution	ورودی	۱	فعال‌سازی لایه‌ی پیچش
Enable_Module	ورودی	۱	فعال‌سازی ماژول
Valid_Conv	خروجی	۱	نشان‌دهنده‌ی معتبر بودن داده‌ی خروجی
Conv_Out_Enable	خروجی	۱	فعال‌سازی خروجی جریان لایه اول
Conv_Out	خروجی	۴	خروجی لایه‌ی پیچشی/ این خروجی به صورت بافر است و ۱۶ مرتبه برای پردازش به لایه‌ی بعدی فرستاده می‌شود.

فرآیند کامل این ماژول به این صورت است که در ابتدا عملیات پیچشی با یک پنجره کشویی شروع می‌شود که در سراسر تصویر ورودی حرکت می‌کند و مقادیر پیکسل را در یک پنجره دو در دو جمع‌آوری می‌کند. سپس این مقادیر در وزن فیلتر مربوطه ضرب می‌شوند و نتایج در دو مرحله جمع‌آوری می‌شوند. یک بایاس به مجموع نهایی اضافه می‌شود و یک تابع فعال‌سازی ReLU اعمال می‌شود که مقدار مثبت یا صفر حاصل را تولید می‌کند. سپس خروجی در یک بافر ذخیره می‌شود. این عمل برای هر گام روی تصویر تکرار می‌شود و نتایج پس از هر گام در بافر ذخیره می‌شود. خروجی‌های بافر شده به صورت متوالی به لایه بعدی ارسال می‌شوند و اطمینان حاصل می‌شود که هر خروجی پردازش شده به درستی در هنگام آماده شدن منتقل می‌شود. کل این فرآیند ۱۶ بار تکرار می‌شود، که مربوط به تعداد کل مناطق دو در دو در تصویر پنج در پنج است، و پس از پردازش همه مناطق، لایه برای عملیات بعدی تنظیم مجدد می‌شود. لازم به ذکر است که داده‌های ورودی این ماژول به صورت هارد-کد[1] پیاده‌سازی شده اند.

[1] Hardcode

۳-۲-۲- ماژول لایه‌ی ادغام

شکل (۳-۵) نشان‌دهنده ماژول لایه‌ی ادغام است. این ماژول برای کاهش ابعاد فضایی نقشه ویژگی ورودی طراحی شده است. این کار با اعمال عملیات حداکثر ادغام انجام می‌شود که شامل انتخاب حداکثر مقدار از یک پنجره با اندازه ثابت است. جزئیات پایه‌های ماژول در جدول (۳-۲) ذکر شده است.

شکل ۳-۵ ماژول لایه‌ی ادغام

جدول ۳-۲ توضیحات سیگنال‌های ماژول لایه‌ی ادغام

سیگنال	نوع سیگنال	طول سیگنال	توضیحات
Pool_Input	ورودی	۴	داده‌ی ورودی لایه‌ی ادغام
Clock	ورودی	۱	سیگنال ساعت ورودی
Reset	ورودی	۱	راه‌اندازی مجدد ماژول
Enable_Pooling	ورودی	۱	فعال‌سازی لایه‌ی ادغام
Enable_Module	ورودی	۱	فعال‌سازی ماژول
Valid_Pool	خروجی	۱	نشان‌دهنده‌ی معتبر بودن داده‌ی خروجی
Pool_Out_Enable	خروجی	۱	فعال‌سازی خروجی جریان لایه دوم
Pool_Out	خروجی	۴	خروجی لایه‌ی ادغام/ این خروجی به صورت بافر است و ۴ مرتبه برای پردازش به لایه‌ی بعدی فرستاده می‌شود.

فرآیند کامل این ماژول به این صورت است که به طور متوالی داده‌های ورودی را در یک پنجره کشویی بارگذاری می‌کند که بر اساس اندازه فیلتر و گام برداشتن در سراسر تصویر حرکت می‌کند. داده‌های پنجره‌ای در یک بافر FIFO[1] ذخیره می‌شود و حداکثر مقدار درون پنجره از طریق مقایسه محاسبه می‌شود. سپس این مقدار حداکثر در یک بافر خروجی ذخیره می‌شود که به لایه بعدی منتقل می‌شود. داده‌ها چندین بار به لایه بعدی منتقل می‌شوند که توسط گام کنترل می‌شود تا زمانی که تمام مقادیر خروجی معتبر تولید شوند. سیگنال‌های کنترل، جابجایی پنجره، عملیات FIFO و تولید خروجی را مدیریت می‌کنند و با مقداردهی اولیه و هماهنگی، عملکرد صحیح را در طول فرآیند تضمین می‌کنند.

۳-۲-۳- ماژول عملیات تسطیح

شکل (۳-۶) نشان‌دهنده ماژول لایه‌ی تسطیح است. این ماژول برای تبدیل یک نقشه ویژگی ورودی چندبعدی به یک بردار یک‌بعدی طراحی شده است. این کار با تغییر شکل نقشه ویژگی انجام می‌شود تا امکان تغذیه آن به لایه‌های کاملاً متصل برای طبقه‌بندی فراهم شود. جزئیات پایه‌های ماژول در جدول (۳-۳) ذکر شده است.

شکل ۳-۶ ماژول عملیات تسطیح

[1] First In First Out

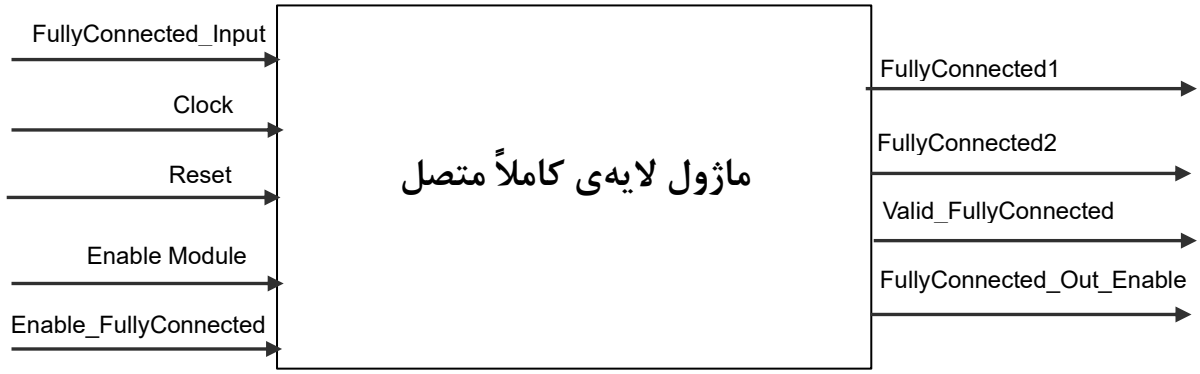
جدول ۳-۳ توضیحات سیگنال‌های ماژول عملیات تسطیح

سیگنال	نوع سیگنال	طول سیگنال	توضیحات
Flatten_Input	ورودی	۴	داده‌ی ورودی عملیات تسطیح
Clock	ورودی	۱	سیگنال ساعت ورودی
Reset	ورودی	۱	راه‌اندازی مجدد ماژول
Enable_Flatten	ورودی	۱	فعال‌سازی عملیات تسطیح
Enable_Module	ورودی	۱	فعال‌سازی ماژول
Valid_Flat	خروجی	۱	نشان‌دهنده‌ی معتبر بودن داده‌ی خروجی
Flat_Out_Enable	خروجی	۱	فعال‌سازی خروجی جریان لایه سوم
Flat_Out	خروجی	۴	خروجی عملیات تسطیح/ این خروجی به صورت بافر است و ۴ مرتبه برای پردازش به لایه‌ی بعدی فرستاده می‌شود.

