عنوان پروژه

دانشجو: علی شایان پور

استاد راهنما: دکتر زین‌العابدین نوابی

دانشکده مهندسی برق و کامپیوتر، دانشگاه تهران

مقدمه

امروزه با توسعه فراگیر و سریع هوش مصنوعی و پیچیدگی مدل‌ها، کتابخانه‌ها و روش‌های زیادی برای ماژولار کردن و ساده‌سازی عملیات‌های هوش مصنوعی به وجود آمده است. این ساده‌سازی‌ها در زبان‌های سطح بالا مانند پایتون در نهایت منجر به از دست دادن سرعت اجرا در محاسبات می‌شود. هر چند کتابخانه‌ها با انجام موازی‌سازی‌هایی در سطح سیستم، باعث تسریع این فرآیند شده‌اند اما این سرعت باز هم برای مدل‌های بسیار پیچیده و مخصوصا برای پردازش‌های Real Time، بسیار پایین است. راهکارهایی که برای این موضوع وجود دارد استفاده حداکثری از پردازش‌های موازی و GPU است اما استفاده از این سخت‌افزارها بسیار هزینه‌بر است.

در این پروژه به پیاده‌سازی سخت‌افزاری شبکه عصبی MLP و CNN پرداخته‌ایم. همچنین به دلیل flexibility زیادی که سخت‌افزار دارد می‌توان با اضافه کردن یا کم کردن واحدهای محاسباتی، به سرعت بیشتری (به بهای هزینه بیشتر) رسید. سعی بر آن است که تمامی پیاده‌سازی‌ها در این پروژه با استفاده از حداقل سخت‌افزار بوده که باعث هزینه کمتر و مصرف توان کمتر می‌شود. از طرفی با وجود کمترین واحدهای محاسباتی همچنان سرعت به طرز چشمگیری بیشتر از نرم‌افزار است.

شرح پروژه

برای بررسی توانایی برد PYNQ-Z2 در بهبود سرعت تشخیص اعداد، دو مدل متداول هوش مصنوعی مورد آزمایش قرار گرفت، که به اختصار شرح داده شده‌اند:

- ❖ Multilayer Perceptron (MLP)
- ❖ Convolutional Neural Network (CNN)

تعداد لایه‌ها و نورون‌های مختلفی برای مدل MLP بررسی شد. در بهترین حالت، مدل مورد بررسی به دقت 97.36% بر روی داده‌های training رسید، در حالی که مشاهده شد بر روی داده‌های test عملکرد خوبی ندارد، که می‌تواند نشانه overfitting باشد.

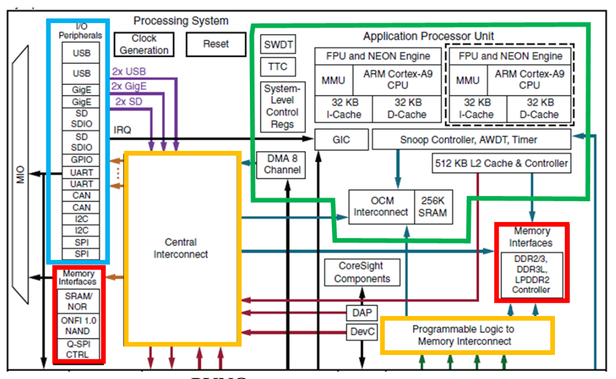
با توجه به تست‌ها و نوع کارکرد شبکه عصبی CNN، این شبکه کاربرد، نتیجه و دقت بسیار بیشتری برای پردازش تصویر و تشخیص اعداد دست‌نوشت MNIST داشتند و برای ادامه کار گزینه مناسب‌تری است.

برای این پروژه، از یک CNN که دو بار بر روی ورودی عملیات کانولوشن و max pooling انجام می‌دهد، و سپس یک لایه‌ی dense و بعد SoftMax روی آن اعمال می‌کند، استفاده شده است. این مدل پس از train شدن به دقت 99.08% بر روی داده‌ّای training رسید، و در عمل نیز نتایج خیلی بهتری بر روی داده‌های واقعی از خود نشان داد.

