



دانشگاه صنعتي امیرکبیر  
 (دانشكده گرمسار)

پایان‌نامه کارشناسی ارشد

مهندسی کامپیوتر گرایش معماری سیستم‌های کامپیوتری

طراحي و پياده‌سازي تسريع‌کننده شبکه‌هاي عصبي پيچشي مبتني بر معماري شبکه روي تراشه سه‌بعدی

نگارنده

محمد عباس‌زاده

استاد راهنما

دکتر حمیدرضا زرندی

تیرماه 1402

صفحه فرم ارزیابی و تصویب پایان­نامه - فرم تأیید اعضاء كميته دفاع

اينجانب محمد عباس‌زاده متعهد مي‌شوم كه مطالب مندرج در اين پایان‌نامه حاصل كار پژوهشي اينجانب تحت نظارت و راهنمايي اساتيد دانشگاه صنعتي اميركبير بوده و به دستاوردهاي ديگران كه در اين پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذكر گرديده است. اين پایان‌نامه قبلاً براي احراز هيچ مدرك هم‌سطح يا بالاتر ارائه نگرديده است.

در صورت اثبات تخلف در هر زمان، مدرك تحصيلي صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پيگيري قانوني خواهد داشت.

كليه نتايج و حقوق حاصل از اين پایان‌نامه متعلق به دانشگاه صنعتي اميركبير مي‌باشد. هرگونه استفاده از نتايج علمي و عملي، واگذاري اطلاعات به ديگران يا چاپ و تكثير، نسخه‌برداري، ترجمه و اقتباس از اين پایان‌نامه بدون موافقت كتبي دانشگاه صنعتي اميركبير ممنوع است.   
نقل مطالب با ذكر مآخذ بلامانع است.

محمد عباس‌زاده



تقدیم به

پدر و مادر عزیزم

که در تمامی مراحل زندگی یار و یاور راهنمایم بوده‌اند و همواره وجودشان مایه افتخارم است.

با سپاس

ازاستاد گرامی آقای دکتر حمیدرضا زرندی

که با راهنمایی‌هایی ارزشمند خود با وجود خوش‌رویی مرا در انجام این پایان‌نامه یاری نمودند.

# چكيده

امروزه شبکه های عصبی با الهام گرفتن از شبکه بیولوژیکی مغز انسان پیشرفته ترین و دقیق ترین الگوریتم ها را برای بسیاری از برنامه های کاربردی هوش مصنوعی ارائه می دهد. شبکه عصبی پیچشی (CNN[[1]](#footnote-1)) عملکرد فوق العاده ای در بسیاری از زمینه ها، مانند تشخیص و دسته بندی اشیا، بینایی ماشین، کنترل و بهینه سازی به دست آورده است. اگرچه شبکه های عصبی پیچشی مزایای بسیاری را برای هوش مصنوعی به ارمغان می آورد، اما زمان محاسبات طولانی تر در کاربرد های بی درنگ و قدرت محاسباتی بیشتر همچنان عملکرد سیستم را به طور قابل توجهی محدود می کند. بنابراین، طراحی شتاب‌دهنده سخت افزاری کارآمد در سال‌های اخیر بسیار مورد توجه قرار گرفته است. با این حال، با افزایش اندازه شبکه عصبی به دلیل محاسبات پیچیده و ارتباطات زیاد بین نورون ها و لایه های شبکه نیازمند یک شبکه منعطف و مقیاس پذیر مثل شبک ههای روی تراشه .از سوی دیگر، اتصال شبکه روی تراشه (NoC) برای حل مشکل ارتباطی پیچیده پیشنهاد شده است، که یک اتصال متقابل جذاب برای ساخت طراحی سخت افزاری کارآمد CNN است.

واژه‌های کلیدی:

شبکه روی تراشه، تسریع‌کننده، شبکه عصبی پیچشی، شبکه روی تراشه سه‌بعدی، شبکه عصبی

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

[فصل 1: مقدمه‌ 1](#_Toc98131323)

[فصل 2: شبکه عصبی و شتاب‌دهنده‌های مبتنی بر پردازش درون حافظه ReRAM 7](#_Toc98131324)

[1-2- یادگیری ماشین و شبکه عصبی 8](#_Toc98131325)

[2-1-2- آموزش و استنتاج 10](#_Toc98131326)

[3-1-2- شبکه عصبی پیچشی عمیق 10](#_Toc98131327)

[4-1-2- محاسبات شبکه عصبی پیچشی 12](#_Toc98131328)

[1-1-2- کاربردهای شبکه عصبی عمیق و چالش‌های موجود 13](#_Toc98131329)

[2-2- شتاب‌دهنده مبتنی بر پردازش درون حافظه ReRAM 14](#_Toc98131330)

[2-2-2- ساختار حافظه ReRAM 15](#_Toc98131331)

[3-2-2- واحد محاسباتی مورد نیاز شبکه عصبی در آرایه متقاطع ReRAM 17](#_Toc98131332)

[4-2-2- تفاوت حافظه ReRAM غیرفرار و حافظه ReRAM محاسباتی 18](#_Toc98131333)

[5-2-2- ساختار شتاب‌دهنده شبکه عصبی عمیق مبتنی بر ReRAM 19](#_Toc98131334)

[6-2-2- چالش‌های شتاب‌دهنده‌های مبتنی بر ReRAM 23](#_Toc98131335)

[3-2- دقت و قابلیت اطمینان در شتاب‌دهنده شبکه عصبی عمیق 25](#_Toc98131336)

[2-3-2- اشکال‌ها در سلول ReRAM 28](#_Toc98131337)

[3-3-2- اشکال‌ها در مدل نورون 31](#_Toc98131338)

[4-3-2- تحمل‌پذیری اشکال 32](#_Toc98131339)

[5-3-2- اهمیت تحمل‌کردن اشکال CNN و فاز استنتاج 33](#_Toc98131340)

[4-2- جمع‌بندی 35](#_Toc98131341)

[فصل 3: پژوهش‌های پیشین: پردازش درون حافظه و قابلیت اطمینان 37](#_Toc98131342)

[1-3- پردازش درون حافظه 39](#_Toc98131343)

[2-3- معماری شتاب‌دهنده‌های شبکه عصبی عمیق مبتنی بر ReRAM 40](#_Toc98131344)

[1-2-3- معماری‌های فاز استنتاج 41](#_Toc98131345)

[2-2-3- معماری‌های فاز آموزش 42](#_Toc98131346)

[3-2-3- معماری‌های قابل بازپیکربندی 46](#_Toc98131347)

[3-3- افزایش دقت و بهبود قابلیت اطمینان 50](#_Toc98131348)

[4-3- تحمل‌پذیری اشکال در شبکه‌های عصبی 50](#_Toc98131349)

[5-3- تشخیص و تحمل‌پذیری اشکال در آرایه متقاطع ReRAM 54](#_Toc98131350)

[1-5-3- تشخیص خطا در آرایه متقاطع 55](#_Toc98131351)

[2-5-3- تحمل‌پذیری اشکال در شتاب‌دهنده مبتنی بر ReRAM 56](#_Toc98131352)

[3-5-3- چالش‌های روش‌های پیشین تحمل‌پذیر اشکال آرایه متقاطع ReRAM 61](#_Toc98131353)

[6-3- چارچوب و شبیه‌سازهای شتاب‌دهنده‌های مبتنی بر ReRAM 62](#_Toc98131354)

[7-3- جمع‌بندی 64](#_Toc98131355)

[فصل 4: پیشنهاد رساله: معماری تحمل‌پذیر اشکال مبتنی بر شتاب‌دهنده ReRAM 67](#_Toc98131356)

[1-4- تحلیل آسیب‌پذیری شبکه عصبی عمیق پیچشی 70](#_Toc98131357)

[1-1-4- بستر شبیه‌سازی و مدل‌های شبکه عصبی 71](#_Toc98131358)

[2-1-4- تزریق اشکال و تحلیل آسیب‌پذیری شبکه عصبی 73](#_Toc98131359)

[2-4- معماری‌ پیشنهادی تحمل‌پذیر اشکال شتاب‌دهنده شبکه عصبی ReRAM 85](#_Toc98131360)

[2-2-4- تحمل‌پذیری اشکال مبتنی بر مصالحه افزونگی زمانی و مکانی 87](#_Toc98131361)

[3-2-4- نگاشت آگاه از اشکال در ‌آرایه متقاطع 91](#_Toc98131362)

[4-2-4- جای‌دهی آگاه از اشکال لایه‌های شبکه عصبی در ‌آرایه متقاطع 95](#_Toc98131363)

[3-4- بستر ارزیابی 96](#_Toc98131364)

[4-4- جمع‌بندی 97](#_Toc98131365)

[5-4- زمان‌بندی انجام پژوهش 98](#_Toc98131366)

|  |  |
| --- | --- |
| فهرست شکل‌ها | صفحه |

[شکل 2‏-‏1 جایگاه شبکه عصبی و یادگیری عمیق در هوش مصنوعی 8](#_Toc98039489)

[شکل 2‏-‏2 مدل شبکه عصبی چند لایه پرسپترون و مدل یک نورون 9](#_Toc98039490)

[شکل 2‏-‏3 معماری شبکه AlexNet 11](#_Toc98039491)

[شکل 2‏-‏4 نمایی از لایه پیچشی در یک لایه 12](#_Toc98039492)

[شکل 2‏-‏5 حافظه‌های دو ترمینالی نوظهور برای ذخیره و محاسبات 14](#_Toc98039493)

[شکل 2‏-‏6 منحنی معمول I-V سلول حافظه ReRAM با ساختار 1T1R 16](#_Toc98039494)

[شکل 2‏-‏7 ساختار آرایه متقاطع ReRAM و انجام عملیات ضرب و جمع آنالوگ 18](#_Toc98039495)

[شکل 2‏-‏8 نگاشت پارمتر‌های شبکه عصبی به آرایه متقاطع ReRAM 20](#_Toc98039496)

[شکل 2‏-‏9 نگاشت شبکه عصبی پیچشی در شتاب‌دهنده مبتنی بر ReRAM 21](#_Toc98039497)