فرآیند دقیق ماژول عملیات تسطیح به این صورت است که این ماژول با پردازش متوالی مقادیر پیکسل با استفاده از یک پنجره کشویی، یک نقشه ویژگی دو بعدی را به یک بردار یک‌بعدی تبدیل می‌کند. این پنجره دو در دو به‌طور مداوم با داده‌های ورودی به‌روز می‌شود و مقادیر را در چهار بافر خروجی ذخیره می‌کند. هنگامی که بافرها پر شده و شرایط مرحله‌ای برآورده شود، ماژول داده‌های بافر را به لایه بعدی ارسال می‌کند. سیگنال‌های کنترل زمان‌بندی انتقال داده را مدیریت می‌کنند و اطمینان می‌دهند که مقادیر پیکسل به‌طور کنترل‌شده و متوالی منتقل شوند. این فرآیند، انتقال روان داده‌ها را از طریق مراحل شبکه تسهیل می‌کند.

۳-۲-۴- ماژول لایه‌ی کاملاً متصل

شکل (۳-۷) نشان‌دهنده ماژول لایه‌ی کاملاً متصل است. این ماژول برای محاسبه خروجی با انجام ضرب ماتریس بین بردار ورودی مسطح و یک ماتریس وزن طراحی شده است. پس از آن، بایاس به جمع مقادیر اضافه شده و یک تابع فعال‌سازی برای تولید مجموعه‌ای از مقادیر خروجی اعمال می‌شود. جزئیات پایه‌های ماژول در جدول (۳-۴) ذکر شده است.



شکل ۳-۷ ماژول لایه‌ی کاملاً متصل

جدول ۳-۴ توضیحات سیگنال‌های ماژول لایه‌ی کاملاً متصل

سیگنال	نوع سیگنال	طول سیگنال	توضیحات
FullyConnected_Input	ورودی	۴	داده‌ی ورودی لایه‌ی کاملاً متصل
Clock	ورودی	۱	سیگنال ساعت ورودی
Reset	ورودی	۱	راه‌اندازی مجدد ماژول
Enable_FullyConnected	ورودی	۱	فعال‌سازی لایه‌ی کاملاً متصل
Enable_Module	ورودی	۱	فعال‌سازی ماژول
Valid_FullyConnected	خروجی	۱	نشان‌دهنده‌ی معتبر بودن داده‌ی خروجی
FullyConnected_Out_Enable	خروجی		فعال‌سازی خروجی جریان لایه‌ی چهارم
FullyConnected1	خروجی	۴	خروجی اول لایه‌ی کاملاً متصل
FullyConnected2	خروجی	۴	خروجی دوم لایه‌ی کاملاً متصل

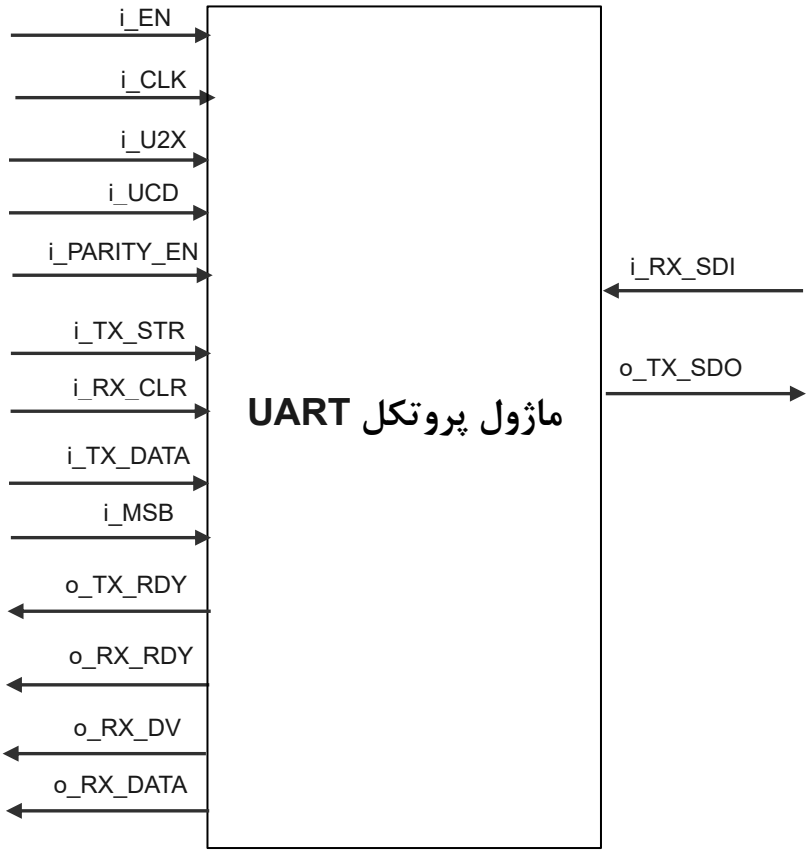
فرآیند دقیق ماژول لایه کاملاً متصل به این صورت است که داده‌های ورودی ابتدا در وزن‌های از پیش تعریف شده ضرب می‌شوند و نتایج ضرب متوسط را ایجاد می‌کنند. سپس این نتایج در دو مرحله جمع می‌شوند تا مشارکت‌ها از همه وزن‌ها جمع شوند. مقادیر جمع شده با اضافه کردن بایاس‌ها تنظیم می‌شوند. سپس، یک تابع فعال‌سازی ReLU به نتایج تعدیل‌شده بایاس اعمال می‌شود و مقادیر منفی را صفر می‌کند. نتایج فعال شده در بافرهای خروجی ذخیره می‌شوند. سیگنال‌های کنترل هر مرحله را مدیریت می‌کنند و از ترتیب‌دهی مناسب و فعال‌سازی عملیات را اطمینان می‌دهد.

۳-۳- مدل معماری پیاده‌سازی شده برای پروتکل UART

در این پژوهش به منظور پیاده‌سازی ارتباط بین دو دستگاه FPGA از پروتکل UART استفاده شده است. برخی از دلایل انتخاب این پروتکل عبارتند از:

۱) پروتکل بسیار ساده و کاربردی است.
۲) مقرون‌به‌صرفه است.
۳) ارتباط بین دو FPGA تنها به دو کانال TX و RX نیاز دارد.
۴) این پروتکل از نوع ارتباط غیرهمگام است، به این معنا که هر FPGA می‌تواند به‌طور مستقل داده ارسال کند.
۵) نیاز به سخت‌افزار پیچیده‌ای ندارد.
۶) برای ارتباطات نقطه به نقطه میان دستگاه‌های FPGA بسیار مناسب است.
۷) پیاده‌سازی این پروتکل نیاز به منابع نرم‌افزاری و سخت‌افزاری کمی دارد.
۸) از نظر مصرف انرژی کارآمد است.

ماژول UART در شکل (۳-۸) نشان داده شده است. در جدول (۳-۵) سیگنال‌ها و پایه‌های ماژول UART شرح داده شده است. ماژول UART از سه زیر ماژول تشکیل شده است.



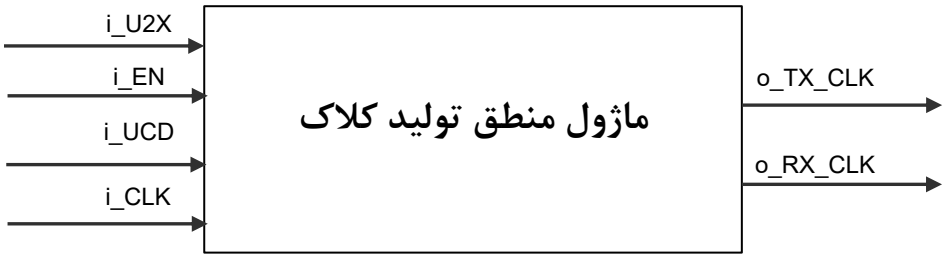
شکل ۳-۸ ماژول UART

جدول ۳-۵ توضیحات سیگنال‌های ماژول UART

سیگنال	نوع سیگنال	طول سیگنال	توضیحات
i_CLK	ورودی	۱	سیگنال ساعت ورودی
i_PARITY_EN	ورودی	۱	فعال‌سازی بیت توازن
i_U2X	ورودی	۱	دو برابر کردن سرعت ارسال
i_TX_STR	ورودی	۱	شروع انتقال داده
i_EN	ورودی	۱	فعال‌سازی ماژول
i_UCD	ورودی	۱۶	تعیین سرعت انتقال
i_RX_CLR	ورودی	۱	خواندن داده‌های دریافت شده
i_TX_DATA	ورودی	۸	داده‌های ارسالی
i_RX_SDI	ورودی	۱	کانال سریال RX
i_MSB	ورودی	۱	تعیین کننده‌ی ترتیب داده‌ها: بیت پرارزش یا کم‌ارزش اول
o_RX_DATA	خروجی	۸	داده‌های دریافتی
o_TX_SDO	خروجی	۱	کانال سریال TX
o_TX_RDY	خروجی	۱	آمادگی ماژول برای ارسال
o_RX_RDY	خروجی	۱	آمادگی برای دریافت داده جدید
o_RX_DV	خروجی	۱	نشان‌دهنده صحت یا خطا در داده‌ی دریافت‌شده

۳-۳-۱- ماژول منطق تولید کلاک

شکل (۳-۹) نشان‌دهنده ماژول تولید کلاک است. این ماژول برای تأمین کلاک مناسب جهت بخش‌های ارسال و دریافت ماژول UART طراحی شده است. جزئیات پایه‌های ماژول در جدول (۳-۶) ذکر شده است..



