**كد رهگيري ثبت پروپوزال: 1599871**

**کد رهگیری ثبت پایان نامه:**





**تعهدنامه اصالت اثر**

اينجانب **سمیّه کیانی** دانشجـوي دوره کارشناسی ارشد رشته **کامپیوتر** دانشكده فنی و مهندسی به شماره دانشجويي **9813224002** كه از پایان نامه خود با عنوان:

**بهبود** **کارایی شبکه‌های عصبی** **عمیق** **با** **استفاده** **از** **حساب** **تقریبی** **و** **سنتز** **سطح** **بالا**

دفاع نموده­ام، بدين وسيله متعهد مي شوم:

- نتايج مندرج در اين پایان نامه توسط اينجانب به‌دست‌آمده و از صحت و اصالت برخوردار است و در مواردي كه از دستاوردهاي علمي و پژوهشي ديگران اعم از پايا ن نامه، كتاب، مقاله و غيره استفاده كرده­ام، رعايت كامل امانت را نموده، مطابق مقررات، آن ها را ارجاع داده و در فهرست منابع و مآخذ اقدام به ذكر آن‌ها نموده­ام.

- تمام يا بخشي از اين پایان نامه تا كنون توسط اينجانب يا فرد ديگري براي دريافت هرگونه مدرك تحصيلي (پايين تر، همسطح يا بالاتر) در هيچ كجا ارائه نگرديده است.

- كليه حقوق مادي و معنوي حاصل از اين پایان نامه متعلق به دانشگاه بوعلي سينا بوده و هرگـونه بهره مندي و يا نـشر دستاوردهاي حاصل از اين پژوهش(و يا به صورت تركيبي با اطلاعات ديگر) اعم از چاپ كتاب، مقاله، ثبت اختراع و غيـره (چه در زمان دانشجـويي و يا پـس از فراغـت از تحصيـل) با همـاهنـگي استـاد (ان) راهنمـا و مشاور و به نام "دانشگاه بوعلي سينا" و یا "**Bu-Ali Sina University**" صورت گيرد.

- در تمامي مقالات حاصل از اين پایان نامه، براي چاپ و ارائه در مجلات داخلي و خارجي، كنفرانس­ها و يا سخنراني­ها آدرس هاي ذيل را درج نمايم:

………………………..……**, Bu-Ali Sina University, Hamedan, Iran.**

**مقالات خارجی**

**...................، گروه ....................، دانشكده ..................، دانشگاه بوعلي سينا، همدان.**

**مقالات خارجی**

- حقوق مادي و معنوي تمام افرادي كه در به دست آمدن نتايج اصلي اين پایان نامه تأثيرگذار بوده­اند را در مقالات مستخرج از پایان نامه رعايت نموده و در تمامي آن ها نام استاد (ان) راهنما و نشاني الكترونيكي دانشگاهي آنان را قيد نمايم.

- در كليه مراحل انجام اين پایان نامه، در مواردي كه به حوزه اطلاعات شخصي افراد دسترسي داشته يا از آن ها استفاده كرده ام، اصل رازداري، ضوابط و اصول اخلاقي پژوهش را رعايت نموده ام.

نام و نام خانوادگي دانشجو:

تاريخ:

امضاء

****

**دانشکده فنی مهندسی**

**گروه آموزشی کامپیوتر**

**پایان‌نامه برای دریافت درجه کارشناسی ارشد در رشته کامپیوتر گرایش معماری سیستم های کامپیوتری**

**عنوان:**

**بهبود** **کارایی شبکه‌های عصبی** **عمیق** **با** **استفاده** از **حساب** **تقریبی** **و** **سنتز** **سطح** **بالا**

**استاد راهنما:**

**دکتر حاتم عبدلی**

**استاد مشاور:**

**دکتر محرم منصوری زاده**

**نگارش:**

**سمیّه کیانی**

**30 آذر 1400**

تقدیر وتشکر

سپاس خدای را که سخنوران، در ستودن او بمانند و شمارندگان، شمردن نعمت های او ندانند و کوشندگان، حق او را گزاردن نتوانند. و سلام و دورد بر محمّد و خاندان پاک او، طاهران معصوم، هم آنان که وجودمان وامدار وجودشان است؛ و نفرین پیوسته بر دشمنان ایشان تا روز رستاخیز…

بدون شک جایگاه و منزلت معلم، اجّل از آن است که در مقام قدردانی از زحمات بی شائبه ی او، با زبان قاصر و دست ناتوان، چیزی بنگاریم.

اما از آنجایی که تجلیل از معلم، سپاس از انسانی است که هدف و غایت آفرینش را تامین می‌کند و سلامت امانت هایی را که به دستش سپرده اند، تضمین؛ بر حسب وظیفه و از باب ” من لم یشکر المنعم من المخلوقین لم یشکر اللَّه عزّ و جلّ” :

ازپدر و مادر عزیزم… این دو معلم بزرگوارم… که همواره بر کوتاهی و درشتی من، قلم عفو کشیده و کریمانه از کنار غفلت هایم گذشته اند و در تمام عرصه های زندگی یار و یاوری بی چشم داشت برای من بوده اند؛

از استاد با کمالات و شایسته؛ جناب آقای دکتر حاتم عبدلی که در کمال سعه صدر، با حسن خلق و فروتنی، از هیچ کمکی در این عرصه بر من دریغ ننمودند و زحمت راهنمایی این پایان نامه را بر عهده گرفتند؛

از استاد صبور و با تقوا ، جناب آقای دکتر محرم منصوری­زاده، که زحمت مشاوره این پایان­نامه را در حالی متقبل شدند که بدون مساعدت ایشان، این پایان نامه به نتیجه مطلوب نمی رسید؛

و از استاد فرزانه و دلسوز؛ جناب آقای دکتر … که زحمت داوری این پایان­نامه را متقبل شدند؛ کمال تشکر و قدردانی را دارم، باشد که این خردترین، بخشی از زحمات آنان را سپاس گوید.

و اینکه در پایان این مقطع تحصیلی ام این دستاورد را تقدیم می‌کنم به:

مادر مهربانم که برای به ثمر رساندن تواناییهای بالقوه ام از شیره جان مایه گذاردند و چون باغبانی دلسوز زحمت مراقبتم را برای رسیدن به این مرحله شکوفایی به جان خریدند. و نیز به همه کسانی که در این راه مشوق و راهنمای من بودند.

شیوه تهیه جدول مربوط به چکیده فارسی

|  |  |  |
| --- | --- | --- |
| **دانشگاه بوعلی سینا**  **مشخصات رساله/پايان نامه تحصيلي** | | |
| **عنوان: بهبود کارایی شبکه‌های عصبی عمیق با استفاده از حساب تقریبی و سنتز سطح بالا** | | |
| **نام نويسنده: سمیّه کیانی** | | |
| **نام استاد/اساتید راهنما: دکتر حاتم عبدلی** | | |
| **نام استاد/اساتید مشاور: دکتر محرم منصوری زاده** | | |
| **دانشكده : فنی و مهندسی** | **گروه آموزشی: کامپیوتر** | |
| **رشته تحصيلي: مهندسی کامپیوتر** | **گرایش تحصیلی: معماری کامپیوتر** | **مقطع تحصيلي: کارشناسی ارشد** |
| **تاريخ تصويب پروپوزال: 08/10/ 1399** | **تاريخ دفاع: 31/06/1400** | **تعداد صفحات: 84** |
| **چكيده:** (**این قسمت حداکثر در دو صفحه تایپ شود. سایز چکیده 10 می باشد که شروع متن چکیده در زیر کلمه و با یک فرورفتگی به فاصله 5/1 سانتی متری(first line) در شروع متن لحاظ گردد.)**  **الگوریتم شبکه‌های عصبی از الگوریتم های یادگیری عمیق هستند. الگوریتمهای متعددی برای حل مشکلات سرعت و زمان اجرای این الگوریتم ها پیشنهاد شده ولی متاسفانه تا کنون به نتیجه ای که بتوان این الگوریتم ها را در زمان مناسب اجرا کرد، نرسیده اند. از آنجایی که این الگوریتم در بسیاری از حوزه­های هوش مصنوعی مانند بینایی ماشین، تشخیص گفتار، تشخیص جسم و ... کاربرد دارند. ما در این جا قصد داریم با استفاده از رایانش تقریبی آنها را تا جایی که ممکن است، بهبود دهیم. در این پایان‌نامه، نسخه‌ای جدید از روش های محاسبات تقریبی برای کاهش زمان اجرا در الگوریتم های شبکه‌های عصبی است، از آنجایی که می دانیم در این الگوریتم ها با مسئله زیاد بودن زمان اجرای الگوریتم­ها و کاربردهای مرتبط با شبکه عصبی مواجه هستیم و می خواهیم این زمان را با استفاده از رایانش تقریبی و سنتز سطح بالا کم کنیم. در واقع دقت را مقداری قربانی زمان اجرای محاسبات می‌کنیم. از طرفی هم مطلع هستیم که شبکه­های عصبی تحمل پذیری خطا دارند، پس لزومی ندارد که پیاده­سازی الگوریتم های محاسباتی خیلی دقیق باشند؛ در نتیجه می خواهیم دقت محاسبات را به اندازه ای که مورد نیاز است و به این الگوریتم لطمه وارد نکند، طراحی کنیم. نتیجه این کار باعث کوچکتر شدن مدار، بالا بردن سرعت پردازش یا فرکانس کار مدار و همچنین کاهش توان مصرفی مدار است. هدف از انجام این پایان نامه، بررسی عوامل موثر در بهبود سرعت و زمان اجرای شبکه‌های عصبی کانولوشن در پردازش تصویر است. شبکه عصبی کانولوشن به عنوان یکی از بهترین شبکه ها در زمینه پردازش و کلاس بندی تصاویر بوده که می‌تواند در بسیاری از کارها و جاهای مختلف استفاده شود. هدف ما در نهایت به حداقل رساندن زمان اجرای الگوریتم های شبکه عصبی با استفاده از محاسبات تقریبی است. این پروپزال براساس مقاله [****[9](#ASNet)] نوشته شده است.** | | |
| **واژه­های کلیدی: شبکه‌های عصبی عمیق، یادگیری عمیق، محاسبات تقریبی، FPGA، ابزارهای سنتز سطح بالا** | | |