[شکل 2‏-‏10 نگاشت وزن‌های مثبت و منفی در دو آرایه متقاطع 21](#_Toc98039498)

[شکل 2‏-‏11 معماری سلسله مراتبی شتاب‌دهنده شبکه عصبی عمیق 22](#_Toc98039499)

[شکل 2‏-‏12 معیار‌های قابلیت اطمینان پایه در شتاب‌دهنده مبتنی برReRAM 27](#_Toc98039500)

[شکل 2‏-‏13 معیار‌های قابلیت اطمینان کاربردی در شتاب‌دهنده ReRAM 27](#_Toc98039501)

[شکل 2‏-‏14 تزریق اشکال در شبکه عصبی در سیستم تشخیص شئ YOLOv3 34](#_Toc98039502)

[شکل 2‏-‏15 تزریق اشکال در شبکه عصبی در سیستم تشخیص شئ در اتومبیل خودران 35](#_Toc98039503)

[شکل 3‏-‏1 معماری شتاب‌دهنده شبکه عصبی ISAAC 42](#_Toc98039504)

[شکل 3‏-‏2 ساختار حافظه اصلی PRIME و مدارهای تغییر یافته/اضافه‌شده 44](#_Toc98039505)

[شکل 3‏-‏3 معماری پیشنهادی PipeLayer 45](#_Toc98039506)

[شکل 3‏-‏4 محاسبات رو‌به‌جلو و رو‌به‌عقب در خط‌لوله پیشنهادی Pipelayer 45](#_Toc98039507)

[شکل 3‏-‏5 معماری تقسیم‌بندی شده قابل‌بازپیکربندی 46](#_Toc98039508)

[شکل 3‏-‏6 جریان کلی واحد محاسبه ضرب نقطه‌ای 47](#_Toc98039509)

[شکل 3‏-‏7 تغییر وزن‌ها در روش بازآموزش شبکه عصبی 59](#_Toc98039510)

[شکل 3‏-‏8 تشخیص اشکال SA1 و SA0 برخط بر اساس مقایسه ولتاژ گوسی 59](#_Toc98039511)

[شکل 3‏-‏9 روش آموزش برخط تحمل‌پذیری اشکال 60](#_Toc98039512)

[شکل 4‏-‏1 معماری شبکه‌های عصبی پیچشی عمیق برای طبقه‌بندی تصاویر 72](#_Toc98039513)

[شکل 4‏-‏2 تأثیر تک اشکال در لایه‌های مختلف 75](#_Toc98039514)

[شکل 4‏-‏3 تأثیر اشکال یک مقدار از فیلتر شبکه عصبی پیچشی بر خروجی لایه 76](#_Toc98039515)

[شکل 4‏-‏4 ماتریس درهم‌ریختگی برای مدل‌های مختلف بدون اشکال و مدل دارای اشکال 78](#_Toc98039516)

[شکل 4‏-‏5 تزریق اشکال بیشینه و کمینه در لایه‌های مختلف LeNet5 80](#_Toc98039517)

[شکل 4‏-‏6 تزریق اشکال بیشینه و کمینه در لایه‌های مختلف SimpleNet 80](#_Toc98039518)

[شکل 4‏-‏7 تزریق همزمان اشکال بیشینه و کمینه 80](#_Toc98039519)

[شکل 4‏-‏8 معماری ‌آرایه متقاطع دوگانه برای نگاشت مقادیر مثبت و منفی وزن‌ها‌ 81](#_Toc98039520)

[شکل 4‏-‏9 تأثیر اشکال SAH یا SAL در آرایه متقاطع ReRAM بر دقت شبکه عصبی عمیق 82](#_Toc98039521)

[شکل 4‏-‏10 درصد تغییرات دقت شبکه عصبی LeNet5 و SimpleNet برای اشکال‌های SAH و SAL 83](#_Toc98039522)

[شکل 4‏-‏11 تأثیر اشکال SAH یا SAL به صورت همزمان در آرایه متقاطع ReRAM بر دقت CNN 83](#_Toc98039523)

[شکل 4‏-‏12 ماتریس درهم‌ریختگی در حضور 10% خطای SAL و 20% اشکال SAH 84](#_Toc98039524)

[شکل 4‏-‏13 فلوچارت بخش‌های مختلف روش پیشنهادی 87](#_Toc98039525)

[شکل 4‏-‏14 مثالی از ساختار تحمل‌پذیر اشکال مبتنی بر افزونگی پیشنهادی 90](#_Toc98039526)

[شکل 4‏-‏15 نگاشت آگاه از اشکال در آرایه متقاطع دوگانه 93](#_Toc98039527)

[شکل 4‏-‏16 نتایج حاصل از تحمل‌پذیری اشکال با استفاده از آرایه متقاطع دوگانه متناظر 95](#_Toc98039528)

|  |  |
| --- | --- |
| فهرست جدول‌ها | صفحه |

[جدول 2‏-‏1 پارامترهای طراحی شتاب‌دهنده شبکه عصبی مبتنی بر ReRAM در سطوح تجرید مختلف 23](#_Toc98039612)

[جدول 3‏-‏1 دسته‌بندی کارهای پیشین در زمینه تزریق اشکال، افزایش دقت و راهکارهای تحمل‌پذیر اشکال 65](#_Toc98039613)

[جدول 4‏-‏1 مشخصه‌های شبکه‌های عصبی پیچشی عمیق برای طبقه‌بندی تصاویر 72](#_Toc98039614)

[جدول 4‏-‏2 میانگین کاهش دقت تمام‌لایه‌ها در مدل‌های مختلف 77](#_Toc98039615)

[جدول 4‏-‏3 تعداد پارامتر‌های هر لایه از شبکه SimpleNet و LeNet5 79](#_Toc98039616)

[جدول 4‏-‏4 تحمل‌پذیری اشکال برای چند نمونه از نقشه اشکال متفاوت با افزونگی پیشنهادی 92](#_Toc98039617)

[جدول 4‏-‏5 مقادیر رسانایی نگاشت آگاه از اشکال توسط ‌آرایه متقاطع دوگانه 93](#_Toc98039618)

[جدول 4‏-‏6 برنامه زمان‌بندی مراحل انجام پس از تصویب موضوع پيشنهادی رساله 98](#_Toc98039619)

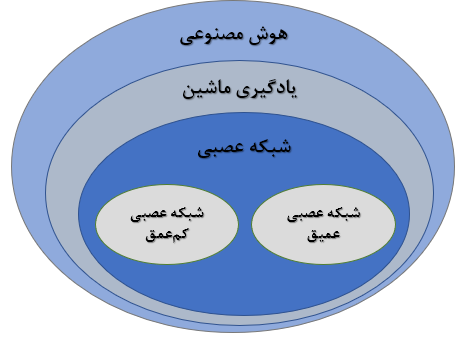
# مقدمه‌

شبکه‎های عصبی پیچشی (CNNs[[2]](#footnote-2)) عملکرد قابل توجهی در بسیاری از زمینه های یادگیری ماشین به دست آورده است، امروزه مدل های شبکه عصبی مانند تشخیص الگو ، پیش بینی، کنترل و بهینه سازی به دست آورده است. اگرچه شبکه های عصبی پیچشی مزایای بسیاری را برای هوش مصنوعی به ارمغان می آورد، اما زمان محاسبات طولانی تر و مصرف توان بیشتر همچنان عملکرد سیستم را به طور قابل توجهی محدود می کند. بنابراین، طراحی تسریع کننده های سخت افزاری برای شبکه های عصبی کارآمدی در سال‌های اخیر بسیار مورد توجه قرار گرفته است. با این حال، به دلیل محاسبات و ارتباطات بسیار پیچیده بین عملیات شبکه های عصبی، اتصال بین هر واحد محاسباتی در داخل لایه های شبکه عصبی پیچشی با افزایش اندازه شبکه پیچیده می‌شود. از سوی دیگر، اتصال شبکه روی تراشه (NoC[[3]](#footnote-3)) برای حل مشکل ارتباطات پیچیده پیشنهاد شده است، که یک اتصال جذاب برای ساخت تسریع کننده های شبکه های عصبی است.

# شبکه عصبی و شتاب‌دهنده‌های مبتنی بر پردازش درون حافظه ReRAM

## یادگیری ماشین و شبکه عصبی

در هوش مصنوعی زیرشاخه بزرگی به نام یادگیری ماشین[[4]](#footnote-4) وجود دارد. آرتور ساموئل[[5]](#footnote-5) در سال 1959 میلادی، یادگیری ماشین را شاخه‌ای از دانش تعریف کرد که به کامپیوتر توانایی یادگیری می‌دهد، بدون آنکه به صورت صریح برنامه‌نویسی شود. به عبارت دیگر، برنامه‌ای که یک‌بار ساخته می‌شود، یاد می‌گیرد که چگونه یک عمل هوشمندانه را خارج از مقوله برنامه‌نویسی انجام دهد. رابطه بین یادگیری عمیق و دیگر شاخه‌های هوش مصنوعی در شکل 2-1 آورده شده است. معماری یادگیری ماشین در دو فاز آموزش و استنتاج کار می‌کند. در فاز آموزش، رابطه بین ورودی و خروجی سیستم مدل‌شده، مشخص می‌شود و در فاز استنتاج، برای هر ورودی، خروجی مورد نظر پیش‌بینی می‌شود.



جایگاه شبکه عصبی و یادگیری عمیق در هوش مصنوعی

|  |  |
| --- | --- |
| (‏2‌-1) |  |

## جمع‌بندی

همانطور که بیان شد، عملیات اصلی در مدل‌های شبکه عصبی، ضرب بردار-ماتریس است که می‌توان این عملیات را به صورت کارا با انرژی کم و سرعت بالا توسط پردازش درون حافظه ReRAM به صورت آنالوگ انجام داد. بنابراین شتاب‌دهنده‌های شبکه عصبی مبتنی بر پردازش درون حافظه ReRAM گزینه مناسبی برای اجرای مدل‌های شبکه عصبی هستند.