طراحی بر روی سخت‌افزار

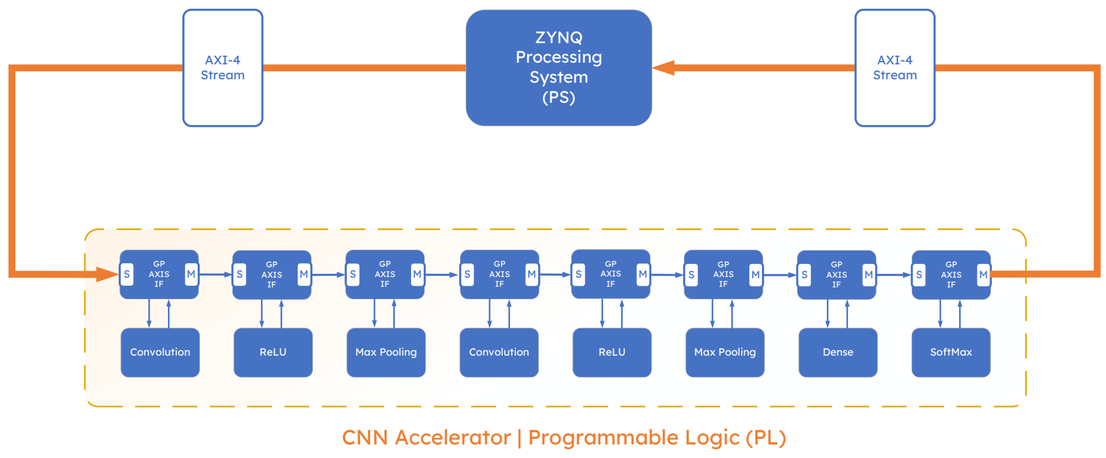
برد PYNQ، که بر پایه‌ی چارچوب Xilinx ZYNQ ساخته شده است، از دو بخش اصلی تشکیل می‌شود:

- Processing System یا PS، که شامل پردازنده‌ی ARM Cortex-A9 دارای سیستم‌عامل Linux است و قابلیت اجرای برنامه‌های به زبان پایتون را دارد. این برنامه‌ها، بخش‌های سخت‌افزاری روی برد PYNQ را مدیریت می‌کنند.
- Programmable Logic یا PL، که قابلیت برنامه‌ریزی به صورت سخت‌افزاری را دارد و مبتنی بر FPGA می‌باشد.



شکل ۴: ساختار درونی برد PYNQ

به طور کلی برای انجام محاسبات سخت‌افزاری از قابلیت‌های مدرن برد PYNQ-Z2 استفاده می‌کنیم. با استفاده از PS هم می‌توان محاسبات را انجام داد و هم می‌توان به صورت کنترلی از آن استفاده کرد و لایه‌ای که میخواهیم روی آن شتاب‌دهی را انجام دهیم، انتخاب کنیم. به صورت کلی داده‌ها از طریق PS دریافت شده و داخل حافظه قرار داده می‌شوند؛ سپس با تنظیم کردن DMA اطلاعات را با پروتکل AXI-4 Stream به سمت شتاب‌دهنده‌های خود ارسال کرده و پس از اتمام پردازش و ارسال جواب توسط آن‌ها داده‌ها را دریافت می‌کنیم. علاوه بر آن می‌توان ماژول‌ها را پشت‌سر هم قرار داد و با استفاده از GP AXIS IF طراحی شده داده‌ها را بدون نیاز به سیگنال کنترلی دیگری به مرحله بعد ارسال کرد و Handshaking به صورت اتومات انجام می‌شود. شکل کلی سیستم در شکل ۵ قابل مشاهده است.



شکل ۵: شکل کلی پیاده‌سازی سخت‌افزاری شبکه عصبی CNN به صورت ترکیبی از کنترل PS و اجرای PL

نتایج

زمان انتقال داده‌ها از Memory به واسطه DMA به یک ماژول AXIS برابر با 1.171ms می‌باشد و زمان دریافت داده‌ها نیز برابر با 1.438 ms می‌باشد. لازم به ذکر است که فرکانس کاری مدار 100 MHz می‌باشد. نتایج بدست آمده نشان می‌دهد که انجام محاسبات زیاد در لایه های مختلف شبکه به صورت سخت‌افزاری نسبت به حالت پیاده‌سازی نرم‌افزاری سرعت بیشتری دارد. با پیاده‌سازی شبکه کانولوشن مدنظر به صورت سخت‌افزاری، در PC با مشخصات Intel Core i7-8750H @ 2.20GHz به اندازه حدود ۷۶ برابر سرعت و در PL به اندازه حدود ۲۰۰۰ برابر افزایش سرعت خواهیم شد؛ در صورتی که صرفا لایه‌های Convolution را به صورت سخت‌افزاری پیاده‌سازی کنیم، در PC به اندازه ۴۰ برابر و در PS به اندازه ۸۶ برابر بهبود سرعت خواهیم داشت.