شکل ۳-۹ ماژول منطق تولید کلاک

جدول ۳-۶ توضیحات سیگنال‌های ماژول منطق تولید کلاک

سیگنال	نوع سیگنال	طول سیگنال	توضیحات
i_CLK	ورودی	۱	سیگنال ساعت ورودی
i_UCD	ورودی	۱	تعیین سرعت انتقال
i_U2X	ورودی	۱	دو برابر کردن سرعت ارسال
i_EN	ورودی	۱	فعال‌سازی ماژول
o_TX_CLK	خروجی	۱	ساعت واحد TX
o_RX_CLK	خروجی	۱	ساعت واحد RX

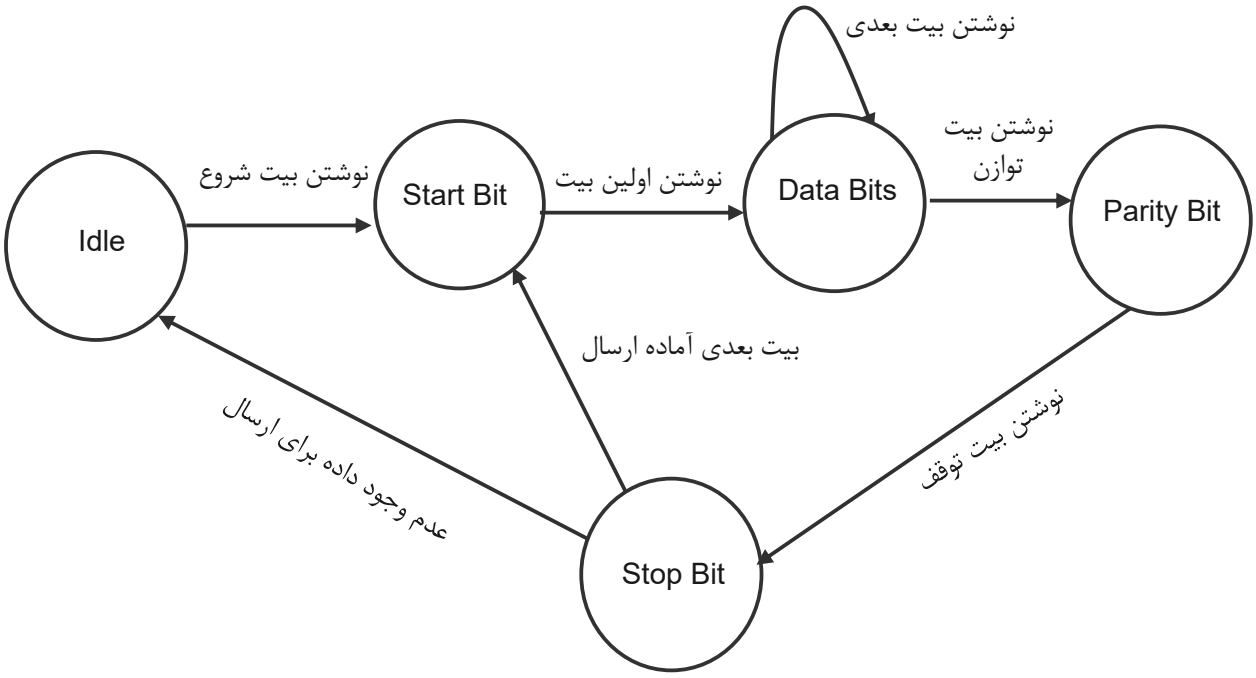
۳-۳-۲- ماژول فرستنده

شکل (۳-۱۰) نشان‌دهنده ماژول انتقال است. این ماژول برای ارسال یک بایت داده به صورت سریال طراحی شده است. جزئیات پایه‌های ماژول در جدول (۳-۷) ذکر شده است. ماشین حالت این ماژول در شکل (۳-۱۱) به تصویر کشیده شده است.

شکل ۳-۱۰ ماژول فرستنده

جدول ۳-۷ توضیحات سیگنال‌های ماژول فرستنده

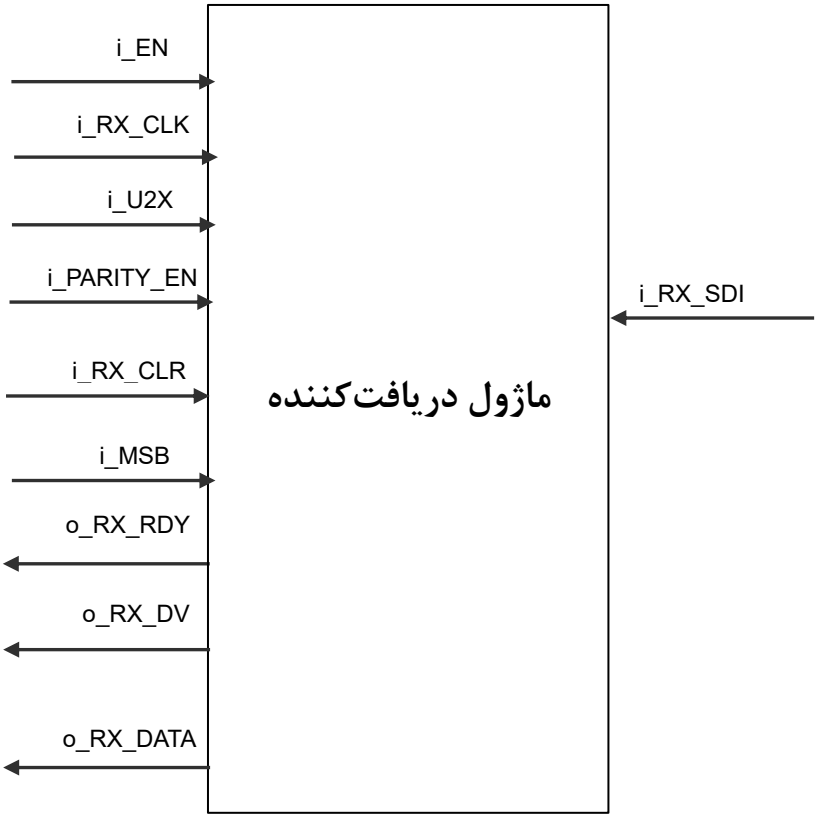
سیگنال	ورودی سیگنال	طول سیگنال	توضیحات
i_EN	ورودی	۱	فعال‌سازی ماژول
i_TX_CLK	ورودی	۱	ساعت واحد TX
i_TX_STR	ورودی	۱	شروع انتقال داده
i_TX_DATA	ورودی	۸	داده‌های ارسالی
i_PARITY_EN	ورودی	۱	فعال‌سازی بیت توازن
i_MSB	ورودی	۱	تعیین کننده‌ی ترتیب داده‌ها: بیت پرارزش یا کم‌ارزش اول
o_TX_RDY	خروجی	۱	آمادگی ماژول برای ارسال
o_TX_SDO	خروجی	۱	کانال سریال TX