### فناوری ساخت حافظه ReRAM در ابتدای کار است

و هنوز به بلوغ نرسیده است. به همین دلیل با چالش‌هایی همراه است که محققان سعی در حل این چالش‌ها دارند.

#### یکی از چالش‌های اساسی

این شتاب‌دهنده‌ها، کاهش دقت شبکه عصبی به دلیل وجود اشکال در آرایه متقاطع ReRAM است و هدف در این پژوهش ارائه یک معماری تحمل‌پذیر اشکال خواهد بود.

# پژوهش‌های پیشین: پردازش درون حافظه و قابلیت اطمینان

در سال‌های اخیر، یادگیری ماشین و به خصوص روش‌های مبتنی بر شبکه عصبی پیچشی توجه بسیاری از محققان در زمینه‌های مختلف را به خود جلب کرده است و توانایی بالقوه‌ای در اغلب رشته‌ها و زمینه‌ها از خود نشان داده‌اند. هر چه مدل‌های CNN پیچیده‌تر شوند، علاوه بر دقت بالای طبقه‌بندی، تعداد پارامترهای شبکه و محاسبات به طور چشمگیری افزایش می‌یابد که باعث مصرف انرژی بالا و زمان محاسبات زیادی می‌شود [18]. به همین دلیل روش‌های بهینه‌سازی مختلفی در سطح نرم‌افزار و سخت‌افزار و معماری‌های خاص‌منظوره پیشنهاد شده تا با کارایی بالا و انرژی مصرفی کم این مدل‌ها را پردازش کند و تأثیری بر روی دقت آن‌ها نگذارد.

(‏3‌-8)

## افزایش دقت و بهبود قابلیت اطمینان

پارامترها و فاکتورهایی که می‌تواند دقت و کارایی مدل شبکه عصبی را در شتاب‌دهنده‌های شبکه عصبی کاهش دهد شامل منابع نویز، تغییرپذیری و وجود اشکال گذرا و دائمی در سلول‌های حافظه، رانش مقاومتی، محدود بودن اندازه ‌آرایه و وضوح مبدل‌های آنالوگ به دیجیتال و دیجیتال به آنالوگ است.

## جمع‌بندی

در این فصل به بررسی روش‌های پیشین در زمینه شتاب‌دهنده‌های شبکه عصبی پرداختیم. در ابتدا مقدمه‌ای از پردازش درون حافظه و موارد استفاده از پردازش درون حافظه ذکر گردید. در ادامه به طراحی‌های شتاب‌دهنده مبتنی بر ReRAM پایه و همچنین طراحی‌هایی که مختص فازهای استنتاج و آموزش هستند و چند نمونه از معماری‌های قابل بازپیکربندی و روش‌های نگاشت و جای‌دهی شبکه عصبی معرفی شدند. معماری‌های ارائه شده تمرکز خود را بر پیاده‌سازی یک واحد محاسبات مبتنی بر پردازش آنالوگ درون حافظه برای انجام عملیات ضرب و جع ماتریسی گذاشته‌اند. همچنین معماری‌هایی برای مدارهای جانبی مانند مبدل‌های DAC و ADC و مدارهایی برای انجام محاسبات توابع مورد نیاز شبکه عصبی، نحوه نگاشت شبکه عصبی به آرایه‌های متقاطع به منظور افزایش قابلیت موازی‌سازی، کاهش تأخیر و انرژی مصرفی پیشنهاد شده است.

در فصل بعد راهکارهایی برای تحمل‌پذیری اشکال شتاب‌دهنده‌های مبتنی بر پردازش درون حافظه ReRAM با در نظر گرفتن رفتار شبکه عصبی و مشخصه‌های حافظه ReRAM ارائه خواهد شد. هدف از این معماری‌ها و راهکارها، کاهش تابع هزینه که همان میزان خطا در دقت شبکه عصبی در حضور اشکال‌های دائمی است، می‌باشد.

# طرح پیشنهادی

اگرچه Noxim یک شبیه‌ساز مبتنی بر شبکه‎های‎ روی‎ تراشه متن باز گسترده است، اما نمی‌تواند از عملکرد محاسبات نورون در هر داخل عناصر پردازشی(PE[[6]](#footnote-6)) پشتیبانی کند، بنابراین، ما شبیه ساز را برای پشتیبانی کردن از شبکه های عصبی پیچشی توسعه و گسترش می دهیم تا تبدیل به یک Noxim برپایه شبکه‎های عصبی CNN شود.(CNN-Noxim) که می تواند رفتار محاسباتی CNN مبتنی بر NoC را کامل شبیه سازی کند.

برای تسهیل محاسبات و عملیات شبکه عصبی مبتنی بر شبکه‎های روی تراشه مراحل زیر را انجام می دهیم:

1) تغییر شکل شبکه عصبی (Neural network reshaping):

2) کنترل جریان محاسباتی عناصر پردازشی (Processing Element computing flow control):

که در بخش بعدی این مراحل به تفضیل توضیح داده می شوند.

در ابتدا باید مدل سطح تراکنش(TLM[[7]](#footnote-7)) را تغییر دهیم تا به صورت ماژولار(Modular) یعنی هر قسمت به صورت مستقل از هم تعریف شده و قابلیت استفاده مجدد داشته باشند.

کانال

ماژول کاشی 3

ماژول کاشی 2

ماژول کاشی 1

ماژول کاشی 0

PE

Router

PE

Router

PE

Router

PE

Router

1. تغییر شکل شبکه عصبی:

در شبکه های بسیار گسترده قرار دادن یک نورون داخل یک عنصر پردازشی باعث انتقالات انبوهی از داده در ارتباطات بین گره های شبکه روی تراشه می شود که این امر تاخیر مبادله بسته ها در شبکه را افزایش می دهد، لذا ملزم به قرار دادن چند نورون به صورت گروهی در داخل عناصر پردازشی هستیم تا عملیات چند نوورن که به هم دیگر وابسته هستند به صورت موازی انجام شوند و در صورت تکمیل عملیات به عنصر پردازشی در گره مقصد فرستاده شود.

همانطور که در شکل فلان دید می شود شبکه های عصبی پیچشی از چند لایه مختلف مثل لایه ادغام (pooling layer) لایه کاملا متصل (fully Connected layer) و لایه پیچشی (convolution layers)

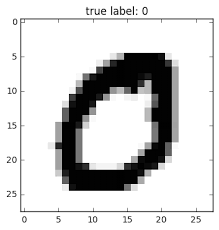
تشکیل شده است.لازم هست ما برای تسهیل عملیات سنگین در این سه لایه، شبکه عصبی را به چند واحد محاسباتی پیوسته تقسیم کنیم )مثل عملیات ساده ضرب و جمع چند نورون همزمان( که با این عمل شبکه عصبی پیچشی را به شبکه عصبی مصنوعی ساده (ANN-Like) تبدیل می کنیم.

بنابراین ما شبکه عصبی پیچشی با چندین لایه ادغام و لایه پیچشی و لایه کاملا متصل به یک شبکه کاملا متصل ساده تبدیل کردیم که ANN-like CNN نامیده می شود.

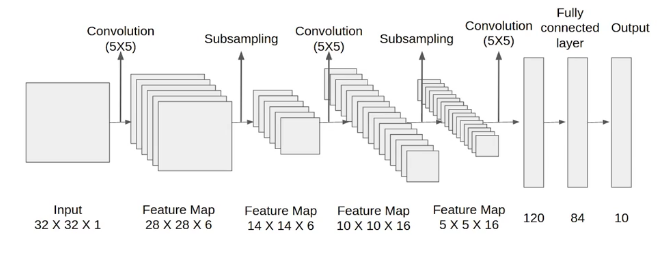
بعد از تغییر شبکه عصبی به یک شبکه عصبی ساده باید این شبکه ساده عصبی را در شبکه روی تراشه نگاشت بکنیم، برای این کار چندین نورون را داخل عناصر پردازشی گروه بندی می کنیم و داخل عنصر پردازشی قرار می دهیم که از نظر سخت افزاری کارکردی کاآمد داشته باشد یعنی دو شرط زیر را داشت باشد:

1. هر نورون داخل همان لایه خود گروه بندی شود.
2. ) نورون هایی که نتیجه محاسباتشان به هم وابسته هستند در در گروه قرار بگیرند.
3. کنترل جریان محاسباتی عناصر پردازشی:

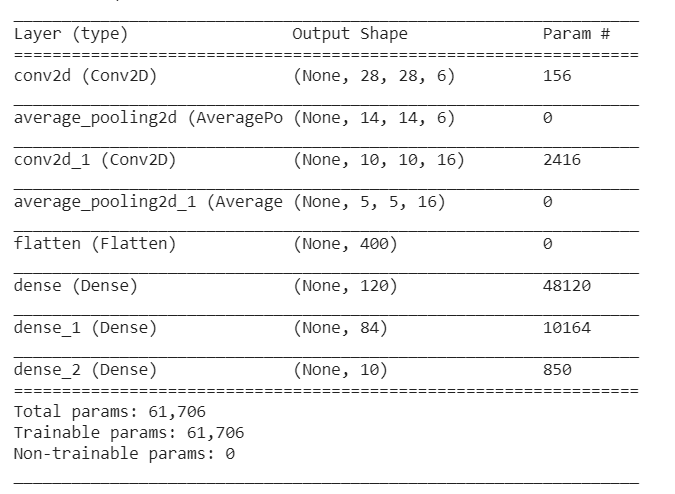
برای پیاده‌سازی این پروژه ابتدا یک شبکه عصبی پیچشی تحت عنوان مدل LeNet که دارای هفت لایه می باشد را در یک پایگاه داده از داده های MNIST دارای 60000 تصویر است را انتخاب کردیم که داده‌های آن از نوع تصویر خاکستری 32\*32 پیکسل است و این تصاویر مربوط به اعداد دست‌نویس صفر تا نه هستند.

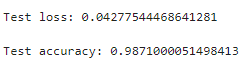


این شبکه عصبی دارای هفت‌لایه هست که هر لایه دارای چندین feature map و فیلتر است که ویژگی‌های مختلف و مهم تصویر از آن استخراج می‌شود.