در مجموع تمامی زمان‌های اجرایی به شرح زیر است:

- کاملا نرم‌افزاری بر روی PC: 1940.8973 میلی‌ثانیه
- کاملا نرم‌افزاری بر روی PS: 51.129 ثانیه
- کاملا سخت‌افزاری بر روی PL: 25.65 میلی‌ثانیه

جدول ۱- زمان‌بندی اجرای لایه‌های مختلف به صورت نرم‌افزاری

Layer	Name	Dimension	PC Time (ms)	PS Time (ms)	Time (ms)
Layer #0	Convolution	in: 784, out: 194688	396.3614	14,135.5	1.95472
Layer #0 activation	ReLU	in: 21632, out: 21632	7.0078	169.7	0.43264
Layer #1	MaxPooling	in: 21632, out: 21632	6.0443	164.6	0.43264
Layer #2	Convolution	in: 5408, out: 2230272	1,520.4479	36,425.4	22.3568
Layer #2 activation	ReLU	in: 7744, out: 7744	2.9997	84.2	0.15488
Layer #3	MaxPooling	in: 7744, out: 6400	2.9995	59.5	0.14144
Layer #4	Flatten	in: 1600, out: 1600	0	1.1	-
Layer #5	Dense	in: 1600, out: 10	5.0367	88.8	0.176
Layer #5 activation	Softmax	in: 10: out: 10	0	0.1	0.005

جمع بندی

در این پروژه به بررسی و مقایسه شبکه‌های عصبی MLP و CNN در کاربرد تشخیص تصویر پرداختیم و ماژول‌ها و لایه‌های مختلف را از لحاظ نرم‌افزاری و سخت‌افزاری مقایسه کردیم.

از نکات مهمی که به آن پی بردیم این بود که چه لایه‌هایی Bottleneck طراحی ما بوده و بهتر است آن را به صورت سخت‌افزاری پیاده‌سازی کرد. از طرفی چه لایه‌ای را می‌توان به صورت نرم‌افزاری پیاده‌سازی کرد تا از پیچیدگی زیاد کاسته شود. به طور خاص در این پروژه با این معماریCNN دیدیم که یکی از Bottleneck های اصلی سیستم، هر دو لایه Convolution بود و زمان بسیار زیادی را صرف محاسبات می‌کرد؛ در طراحی‌های Real Time این موضوع خیلی بیشتر خود را نشان می دهد. یکی از تکنیک های مورد استفاده برای حل این مشکل طراحی و پیاده سازی سخت افزاری لایه زمانبر و قرار دادن آن به صورت Accelerator در کنار واحد CPU می باشد.

در آینده می‌توان علاوه بر قسمت Evaluation، قسمت train کردن را نیز به صورت سخت‌افزاری پیاده‌سازی کرد؛ زیرا یک بخش عظیمی از زمان در train کردن سپری می‌شود و می‌توان از این موضوع به خوبی استفاده کرد و زمان train کردن را فوق‌العاده کاهش داد. همچنین با استفاده از پارامتری بودن تمام طراحی‌های انجام شده می‌توان با تغییر دادن پارامترها و کم و زیاد کردن لایه‌ها و همچنین استفاده از شبکه‌ها و معماری‌های دیگر، زمان اجرا را برای حالت‌های دیگر محاسبه کرد تا این اطلاعات دید خوبی برای طراحان داشته باشد.

مراجع اصلی

1. http://www.pynq.io/
2. https://zynq.io/
3. https://www.xilinx.com/support/university/xup-boards/XUPPYNQ-Z2.html
4. https://pynq.readthedocs.io/en/v2.3/
5. https://github.com/alexforencich/verilog-axis
6. https://www.hackster.io/whitney-knitter/axi4-lite-interface-wrapper-for-custom-rtl-in-vivado-2021-2-8a7009
7. https://www.xilinx.com/video/software/axi4-stream-interfaces.html