تنها در 3 سال، محققان از 8 لایهAlexNet به 152 لایه ResNet رسیدند و میزان خطای چالش ImageNet را به کمتر از **4٪** کاهش دادند.

**فهرست جدول ها ·**

[جدول ‏2‑2مقایسه شبکه‌های مختلف کانولوشنی 35](#_Toc92200591)

[جدول ‎1‑1تنظیمات مجموعه داده‌ها و پارامترها 70](#_Toc92200592)

[جدول ‎1‑2 Experimental Statistics 70](#_Toc92200593)

[جدول ‎1‑3نتایج: روش تقریب حالت کانولوشن تکی 70](#_Toc92200594)

**فهرست مطالب**

[1 فصل اول: مقدمه 1](#_Toc93634552)

[1-1 مقدمه 1](#_Toc93634553)

[1-2 تعریف مسئله و ضرورت انجام کار 1](#_Toc93634554)

[1-3 مفروضات پژوهش 2](#_Toc93634555)

[1-4 اهداف پژوهش 2](#_Toc93634556)

[1-5 کاربرد‌های پژوهش 3](#_Toc93634557)

[1-6 جمع‌بندی 3](#_Toc93634558)

[2 فصل دوم: مبانی نظری وادبیات پژوهش 4](#_Toc93634559)

[1-7 مروری بر شبکه‌های عصبی مصنوعی 5](#_Toc93634560)

[1-7-1 کاربردهای شبکه‌های عصبی 5](#_Toc93634561)

[1-7-2 انواع شبکه عصبی مصنوعی 5](#_Toc93634562)

[1-7-2-1 شبکه‌های عصبی چگونه کار می‌کنند؟ 6](#_Toc93634563)

[1-7-3 شبکه‌های عصبی کانولوشنی CNN 8](#_Toc93634564)

[1-7-3-1 کاربرد شبکه‌های عصبی کانولوشن 9](#_Toc93634565)

[1-7-3-2 جزئیات پردازش تصویر با استفاده از شبکه‌های عصبی کانولوشنیCNN 11](#_Toc93634566)

[1-7-3-2-1 معماری CNN سنتی 11](#_Toc93634567)

[1-7-3-2-2 شبکه‌های عصبی کانولوشنی LexNet 12](#_Toc93634568)

[1-7-3-2-3 شبکه‌های عصبی کانولوشنی AlexNet 13](#_Toc93634569)

[1-7-3-2-1 شبکه‌های عصبی کانولوشنی ResNet 17](#_Toc93634570)

[1-8 مروری بر رایانش تقریبی approximate computing 23](#_Toc93634571)

[1-9 مروری بر حساب تقریبی approximate arithmetic 25](#_Toc93634572)

[1-10 جمع‌بندی 28](#_Toc93634573)

[3 فصل سوم: پیشینه تحقیق 30](#_Toc93634574)

[3-1 مروری بر تئوری‌ها و مطالعات پیشین 32](#_Toc93634575)

[3-1-1 بهبود شبکه‌های عصبی مصنوعی عمیق با استفاده از ابزارهای سنتز سطح بالا 32](#_Toc93634576)

[3-1-2 بهبود شبکه‌های عصبی مصنوعی عمیق با اجرای موازی آنها 36](#_Toc93634577)

[3-1-3 بهبود شبکه‌های عصبی مصنوعی عمیق با استفاده از محاسبات تقریبی 45](#_Toc93634578)

[3-1-4 بیان پژوهش جاری 65](#_Toc93634579)

[3-1-5 خلاصه پژوهش 65](#_Toc93634580)

[4 فصل چهارم: روش پیشنهادی 67](#_Toc93634581)

[5 فصل پنجم: نتیجه و ارزیابی 69](#_Toc93634582)

[6 فصل ششم: نتیجه گیری 71](#_Toc93634583)

[7 مراجع 73](#_Toc93634584)

**فهرست شکل ها ·**

[شکل ‏2‑2 همپوشانی اسکن مغزی مؤسسه‌ ملی NIDA و مؤسسه‌ ملی سلامت آمریکا با شبکه عصبی 5](#_Toc92204745)

[شکل ‏2‑3 مقایسه هوش مصنوعی، یادگیری ماشین، یادگیری عمیق و شبکه عصبی 6](#_Toc92204746)

[شکل ‏2‑6 تصویر نورون متعلق‌ به مؤسسه ملی NIDA سلامت آمریکا 7](#_Toc92204747)

[شکل ‏2‑9 ساختار شبکه‌های عصبی 10](#_Toc92204748)

[شکل ‏2‑11 نمای کلی شبکه عصبی مصنوعی عمیق 11](#_Toc92204749)

[شکل ‏2‑12اعمال تصویر ورودی به شبکه MLP؛ تصویر به بردار تبدیل و تحویل به شبکه MLP 11](#_Toc92204750)

[شکل ‏2‑14 ساختار لایه‌های شبکه‌های عصبی کانولوشنی 13](#_Toc92204751)

[شکل ‏2‑15 ویژگی‌های مختلف تشخیص لایه‌های ورودی 13](#_Toc92204752)

[شکل ‏2‑16 شبکه کانولوشنی برای پردازش تصویر 17](#_Toc92204753)

[شکل ‏2‑17نمونه ای از عملكرد و تجمع 18](#_Toc92204754)

[شکل ‏2‑18 معماری CNN سنتی 19](#_Toc92204755)

[شکل ‏2‑19 مراحل شبکه‌های عصبی کانولوشنی در پردازش تصویر 19](#_Toc92204756)

[شکل ‏2‑20معماری LeNet 20](#_Toc92204757)

[شکل ‏2‑21ساختار شبکه عصبی LexNet 20](#_Toc92204758)

[شکل ‏2‑22ساختار شبکه کانولوشنی 21](#_Toc92204759)

[شکل ‏2‑23 ساختار شبکه AlexNet 21](#_Toc92204760)

[شکل ‏2‑24معماری کلی CNN شامل یک لایه ورودی، چندین لایه متراکم متناوب و حداکثر ترکیب ، یک لایه کاملاً متصل و یک لایه طبقه‌بندی است. 23](#_Toc92204761)

[شکل ‏2‑25جزئیات پیاده‌سازی شبکه‌های عصبی کانولوشنی AlexNet 24](#_Toc92204762)

[شکل ‏2‑26خطای تست و آموزش شبکه 29](#_Toc92204763)

[شکل ‏2‑28 نمایش مزایا در قالب خلاصه 39](#_Toc92204764)

[شکل ‏2‑29 نمودار کارایی روش محاسبات تقریبی 40](#_Toc92204765)

[شکل ‎1‑4 The compilation flow with the proposed ALAMO RTL compiler 47](#_Toc92204766)

[شکل ‎1‑5 محاسبات تقریبی برای DNN ها 58](#_Toc92204767)

[شکل ‎1‑6 نمای بهینه‌سازی بین لایه ای برای یادگیری عمیق 63](#_Toc92204768)

[شکل ‎1‑7 نمای بهینه‌سازی بین لایه ای برای یادگیری عمیق به صورت جزیی 65](#_Toc92204769)

[شکل ‎1‑1 مقایسه بین ApproxANN و AxNN 69](#_Toc92204770)

[شکل ‎1‑1 مقایسه بین ApproxANN و AxNN 70](#_Toc92204771)

[شکل ‎1‑2Illustration of typical approximations of computational path of DNNs.The methods targeted in this work are marked. 71](#_Toc92204772)

[شکل ‎1‑3 سناریوهای مختلف محاسبات تقریبی 72](#_Toc92204773)

فهرست علائم اختصاري ·

<https://github.com/BVLC/caffe>

<https://nbviewer.jupyter.org/github/BVLC/caffe/blob/master/examples/01-learning-lenet.ipynb>

<https://nbviewer.jupyter.org/github/BVLC/caffe/blob/master/examples/00-classification.ipynb>

پیوست ها ·

فهرست مراجع و یا فهرست منابع ·

چکیده به زبان انگلیسی

# فصل اول: مقدمه

## مقدمه

در فصل اول، ابتدا مسأله تعریف و ضرورت انجام پژوهش مربوط به بهبود کارایی شبکه‌های عصبی عمیق با استفاده از ابزار سنتز سطح بالا به صورت خلاصه مطرح و سپس، مفروضات، اهداف و کاربرد‌های پژوهش بیان می‌شود. نهایتاً؛ روند موجود در این پژوهش، به طور مختصر تشریح می‌گردد.