شکل بالا هفت‌لایه مختلف این شبکه را نشان می‌دهد. ابتدا با زبان پایتون و کتابخانه Keras این شبکه عصبی را شبیه‌سازی می‌کنیم و با توجه با میزان دقت و خطا سعی می‌کنیم تا با تعداد تکرار مناسب دقت را به حداکثر برسانیم.





سپس داده‌های ورودی و وزن‌های ضرب شده و مدل به‌دست‌آمده را ذخیره می‌کنیم.

حال باید این داده‌های شبکه عصبی را روی شبکه روی تراشه سه‌بعدی توسط ابزار شبیه‌ساز Noxim نگاشت بکنیم.

در اینجا با چالش‌هایی روبرو هستیم:

1. نگاشت به چه صورت انجام شود تا به همان دقت برسیم؟
2. کنترل جریان به چه صورت رعایت شود تا تداخل صورت نگیرد و نورونی منتظر جواب لایه قبل نباشد؟
3. چه الگوریتم مسیریابی انتخاب شده؟ چه تعداد نورون داخل عناصر پردازشی گروه‌بندی شود؟
4. اندازه شبکه روی تراشه چند است و چه تأثیری روی پارامترهای نهایی دارد؟

طبق محاسبات در لایه‌های مختلف این شبکه متوجه می‌شویم که تعداد پارامترها در عملیات عصبی که ضرب و جمع ماتریسی هستند بسیار زیاد است که منجر به محاسبات زمان‌بری می‌شود پس باید این محاسبات را توسط systolic array ها در شبکه روی تراشه بطور موازی انجام دهیم تا در زمان بهبود حاصل شود.

برای مثال در لایه اول شبکه عصبی داریم:

**لایه اول**

n = 6 =>تعداد فیلتر

=> F =5سایز فیلتر

Padding => p =0لایه‌گذاری

Stride => s =1گام

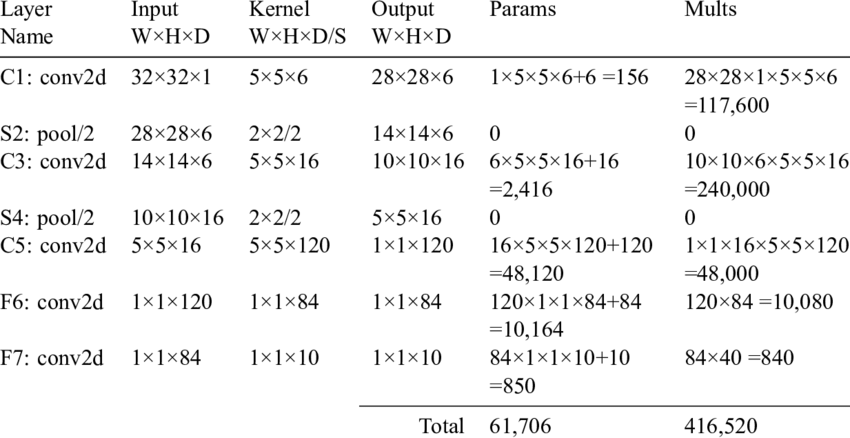
5 x 5 x 1 x 6 + 6 =156= تعداد پارامتر

28 x 28 x 156 = 122304= تعداد اتصالات

که در کل طی عملیات در هفت‌لایه تعداد اتصالات به مقدار زیر است می‌رسیم:

122304 + 5880 + 151600 + 2000 = 281784

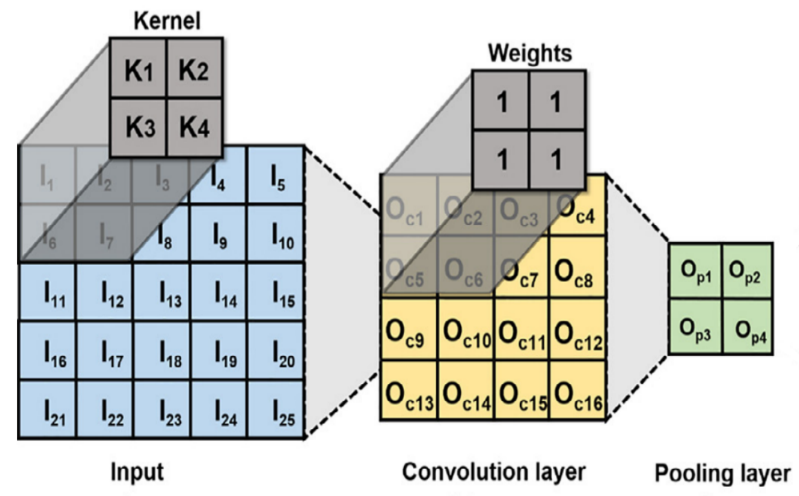
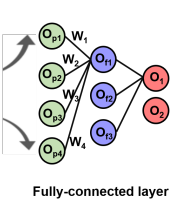
برای هر لایه داریم:



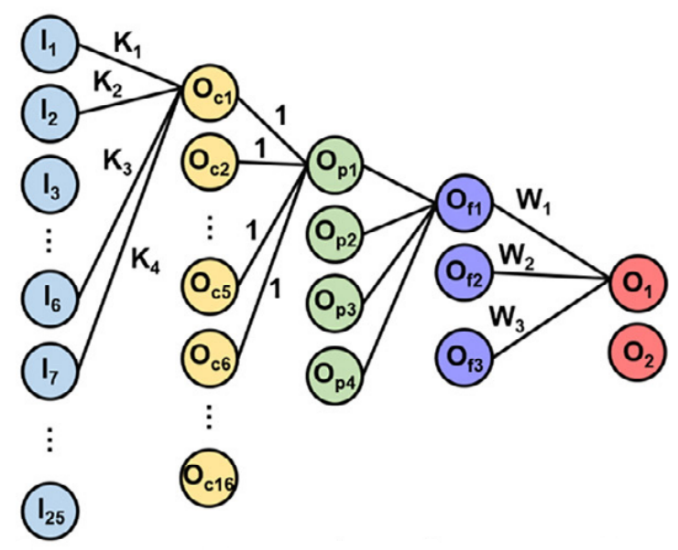
قبل از نگاشت روی شبکه روی تراشه باید تعداد اتصالات را به حداقل برسانیم.

**راه‌حل:** کاهش تعداد اتصالات و محاسبات نورونی reshape کردن شبکه عصبی است که به یک شبکه کاملاً متصل تبدیل می‌شود. برای نگاشت این شبکه عصبی روی شبکه روی تراشه سه‌بعدی ابتدا شبکه عصبی پیچشی را که دارای ماتریس ورودی و ماتریس وزن‌ها هست به یک ماتریس تک ردیفی مثل بردار تبدیل می‌کنیم با این تغییر شبکه عصبی پیچشی به شبکه همانند یک شبکه عصبی مصنوعی ساده تبدیل می‌شود که تمام لایه‌ها تماماً متصل هستند که به این کار مسطح کردن می‌گویند (ANN Like)

یعنی شبکه عصبی پیچشی زیر:



تبدیل به شبکه عصبی کاملا متصل مثل شکل زیر می‌شود:



اکنون شبکه ساده‌تر شده و با توجه نوع لایه (کانولوشنال - ادغام - کاملاً متصل) عملیات ضرب و جمع داخل عناصر پردازشی طبق فلوچارت زیر انجام می‌شود:

بله

حفظ اطلاعات نورون و عملیات (3)

آیا لایه کاملا متصل هست

خیر

خیر

عملیات ادغام روی نتایج لایه کانولوشنال قبلی عملیات (2)

آیا لایه کانولوشنال هست؟

بله

بله

عملیات کانولوشنال

عملیات (1)

لایه بعدی؟

خیر

در لایه convolution layer

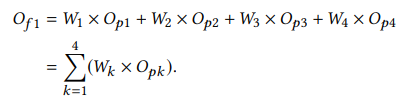
(1)

در لایه pooling layer

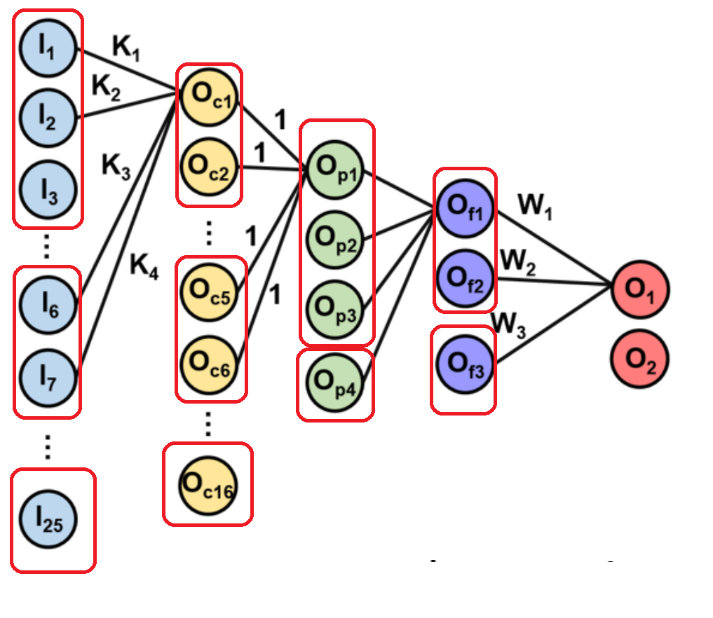
(2)

در لایه fully-connected layer

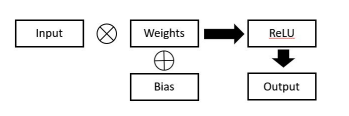
(3)



اکنون باتوجه‌به تعداد داده ورودی و اندازه ماتریس وزن‌ها و feature map تعداد نورون داخل عناصر پردازشی را انتخاب می‌کنیم با این شرط که هر نورون داخل همان لایه خود گروه‌بندی بشود تا در صورت محاسبات ضرب و جمع ماتریسی عنصر پردازشی منتظر پاسخ نورون لایه قبل نشود.



طبق الگوریتم زیر محاسبات داخل عناصر پردازشی صورت می‌گیرد.





