بسم الله الرحمن الرحیم

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

تمرین درس هوش مصنوعی در سیستم‌های نهفته – آبان ۱۴۰۲

اهداف

هدف این تمرین، آشنایی با روال پیاده‌سازی و سنتز[۱] شبکه‌های عصبی روی پلتفرم‌هایی با منابع محدود است. این روال شامل پیاده‌سازی سطح بالا (کد پایتان[۲])، فشرده‌سازی مدل و استفاده از نتایج آن برای پیاده‌سازی سطح پایین (وریلاگ[۳]) و در نهایت سنتز با استفاده از ابزارهای سنتز موجود است.

۱- مقدمه

برای دستیابی به هدف تعیین شده، از میان طیف گسترده‌ای از انواع شبکه‌های عصبی موجود، ما یک مثال ساده و کاربردی از شبکه‌ی MLP درنظر گرفته‌ایم. در ادامه توضیح مختصری درباره‌ی این دسته از شبکه‌ها داده می شود و سپس در بخش‌های بعدی نیازمندی‌های لازم برای انجام تمرین ذکر می شود.

۱-۱- شبکه MLP

شبکه عصبی چند لایه پرسپترون[۴] (MLP) یکی از شبکه‌های عصبی مصنوعی پرکاربرد است که در یادگیری ماشین و یادگیری عمیق استفاده می‌شود. شبکه های MLP به خاطر توانایی در مدیریت یک دسته وسیع از وظایف، شامل مسائل دسته‌بندی و رگرسیون، شناخته می‌شوند. به دلیل سبک وزن بودن و محاسبات کم، شبکه‌های MLP، با استفاده از ماژول‌های MAC[۵] به راحتی قابل پیاده‌سازی برروی سخت‌افزار هستند. این مدل‌ها، در کاربردهای زیادی در زمینه‌های بسیار متنوع، از جمله پردازش تصویر مثل تشخیص لبه[۶] تصویر، جهت حل مسائل راه یافته اند.

۱-۲- مدل طبقه بندی

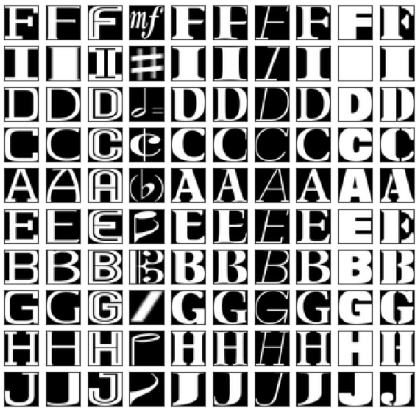
یکی از کاربردهای بسیار مهم و رایج شبکه‌های MLP، طبقه‌بندی یا برچسب‌گذاری ورودی است. این مدل‌ها ابتدا ورودی مورد نظر را دریافت می‌کند، سپس پردازش و تحلیل این داده‌ها را با استفاده از لایه‌های میانی(مخفی) انجام می‌دهد و در نهایت کلاس مرتبط با ورودی داده شده را در خروجی نشان می‌دهد.

۱-۳- دیتاست

در این تمرین ما از یک شبکه عصبی MLP برای طبقه بندی مجموعه داده[۷]های NotMNIST [1] (فقط حروف A، B، C) استفاده می‌کنیم. در شکل 1 نمونه‌هایی از داده های NotMNIST نشان داده شده است.

---

۱ synthesis
۲ Python
۳ Verilog
۴ Multi-layer Perceptron
۵ Multiply And Accumulator
۶ edge
۷ Dataset



شکل ۱- بخشی از مجموعه داده NotMNIST

۴-۱- فشرده‌سازی

تاکنون با شبکه‌های MLP و یکی از کاربردهای آن آشنا شدید. اگرچه این مدل‌ها کوچک هستند و به راحتی با دقت بیتی بالا بر روی پلتفرم‌هایی با منابع محدود قابل پیاده‌سازی هستند، ولی در دنیای واقعی شبکه‌های بزرگی وجود دارند که پیاده‌سازی آن‌ها با دقت بیتی بالا، به دلیل حجم بسیار زیاد پارامترهای موجود، که نیازمند تبادلات حافظه‌ای زیاد و محاسبات سنگین ناشی از آن هستند، بر روی چنین پلتفرم‌هایی امکان‌پذیر نیست یا هزینه بسیار زیادی را تحمیل می‌کند. یکی از ساده‌ترین راه‌کارهای موجود می‌تواند کاهش حجم مدل‌های شبکه عصبی با تغییر پارامترهای شبکه، مثل کاهش ابعاد ورودی باشد. به طوری که حداقل افت صحت یا افزایش خطا رخ دهد. اما این روش نیازمند تغییرات به ازای شبکه‌ها و مدل‌های مختلف است.