شکل ۳-۱۱ ماشین حالت ماژول فرستنده

۳-۳-۳- ماژول دریافت‌کننده

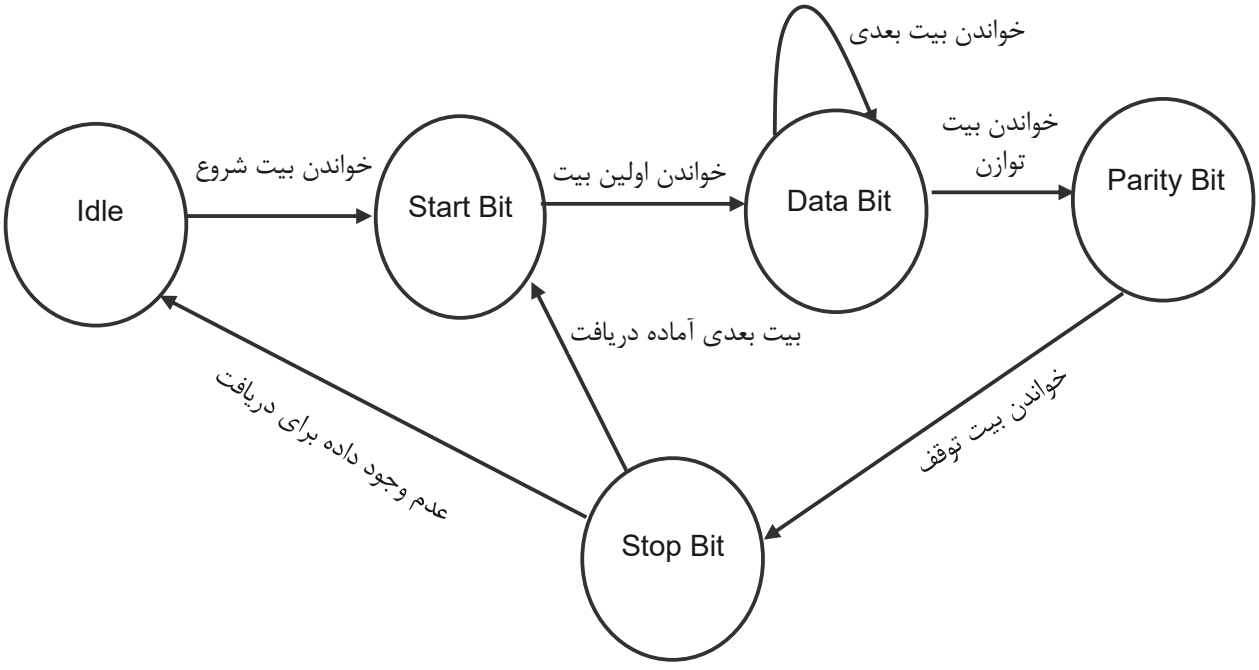
شکل (۳-۱۲) نشان‌دهنده ماژول دریافت کننده است. این ماژول برای دریافت یک بایت داده به صورت سریال طراحی شده است. جزئیات پایه‌های ماژول در جدول (۳-۸) ذکر شده است. ماشین حالت این ماژول در شکل (۳-۱۳) به تصویرکشیده شده است.



شکل ۳-۱۲ ماژول دریافت‌کننده

جدول ۳-۸ توضیحات سیگنال‌های ماژول دریافت‌کننده

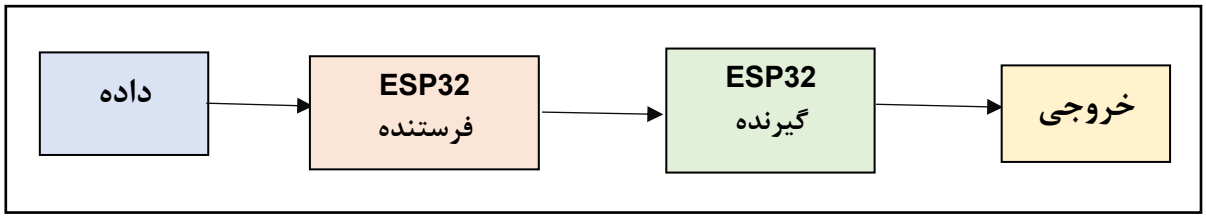
سیگنال	ورودی سیگنال	طول سیگنال	توضیحات
i_EN	ورودی	۱	فعال‌سازی ماژول
i_RX_CLK	ورودی	۱	ساعت واحد RX
i_U2X	ورودی	۱	دو برابر کردن سرعت ارسال
i_PARITY_EN	ورودی	۱	فعال‌سازی بیت توازن
i_RX_CLR	ورودی	۱	خواندن داده‌های دریافت شده
i_RX_SDI	ورودی	۱	کانال سریال RX
i_MSB	ورودی	۱	تعیین کننده‌ی ترتیب داده‌ها: بیت پرارزش یا کم‌ارزش اول
o_RX_RDY	خروجی	۱	آمادگی برای دریافت داده جدید
o_RX_DV	خروجی	۱	نشان‌دهنده صحت یا خطا در داده‌ی دریافت‌شده
o_RX_DATA	خروجی	۸	داده‌های دریافتی



شکل ۳-۱۳ ماشین حالت ماژول دریافت‌کننده

۳-۴- پیاده‌سازی ارتباط میان دو ماژول ESP32

در این قسمت به نحوه‌ی پیاده‌سازی ارتباط میان دو ESP32 می‌پردازیم. از این ارتباط در پیاده‌سازی حالت بی‌سیم سیستم استفاده می‌شود. در شکل (۳-۱۴) نمای کلی از فرآیند این ارتباط مشاهده می‌شود.



شکل ۳-۱۴ نمودار بلوکی تعامل میان دو ESP32

ارتباط بین دو دستگاه ESP32 شامل ترکیبی از پروتکل‌های UART و وای‌فای است. اولین دستگاه از طریق رابط UART خود داده‌ها را از یک منبع خارجی مانند FPGA دریافت می‌کند. هنگامی که داده‌ها روی پین دریافت UART خود می‌رسند، ESP32 فرستنده این بایت‌ها را می‌گیرد و آنها را برای انتقال بی‌سیم آماده می‌کند. این دستگاه با استفاده از قابلیت‌های وای‌فای خود، یک بسته[۱] UDP[۲] ایجاد می‌کند تا داده‌های دریافتی را کپسوله کند. سپس این بسته از طریق شبکه وای‌فای به آدرس ای‌پی[۳] و پورت خاصی که دستگاه دوم منتظر دریافت آن است ارسال می‌شود.

برای اینکه دستگاه‌ها به طور موثر ارتباط برقرار کنند، ابتدا باید یکدیگر را بشناسند و ارتباط برقرار کنند. دستگاه اول به شبکه وای‌فای ارائه شده توسط دستگاه دوم متصل می‌شود که به عنوان نقطه دسترسی وای‌فای[۴] عمل می‌کند. در طی این فرآیند، دستگاه اول SSID[۵] شبکه خاصی را که توسط دومی پخش شده است، با استفاده از اعتبار[۶] از پیش به اشتراک گذاشته شده برای احراز هویت و ایجاد یک پیوند امن جستجو می‌کند. پس از اتصال، این دو دستگاه اکنون می‌توانند به طور قابل اعتمادی تعامل داشته باشند. دستگاه اول داده‌ها را به آدرس ای‌پی و پورت مرتبط با دومی ارسال می‌کند که تشخیص و اتصال بین آنها را تأیید می‌کند. در شکل (۳-۱۵) ساختار این ارتباط مشاهده می‌شود.