## تعریف مسئله و ضرورت انجام کار

در الگوریتم‌های شبکه‌های عصبی و کاربردهای مختلف مرتبط با آن، یکی از چالش‌های اصلی، زیاد بودن زمان اجرای محاسبات است. برای سرعت بخشیدن به اجرای پردازش‌های مربوط به شبکه عصبی و یادگیری عمیق، بسترهای سخت‌افزاری مختلفی مانند استفاده ازCPU،TPU و GPU موردِاستفاده و ارزیابی قرار گرفته است[1] و [2]. با این حال هنوز در بسیاری از کاربردها زمان اجرای طولانی مهمترین معضل است. شبکه‌های عصبی در زمینه‌های مختلف مورد استفاده قرار گرفته است، بخصوص کاربردهایی که نسبت به خطا تحمل‌پذیر هستند. بنابراین با توجه به اینکه در این کاربردها، شبکه‌های عصبی ذاتاً با کمی خطا سروکار دارند، لزومی ندارد که پیاده‌سازی الگوریتم‌های محاسباتی آنها هم خیلی دقیق باشد[3]. در نتیجه می‌خواهیم یک مقداری از دقت محاسبات کم کنیم و سرعت را در عوض بالا ببریم یا به عبارت دیگر، دقت محاسبات را به اندازه‌ای که مورد نیاز است طراحی کنیم. به همین دلیل از این ویژگی شبکه‌های عصبی استفاده شده تا با استفاده از حساب تقریبی، حجم محاسبات و در نتیجه زمان اجرای الگوریتم‌ها را کاهش دهیم. در شبکه‌های عصبی مقدار زیادی عملیات حسابی مشاهده می‌کنیم که می‌توان با استفاده از حساب تقریبی دقت را تا حدی قربانی سرعت و بهره‌وری انرژی کرد و بین دقت خروجی و سرعت محاسبات مصالحه برقرار کرد.

پردازنده‌های GPU برای کارهای محاسباتی تکراری و موازی مناسب است و کارایی آن در یادگیری ماشین و بسیاری از انواع محاسبات علمی مناسب است. از طرفی زمان اجرای شبکه‌های عصبی بسیار طولانی است، مخصوصاً برای سیستم‌های تعبیه شده[[1]](#footnote-1) و سیستم‌هایی که از CPU استفاده می‌کنند، بسیار کندند. متأسفانه، از آنجا که GPU ها انرژی زیادی مصرف می‌کنند، برای همه سیستم‌های تعبیه شده مناسب نیستند. بنابراین انواع پیاده‌سازی مبتنی بر FPGA برای سیستم‌های تعبیه شده بیدرنگ و کم‌مصرف ارائه شده است. قابلیت برنامه‌ریزی و تنظیم مجدد FPGA ها امکان ارزیابی سریع‌تر طراحی سفارشی نسبت به ASIC را فراهم می‌کند و بنابراین آنها گزینه‌های پیاده‌سازی بهتری هستند. FPGA ها می‌توانند عملکرد بهتری با مصرف انرژی کمتر ارائه دهند که برای سیستم‌عامل‌های تعبیه شده ضروری است. با این حال، توپولوژی و اندازه شبکه می‌تواند از نظر منابع و اتصالات به یک مشکل برای FPGA تبدیل شود. از طرفی فرآیند آموزش برای یادگیری عمیق در FPGA‌ها دشوار است؛ زیرا به دلیل تغییرات همزمان چندین پارامتر، بسیاری از عملیات باید به صورت کلی و همزمان انجام شود. پیاده‌سازی‌های قبلی FPGA مبتنی بر ابزارهای سنتز سطح بالا (HLS) به انعطاف‌پذیری خوب، برنامه‌ریزی آسان و زمان طراحی کوتاه دست یافته‌اند، اما استفاده از سخت‌افزار و حافظه آنها ناکارآمد است و ممکن است اجازه ندهد از ساختارهای سخت‌افزاری سطح پایین برای دستیابی به عملکرد و توان بالاتر استفاده شود[4].

ریزپردازنده در حال ظهور VPU نوع خاصی از شتاب‌دهنده هوش مصنوعی است که برای سرعت‌بخشیدن به وظایف بینایی ماشین طراحی شده است. این معماری اختصاصی، برای کاهش میزان حافظه و منابع محاسباتی (ضرب ماتریس وزن و تابع فعالسازی) مورد نیاز برای پیش بینی‌ها در مدل‌های شبکه عصبی، از کوانتیزیشن استفاده می‌کند. اینتل با محاسبات تعبیه‌شده و لبه‌ای، واحد پردازش بینایی (Movidius Myriad VPU) را پیشنهاد کرد، که یک تراشه قابل برنامه‌ریزی با یک شتاب­دهنده سخت‌افزاری اختصاصی برای استنتاج‌های شبکه عصبی عمیق در بینایی کامیپوتر است[5].

با وجود پژوهش‌های گسترده‌ای که برای افزایش کارایی و کاهش انرژی مصرفی سخت‌افزار شبکه­های عصبی انجام شده است، هنوز هم این چالش‌ها باعث شده که به‌کارگیری بعضی از کاربردهای مبتنی بر شبکه‌های عصبی، بخصوص در دستگاه‌هایی که منابع و توان محاسباتی محدودتری دارند، غیرممکن یا پرهزینه و کند باشد. بنابراین در این پژوهش با هدف کاهش زمان پردازش محاسبات شبکه‌های عصبی، دقت محاسبات را تا حدِممکن در سخت‌افزار مدار محاسباتی کاهش می‌دهیم تا مدارمان کوچکتر و حداکثر فرکانس کاری مدار بیشتر شود. درواقع با هدف کاهش زمان پردازش محاسبات شبکه‌های عصبی، قصد داریم که با استفاده از روش‌های حساب کامپیوتری و حساب تقریبی، مساحت و تاخیر مدار را کاهش دهیم. در کنار افزایش سرعت انجام محاسبات، مزیت دیگر این کار کاهش توان مصرفی مدار است.

## مفروضات پژوهش

برخی از مفروضات این پژوهش به صورت زیر است:

* استفاده از محاسبات تقریبی[[2]](#footnote-2) با وجود اینکه کمی از دقت را کم می‌کند ولی باعث افزایش کارایی، سرعت محاسبات و بهره‌وری انرژی سخت‌افزار مورد استفاده در شبکه‌های عصبی می‌شود.
* استفاده از HLS به طراحی بهینه‌تر و سریع‌تر سخت‌افزار شبکه‌های عصبی کمک می‌کند.

## اهداف پژوهش

در این تحقیق سعی می‌کنیم با استفاده از روش‌های مختلف حساب کامپیوتری سرعت و زمان اجرای الگوریتم‌های مختلف شبکه عصبی را بهبود دهیم. با توجه به مرور ادبیات انجام شده، جنبه‌های نوآوری تحقیق به صورت ذیل خواهد بود:

* با استفاده از سنتز سطح بالا و حساب تقریبی، سخت‌افزار محاسبات شبکه‌های عصبی عمیق با حداقل دقت مورد نیاز ولی با سرعت پردازش بالاتر طراحی خواهد شد که البته این کاهش دقت محاسبات، نباید مشکلی در اجرای الگوریتم‌ها ایجاد کند.
* استفاده از اپراتورهای حساب تقریبی برای کاهش توان مصرفی مدار سخت‌افزاری شبکه عصبی در کنار افزایش کارایی مدار.
* ارائه روش محاسباتی روی عملیات ریاضی مانند ضرب برای حداقل کردن هزینه‌های مربوط به مسأله.