یکی دیگر از روش‌های بسیار پرکاربرد و کم خطا، استفاده از کوانتیزاسیون[۸] است. کوانتیزاسیون فرآیندی است که مقادیر پیوسته را به مجموعه محدودی از مقادیر گسسته یا سطوح تقریب می‌دهد یا گرد می‌کند. این فرآیند به طور معمول در پردازش سیگنال دیجیتال و فشرده‌سازی داده‌ها برای نمایش و ذخیره سازی داده‌ها با کارآیی بیشتر استفاده می‌شود. در مدل‌های شبکه عصبی، عموما پارامترهای وزن مدل را از اعداد ممیز شناور[۹] ۳۲ بیت به اعداد ممیز ثابت[۱۰] ۱۶ بیت یا کمتر گرد می‌کنند؛ به عنوان مثال، با تبدیل وزن‌ها از ۳۲ به ۸ بیت ممیز ثابت، حجم مدل تا چهار برابر کاهش می‌یابد، که مقدار قابل توجهی است [2].

۲- پیش نیازهای انجام تمرین

۱. آشنایی اولیه با پایتان و شبکه‌های عصبی
۲. نرم افزار شبیه سازی زبان توصیف سخت افزار مانند Modelsim
۳. ابزار سنتز yosys [۱۱]

۳- مراحل انجام تمرین

در این تمرین هدف ساخت یک طبقه‌بند با حداقل حجم ممکن و حفظ صحت[۱۲] مدل تا حد امکان، و در نهایت شبیه‌سازی آن با زبان توصیف سخت‌افزار (در اینجا وریلاگ) و سنتز آن با استفاده از ابزار yosys است. بنابراین تمرین شامل دو مرحله اصلی است؛ در مرحله اول با توجه به مواردی که در تمرین اول آموزش دیدید، یک مدل شبکه عصبی طبقه‌بند را طراحی خواهید کرد. در مرحله دوم

---

[۸] Quantization
[۹] Floating point
[۱۰] Fixed Point
[۱۱] https://github.com/YosysHQ/yosys
[۱۲] accuracy

نیز پیاده‌سازی مدل نهایی بدست آمده را با زبان توصیف سخت‌افزار وریلاگ انجام می‌دهید، شبیه‌سازی می‌کنید و در نهایت سنتز خواهید کرد.

۱-۳- مرحله اول:

یک فایل کد پایتان در اختیار شما قرار داده شده است:

۱. فایل hw-notmnist.ipynb مربوط به طراحی طبقه‌بند خودکار برای داده‌های NotMNIST.
۲. لینک دیتاست NotMNIST در زیر آمده است:

https://drive.google.com/file/d/1Q6D2L25xSwKdiBBnE8-uQ8cEjGntky8P/view

کد مراحل دریافت و آماده‌سازی داده‌ها در فایل موجود است. داده‌ها از ابتدا بین ۰ و ۱ نرمال شده‌اند لذا نیازی به نرمال‌سازی مجدد نیست.

موارد زیر را انجام دهید:

۱) با استفاده از لایه‌های MLP، مدلی طراحی کنید که سه حرف A، B و C را با دقت حداقل ۹۶ درصد تشخیص دهد؛ یعنی دقت برروی داده‌های تست یا ارزیابی از ۹۶ کمتر نباشد. همچنین نمودار دقت و خطا در طول آموزش را گزارش کنید.

۲) با استفاده از classification_report در sklearn، مقادیر precision، recall و f1-score را برای مدل نهایی گزارش و تحلیل کنید.

۳) حجم مدل را بر حسب کیلوبایت گزارش کنید. می‌توانید تابعی بنویسید که با دریافت اطلاعات یک مدل از ورودی (تعداد نورون‌های هر لایه و تعداد بیت)، حجم نهایی مدل را بر حسب KB، با توجه به ماتریس وزن هر لایه، محاسبه کند.

۴) ابعاد تصویر را تا حد امکان کوچک کرده (با استفاده از تابع resize_images) و سپس مدل جدیدی برای آن طراحی کرده و آموزش دهید. حداکثر ۴ لایه استفاده کنید (با احتساب لایه طبقه‌بند)؛ همچنین حداکثر افت دقت مجاز ۳ درصد است. درواقع باید با تغییر انداز ورودی، از خروجی دقت مدل تشخیص دهید که تا چه اندازه کاهش ابعاد مناسب است (ممکن است چندین بار نیاز به تغییر پارامترها و لایه ها داشته باشید).