در حین عملیات محاسبه شبکه عصبی کنترل جریان باید صورت گیرد تا اگر عملیات نورونی به طور کامل انجام نشده داده را به نورون یا گروه بعدی ارسال نکند. این چرخه برای تمام داده‌ها صورت می‌گیرد.

اگر تعداد گروه نورونی بیشتر از تعداد گره در لایه اول باشد باید بقیه گروه های نورونی را روی لایه بالایی نگاشت بکنیم طریقه نگاشت روی شبکه روی تراشه توسط 5 الگوریتم زیر انجام می شود:

نگاشت مستقیم:

نگاشت باتوجه‌به محور x:

به طور مستقیم و پیوسته گروه‌های نورونی را باتوجه‌به شناسه گروه در عناصر پردازشی نگاشت می‌کند. یعنی در جهت محور x نگاشت می‌کند. اگر تعداد گروه نورونی از تعداد عناصر پردازشی بیشتر باشد گروه‌بندی از لایه زیرین شروع به نگاشت می‌شود؛ یعنی بعد Z تا تمام گروه نورونی جای‌گذاری شوند.

PE

3

2

4

1

…

5

نورون

نگاشت باتوجه‌به محور y:

به طور مستقیم و پیوسته گروه‌های نورونی را باتوجه‌به شناسه گروه در عناصر پردازشی نگاشت می‌کند؛ یعنی در جهت محور y نگاشت می‌کند. اگر تعداد گروه نورونی از تعداد عناصر پردازشی بیشتر باشد گروه‌بندی از لایه زیرین شروع به نگاشت می‌شود؛ یعنی بعد Z تا تمام گروه نورونی جای‌گذاری شوند.

1

5

2

4

3

نگاشت مستقیم مبتنی بر لایه Y:

گروه نورونی را باتوجه‌به شماره لایه‌ای که گروه نورونی در آن قرار دارد به عناصر پردازشی نگاشت می‌کند در جهت محور Y یعنی شکل شبکه عصبی تغییر نمی‌کند هر نورون در موقعیت خودش گروه‌بندی می‌شود.

5

6

3

4

2

1

نگاشت مستقیم مبتنی بر لایه X:

گروه نورونی را باتوجه‌به شماره لایه‌ای که گروه نورونی در آن قرار دارد به عناصر پردازشی نگاشت می‌کند در جهت محور X یعنی شکل شبکه عصبی تغییر نمی‌کند هر نورون در موقعیت خودش گروه‌بندی می‌شود.

1

5

2

6

4

3

نگاشت به‌صورت تصادفی:

گروه‌های نورونی به‌صورت تصادفی داخل عناصر پردازی با شرط رعایت تعداد ظرفیت عناصر انجام می‌شود.

1

4

3

2

5

در این الگوریتم ممکن است عناصر به هم نزدیک باشند یا ممکن است خیلی از هم دور باشند و دسترسی سخت باشد در نتیجه تأخیر بیشتر صورت می‌گیرد.

نگاشت براساس بارکاری work load:

(واریانس حرارتی – اختلاف بزرگ ترین دما از متوسط)

نگاشت آگاه:

براساس ایجاد جدول و مدل نگاشت دلخواه که کمترین connection ایجاد شود

برای پیاده‌سازی این شبکه عصبی روی یک شبکه با سایز 8x8x8 با تعداد بسته و فلیت متفاوت با تعداد گروه نورونی ۳۲تایی و ۶۴تایی و ۱۲۸تایی با سایز بافر 16 به شکل زیر گروه‌بندی و خوشه‌بندی شدند:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  | |  |  |
| تأخیر شبکه با الگوریتم نگاشت متفاوت (سیکل) | | | | | | | | تعداد بسته/ فلیت | اندازه گروه نورونی |
| تصادفی | لایه Y | | لایه X | | محور Y | | محور X |
| 182 | 183 | | 181 | | 183 | | 180 | 14/453 | 32 |
| 156 | 142 | | 143 | | 144 | | 144 | 8/441 | 64 |
| 124 | 94 | | 94 | | 94 | | 94 | 5/435 | 128 |
| 19 | 8 | | 8 | | 8 | | 8 | 2/429 | 512 |
| 18 | 9 | | 9 | | 9 | | 9 | 2/429 | 800 |
| 15 | 9 | | 9 | | 9 | | 9 | 1/429 | 1024 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  | |  |  |
| بازدهی شبکه با الگوریتم نگاشت متفاوت (فلیت/سیکل) | | | | | | | | تعداد بسته/ فلیت | اندازه گروه نورونی |
| تصادفی | لایه Y | | لایه X | | محور Y | | محور X |
| 2.09 | 3.2 | | 2.7 | | 2.81 | | 3.05 | 14/453 | 32 |
| 1.70 | 2.7 | | 2.6 | | 2.56 | | 2.56 | 8/441 | 64 |
| 1.48 | 2.02 | | 2.02 | | 2.00 | | 2.00 | 5/435 | 128 |
| 0.50 | 0.51 | | 0.51 | | 0.52 | | 0.53 | 2/429 | 512 |
| 0.99 | 0.52 | | 0.52 | | 0.53 | | 0.53 | 2/429 | 800 |
| 0.51 | 0.52 | | 0.52 | | 0.52 | | 0.52 | 1/429 | 1024 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  | |  |  |
| توان مصرفی شبکه با الگوریتم نگاشت متفاوت (فلیت/سیکل) | | | | | | | | تعداد بسته/ فلیت | اندازه گروه نورونی |
| تصادفی | لایه Y | | لایه X | | محور Y | | محور X |
| 8.96 | 10.01 | | 9.01 | | 9.03 | | 9.02 | 14/453 | 32 |
| 9.05 | 9.08 | | 9.01 | | 9.08 | | 9.08 | 8/441 | 64 |
| 9.04 | 9.06 | | 8.09 | | 9.01 | | 9.01 | 5/435 | 128 |
| 9.08 | 8.94 | | 8.94 | | 8.94 | | 8.94 | 2/429 | 512 |
| 8.87 | 8.94 | | 8.94 | | 8.94 | | 8.94 | 2/429 | 800 |
| 8.85 | 8.94 | | 8.94 | | 8.94 | | 8.94 | 1/429 | 1024 |

مدل شبکه VGG-16 :

این مدل یک تصویر سه کاناله با 3× 224×224 پیکسل می باشد

برای مدل شبکه VGG-16 چون تعداد لایه ها بیشتر است بنابراین عملیات ضرب و جمع افزایش می یابد تاخیر عناصر پردازشی و تاخیر تحویل بسته در روی شبکه روی تراشه جداگانه محاسبه شده و تاخیر کل نیز بررسی می شود.

کل پارامتر های این مدل در 16 لایه (13 لایه کانولوشنال 3 لایه کاملا متصل) :

تعداد پارامتر: 15087080

که بعد از گروه بندی تعداد این اتصالات و عملیات ضرب و جمع به 8988 می رسد برای سه شبکه 16×16×16 و32×32×32 و 64×64×64 میزان تاخیر محاسبه شده است:

برای بهبود در میزان دما در شبکه‌های روی تراشه دو الگوریتم مسیریابی xyz ,zxy نیز ارائه شده است که تفاوت دو الگوریتم را در سه شبکه متفاوت در شکل زیر قابل‌مشاهده است.

شکل 1مقایسه دو الگوریتم پیشنهادی

میانگین کاهش دقت تمام‌لایه‌ها در مدل‌های مختلف

|  |  |  |  |
| --- | --- | --- | --- |
|  | **BaseLine Accuracy** | **Average of all faulty layers** | **Accuracy Degradation** |
| **SimpleNet** | 99.42% | 97.41% | 2.01% |
| **LeNet** | 99.26% | 97.25% | 2.01% |
| **AlexNet** | 86.82% | 86.61% | 0.21% |
| **ResNet18** | 89.41% | 56.87% | 32.54% |
| **VGG19** | 88.40% | 57.72% | 30.68% |

همانطور که گفته شد،

## جمع‌بندی

در این فصل، ابتدا آسیب‌پذیری الگوریتم‌های شبکه عصبی در برابر اشکال در آرایه‌های متقاطع ReRAM و تأثیر رخداد اشکال بر روی دقت نهایی شبکه عصبی در سه مرحله مختلف مورد تحلیل و ارزیابی قرار گرفت.

## زمان‌بندی انجام پژوهش

برنامه زمان‌بندی انجام تحقيق پس از تصویب موضوع پيشنهادی رساله در جدول 4-6 ارائه شده است. لازم به ذکر است که شبيه‌سازی‌های مربوط به سه بخش نخست در جدول، انجام شده است که نتایج آن در زیر فصل 4-2 ارائه شده است.

برنامه زمان‌بندی مراحل انجام پس از تصویب موضوع پيشنهادی رساله

**منابع و مراجع**

[1] L. Song, X. Qian, H. Li, and Y. Chen, “PipeLayer: A Pipelined ReRAM-based Accelerator for Deep Learning,” in *Proceedings of the  IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2017, pp. 541–552. doi: 10.1109/HPCA.2017.55.

[2] F. Khalid, M. A. Hanif, S. Rehman, and M. Shafique, “Security for Machine Learning-based Systems: Attacks and Challenges During Training and Inference,” in *Proceedings of the International Conference on Frontiers of Information Technology (FIT)*, Dec. 2018, pp. 327–332. doi: 10.1109/FIT.2018.00064.

[3] W. Xu, J. Wang, and X. Yan, “Advances in Memristor-Based Neural Networks,” *Frontiers in Nanotechnology*, vol. 3, p. 645995, Mar. 2021, doi: 10.3389/fnano.2021.645995.

[4] D. Fan and S. Angizi, “Energy Efficient In-Memory Binary Deep Neural Network Accelerator with Dual-Mode SOT-MRAM,” in *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, Nov. 2017, pp. 609–612. doi: 10.1109/ICCD.2017.107.

[5] J. Chen and X. Ran, “Deep Learning with Edge Computing: A Review,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.

[6] W. Shi and S. Dustdar, “The Promise of Edge Computing,” *Computer (Long Beach Calif)*, vol. 49, no. 5, pp. 78–81, May 2016, doi: 10.1109/MC.2016.145.