## کاربرد‌های پژوهش

کاربرد‌های این پژوهش در موارد زیر است:

- بهبود سرعت پردازش سخت‌افزار شبکه‌های عصبی و شبکه‌های عصبی عمیق

- بهبود کارایی و بهره‌وری انرژی در پردازش‌های مرتبط با هوش مصنوعی در تلفن همراه و دستگاه‌هایی که سرعت محاسبات و توان مصرفی از اهمیت ویژه‌ای برخوردار است.

## جمع‌بندی

در این پایان‌نامه روشی برای بهبود سرعت اجرای الگوریتم‌های شبکه عصبی عمیق با در نظر گرفتن روابط ضرب‌ها مورد مطالعه قرار می‌گیرد. در فصل اول، تعریف مساله و ضرورت انجام کار، مفروضات و اهداف پژوهش، کاربرد‌های پایان‌نامه را مطرح کردیم.

مراحل بعدی تحقیق به شرح زیر است: در فصل دوم؛ مطالعات مرتبط با موضوع پژوهش شامل شبکه‌های عصبی عمیق، رایانش تقریبی و حساب تقریبی بیان می‌شود. در این فصل همچنین درباره انواع مدل‌های شبکه عصبی مختلف و روش‌های مختلف محاسبات تقریبی و کاربردهای آن توضیح داده می‌شود. در فصل سوم، درباره پیشینه تحقیق از جمله مروری بر تئوری‌ها و مطالعات پیشین، بیان پژوهش جاری توضیحاتی داده می‌شود. در فصل چهارم درباره روش پیشنهادی توضیح داده می‌شود، در فصل پنجم یک مدل شبکه عصبی در نظر گرفته شده و یک الگوریتم محاسبات تقریبی برای حل مشکل سرعت اجرای آن با توجه به شرح مساله و مفروضات مطرح شده پیشنهاد می‌شود و الگوریتم پیشنهادی با یک الگوریتم مناسب در زمینه حساب تقریبی و رایانش تقریبی با استفاده از شاخص‌هایی مقایسه می‌شود. سپس در نهایت در فصل ششم که فصل آخر می‌باشد، نتیجه‌گیری است و از این پژوهش نتیجه‌گیری­های لازم گرفته می‌شود و برای مطالعات آتی به محققین پیشنهاد‌هایی مطرح می‌شود.

# فصل دوم: مبانی نظری وادبیات پژوهش

## مروری بر شبکه‌های عصبی مصنوعی

شبکه‌های عصبی مصنوعیANN سیستم‌های اتصالگر، سیستم‌های محاسبه‌کننده‌ای هستند که از شبکه‌های عصبی زیستی الهام گرفته شده‌اند. این سیستم‌ها، با بررسی مثال‌ها، فعالیت‌ها را یاد می‌گیرند (به عبارت دیگر عملکرد خود را در انجام فعالیت‌ها به مرور بهبود می‌دهند) و عموماً این اتفاق بدون هیچ برنامه‌نویسی مختص به فعالیت انجام نمی‌شود[6].

علی‌رغم مزایای بسیار زیادی که این شبکه‌ها دارند، سرعت اجرا و زمان اجرای بهینه‌ای ندارند و زمان زیادی برای اجرای آنها صرف می‌شود؛ برای این منظور راهکارهای فراوانی برای بهبود این الگوریتم‌ها به کار گرفته شده است و مسائلی که در این زمینه وجود دارند بسیار متنوع هستند؛ تاکنون هم از راه‌های تقریب نرم افزاری و هم از تقریب‌سخت‌افزاری متفاوتی استفاده شده است، با توجه به اهمیت شبکه‌های عصبی عمیق و پردازش تصویر تاکنون کار‌های تحقیقاتی بسیار زیادی در مورد آن انجام شده است که ما در این جا قصد داریم این مسائل را با تقریب‌های سخت‌افزاری بهبود دهیم. مطالعاتی در زمینه شبکه‌های عصبی و نحوه بهبود آنها وجود دارند؛ که بعد از معرفی شبکه‌های عصبی، نمونه‌هایی از آنها در ادامه آمده است[7].

هدف آغازین روش شبکه‌های عصبی، حل مسئله به روش ذهن انسان بود. به مرور زمان، توجه به تطبیق برخی توانایی‌های ذهنی خاص معطوف و از مسیر اصلی زیست شناختی منحرف شد.توسط الگوریتمِ شبکه‌های عصبی، می‌توان مدل‌های مختلف و پیچیده‌ای را شناخت. شبکه‌های عصبی در فعالیت‌های متنوعی استفاده شده‌اند، از جمله بینایی ماشین، شناسایی گفتار، ترجمه ماشینی، فیلترینگ شبکه‌های اجتماعی، بازی‌های رومیزی و بازی‌های ویدیویی و تشخیص پزشکی.

کاربردهای شبکه‌های عصبی این است که در سال‌های اخیر مطالعاتی انجام شده‌اند که از الگوریتم‌های یادگیری عمیق برای داده‌های عصبی استفاده می‌کنند. خطوط لوله مورد استفاده در این مطالعات بیشتر به مراحل مختلف پردازش برای استخراج ویژگی[[3]](#footnote-3) نیاز دارند، اگرچه پیشرفت‌های مدرن در یادگیری عمیق برای طبقه‌بندی تصویر[[4]](#footnote-4) می‌تواند چارچوبی قدرتمند برای ایجاد ویژگی‌های خودکار و تجزیه و تحلیل ساده‌تر ارائه دهد.

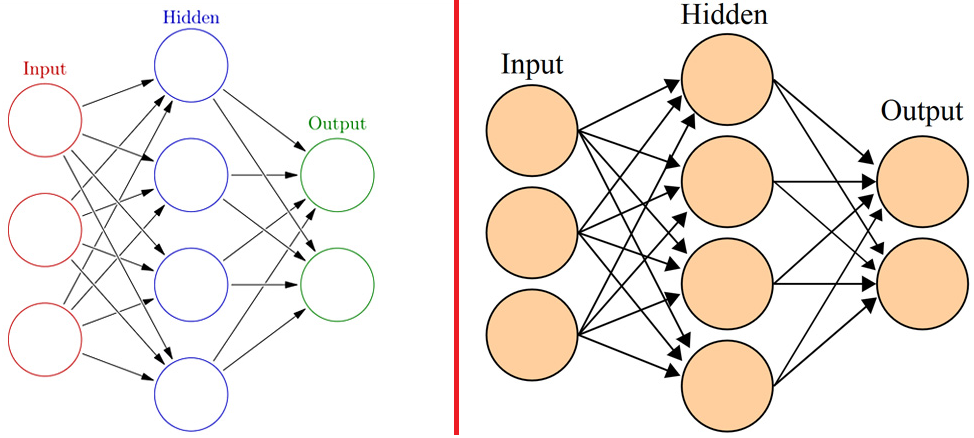
امروز به قدری استفاده از سیستم‌های هوشمند و به ویژه شبکه عصبی مصنوعی گسترده شده است که می‌توان این ابزارها را در ردیف عملیات پایه ریاضی و به عنوان ابزارهای عمومی و مشترک، طبقه‌بندی کرد[8].

Residual and plain convolutional neural networks for 3D brain MRI classification

### انواع شبکه عصبی مصنوعی

یادگیری عمیق در چند سال گذشته موفقیت‌های عظیمی را در حوزه‌های مختلف برنامه نشان داده است. این زمینه جدید از یادگیری ماشین به سرعت در حال رشد است و در بیشتر حوزه برنامه‌های کاربردی با برخی از شیوه‌های جدید برنامه کاربردی، که به باز شدن فرصت‌های جدید کمک می‌کند. روشهای متفاوتی در زمینه‌های مختلف روشهای یادگیری ارائه شده که شامل یادگیری تحت نظارت، نیمه تحت نظارت و بدون نظارت است. نتایج تجربی نشان‌دهنده عملکرد پیشرفته یادگیری عمیق نسبت به روشهای سنتی یادگیری ماشین در زمینه پردازش تصویر، بینایی رایانه‌ای، تشخیص گفتار، ترجمه ماشینی، تصویربرداری پزشکی، پردازش اطلاعات پزشکی، رباتیک و کنترل، بیو‌اطلاعات، پردازش زبان طبیعی (NLP) ، امنیت سایبری و بسیاری دیگر است[9].