۵) برای مدل نهایی مرحله ۴، نمودار دقت و خطا در طول آموزش را گزارش کنید. همچنین precision، recall و f1-score را نیز گزارش و با مدل قبل مقایسه کنید.

۶) نمودار هیستوگرام یکی از نمودارهای مفیدی است که می‌تواند بازه اعداد وزن‌ها و تعداد تکرار هریک را برای هر لایه یا ورودی‌ها نشان دهد و به تعیین تعداد بیت مورد نیاز برای کوانتایز نمودن مدل کمک کند. نمودار هیستوگرام برای وزن‌های هر لایه را گزارش کنید. به نظر شما بهترین تعداد بیتی که می‌توان برای کوانتایز کردن وزن ها **بدون افت دقت** در نظر گرفت، به ازای هر لایه چند بیت است؟ (می‌تواند ترکیبی باشد) برای ورودی ها و خروجی‌ها چطور؟ بر این اساس شبکه را کوانتایز کنید و آموزش دهید.

۷) اگر در جوابی که برای مرحله ۶ بدست آورده اید، همه لایه‌ها ۸ بیتی هستند، از این سوال گذر کنید. مدل را با ۸ بیت کوانتایز کنید (ورودی و خروجی را می توانید دلخواه درنظر بگیرید) و آموزش دهید. نمودار دقت و خطا در طول آموزش را گزارش کنید. همچنین precision، recall و f1-score را نیز گزارش و با مدل قبل مقایسه کنید.

۸) مرحله ۷ را با ۶ بیت تکرار کنید. سپس با مدل ۸ بیتی و مدل اصلی مرحله ۴ مقایسه کنید.

۹) مرحله ۷ را با ۴ بیت نیز تکرار کنید. سپس با مدل ۸ بیتی، ۶ بیتی و مدل اصلی مرحله ۴ مقایسه کنید.

۱۰) یک نمودار میله‌ای رسم کنید که دقت و حجم (بر حسب MB) هریک از مدل‌های بدست آمده از ابتدا تا کنون را با یکدیگر مقایسه کند.

۱۱) مدل اصلی نسبت به بهترین و فشرده‌ترین مدل بدست آمده چند برابر است؟ (در صورتی که دقت مدلی زیر ۹۱ درصد است، از آن صرف نظر کنید)

۱۲) یک نمونه از ورودی‌های کوانتایز شده و همه وزن‌های کوانتایز شده از مدل ۸ بیت و ۶ بیت را در فایل‌های mem داده شده ذخیره کنید. اگر ورودی مدل بیش از ۱۰۰ بعد دارد (تصاویر با ابعاد ۱۰×۱۰ یا بیشتر)، ابتدا ورودی را به لایه اول مدل بدهید (با احتساب لایه activation پس از آن) و خروجی آن را در فایل input.txt ذخیره کنید؛ در این صورت مدل را از لایه ۲ به بعد در مرحله دو تمرین پیاده سازی خواهید کرد.

۲-۳- مرحله دوم:

در این قسمت با نحوه پیاده‌سازی و سنتز مدل بدست آمده از مرحله قبل آشنا می‌شوید. دو فایل وریلاگ با اسم node.v و weights_memory.v به شما داده شده است. این فایل پیاده‌سازی یک نورون از لایه MLP است که درواقع عملیات MAC را انجام می‌دهد؛ یعنی هریک از ورودی‌های آن نورون را در وزن متناظر خود ضرب می کند و در نهایت با مقدار بایاس جمع می‌کند. برای سادگی، زمان انجام عملیات یک clock cycle در نظر گرفته شده است. بر اساس فرض این تمرین، در ادامه ماژول‌های relu و clip و layer را تعریف خواهید کرد، به طوری که هر لایه شامل node، relu و clip (هر کدام در یک clock cycle) است. بنابراین خروجی هر لایه پس از سه clock cycle آماده خواهد بود. توجه کنید که بیت علامت باید جدا در نظر گرفته شود، به عنوان مثال عدد ۸ بیتی با ۷ بیت اعشاری، درواقع ۹ بیت خواهد بود. همه ماژول‌ها را با وریلاگ طراحی کنید.