[1] Packet
[2] User Datagram Protocol
[3] IP Address
[4] Wi-Fi Access Point
[5] Service Set Identifier
[6] Credentials

```
const char* ssid = "ESP32_AP";
const char* password = "12345678";
const char* udpAddress = "192.168.4.1";
const int udpPort = 1234;
```

شکل ۳-۱۵ ساختار برقراری ارتباط میان دو ماژول ESP32

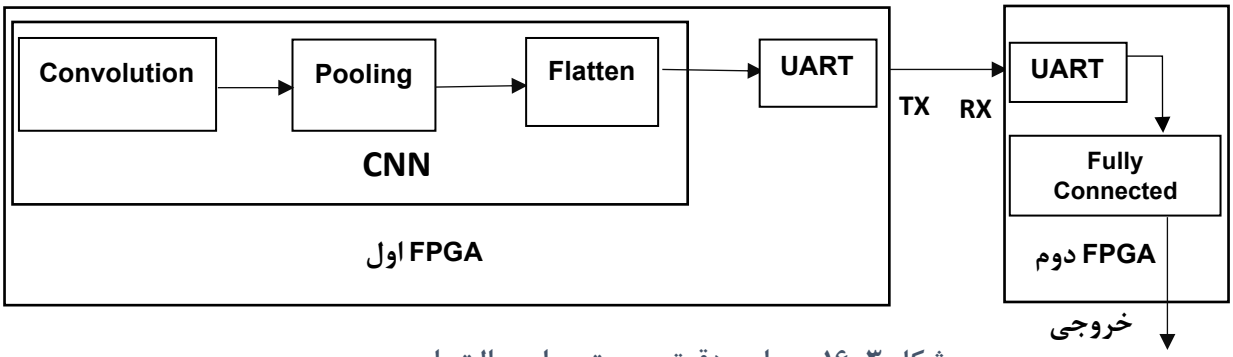
در انتهای گیرنده، ESP32 دوم به عنوان یک نقطه دسترسی وای‌فای برای مدیریت اتصالات ورودی تنظیم شده است. به طور مداوم درگاه UDP تعیین شده خود را برای هر بسته داده ورودی از واحد اول نظارت می‌کند. پس از دریافت بسته، دستگاه داده‌های محصور شده را استخراج می‌کند و آن را به رابط UART خود یا پردازش بیشتر، در صورت نیاز، هدایت می‌کند. این ترتیب انتقال بدون درز و بی‌سیم داده‌هایی را که ابتدا از طریق UART دریافت می‌شد، تسهیل می‌کند، و این دو واحد را قادر می‌سازد تا به طور موثر در سراسر یک شبکه ارتباط برقرار کنند، حتی زمانی که از نظر فیزیکی از هم جدا هستند.

۳-۵- پیاده‌سازی سیستم

در بخش ۳-۱، معماری کلی سیستم شرح داده شد. در این قسمت، پیاده‌سازی سیستم در دو حالت باسیم و بی‌سیم به صورت دقیق مورد بررسی قرار می‌گیرد. ابتدا پیاده‌سازی سیستم در حالت باسیم توضیح داده خواهد شد، سپس تفاوت‌ها و افزونه‌های لازم برای پیاده‌سازی در حالت بی‌سیم مورد بحث قرار خواهند گرفت. این تحلیل جامع، زمینه‌ای مناسب برای درک کامل از نحوه‌ی عملکرد سیستم در شرایط مختلف فراهم می‌آورد.

۳-۵-۱ پیاده‌سازی سیستم برای حالت باسیم

در این بخش، نحوه پیاده‌سازی سیستم برای حالت باسیم به طور کامل شرح داده می‌شود. برای برقراری توازن میان دو دستگاه FPGA، محاسبات شبکه عصبی پیچشی میان آن دو تقسیم شده است؛ به‌طوری که FPGA اول لایه‌ی پیچشی، لایه‌ی ادغام، و عملیات تسطیح و در مقابل FPGA دوم لایه‌ی کاملاً متصل را محاسبه می‌کند. فرآیند کلی سیستم در شکل (۳-۱۶) نشان داده شده است.



شکل ۳-۱۶ معماری دقیق سیستم برای حالت باسیم

پردازش در این سیستم با ماژول لایه پیچشی آغاز می‌شود. پس از اتمام پردازش در این لایه، لایه ادغام شروع به کار کرده و ابعاد تصویر را کاهش می‌دهد. خروجی‌های به‌دست‌آمده به‌صورت متوالی به ماژول عملیات تسطیح ارسال می‌شوند. با پایان یافتن محاسبات در این ماژول، محاسبات شبکه عصبی در FPGA اول به پایان می‌رسد. برای انتقال نتایج به FPGA دوم، از ماژول UART استفاده می‌شود. این ماژول، نتایج به‌دست‌آمده را بیت به بیت از طریق خط TX ارسال می‌کند. این انتقال با یک بیت شروع آغاز شده، سپس بیت‌های داده ارسال می‌شوند و در نهایت با یک بیت توقف به پایان می‌رسد. دستگاه دریافت‌کننده خط RX را برای شناسایی این بیت‌ها نظارت کرده و هنگام دریافت، داده‌های اصلی را بازسازی می‌کند. بیت‌های شروع و توقف بسیار مهم هستند، زیرا آغاز و پایان هر بسته داده را علامت‌گذاری کرده و از تفسیر صحیح جریان داده توسط دستگاه گیرنده اطمینان حاصل می‌کنند. سپس این داده‌ها به‌عنوان ورودی به ماژول کاملاً متصل ارسال شده و این ماژول محاسبات لازم را انجام داده و خروجی نهایی را تولید می‌کند. در ادامه، پایه‌های سیستم برای FPGA اول و FPGA دوم به ترتیب در جدول (۳-۹) و (۳-۱۰) توضیح داده شده است.

جدول ۳-۹ توضیحات سیگنال‌های FPGA اول

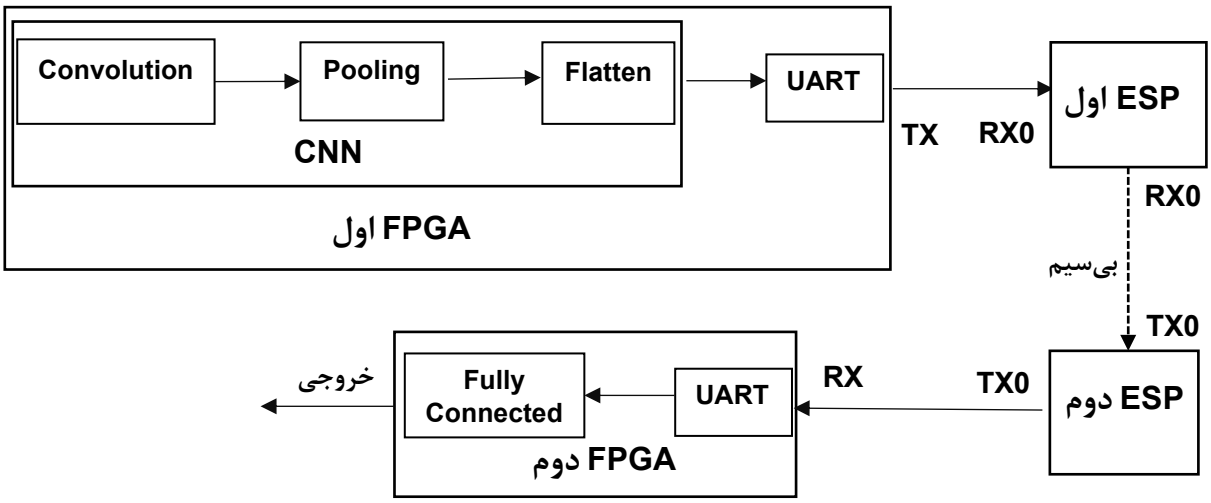
سیگنال	ورودی سیگنال	طول سیگنال	توضیحات
Clock	ورودی	۱	سیگنال ساعت ورودی
Start	ورودی	۱	شروع پردازش داده‌های شبکه عصبی
Reset	ورودی	۱	راه‌اندازی مجدد ماژول
Output11	ورودی	۱	خروجی اول FPGA فرستنده
Output12	خروجی	۱	خروجی دوم FPGA فرستنده
Output13	خروجی	۱	خروجی سوم FPGA فرستنده
Outpu14	خروجی	۸	خروجی چهارم FPGA فرستنده
Valid_Data_1	خروجی	۱	بررسی آماده بودن داده‌ها برای ارسال
EnableData	خروجی	۱	فعال‌سازی جریان داده‌ها
TX	خروجی	۱	کانال سریال TX
RX	ورودی	۱	کانال سریال RX

جدول ۳-۱۰ توضیحات سیگنال‌های FPGA دوم

سیگنال	ورودی سیگنال	طول سیگنال	توضیحات
Clock	ورودی	۱	سیگنال ساعت ورودی
RX	خروجی	۱	کانال سریال RX
TX	ورودی	۱	کانال سریال TX
Reset	خروجی	۱	راه‌اندازی مجدد عملیات ماژول
Output21	خروجی	۱	خروجی اول FPGA گیرنده
Output22	خروجی	۱	خروجی دوم FPGA گیرنده
Valid_Data_2	خروجی	۱	آمادگی برای مرحله بعدی پردازش

۳-۵-۲ پیاده‌سازی سیستم برای حالت بی‌سیم

پیاده‌سازی سیستم در حالت بی‌سیم مشابه حالت باسیم است که در بخش۳-۵-۱ شرح داده شد. تفاوت اصلی در نحوه انتقال داده‌های میانی بین دو FPGA است که در اینجا از ارتباط بی‌سیم استفاده شده است. فرآیند کلی سیستم در شکل (۳-۱۷) نشان داده شده است. لازم به ذکر است که پایه‌های پورت‌ها و سیگنال‌ها برای این پیاده‌سازی همان جدول‌های ارائه‌شده در بخش ۳-۵-۱ هستند (جدول‌های ۳-۹ و ۳-۱۰).