The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches



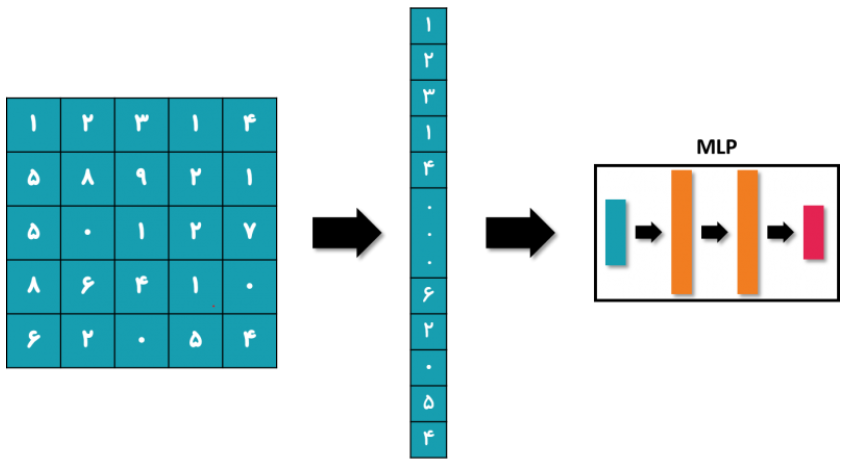
شکل ‏2‑9 ساختار شبکه‌های عصبی

شبکه‌های عصبی داده‌ها را از ورودی دریافت و در لایه‌های مخفی خود آن‌ها را تحلیل می‌کنند تا نهایتا یک خروجی ارائه دهند. این داده‌ها می‌توانند گروهی از تصاویر، صداها، نوشته‌ها و… باشند که باید ترجمه و برای یک ماشین قابل درک شوند. به کمک شبکه‌های عصبی، اطلاعات را طبقه‌بندی می‌کنیم؛ اطلاعات مختلف می‌توانند بر اساس شباهت به مثالی مشخص، گروه‌بندی شوند. آن‌ها حتی می‌توانند امکانات و داده‌های لازم برای تغذیه به یک الگوریتم دیگر را هم فراهم و طبقه‌بندی کنند[10].

شبکه عصبی عمیق، شبکه عصبی‌‌ است که از سه لایه یا بیشتر نورون تشکیل شده باشد. همان‌طور که در شکل ‏2‑11 مشاهده می‌شود، لایه‌‌های مخفی می‌توانند تعداد زیادی زیرلایه را در خود جای دهد. در چنین حالتی آن را شبکه عصبی عمیق می‌نامند. به کمک شبکه‌های عصبی عمیق، مسائل به شدت پیچیده در زمینه‌های مختلف، مانند مسائل ساده قابل حل می‌شوند[11].

### شبکه‌های عصبی کانولوشنی CNN

تفاوت شبکه عصبی MLP و کانولوشن در ورودی است، مثلا تصویر... تصاویر معمولا به شکل یک ماتریس دوبعدی از اعداد نمایش داده می‌شوند. هر درایه در این ماتریس دوبعدی معادل با یک پیکسل هست. اگر یک تصویر 100×100 داشته باشیم، یعنی 10000 پیکسل داریم که به صورت دوبعدی به‌صورت مسالمت‌آمیز کنار هم‌دیگر نشسته‌اند. حال تصور کنید بخواهیم یک لایه ورودی برای این 10000 پیکسل بسازیم؛ باید 10000 نورون برای لایه ورودی شبکه MLP درنظر بگیریم. اضافه کردن نورون و لایه بیشتر به این شبکه MLP باعث می‌شود، شبکه ما شامل حجم بزرگی از پارامترها شود، محاسباتش هزینه‌بر باشد و البته Overfitting اتفاق بیفتد.



شکل ‏2‑12اعمال تصویر ورودی به شبکه MLP؛ تصویر به بردار تبدیل و تحویل به شبکه MLP

نقطه قوت شبکه CNN، در اینجاست که این شبکه‌ها به‌گونه‌ای طراحی شده‌اند که برای ورودی‌های با ساختار ماتریسی (دوبعدی و سه‌بعدی) به‌خوبی کار می‌کنند. شبکه MLP، ساختار داده‌های ورودی را عوض می‌کند و یک ماتریس دوبعدی 100×100 را تبدیل به یک بردار به ابعاد 10000 می‌کند. اما شبکه CNN ساختار ورودی را عوض نمی‌کند و به ارتباط بین پیکسل‌های همسایه اهمیت می‌دهد.

باید به ارتباط بین پیکسل‌های همسایه اهمیت دهیم، تا ساختار تصویر یا ورودی را عوض نکنیم به‌عبارتی تبدیل به بردار نکنیم و همان ورودی اصلی را به شبکه بدهیم. دقیقاً همان کاری که شبکهCNN انجام می‌دهد.

شبکه CNN برای تصویر چند‌بعدی داده‌های زیر انواع داده‌ای ممکن برای شبکه CNN مناسب است:

* **داده‌های یک بعدی**: سیگنال و sequence مثلا یک sequence از کلمات
* **داده‌های دوبعدی:** تصویر و طیف (spectogram) صوت
* **داده‌های سه بعدی:** ویدئو و تصاویر حجمی مثلا تصاویر MRI
* **داده‌های چهاربعدی**: تصاویر حجمی همراه با زمان مانند fMRI

بعد از آشنایی با مزایای شبکه کانولوشن و بررسی تفاوتش با شبکه MLP، باید معماری این شبکه را بررسی کنیم.

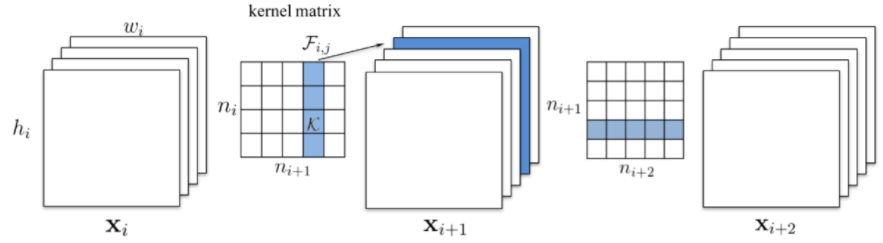
از سال 2011 به بعد، شبکه‌های مدرنِ یادگیریِ عمیق، لایه‌های کانولوشنی را با لایه‌های حداکثر تجمع جابه‌جا کرده اند، که بالای آنها چند لایه با اتصال کامل یا تُنُک، و در فوقانی‌ترین سطح یک لایه دسته‌بندی[[5]](#footnote-5) قرار می‌گیرد. یادگیری معمولاً بدون پیش-آموزش نظارت نشده انجام می‌شود.

این روش‌های یادگیری عمیق نظارت شده، اولین‌هایی بودند که در برخی فعالیت‌ها عملکردی قابل رقابت با انسان را داشتند[12].

شبکه‌های عصبی کانولوشن CNN یکی از مهمترین روش‌های یادگیری عمیق هستند که در آنها چندین لایه با روشی قدرتمند آموزش می‌بینند؛ این روش بسیار موثر بوده و یکی از رایج‌ترین روشها در کاربردهای مختلف بینایی کامپیوتر است. به سه دلیل شبکه‌های کانولوشنی بیشتر برای پردازش تصویر و بینایی کامپیوتر انتخاب می‌شوند: نسبت به جابه‌جایی و اندازه ویژگی‌ها حساس نیستند و دارای پیاده‌سازی سریع هستند.

CNN ها برای پردازش تصاویر و سایر داده‌های دو بعدی مناسب هستند و نتایج برتری را هم در کاربردهای تصویر و هم در گفتار نشان داده‌اند. می‌توان آن ها را با پس-نشر استاندارد آموزش داد. آموزش CNN ها از سایر شبکه‌های عصبی منظم، پیش‌نگر و عمیق آسان‌تر است و در آن پارامترهای بسیار کمتری نیاز به تخمین‌زدن دارند[13].

تصویر کلی یک معماری شبکه عصبی کانولوشن در شکل زیر نمایش داده شده است.

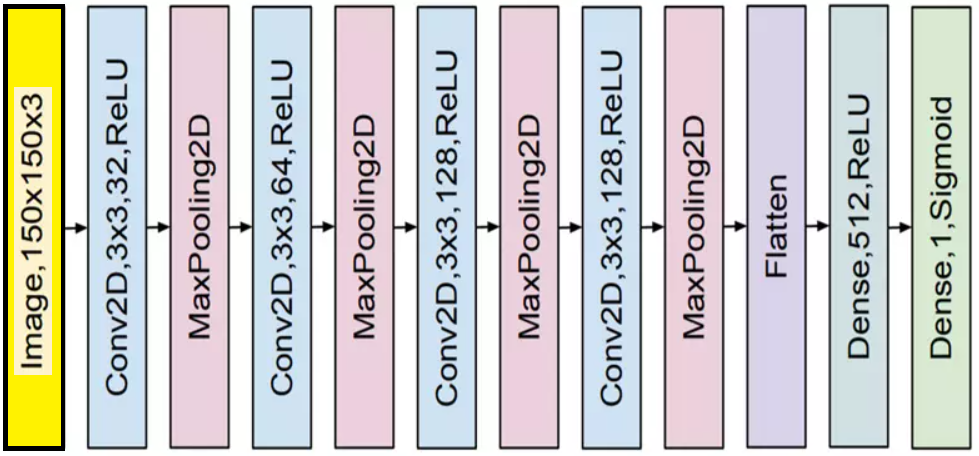


شکل ‏2‑14 ساختار لایه‌های شبکه‌های عصبی کانولوشنی

بطور کلی، یک شبکه CNN از سه لایه اصلی تشکیل می‌شود که عبارتند از: لایه کانولوشن، لایه Pooling و لایه کاملاً متصل. لایه‌های مختلف وظایف مختلفی را انجام می‌دهند.