[7] “NVIDIA Turing architecture based GPUs,” *https://www.nvidia.com/en-us/designvisualization/ technologies/turing-architecture/*. [Online]. Available: https://www.nvidia.com/en-us/design-visualization/technologies/turing-architecture/

[8] N. P. Jouppi *et al.*, “In-Datacenter Performance Analysis of a Tensor Processing Unit,” in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2017, pp. 1–12. doi: 10.1145/3079856.3080246.

[9] E. Chung *et al.*, “Serving DNNs in Real Time at Datacenter Scale with Project Brainwave,” *IEEE Micro*, vol. 38, no. 2, pp. 8–20, Mar. 2018, doi: 10.1109/MM.2018.022071131.

[10] B. Hickmann, J. Chen, M. Rotzin, A. Yang, M. Urbanski, and S. Avancha, “Intel Nervana Neural Network Processor-T (NNP-T) Fused Floating Point Many-Term Dot Product,” in *Proceedings of the IEEE Symposium on Computer Arithmetic (ARITH)*, Jun. 2020, pp. 133–136. doi: 10.1109/ARITH48897.2020.00029.

[11] S. Jain *et al.*, “Neural network accelerator design with resistive crossbars: Opportunities and challenges,” *IBM Journal of Research and Development*, vol. 63, no. 6, 2019, doi: 10.1147/JRD.2019.2947011.

[12] G. Charan, A. Mohanty, X. Du, G. Krishnan, R. v Joshi, and Y. Cao, “Accurate Inference With Inaccurate RRAM Devices: A Joint Algorithm-Design Solution,” *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 6, no. 1, pp. 27–35, Jun. 2020, doi: 10.1109/JXCDC.2020.2987605.

[13] Q. Wang, Y. Park, and W. D. Lu, “Device Non-Ideality Effects and Architecture-Aware Training in RRAM In-Memory Computing Modules,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021, pp. 1–5. doi: 10.1109/ISCAS51556.2021.9401307.

[14] H. Jeong and L. Shi, “Memristor Devices for Neural Networks,” *Journal of Physics D: Applied Physics*, vol. 52, no. 2, p. 23003, Oct. 2018, doi: 10.1088/1361-6463/aae223.

[15] P. Chi *et al.*, “PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory,” in *Proceedings of the ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2016, pp. 27–39. doi: 10.1109/ISCA.2016.13.

[16] A. Shafiee *et al.*, “ISAAC: a Convolutional Neural Network Accelerator with in-Situ Analog Arithmetic in Crossbars,” *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, Jun. 2016, doi: 10.1145/3007787.3001139.

[17] C. Lammie and M. R. Azghadi, “MemTorch: A Simulation Framework for Deep Memristive Crossbar Architectures,” in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, Oct. 2020, pp. 1–5. doi: 10.1109/ISCAS45731.2020.9180810.

[18] Z. Zhu *et al.*, “MNSIM 2.0: A Behavior-Level Modeling Tool for Memristor-based Neuromorphic Computing Systems,” in *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, Sep. 2020, pp. 83–88. doi: 10.1145/3386263.3407647.

[19] S. Mittal, “A Survey of ReRAM-Based Architectures for Processing-In-Memory and Neural Networks,” *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 75–114, Apr. 2018, doi: 10.3390/make1010005.

[20] J. Li, M. Mao, and C. Chakrabarti, “Improving Reliability of ReRAM-Based DNN Implementation through Novel Weight Distribution,” in *Proceedings of the IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct. 2019, pp. 189–194. doi: 10.1109/SiPS47522.2019.9020318.

[21] I. Chakraborty, M. Fayez Ali, D. Eun Kim, A. Ankit, and K. Roy, “GENIEx: A Generalized Approach to Emulating Non-Ideality in Memristive Xbars using Neural Networks,” in *ACM/IEEE Design Automation Conference (DAC)*, Jul. 2020, pp. 1–6. doi: 10.1109/DAC18072.2020.9218688.

[22] Z. Song *et al.*, “ITT-RNA: Imperfection Tolerable Training for RRAM-Crossbar-based Deep Neural-Network Accelerator,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 1, pp. 129–142, Jan. 2021, doi: 10.1109/TCAD.2020.2989373.

[23] Y. Long, X. She, and S. Mukhopadhyay, “Design of Reliable DNN Accelerator with Un-reliable ReRAM,” in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2019, pp. 1769–1774. doi: 10.23919/DATE.2019.8715178.

[24] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional Networks and Applications in Vision,” in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2010, pp. 253–256. doi: 10.1109/ISCAS.2010.5537907.

[25] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, “Efficient Processing of Deep Neural Networks: A Tutorial and Survey,” *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017, doi: 10.1109/JPROC.2017.2761740.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2012, vol. 25. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[27] W. Zhang *et al.*, “Design guidelines of RRAM based neural-processing-unit: A joint device-circuit-algorithm analysis,” 2019. doi: 10.1145/3316781.3317797.

[28] S. Ghose, A. Boroumand, J. S. Kim, J. Gomez-Luna, and O. Mutlu, “Processing-in-Memory: A Workload-Driven Perspective,” *IBM Journal of Research and Development*, vol. 63, no. 6, pp. 3:1–3:19, Nov. 2019, doi: 10.1147/JRD.2019.2934048.

[29] D. Ielmini and G. Pedretti, “Device and Circuit Architectures for In‐Memory Computing,” *Advanced Intelligent Systems*, vol. 2, no. 7, p. 2000040, Jul. 2020, doi: 10.1002/aisy.202000040.

[30] M. Hu *et al.*, “Dot-Product Engine for Neuromorphic Computing: Programming 1T1M Crossbar to Accelerate Matrix-Vector Multiplication,” in *Proceedings of the Annual Design Automation Conference on (DAC)*, 2016, pp. 1–6. doi: 10.1145/2897937.2898010.

[31] F. Zahoor, T. Z. Azni Zulkifli, and F. A. Khanday, “Resistive Random Access Memory (RRAM): an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (MLC) Storage, Modeling, and Applications,” *Nanoscale Research Letters*, vol. 15, no. 1, p. 90, Apr. 2020, doi: 10.1186/s11671-020-03299-9.

[32] G. Pedretti and D. Ielmini, “In-Memory Computing with Resistive Memory Circuits: Status and Outlook,” *Electronics (Basel)*, vol. 10, no. 9, p. 1063, Apr. 2021, doi: 10.3390/electronics10091063.

[33] G. Pedretti, E. Ambrosi, and D. Ielmini, “Conductance Variations and their Impact on the Precision of in-Memory Computing with Resistive Switching Memory (RRAM),” in *Proceedings of the IEEE International Reliability Physics Symposium (IRPS)*, Mar. 2021, pp. 1–8. doi: 10.1109/IRPS46558.2021.9405130.

[34] Y. Guo, H. Wu, B. Gao, and H. Qian, “Unsupervised Learning on Resistive Memory Array Based Spiking Neural Networks,” *Frontiers in Neuroscience*, vol. 13, p. 812, Aug. 2019, doi: 10.3389/fnins.2019.00812.

[35] L. Xia *et al.*, “MNSIM: Simulation Platform for Memristor-based Neuromorphic Computing System,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, p. 1, 2017, doi: 10.1109/TCAD.2017.2729466.

[36] S. Huang *et al.*, “Mixed Precision Quantization for ReRAM-based DNN Inference Accelerators,” in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2021, pp. 372–377. doi: 10.1145/3394885.3431554.

[37] Y. Sun *et al.*, “Unary Coding and Variation-Aware Optimal Mapping Scheme for Reliable ReRAM-based Neuromorphic Computing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, p. 1, 2021, doi: 10.1109/TCAD.2021.3051856.

[38] B. Yan *et al.*, “Resistive Memory‐Based In‐Memory Computing: From Device and Large‐Scale Integration System Perspectives,” *Advanced Intelligent Systems*, vol. 1, no. 7, p. 1900068, Nov. 2019, doi: 10.1002/aisy.201900068.

[39] V. Milo *et al.*, “Optimized Programming Algorithms for Multilevel RRAM in Hardware Neural Networks,” in *Proceedings of the IEEE International Reliability Physics Symposium (IRPS)*, Mar. 2021, pp. 1–6. doi: 10.1109/IRPS46558.2021.9405119.

[40] M. Zhao, B. Gao, J. Tang, H. Qian, and H. Wu, “Reliability of Analog Resistive Switching Memory for Neuromorphic Computing,” *Applied Physics Reviews*, vol. 7, no. 1, p. 11301, Mar. 2020, doi: 10.1063/1.5124915.

[41] I. Yeo, M. Chu, S.-G. Gi, H. Hwang, and B.-G. Lee, “Stuck-at-Fault Tolerant Schemes for Memristor Crossbar Array-Based Neural Networks,” *IEEE Transactions on Electron Devices*, vol. 66, no. 7, pp. 2937–2945, Jul. 2019, doi: 10.1109/TED.2019.2914460.

[42] O. Tunali and M. Altun, “Permanent and Transient Fault Tolerance for Reconfigurable Nano-Crossbar Arrays,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 5, pp. 747–760, May 2017, doi: 10.1109/TCAD.2016.2602804.

[43] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang, “Fault-Tolerant Training with On-Line Fault Detection for RRAM-based Neural Computing Systems,” in *Proceedings of the Annual Design Automation Conference (DAC)*, Jun. 2017, pp. 1–6. doi: 10.1145/3061639.3062248.

[44] L. Xia *et al.*, “Stuck-at Fault Tolerance in RRAM Computing Systems,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 102–115, Mar. 2018, doi: 10.1109/JETCAS.2017.2776980.

[45] Ching-Yi Chen *et al.*, “RRAM Defect Modeling and Failure Analysis Based on March Test and a Novel Squeeze-Search Scheme,” *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 180–190, Jan. 2015, doi: 10.1109/TC.2014.12.