توضیح ورودی‌ها و خروجی‌های ماژول node:

پارامتر LAYERID: شماره لایه‌ای که نورون در آن قرار دارد

پارامتر NODEID: شماره نورون در یک لایه

پارامتر WIDTH: عرض بیت هر یک از ورودی های نورون

پارامتر INPUT_NUM: تعداد ورودی های نورون

پارامتر OUTPUT_WIDTH: عرض بیت خروجی نورون

پارامتر FRACTION: تعداد بیت های اعشاری خروجی

clk: سیگنال یک بیتی نشان دهنده وضعیت clock

in_ready: ورودی یک بیتی نشان دهنده آماده بودن ورودی ماژول

X: مجموعه‌ی ورودی‌های نورون که در INPUT_NUM * WIDTH بیت قرار گرفته اند

out: خروجی ماژول با عرض بیت OUTPUT_WIDTH بیت

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی

۱) **ماژول relu**: همانطور که در مدل دیدید، از تابع فعال‌ساز relu برای هر لایه استفاده شده است. این تابع فقط ورودی‌های مثبت را فیلتر می‌کند؛ به طوری که اگر خروجی منفی باشد، صفر بازگردانده شده و در غیر اینصورت همان عدد در خروجی ظاهر می‌شود. ماژولی به اسم ReLu طراحی کنید که در یک clock cycle عملیات مورد نظر را انجام دهد. این ماژول باید شامل ورودی‌ها و خروجی‌های زیر باشد:

clk: سیگنال یک بیتی نشان دهنده وضعیت clock

valid: ورودی یک بیتی نشان‌دهنده معتبر بودن ورودی

x: ورودی اصلی به ماژول

out: خروجی ماژول

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی ماژول

۲) **ماژول clip**: این ماژول ورودی w بیتی را دریافت می کند و خروجی out_w بیتی را برمی‌گرداند؛ بدین صورت که ورودی را با بیشترین و کمترین عدد در out_w بیتی مقایسه می کند، اگر بیشتر از این بازه باشد، بیشترین عدد آن، اگر کمتر باشد، کمترین عدد آن و در غیر اینصورت خود ورودی را برمیگرداند.

راهنمایی: در صورتی که ورودی در بین بازه مورد نظر با out_w بیت باشد، باید خروجی به صورت زیر انتخاب شود (عدد ۱۷ بیتی زیر با ۱۴ بیت اعشار را در نظر بگیرید):

00110110001011001

Integer part: **0**01

Fraction part: 10110001011001

در صورتی که خروجی مورد نظر ۹ بیتی با ۷ بیت اعشار باشد، از بخش صحیح بیت اول از راست به همراه بیت علامت آن را جدا می کنیم و برای بخش صحیح خروجی قرار می‌دهیم. برای بخش اعشاری نیز از سمت چپ به تعداد مورد نیاز یعنی ۷ بیت را جدا می کنیم و به عنوان بخش اعشاری خروجی قرار میدهیم:

01 1011000

این ماژول باید شامل ورودی‌ها و خروجی‌های زیر باشد:

clk: یک ورودی یک بیتی clock

valid: ورودی یک بیتی نشان‌دهنده معتبر بودن ورودی

x: ورودی اصلی به ماژول

out: خروجی ماژول

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی ماژول

۳) **ماژول layer**: در این ماژول به تعداد لازم از نورون، ReLu و clip نمونه‌سازی می‌شود. این کار را با استفاده از generate در verilog و یک حلقه for انجام دهید. این ماژول باید شامل ورودی‌ها و خروجی‌های زیر باشد:

clk: یک ورودی یک بیتی clock

valid: ورودی یک بیتی نشان‌دهنده معتبر بودن ورودی

x: ورودی اصلی به ماژول (دقیقاً مشابه ورودی به هر نورون)

out: خروجی ماژول که مانند ورودی هر نورون که عرض آن OUTPUT_NUM * OUT_WIDTH بیتی باشد.

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی ماژول

۴) **ماژول max_index**: این ماژول، تعدادی ورودی را دریافت می‌کند (از لایه آخر) و سپس اندیس بیشترین عدد را در خروجی می‌دهد. این ماژول می‌تواند شامل ورودی‌ها و خروجی‌های زیر باشد:

clk: یک ورودی یک بیتی clock

valid: ورودی یک بیتی نشان‌دهنده معتبر بودن ورودی

x: ورودی اصلی به ماژول (دقیقاً مشابه ورودی به هر نورون)

out: خروجی ماژول که یک آرایه از خروجی‌ها به ازای هر نورون است.