در شکل 7 یک مثال ساده از معماری شبکه عصبی کانولوشن نشان داده شده است. شبکه کانولوشن معمولا از بلوک‌های مختلفی تشکیل شده است. لایه‌ها یا بلوک‌های مختلف در شبکه CNN عبارتنداز:

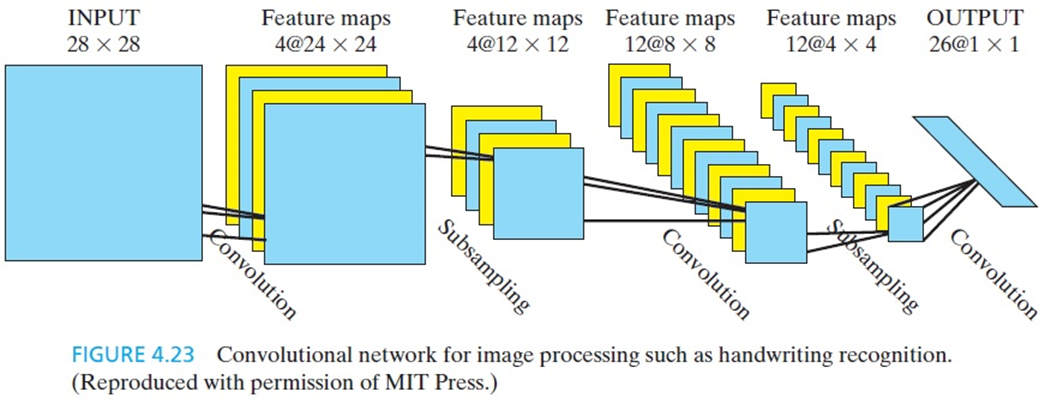
لایه ورودی(Input layer) ، لایه کانولوشن (Convolutional layer)، لایه غیرخطی[[6]](#footnote-6) (Non-linear activation function)، لایه پولینگ، لایه فولی کانکتد (Fully connected layer)



شکل 7: یک مثال ساده از معماری شبکه عصبی کانولوشن

ابتدا لایه ورودی را بررسی می‌کنیم، سپس به لایه‌های شبکه کانولوشن می‌رسیم. شبکه CNN ورودی‌های متنوعی را می‌پذیرد. ابتدا معروف‌ترین داده ورودی به شبکه CNN، یعنی تصویر را بررسی کنم.

در شکل زیر یک معماری کلی از شبکه عصبی کانولوشن برای دسته‌بندی تصاویر بصورت لایه‌به‌لایه نمایش داده شده است.



شکل ‏2‑16 شبکه کانولوشنی برای پردازش تصویر

در هر شبکه عصبی کانولوشن دو مرحله برای آموزش وجود دارد. مرحله feed forward و مرحله backpropagation یا پس انتشار. در مرحله اول تصویر ورودی به شبکه تغذیه می‌شود و این عمل چیزی جز ضرب نقطه‌ای بین ورودی و پارامترهای هر نورون و نهایتاً اعمال عملیات کانولوشن در هر لایه نیست. سپس خروجی شبکه محاسبه می‌شود. در این جا به منظور تنظیم پارامترهای شبکه و یا به عبارت دیگر همان آموزش شبکه، از نتیجه خروجی جهت محاسبه میزان خطای شبکه استفاده می‌شود. برای اینکار خروجی شبکه را با استفاده از یک تابع خطا[[7]](#footnote-7) با پاسخ صحیح مقایسه کرده و اینطور میزان خطا محاسبه می‌شود. در مرحله بعدی بر اساس میزان خطای محاسبه شده مرحله backpropagation آغاز می‌شود. در این مرحله گرادیانت هر پارامتر با توجه به قائده chain rule محاسبه می‌شود و تمامی پارامترها با توجه به تاثیری که بر خطای ایجاد شده در شبکه دارند، تغییر پیدا می‌کنند. بعد از بروزسانی‌شدن پارامترها مرحله بعدی feed-forward شروع می‌شود. بعد از تکرار تعداد مناسبی از این مراحل آموزش شبکه پایان می‌یابد [14]**[[6](#Machines)]**.

**کانولوشن چیست؟**

در عملگر کانولوشن، چهار مولفه مهم وجود دارد که عبارتنداز:

* ماتریس یا تصویر ورودی (Input)
* فیلتر یا کرنل کانولوشنی (Convolution Filter)
* عملگر کانولوشن (\*)
* ویژگی خروجی کانولوشن (Output)

**لایه کانولوشن در شبکه عصبی کانولوشن (Convolutional Layer)**

هسته اصلی شبکه CNN لایه کانولوشنی است که درصد اعظم محاسبات شبکه عصبی کانولوشن را به خود اختصاص داده است. هر لایه کانولوشن در شبکه عصبی کانولوشن شامل مجموعه‌ای فیلتر است و از کانولوشن بین فیلترها و لایه ورودی است که خروجی ساخته می‌شود. به خروجی لایه کانولوشنی، Feature Map گفته می‌شود.

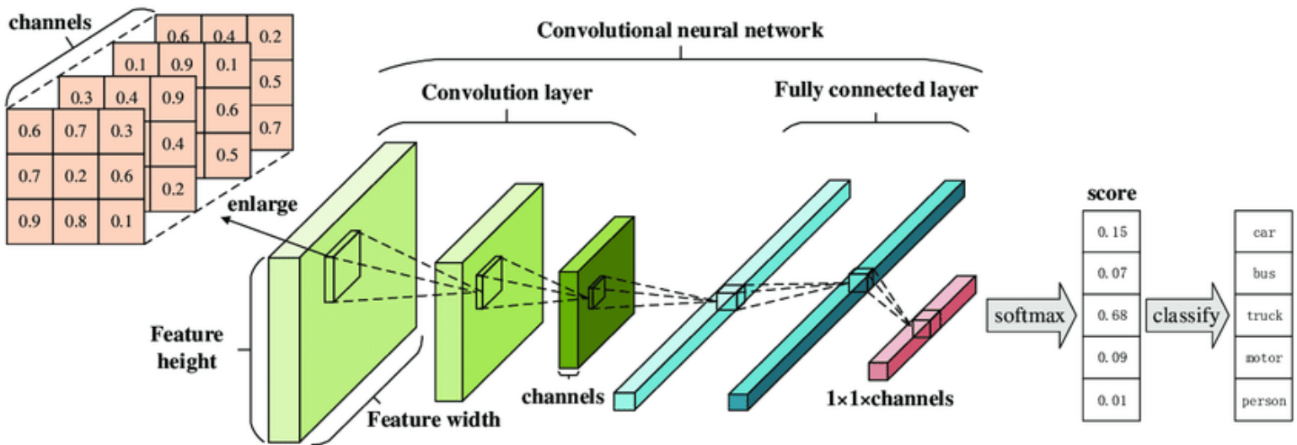
یک فیلتر می‌تواند شامل یک الگویی خاص باشد و در تصویر به دنبال آن الگو باشد. اتفاقاً در فرآیند آموزش شبکه، به دنبال این هستیم که این فیلترها الگوهای معناداری از هر تصویر استخراج کنند. مثلاً، فیلتری داریم که شامل الگوی گوش گربه است و می‌تواند حضور یک گربه در تصویر ورودی را تشخیص دهد.

این همان کاری است که در لایه کانولوشنی انجام می‌شود. جستجو در تصویر برای یافتن تنها یک الگو منجر به نتایج خوبی نمی‌شود و باعث می‌شود شبکه از لحاظ کارایی محدود باشد. برای حل این مشکل، نیاز است که لایه کانولوشنی چندین فیلتر داشته باشد. هریک از فیلترها به تنهایی یک الگوی خاص داشته باشند و خروجی لایه کانولوشنی مجموعه‌ای از الگوهای مختلف باشد.