[46] K. Beckmann, J. Holt, H. Manem, J. van Nostrand, and N. C. Cady, “Nanoscale Hafnium Oxide RRAM Devices Exhibit Pulse Dependent Behavior and Multi-level Resistance Capability,” *MRS Advances*, vol. 1, no. 49, pp. 3355–3360, Oct. 2016, doi: 10.1557/adv.2016.377.

[47] A. Grossi *et al.*, “Resistive RAM Endurance: Array-Level Characterization and Correction Techniques Targeting Deep Learning Applications,” *IEEE Transactions on Electron Devices*, vol. 66, no. 3, pp. 1281–1288, Mar. 2019, doi: 10.1109/TED.2019.2894387.

[48] A. Chaudhuri, J. Talukdar, F. Su, and K. Chakrabarty, “Functional Criticality Classification of Structural Faults in AI Accelerators,” in *Proceedings of the  IEEE International Test Conference (ITC)*, Nov. 2020, pp. 1–5. doi: 10.1109/ITC44778.2020.9325272.

[49] B. K. Joardar, J. R. Doppa, H. Li, K. Chakrabarty, and P. P. Pande, “Learning to Train CNNs on Faulty ReRAM-based Manycore Accelerators,” *ACM Transactions on Embedded Computing Systems*, vol. 20, no. 5s, pp. 1–23, Oct. 2021, doi: 10.1145/3476986.

[50] G. Li *et al.*, “Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2017, pp. 1–12. doi: 10.1145/3126908.3126964.

[51] T.-C. Chang, K.-C. Chang, T.-M. Tsai, T.-J. Chu, and S. M. Sze, “Resistance Random Access Memory,” *Materials Today*, vol. 19, no. 5, pp. 254–264, Jun. 2016, doi: 10.1016/j.mattod.2015.11.009.

[52] C. Torres-Huitzil and B. Girau, “Fault and Error Tolerance in Neural Networks: A Review,” *IEEE Access*, vol. 5, pp. 17322–17341, 2017, doi: 10.1109/ACCESS.2017.2742698.

[53] E. M. el Mhamdi and R. Guerraoui, “When Neurons Fail,” in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2017, pp. 1028–1037. doi: 10.1109/IPDPS.2017.66.

[54] C. Neti, M. H. Schneider, and E. D. Young, “Maximally Fault Tolerant Neural Networks,” *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 14–23, Jan. 1992, doi: 10.1109/72.105414.

[55] K. Zhao *et al.*, “FT-CNN: Algorithm-based Fault Tolerance for Convolutional Neural Networks,” *IEEE Transactions on Parallel and Distributed Systems*, p. 1, 2021, doi: 10.1109/TPDS.2020.3043449.

[56] S. A. El-Sayed *et al.*, “Spiking Neuron Hardware-Level Fault Modeling,” in *Proceedings of the IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Jul. 2020, pp. 1–4. doi: 10.1109/IOLTS50870.2020.9159745.

[57] A. Mahmoud *et al.*, “PyTorchFI: A Runtime Perturbation Tool for DNNs,” in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Jun. 2020, pp. 25–31. doi: 10.1109/DSN-W50199.2020.00014.

[58] H. Guo, L. Peng, J. Zhang, Q. Chen, and T. D. LeCompte, “ATT: A Fault-Tolerant ReRAM Accelerator for Attention-based Neural Networks,” in *Proceedings of the International Conference on Computer Design (ICCD)*, Oct. 2020, pp. 213–221. doi: 10.1109/ICCD50377.2020.00047.

[59] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, “Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping,” in *Proceedings of the Annual Design Automation Conference (DAC)*, Jun. 2019, pp. 1–6. doi: 10.1145/3316781.3317870.

[60] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, “DRISA: a DRAM-based Reconfigurable In-Situ Accelerator,” in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct. 2017, pp. 288–301. doi: 10.1145/3123939.3123977.

[61] M. Zabihi, Z. I. Chowdhury, Z. Zhao, U. R. Karpuzcu, J.-P. Wang, and S. S. Sapatnekar, “In-Memory Processing on the Spintronic CRAM: From Hardware Design to Application Mapping,” *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1159–1173, Aug. 2019, doi: 10.1109/TC.2018.2858251.

[62] L. Cheng, H.-X. Zheng, Y. Li, T.-C. Chang, S. M. Sze, and X. Miao, “In-Memory Digital Comparator Based on a Single Multivalued One-Transistor-One-Resistor Memristor,” *IEEE Transactions on Electron Devices*, vol. 67, no. 3, pp. 1293–1296, Mar. 2020, doi: 10.1109/TED.2020.2967401.

[63] B. Li, P. Gu, Y. Shan, Y. Wang, Y. Chen, and H. Yang, “RRAM-Based Analog Approximate Computing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 12, pp. 1905–1917, Dec. 2015, doi: 10.1109/TCAD.2015.2445741.

[64] M. Zhou, M. Imani, S. Gupta, Y. Kim, and T. Rosing, “GRAM: Graph Processing in a ReRAM-based Computational Memory,” in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2019, pp. 591–596. doi: 10.1145/3287624.3287711.

[65] L. Han, Z. Shen, D. Liu, Z. Shao, H. H. Huang, and T. Li, “A Novel ReRAM-Based Processing-in-Memory Architecture for Graph Traversal,” *ACM Transactions on Storage*, vol. 14, no. 1, pp. 1–26, Apr. 2018, doi: 10.1145/3177916.

[66] Y. Chen *et al.*, “DaDianNao: A Machine-Learning Supercomputer,” in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2014, pp. 609–622. doi: 10.1109/MICRO.2014.58.

[67] F. Chen, L. Song, H. H. Li, and Y. Chen, “ZARA: A Novel Zero-free Dataflow Accelerator for Generative Adversarial Networks in 3D ReRAM,” in *Proceedings of the Annual Conference on Design Automation (DAC)*, Jun. 2019, pp. 1–6. doi: 10.1145/3316781.3317936.

[68] M. A. Hanif, M. Zuhaib Akbar, R. Ahmed, S. Rehman, A. Jantsch, and M. Shafique, “MemGANs: Memory Management for Energy-Efficient Acceleration of Complex Computations in Hardware Architectures for Generative Adversarial Networks,” in *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Jul. 2019, pp. 1–6. doi: 10.1109/ISLPED.2019.8824833.

[69] Q. Wang, X. Wang, S. H. Lee, F.-H. Meng, and W. D. Lu, “A Deep Neural Network Accelerator Based on Tiled RRAM Architecture,” in *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*, Dec. 2019, pp. 14.4.1–14.4.4. doi: 10.1109/IEDM19573.2019.8993641.

[70] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, “Vortex: Variation-Aware Training for Memristor X-Bar,” in *Proceedings of the Annual Design Automation Conference (DAC)*, Jun. 2015, pp. 1–6. doi: 10.1145/2744769.2744930.

[71] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, “Training and Operation of an Integrated Neuromorphic Network based on Metal-Oxide Memristors,” *Nature*, vol. 521, no. 7550, pp. 61–64, May 2015, doi: 10.1038/nature14441.

[72] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, “Memristor crossbar-based neuromorphic computing system: A case study,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 10, 2014, doi: 10.1109/TNNLS.2013.2296777.

[73] K. Vandoorne *et al.*, “Experimental Demonstration of Reservoir Computing on a Silicon Photonics Chip,” *Nature Communications*, vol. 5, no. 1, Dec. 2014, doi: 10.1038/ncomms4541.

[74] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, “High Precision Tuning of State for Memristive Devices by Adaptable Variation-Tolerant Algorithm,” *Nanotechnology*, vol. 23, no. 7, p. 75201, Jan. 2012, doi: 10.1088/0957-4484/23/7/075201.

[75] S. Feng, S. Gupta, A. Ansari, and S. Mahlke, “Shoestring: Probabilistic Soft Error Reliability on the Cheap,” in *Proceedings of the International Conference on Architectural support for programming languages and operating systems (ASPLOS)*, Mar. 2010, pp. 385–396. doi: 10.1145/1736020.1736063.

[76] G. Li, Q. Lu, and K. Pattabiraman, “Fine-Grained Characterization of Faults Causing Long Latency Crashes in Programs,” in *Proceedings of the Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-W)*, Jun. 2015, pp. 450–461. doi: 10.1109/DSN.2015.36.

[77] S. K. S. Hari, S. v Adve, and H. Naeimi, “Low-Cost Program-Level Detectors for Reducing Silent Data Corruptions,” in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2012, pp. 1–12. doi: 10.1109/DSN.2012.6263960.

[78] S. A. El-Sayed, L. A. Camunas-Mesa, B. Linares-Barranco, and H.-G. Stratigopoulos, “Self-Testing Analog Spiking Neuron Circuit,” in *Proceedings of the International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Jul. 2019, pp. 81–84. doi: 10.1109/SMACD.2019.8795234.

[79] O. Temam, “A Defect-Tolerant Accelerator for Emerging High-Performance Applications,” in *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2012, pp. 356–367. doi: 10.1109/ISCA.2012.6237031.

[80] E.-I. Vatajelu, G. di Natale, and L. Anghel, “Special Session: Reliability of Hardware-Implemented Spiking Neural Networks (SNN),” in *Proceedings of the IEEE VLSI Test Symposium (VTS)*, Apr. 2019, pp. 1–8. doi: 10.1109/VTS.2019.8758653.

[81] B. Reagen *et al.*, “Ares: A framework for Quantifying the Resilience of Deep Neural Networks,” in *Proceedings of the Design Automation Conference (DAC)*, Jun. 2018, pp. 1–6. doi: 10.1109/DAC.2018.8465834.

[82] M. A. Neggaz, I. Alouani, S. Niar, and F. Kurdahi, “Are CNNs reliable enough for critical applications? An exploratory study,” *IEEE Design and Test*, vol. 37, no. 2, 2020, doi: 10.1109/MDAT.2019.2952336.

[83] S. K. S. Hari, T. Tsai, M. Stephenson, S. W. Keckler, and J. Emer, “SASSIFI: An Architecture-Level Fault Injection Tool for GPU Application Resilience Evaluation,” in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2017, pp. 249–258. doi: 10.1109/ISPASS.2017.7975296.