شکل ۳-۱۷ معماری دقیق سیستم برای حالت باسیم

فرآیند این سیستم با انجام عملیات ریاضی همانند روش سیمی در دستگاه اول آغاز می‌شود. پس از پایان محاسبات در FPGA اول، به جای ارسال مستقیم نتایج میانی به صورت سیمی به FPGA دوم، این نتایج ابتدا از طریق خط TX ماژول UART به ورودی RX0 اولین ماژول ESP32 منتقل می‌شوند. این ماژول پس از دریافت داده‌ها، آن‌ها را از طریق شبکه بی‌سیم به ماژول ESP32 دوم ارسال می‌کند. ماژول دوم نیز داده‌های دریافتی را از طریق خروجی TX0 خود به ورودی RX ماژول UART متصل به FPGA دوم منتقل می‌کند. این انتقال بی‌سیم امکان برقراری ارتباط بین FPGAها را بدون نیاز به سیم‌کشی مستقیم فراهم می‌آورد. پس از دریافت داده‌ها توسط FPGA دوم، فرآیند پردازش مشابه حالت باسیم ادامه می‌یابد و محاسبات لایه کاملاً متصل انجام می‌شود تا خروجی نهایی تولید گردد.

۳-۶- خلاصه

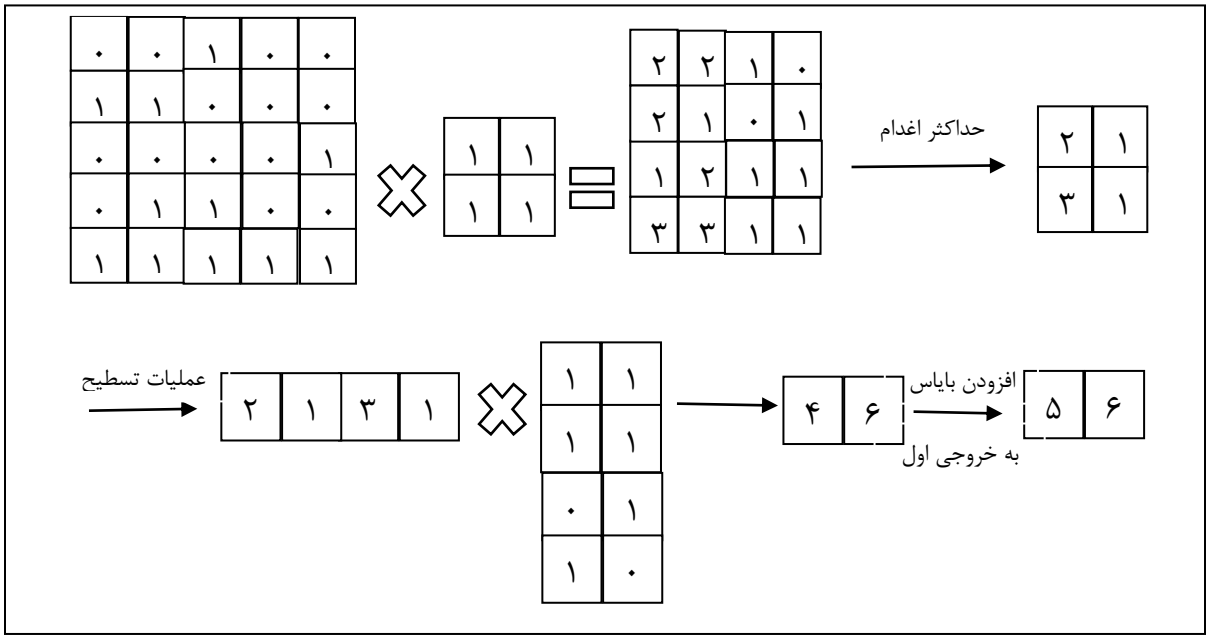
در این بخش، نحوه‌ی پیاده‌سازی پروژه مورد بحث قرار گرفت. ابتدا ساختار کلی سیستم بررسی شد، سپس ساختار ماژول‌های شبکه عصبی پیچشی و UART توضیح داده شد. در ادامه، نحوه‌ی تعامل میان دو ESP32 تشریح گردید. در پایان، معماری و پیاده‌سازی سیستم در دو حالت باسیم و بی‌سیم تشریح شد.

فصل چهارم : نتایج پیاده‌سازی

در این قسمت تصاویر پیاده‌سازی پروژه، نتایج حاصل شده و مقایسه دو حالت پروژه بررسی می‌گردد.

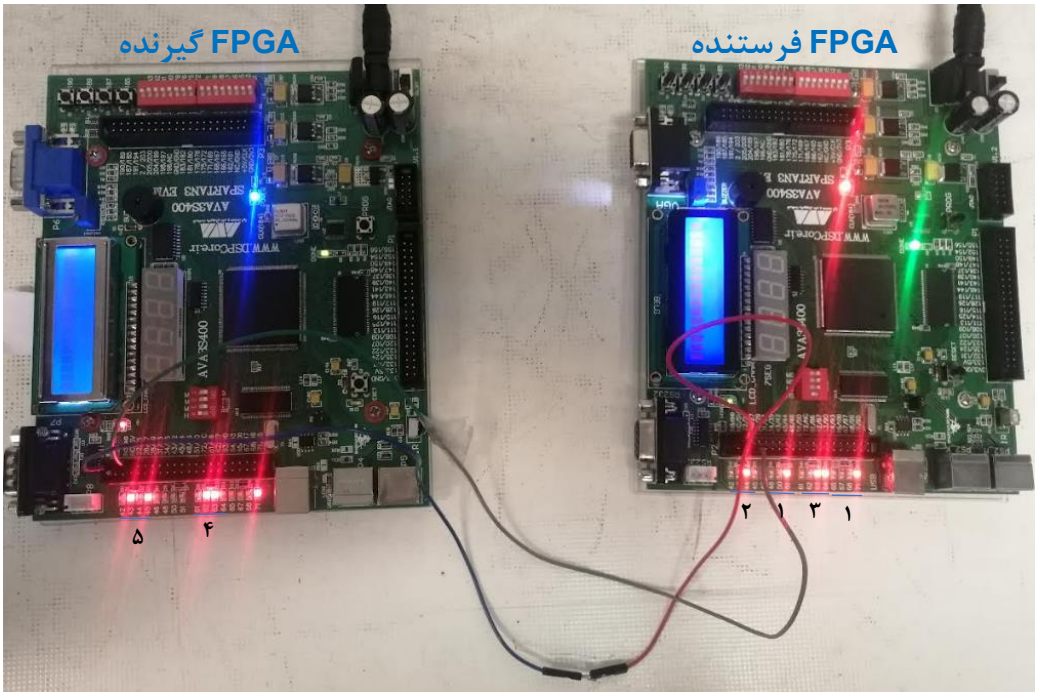
۴-۱- تصاویر پیاده‌سازی و نتایج سخت‌افزاری پژوهش

در این بخش، به بررسی نمونه محاسبات مورد استفاده در پیاده‌سازی سخت‌افزاری پروژه می‌پردازیم. همان‌طور که در فصل سوم توضیح داده شد، ابتدا FPGA فرستنده محاسبات مربوط به لایه‌ی پیچشی، لایه‌ی ادغام، و عملیات تسطیح را انجام می‌دهد و سپس نتایج میانی را به FPGA دریافت‌کننده ارسال می‌کند. FPGA دریافت‌کننده وظیفه‌ی انجام عملیات لایه‌ی کاملاً متصل را بر عهده دارد و خروجی نهایی را تولید می‌کند. در شکل (۴-۱) محاسابات و مقادیر استفاده شده مشاهده می‌شود.