#### انواع شبکه‌های عصبی کانولوشنی

##### معماری CNN سنتی

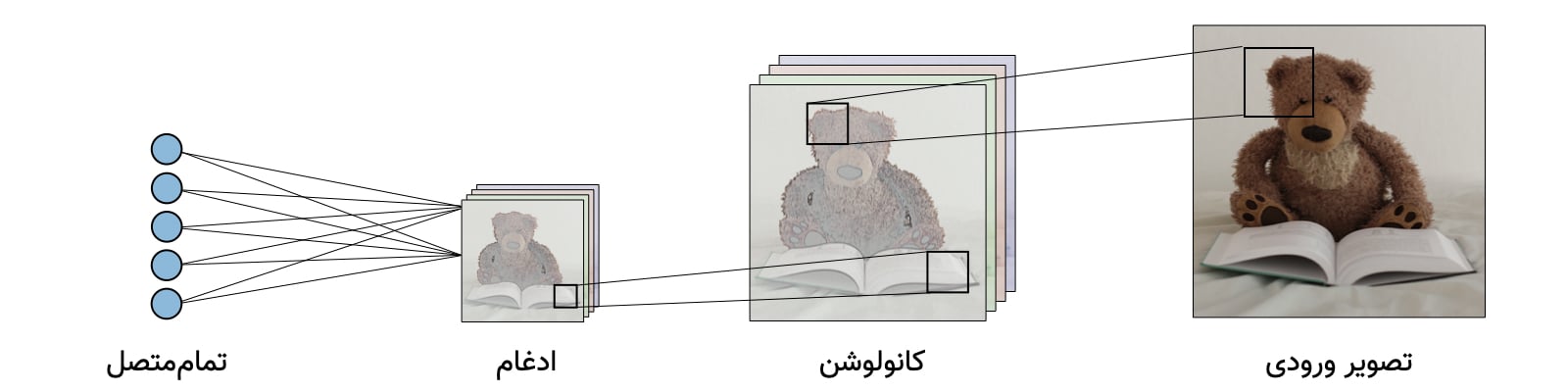
ساختار سنتی CNN عمدتاً از لایه‌های کانولوشنی، لایه‌های کانولوشن، لایه‌های کاملاً متصل و برخی از توابع‌ فعال‌سازی تشکیل شده است. هر هسته کانولوشن به قسمتی از نقشه‌های ویژگی متصل می‌شود. ورودی به تمام عناصر خروجی در لایه کاملاً متصل است.



شکل ‏2‑18 معماری CNN سنتی

شبکه‌های عصبی مصنوعی کانولوشنی، که همچنین با عنوان CNN شناخته می‌شوند، نوع خاصی از شبکه‌های عصبی هستند که عموماً از لایه‌های زیر تشکیل شده‌اند[15]:

**لایه کانولوشنیCONV، لایه ادغام POOL، تمام‌ متصل FC**

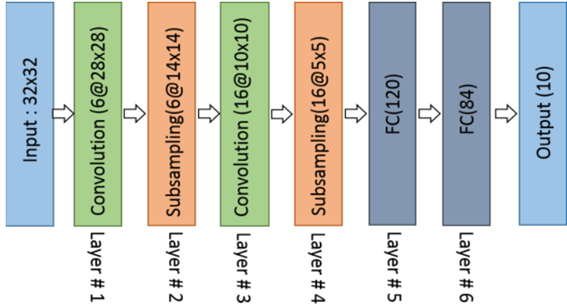


شکل ‏2‑19 مراحل شبکه‌های عصبی کانولوشنی در پردازش تصویر

با توجه به کاربرد گسترده شبکه‌های عصبی کانولوشنی محققان متوجه زمان اجرای زیاد آنها شدند و برای همین برای بهبود زمان اجرای این شبکه‌ها اقدامات مختلفی انجام دادند؛ حالا به سراغ راهکارهایی که محققین تاکنون برای بهبود شبکه‌های عصبی عمیق ارائه داده‌اند، می‌رویم.

##### شبکه‌های عصبی کانولوشنی LexNet

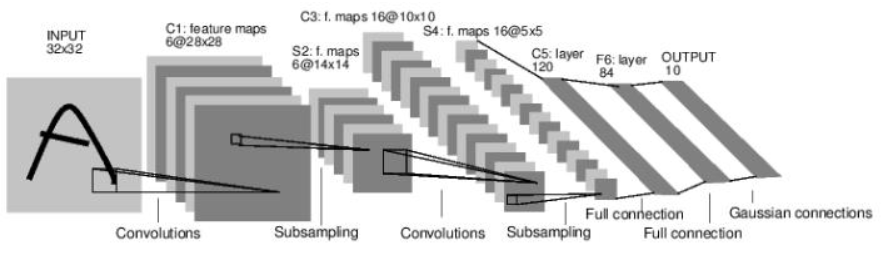
به صورت کلی معماری LeNet به صورت زیر است[9]:



شکل ‏2‑20معماری LeNet

The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches

اولین شبکه‌ی کانولوشنی که بسیار مطرح شد، شبکه‌ی LeNet-5 بود و در سال 1998 ارائه شد. این شبکه را می‌توان به عنوان اولین شبکه که فیلتر‌های‌کانولوشنی در آن استفاده شده ‌‌نام برد. هدف این شبکه، خواندن کد پستی از پشت بسته‌های پستی بود[16].



شکل ‏2‑21ساختار شبکه عصبی LexNet

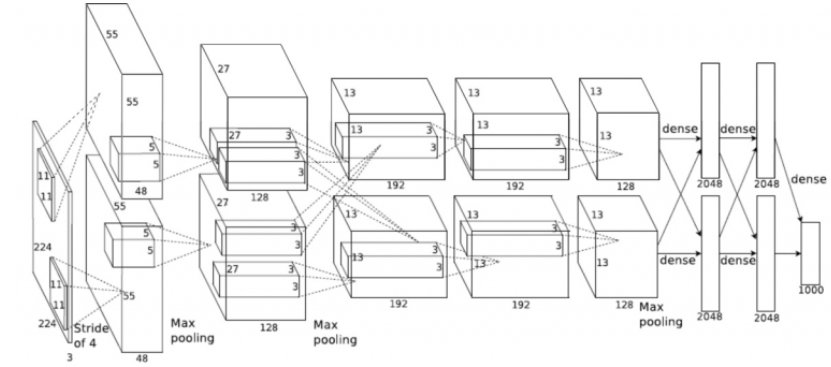
##### شبکه‌های عصبی کانولوشنی AlexNet

دومین شبکه نورونی کانولوشنی مهم شبکه AlexNet است که توسط Alex Krizhevsky و حدود چهارده سال بعد از LeNet یعنی در سال 2012 معرفی شد. این شبکه دارای 8 لایه )پنج لایه کانولوشنی و سه لایه (fully connected است که موجب شده بعضی اوقات در دسته شبکه‌های کم‌عمق (Shallow) قرار گیرد.

طبقه‌بندی تصویر[[8]](#footnote-8) یکی از مشکلات مهم در زمینه یادگیری ماشین است. معماری یادگیری عمیق در بسیاری از برنامه‌های یادگیری ماشین مانند طبقه‌بندی تصویر و تشخیص شیء استفاده می‌شود. توانایی دستکاری خوشه‌های[[9]](#footnote-9) بزرگ تصویر و پیاده‌سازی سریع آنها، یادگیری عمیق را به روشی محبوب در طبقه‌بندی تصاویر تبدیل می‌کند. این مطالعه به موفقیت شبکه‌های عصبی کانولوشنال که معماری یادگیری عمیق است، در حل مشکلات طبقه‌بندی تصویر اشاره می‌کند[17]. در زیر یک تصویر از عملکرد این معماری را می‌بینید.