[84] T. Tsai, S. K. S. Hari, M. Sullivan, O. Villa, and S. W. Keckler, “NVBitFI: Dynamic Fault Injection for GPUs,” in *Proceedings of the Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2021, pp. 284–291. doi: 10.1109/DSN48987.2021.00041.

[85] Z. Chen, N. Narayanan, B. Fang, G. Li, K. Pattabiraman, and N. DeBardeleben, “TensorFI: A Flexible Fault Injection Framework for TensorFlow Applications,” in *IEEE International Symposium on Software Reliability Engineering (ISSRE)*, Oct. 2020, pp. 426–435. doi: 10.1109/ISSRE5003.2020.00047.

[86] J. J. Zhang, K. Basu, and S. Garg, “Fault-Tolerant Systolic Array based Accelerators for Deep Neural Network Execution,” *IEEE Design & Test*, vol. 36, no. 5, pp. 44–53, Oct. 2019, doi: 10.1109/MDAT.2019.2915656.

[87] G. Gambardella *et al.*, “Efficient Error-Tolerant Quantized Neural Network Accelerators,” in *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2019, pp. 1–6. doi: 10.1109/DFT.2019.8875314.

[88] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Adv Neural Inf Process Syst*, vol. 32, pp. 8026–8037, 2019.

[89] Y. H. Chen, J. Emer, and V. Sze, “Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks,” 2016. doi: 10.1109/ISCA.2016.40.

[90] R. Hasan, T. M. Taha, and C. Yakopcic, “On-Chip Training of Memristor based Deep Neural Networks,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 3527–3534. doi: 10.1109/IJCNN.2017.7966300.

[91] O. Krestinskaya, K. N. Salama, and A. P. James, “Learning in Memristive Neural Network Architectures Using Analog Backpropagation Circuits,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 719–732, Feb. 2019, doi: 10.1109/TCSI.2018.2866510.

[92] A. Mehonic, D. Joksas, W. H. Ng, M. Buckwell, and A. J. Kenyon, “Simulation of Inference Accuracy Using Realistic RRAM Devices,” *Frontiers in Neuroscience*, vol. 13, p. 593, Jun. 2019, doi: 10.3389/fnins.2019.00593.

[93] A. Ankit *et al.*, “PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference,” in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Apr. 2019, pp. 715–731. doi: 10.1145/3297858.3304049.

[94] H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr, “Recent Progress in Analog Memory-based Accelerators for Deep Learning,” *Journal of Physics D: Applied Physics*, vol. 51, no. 28, p. 283001, Jun. 2018, doi: 10.1088/1361-6463/aac8a5.

[95] A. van de Goor and Y. Zorian, “Effective March Algorithms for Testing Single-Order Addressed Memories,” in *Proceedings of the European Conference on Design Automation with the European Event in ASIC Design*, 1993, pp. 499–505. doi: 10.1109/EDAC.1993.386425.

[96] Y.-X. Chen and J.-F. Li, “Fault Modeling and Testing of 1T1R Memristor Memories,” in *Proceedings of the IEEE VLSI Test Symposium (VTS)*, Apr. 2015, pp. 1–6. doi: 10.1109/VTS.2015.7116247.

[97] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, “Modeling, Detection, and Diagnosis of Faults in Multilevel Memristor Memories,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 822–834, May 2015, doi: 10.1109/TCAD.2015.2394434.

[98] M. Wang, N. Deng, H. Wu, and Q. He, “Theory Study and Implementation of Configurable ECC on RRAM Memory,” in *Proceedings of the Non-Volatile Memory Technology Symposium (NVMTS)*, Oct. 2015, pp. 1–3. doi: 10.1109/NVMTS.2015.7457488.

[99] M. Liu and K. Chakrabarty, “Online Fault Detection in ReRAM-based Computing Systems by Monitoring Dynamic Power Consumption,” in *Proceedings of the IEEE International Test Conference (ITC)*, Nov. 2020, pp. 1–10. doi: 10.1109/ITC44778.2020.9325259.

[100] M. Liu, L. Xia, Y. Wang, and K. Chakrabarty, “Fault Tolerance for RRAM-Based Matrix Operations,” in *Proceedings of the IEEE International Test Conference (ITC)*, Oct. 2018, pp. 1–10. doi: 10.1109/TEST.2018.8624687.

[101] C. Liu, M. Hu, J. P. Strachan, and H. (Helen) Li, “Rescuing Memristor-based Neuromorphic Design with High Defects,” in *Proceedings of the Design Automation Conference (DAC)*, Jun. 2017, pp. 1–6. doi: 10.1145/3061639.3062310.

[102] Q. Lou, T. Gao, P. Faley, M. Niemier, X. S. Hu, and S. Joshi, “Embedding Error Correction into Crossbars for Reliable Matrix Vector Multiplication Using Emerging Devices,” in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, Aug. 2020, pp. 139–144. doi: 10.1145/3370748.3406583.

[103] C.-F. Lee, H.-J. Lin, C.-W. Lien, Y.-D. Chih, and J. Chang, “A 1.4Mb 40-nm Embedded ReRAM Macro with 0.07um ^2 Bit Cell, 2.7mA/100MHz Low-Power Read and Hybrid Write Verify for High Endurance Application,” in *Proceedings of the IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov. 2017, pp. 9–12. doi: 10.1109/ASSCC.2017.8240203.

[104] B. Zhang, N. Uysal, D. Fan, and R. Ewetz, “Redundant Neurons and Shared Redundant Synapses for Robust Memristor-based DNNs with Reduced Overhead,” in *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*, Sep. 2020, pp. 339–344. doi: 10.1145/3386263.3406910.

[105] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, “Detection, diagnosis, and repair of faults in memristor-based memories,” in *Proceedings of the IEEE VLSI Test Symposium (VTS)*, Apr. 2014, pp. 1–6. doi: 10.1109/VTS.2014.6818762.

[106] L. Chen *et al.*, “Accelerator-Friendly Neural-Network Training: Learning Variations and Defects in RRAM Crossbar,” in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2017, pp. 19–24. doi: 10.23919/DATE.2017.7926952.

[107] W. Li, Y. Wang, H. Li, and X. Li, “RRAMedy: Protecting ReRAM-Based Neural Network from Permanent and Soft Faults During Its Lifetime,” in *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, Nov. 2019, pp. 91–99. doi: 10.1109/ICCD46524.2019.00020.

[108] B. Zhang, N. Uysal, D. Fan, and R. Ewetz, “Handling Stuck-at-faults in Memristor Crossbar Arrays using Matrix Transformations,” in *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, Jan. 2019, pp. 474–479. doi: 10.1145/3287624.3287707.

[109] H. Shin, M. Kang, and L. S. Kim, “Fault-free: A Fault-resilient Deep Neural Network Accelerator based on Realistic ReRAM Devices,” in *Proceedings - Design Automation Conference*, Dec. 2021, vol. 2021-December, pp. 1039–1044. doi: 10.1109/DAC18074.2021.9586286.

[110] N. Binkert *et al.*, “The gem5 simulator,” *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, p. 1, Aug. 2011, doi: 10.1145/2024716.2024718.

[111] Xiangyu Dong, Cong Xu, Yuan Xie, and N. P. Jouppi, “NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, Jul. 2012, doi: 10.1109/TCAD.2012.2185930.

[112] M. Poremba, T. Zhang, and Y. Xie, “NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems,” *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 140–143, Jul. 2015, doi: 10.1109/LCA.2015.2402435.

[113] W. Fei, H. Yu, W. Zhang, and K. S. Yeo, “Design Exploration of Hybrid CMOS and Memristor Circuit by New Modified Nodal Analysis,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1012–1025, Jun. 2012, doi: 10.1109/TVLSI.2011.2136443.

[114] M.-Y. Lin *et al.*, “DL-RSIM: a Simulation Framework to Enable Reliable ReRAM-based Accelerators for Deep Learning,” in *Proceedings of the International Conference on Computer-Aided Design (ISCAD)*, Nov. 2018, pp. 1–8. doi: 10.1145/3240765.3240800.

[115] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, “DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies,” in *Proceedings of the  IEEE International Electron Devices Meeting (IEDM)*, Dec. 2019, pp. 32.5.1–32.5.4. doi: 10.1109/IEDM19573.2019.8993491.

[116] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, “DNN+NeuroSim V2.0: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators for On-Chip Training,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 11, pp. 2306–2319, Nov. 2021, doi: 10.1109/TCAD.2020.3043731.

[117] T. Liu, W. Wen, L. Jiang, Y. Wang, C. Yang, and G. Quan, “A Fault-Tolerant Neural Network Architecture,” in *Proceedings of the Annual Design Automation Conference (DAC)*, Jun. 2019, pp. 1–6. doi: 10.1145/3316781.3317742.

[118] Y. Long, T. Na, and S. Mukhopadhyay, “ReRAM-Based Processing-in-Memory Architecture for Recurrent Neural Network Acceleration,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 12, pp. 2781–2794, Dec. 2018, doi: 10.1109/TVLSI.2018.2819190.

[119] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based Learning Applied to Document Recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[120] A. Krizhevsky, V. Nair, and G. Hinton, “CIFAR-10 (Canadian Institute for Advanced Research)”, [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[121] “Examples/mnist at master · pytorch/examples.” [Online]. Available: https://github.com/pytorch/examples

[122] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *in Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[123] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” May 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[124] C. Lammie, W. Xiang, B. Linares-Barranco, and M. R. Azghadi, “MemTorch: An Open-source Simulation Framework for Memristive Deep Learning Systems,” *arXiv:2004.10971 [cs]*, Apr. 2021, [Online]. Available: http://arxiv.org/abs/2004.10971

1. convolutional neural network [↑](#footnote-ref-1)
2. Convolutional Neural Networks [↑](#footnote-ref-2)
3. Network on Chip [↑](#footnote-ref-3)
4. Machine Learning (ML) [↑](#footnote-ref-4)
5. Arthur Samuel [↑](#footnote-ref-5)
6. Processing Element [↑](#footnote-ref-6)
7. Transaction Level Model [↑](#footnote-ref-7)