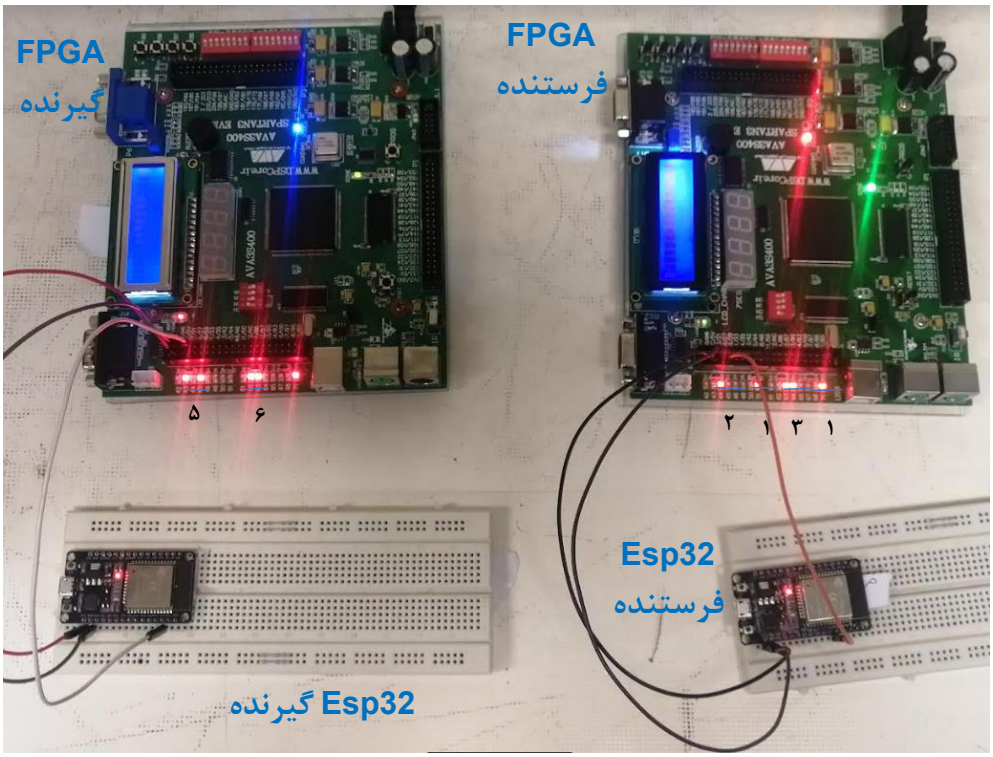


شکل ۴-۱ نمونه محاسبات استفاده شده در پیاده‌سازی سخت‌افزاری پروژه

با توجه به شکل (۴-۱)، انتظار می‌رود که FPGA اول چهار عدد خروجی و FPGA دوم دو عدد خروجی تولید کنند. در ادامه، نتایج پیاده‌سازی بر روی بردهای FPGA برای حالت‌های باسیم و بی‌سیم به ترتیب در شکل‌های (۴-۲) و (۴-۳) قابل مشاهده است.



شکل ۴-۲ پیاده‌سازی سخت‌افزاری حالت باسیم پروژه



شکل ۴-۳ پیاده‌سازی سخت‌افزاری حالت بی‌سیم پروژه

با توجه به خروجی‌های شکل‌های (۴-۲) و (۴-۳) و تطابق آن‌ها با خروجی‌های شکل (۴-۱)، می‌توان به صحت عملکرد پروژه نتیجه‌گیری کرد.

۴-۲- مقایسه دو حالت پروژه

در این بخش، به مقایسه‌ی دو حالت پیاده‌سازی شده با استفاده از ویژگی‌ها و پارامترهای مختلف می‌پردازیم. مقایسه‌ی بین این دو حالت سیستم در جدول (۴-۱) ارائه شده است.

سرعت سیستم و مدت زمان انتقال با استفاده از فرمول‌های زیر محاسبه شده است:

$$Baud = \frac{CF}{8(UBRRn+1)} \qquad Transmission\ time = \frac{1}{Baud} \tag{۱-۴}$$

سرعت سیستم و مدت زمان انتقال به ترتیب از رابطه‌های نرخ بادی و زمان انتقال به دست می‌آید. در این رابطه‌ها، CF به معنی فرکانس ساعت سیستم و UBRRn نشان‌دهنده مقدار ثبت‌شده در رجیستر برای تنظیم نرخ بادی است.

جدول ۴-۱ مقایسه ویژگی‌های دو حالت پیاده‌سازی شده

ویژگی	ارتباط باسیم	ارتباط بی‌سیم
تعداد سیم استفاده شده	دو عدد سیم	شش عدد سیم
هزینه	پایین‌تر، زیرا فقط کابل‌ها و اتصالات مورد نیاز است.	بالاتر، با توجه به هزینه ماژول‌های ESP32 و بردبورد[1]
مقیاس‌پذیری	محدود به تعداد پورت‌های UART و اتصالات فیزیکی موجود	مقیاس پذیرتر، اضافه کردن دستگاه‌های بیشتر بدون نیاز به سیم‌کشی اضافی آسان‌تر است.
سرعت	۷۶،۹۲۳ bps	۳۸،۴۶۱.۵ bps
مدت زمان انتقال	۲۰۸ µs	۴۱۶ µs
مصرف برق	پایین‌تر	بالاتر، به دلیل برق اضافی مورد نیاز برای ارتباطات بی‌سیم
ایمنی در برابر نویز	مصونیت بالا در برابر تداخل	ایمنی کمتر در برابر تداخل
محدوده[2]	محدود به طول سیم	دامنه‌ی گسترده‌تر
قابلیت اطمینان	قابلیت اطمینان بسیار بالا، تضمین نتایج دقیق بدون از دست دادن اطلاعات	احتمال از دست دادن داده به علت وجود نویز
سهولت اجرا	اتصالات ساده و پایدار و مستقیم	تنظیمات پیچیده‌تری مورد نیاز است، به‌ویژه در مدیریت اتصالات بی‌سیم و همگام‌سازی پردازش شبکه عصبی پیچشی در سراسر FPGA
راه‌اندازی و نگهداری	راه‌اندازی و نگهداری آسان‌تر، مشکلات کمتری در حین اجرا	چالش‌برانگیزتر، نیاز به عیب‌یابی اتصال بی‌سیم و همگام‌سازی بین فرآیندهای شبکه عصبی پیچشی در هر دو FPGA
امنیت	ایمن‌تر با حداقل خطر، زیرا نیاز به دسترسی فیزیکی به سیم دارد.	خطرات امنیتی بالقوه ناشی از ارتباطات بی‌سیم مشاهده شد، اگرچه رمزگذاری برخی از این نگرانی‌ها را کاهش داد.
محدودیت‌های فیزیکی	فضای فیزیکی لازم برای کابل‌کشی میان FPGAها	بدون نیاز به اتصال فیزیکی بین FPGAها، اما همچنان به سیم‌کشی بین ESP32 و FPGA نیاز است.

با توجه به توضیحات جدول فوق، می‌توان نتیجه گرفت که حالت باسیم از عملکرد بهتری برخوردار است.

۴-۳- خلاصه

در این فصل، ابتدا نمونه‌هایی از محاسبات به‌کاررفته در پیاده‌سازی پروژه ارائه شد. سپس، صحت عملکرد سخت‌افزاری پروژه در دو حالت باسیم و بی‌سیم مورد بررسی قرار گرفته است. در نهایت، ویژگی‌های هر دو حالت سیستم با یکدیگر مقایسه شده و نتیجه‌گیری می‌شود که حالت باسیم از عملکرد بهتری برخوردار بوده است.