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی ماژول

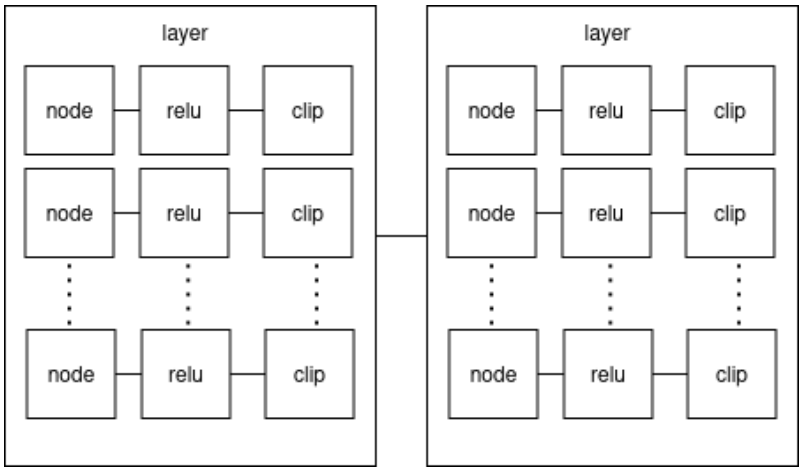
۵) در نهایت در یک فایل model.v، به تعداد لایه‌های طبقه‌بند نهایی که وزن‌های آن را ذخیره کردید، از ماژول layer نمونه‌سازی کنید و با متصل کردن ورودی و خروجی‌های آن‌ها به یکدیگر به شکل مناسب، خروجی ها را تولید کنید.

۶) یک فایل به نام testbench.sv ایجاد کنید و در آن از ماژول model یک نمونه بسازید. در این ماژول باید در بخش initial فایل ورودی ذخیره شده از مرحله اول تمرین را بخوانید و به عنوان ورودی لایه اول (یا لایه دوم، همانطور که در سوال ۱۲ مرحله اول توضیح داده شد) در رجیستر مناسب ذخیره کنید.

۷) نتایج شبیه‌سازی را یک بار برای مدل ۸ بیتی و یک بار برای مدل ۶ بیتی، گزارش کرده و نتایج را توضیح دهید. با تست سه ورودی نشان دهید که مدل طبقه‌بند سخت‌افزاری کلاس (برچسب) مورد نظر ورودی را همانند خروجی مدل مرحله اول

برچسب گذاری می‌کند (حتی اگر خروجی را به اشتباه پیش‌بینی می‌کنند، هردو مدل باید همان اشتباه مشابه را در خروجی نمایش دهند).
شکل 2 شمایی از معماری که باید پیاده‌سازی شود را نشان می‌دهد.

۸) با استفاده از ابزار yosys، هر دو مدل خود را سنتز کنید، این کار را با هر سه دستور synth انجام دهید و در نهایت با دستور stat، خروجی مربوط به تعداد cell ها و تعداد سیم‌ها را برای هر یک را گزارش و با یکدیگر مقایسه کنید.
نمره اضافی: برای هر یک از مدل‌ها، دستورهای synth_xilinx و synth_intel را نیز امتحان و سپس نتایج آن‌ها را با یکدیگر مقایسه کنید؛ یعنی موارد موجود در خروجی stat را باید توضیح دهید و تحلیل کنید (تعداد DFF، تعداد LUT و ...).



شکل ۲- شمای کلی معماری مورد نیاز

لازم است موارد زیر جهت تحویل تمرین و ارائه‌ی گزارش رعایت شوند:

- گزارش خود را در بخش‌های مجزا شامل چکیده، نحوه‌ی انجام کار، نتایج به دست آمده، تحلیل نتایج، نتیجه‌گیری و ضمائم بیاورید. فایل گزارش باید بر اساس فرمت قرار داده شده در سایت درس باشد.
- در پیاده‌سازی های سخت‌افزاری، لازم است همه ماژول‌ها را تا حد امکان پارامتری بنویسید (به عنوان مثال تعداد نورون‌های ورودی لایه، عرض بیت و ... پارامتر باشند).
- درصورت استفاده از تکنیک‌های اضافه برای فشرده‌سازی، در گزارش توضیح دهید.
- فایل گزارش به صورت doc باشد. کد خود را نیز آپلود کنید.
- تمرین را با فرمت YourName_StudentNo_EAI2.rar آپلود کنید.
- گروه ها حتما دو نفره باشند.
- بارگذاری فایل‌های گزارش توسط یکی از اعضای گروه کافی است.
- نمره از 100 محاسبه می شود و به ازای هر روز تاخیر در اپلود تمرین، به اندازه 2^{x} (x تعداد روز تاخیر) از نمره شما کسر می‌شود.
- در صورت مشاهده تشابه زیاد در کدها و گزارش، نمره 100- برای هر دو گروه اعمال خواهد شد.
- تمرین تحویل حضوری دارد که زمان آن بعدا اعلام خواهد شد.

۴- مراجع

[1] Available online at: https://www.kaggle.com/datasets/jwjohnson314/notmnist

[2] Krishnamoorthi, R., 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.