[1] Breadboard
[2] Range

فصل پنجم: نتیجه‌گیری و پیشنهادها

در آخرین فصل از این پروژه، به جمع‌بندی فصول گذشته و نتیجه‌گیری می‌پردازیم. سپس در انتها پیشنهادات برای کارهای آتی مطرح می‌گردد.

۵-۱- جمع‌بندی و نتیجه‌گیری

در این پروژه بر پیاده‌سازی و مقایسه ارتباط باسیم و بی‌سیم بین دو FPGA با استفاده از چارچوب شبکه عصبی پیچشی تمرکز شد. هدف اصلی ارزیابی عملکرد، قابلیت اطمینان و عملی بودن هر دو روش ارتباطی در یک محیط FPGA در دنیای واقعی، با هدف تعیین مناسب‌ترین رویکرد برای کاربردهای با کارایی بالا بود.

در ابتدا با ایجاد یک پایه نظری قوی، کاوش در مفاهیم اساسی مانند معماری شبکه عصبی پیچشی، توابع لایه، و محاسبات پیچیده درگیر در عملیات شبکه عصبی صحبت کردیم. سپس، پروتکل UART را برای پیاده‌سازی ارتباطات پروژه بررسی کردیم. درک این مفاهیم اساسی برای مراحل بعدی پیاده‌سازی بسیار مهم بود و زمینه لازم را برای مقایسه روش‌های ارتباطی فراهم کرد.

سپس به طراحی دقیق و توسعه ماژول های سیستم، از جمله توضیح عمیق هر جزء و معماری کلی سیستم پرداختیم. این فرآیند شامل طراحی ماژول‌های شبکه عصبی پیچشی بر روی FPGA، تشریح ماژول ارتباطی UART و مدیریت سیگنال‌های کلیدی بود که ارتباط بین دو FPGA را تسهیل می‌کرد. این تفکیک دقیق برای حصول اطمینان از اینکه سیستم به طور موثر عمل می‌کند و درک روشنی از نحوه تعامل هر جزء برای دستیابی به ارتباط یکپارچه ارائه می دهد، حیاتی بود.

در مرحله آخر، روش‌های ارتباط سیمی و بی‌سیم را با استفاده از یک مثال عملی برای نمایش عملکرد شبکه عصبی پیچشی در FPGA نشان دادیم. تجزیه و تحلیل مقایسه‌ای نشان داد که روش ارتباط سیمی از روش بی‌سیم بهتر عمل می‌کند. اگرچه رویکرد بی‌سیم انعطاف‌پذیری را ارائه می‌دهد، اما با چالش‌هایی با نرخ انتقال داده و پایداری سیگنال، به‌ویژه در محیط‌هایی با تداخل بالقوه مواجه می‌شود. این پروژه بر اهمیت انتخاب روش ارتباطی مناسب بر اساس نیازها و محدودیت‌های خاص برنامه تأکید می‌کند و بر مبادلات بین عملکرد و انعطاف‌پذیری که مهندسان باید در طراحی سیستم‌های مبتنی بر FPGA در نظر بگیرند، تأکید می‌کند.

۵-۲- پیشنهادات

در این بخش به ایده‌هایی می‌پردازیم که می‌توانند موضوع این پروژه را در آینده گسترش دهند و به نتایج بهتری منجر شوند. هدف از این پیشنهادات، بهبود عملکرد پروژه و ارتقای کاربردهای عملی آن است. یکی از اقدامات مهمی که برای بهبود این پروژه می‌توان انجام داد، استفاده از مدل‌های شبکه عصبی پیچشی پیشرفته‌تری مانند

LeNet5 است. این مدل، به دلیل ساختار پیچیده‌تر و تعداد لایه‌های بیشتر، گزینه‌ای مناسبی برای گسترش این پژوهش خواهد بود. بهره‌گیری از LeNet5 می‌تواند موجب توزیع متوازن‌تر حجم محاسبات میان دو FPGA شود و بار کاری هر دو دستگاه را به طور مؤثرتری افزایش دهد. علاوه بر این، به کارگیری این مدل می‌تواند دقت و کارایی سیستم را به شکل قابل‌توجهی بهبود بخشد.

همچنین، بررسی استفاده از پروتکل‌های ارتباطی دیگری مانند SPI[1] و I2C[2] برای پیاده‌سازی این پژوهش می‌تواند به‌طور جدی مورد توجه قرار گیرد. این پروتکل‌ها با توجه به ویژگی‌های خاص خود، می‌توانند گزینه‌های مناسبی برای انتقال داده‌ها در میان اجزای مختلف سیستم باشند. ارزیابی دقیق‌تر این پروتکل‌ها می‌تواند به انتخاب بهینه‌تر روش‌های ارتباطی برای پروژه‌های آینده کمک کند و در نهایت منجر به بهبود کارایی سیستم شود.

در نهایت، یکی دیگر از راهکارهای پیشنهادی برای گسترش این پژوهش، افزایش تعداد بردهای FPGA مورد استفاده است. این افزایش به گونه‌ای طراحی شود که محاسبات هر لایه از شبکه عصبی پیچشی به یک برد اختصاص یابد. با این روش، توزیع محاسبات میان بردهای مختلف بهینه‌تر می‌شود و علاوه بر افزایش سرعت پردازش، امکان استفاده از شبکه‌های عصبی پیچیده‌تر و با لایه‌های بیشتر نیز فراهم می‌شود. این اقدام می‌تواند ظرفیت پروژه را برای انجام محاسبات پیچیده‌تر و حجیم‌تر به میزان چشمگیری افزایش دهد.

[1] Serial Peripheral Interface
[2] Inter-Integrated Circuit

منابع و مراجع

[1] Haykin, S. (2009). *Neural Networks and Learning Machines* (3rd ed.). Pearson Education India.

[2] Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic Concepts of Artificial Neural Network (ANN) Modeling and Its Application in Pharmaceutical Research. *Journal of Pharmaceutical and Biomedical Analysis, 22*(5), 717-727.

[3] Bengio, Y., Goodfellow, I., & Courville, A. (2017). *Deep learning* (Vol. 1). MIT Press.

[4] Phung, V. H., & Rhee, E. J. (2019). A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences, 9*(21), 4500.

[5] Aghdam, H. H., & Heravi, E. J. (2017). Guide to Convolutional Neural Networks. *Springer.*

[6] Shyam, R. (2021). Convolutional Neural Network and Its Architectures. *Journal of Computer Technology & Applications, 12*(2), 6-14.

[7] Véstias, M. P. (2021). Convolutional Neural Network. In M. Khosrow-Pour (Ed.), *Encyclopedia of Information Science and Technology* (5th ed., pp. 12-26). IGI Global.

[8] Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A Review of Convolutional Neural Networks in Computer Vision. *Artificial Intelligence Review, 57*(4), 99.

[9] Jayawardana, R., & Bandaranayake, T. S. (2021). Analysis of Optimizing Neural Networks and Artificial Intelligent Models for Guidance, Control, and Navigation Systems. *International Research Journal of Modernization in Engineering, Technology and Science, 3*(3), 743-759.

[10] Yani, M., Budhi Irawan, S., & Setiningsih, S. C. (2019). Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. In *Journal of Physics: Conference Series* (Vol. 1235, No. 1, p. 012001). IOP Publishing.

[11] Guissous, A. E. (2019). Skin Lesion Classification Using Deep Neural Network. *arXiv preprint arXiv:1911.07817*.

[12] Mishra, V., & Kane, L. (2023). A Survey of Designing Convolutional Neural Network Using Evolutionary Algorithms. *Artificial Intelligence Review, 56*(6), 5095-5132.

[13] Lambert, T. R. (2017). An Introduction to Microcontrollers and Embedded Systems. *Auburn University. July*, *344*.

[14] Gupta, A., & Gupta, A. (2019). UART Communication. In *The IoT hacker's handbook: A practical guide to hacking the Internet of Things* (pp. 59-80).

[15] Ramdeane, A., & Lynch, L. (2020). Low-Cost Seismic Data Acquisition System Based on Open-Source Hardware and Software Tools. *The University of the West indies*.