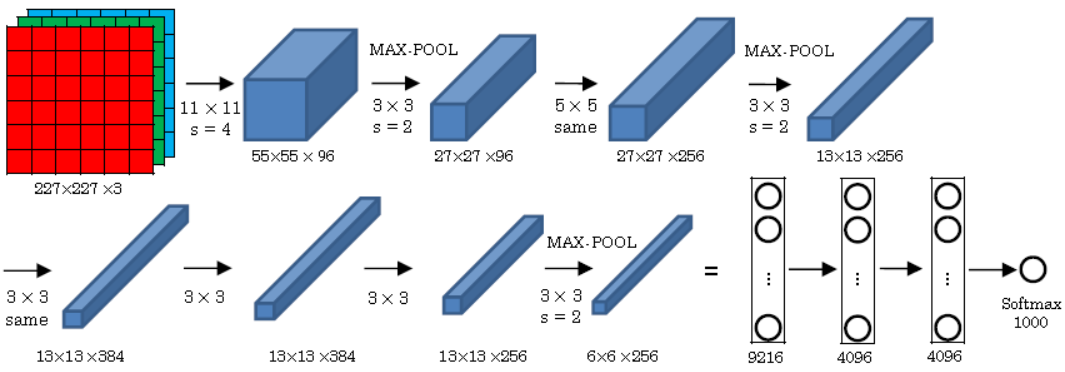
<https://caffe.berkeleyvision.org/>

Image classification with caffe deep learning framework مقاله و نت بوکاش



شکل ‏2‑23 ساختار شبکه AlexNet

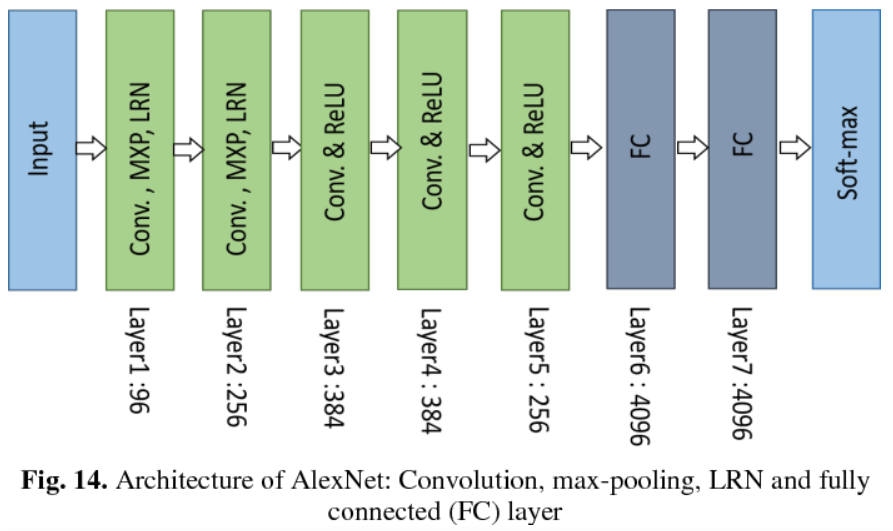
دلیل این که لایه‌ها دوطبقه‌ای هستند، استفاده از دو GPU برای پردازش آن است) درشکل ورودی شبکه یک عکس 3\*224\*224 است اما ورودی دقیق 3\*227\*227 است. در ابتدا 96 فیلتر کانولوشنی 11\*11 همراه با 4 گام و padding=0 اعمال می‌شود. خروجی آرایه 96\*55\*55 است[[10]](#footnote-10). در مرحله بعد از لایه maxpool 3\*3 با اندازه گام 2 عبور داده می‌شود که خروجی 96\*27\*27 می‌شود.( تعداد پارامتر‌های قابل یادگیری در maxpool صفراست) در مرحله بعد از 256 فیلتر کانولوشنی 5\*5 با گام یک و padding=2 عبور می‌دهیم که خروجی 256\*27\*27 است. در مرحله بعد دوباره از فیلتر maxpool 3\*3 با گام 2 عبور می‌دهیم که خروجی 13\*13\*96 است. در سه مرحله بعد از فیلتر‌های کانولوشنی با گام و padding یک استفاده می‌کنیم که تعداد آنها به ترتیب 384، 384، 256 است که خروجی نهایتاً 256\*13\*13 می‌شود. سپس از یک لایه maxpool 3\*3 با گام یک عبور می‌دهیم و خروجی آن 256\*6\*6 می‌شود. در آخر نیز سه لایه fully connected قرار دارد. دو لایه 4096 تایی و یک لایه 1000 تایی با تابع فعالیت softmax برای تعیین کلاس‌های موجود در مساله. (به طور مثال اگر قصد انتخاب بین سه کلاس سگ، گربه، موش را داشته باشیم اندازه لایه آخر را سه در نظر می‌گیریم.)



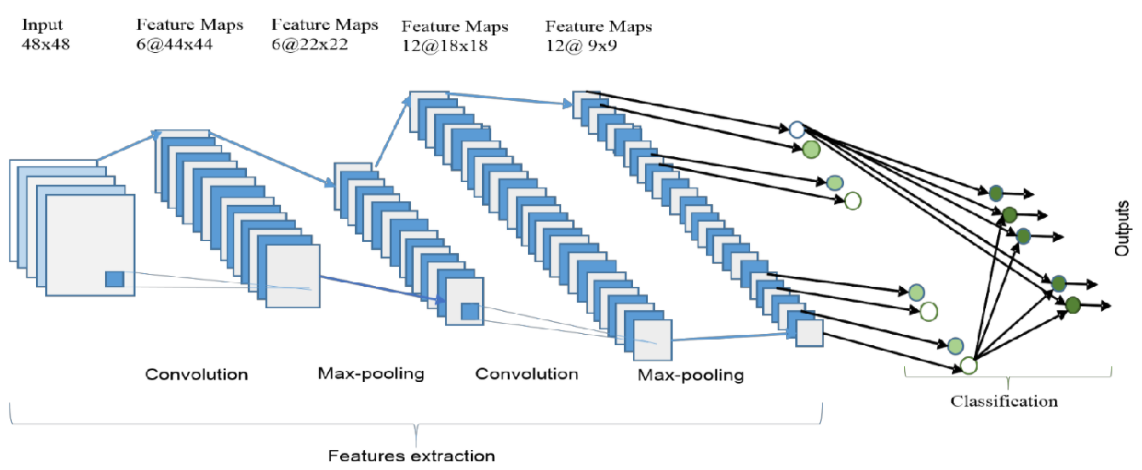
وقتی AlexNet را باز می‌کنیم، به نظر یک معماری ساده با لایه‌های پیچیده است که روی هم سوار شده‌اند و کاملاً به لایه‌های بالایی متصل هستند. چیزی که این مدل را متفاوت می‌کند، سرعت انجام وظیفه و استفاده از GPU برای یادگیری است. در دهه‌ 80 میلادی، برای یادگیری یک شبکه‌ عصبی ازCPU استفاده می‌کردند ولی AlexNet با استفاده از GPU سرعت این یادگیری را ده برابر کرد. [9].

The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches

<http://blog.class.vision/1397/03/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet/>



شکل 14. معماری AlexNet: Convolution ، max-pooling ،LRN و لایه کاملاً متصل FC



شکل ‏2‑24معماری کلی CNN شامل یک لایه ورودی، چندین لایه متراکم متناوب و حداکثر ترکیب ، یک لایه کاملاً متصل و یک لایه طبقه‌بندی است.

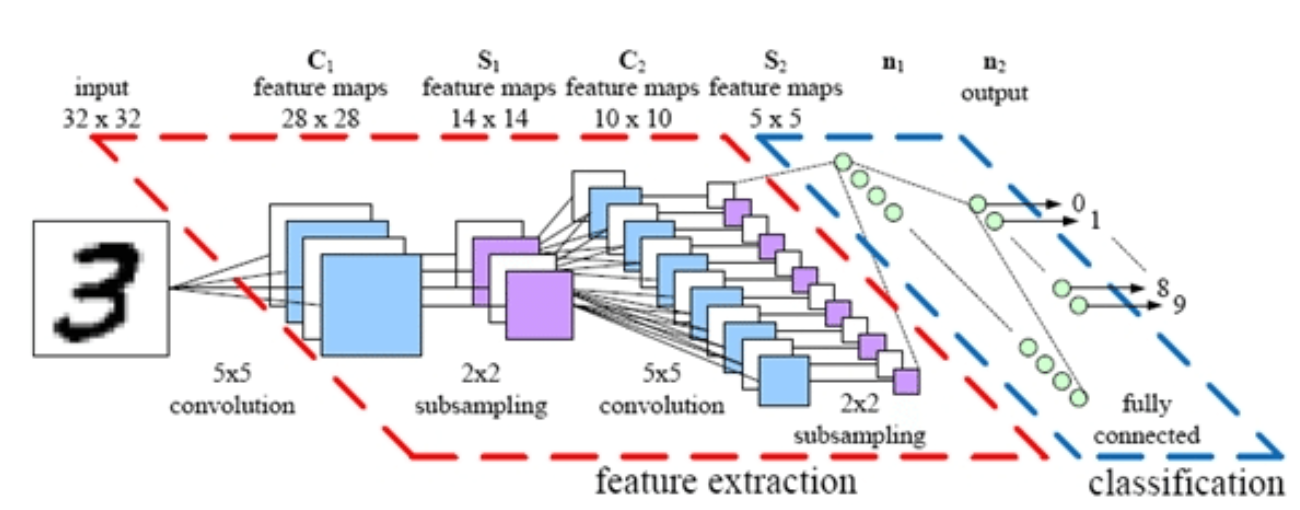
شکل 11. معماری کلی CNN شامل یک لایه ورودی، چندین لایه متراکم متناوب[[11]](#footnote-11) و حداکثر ترکیب[[12]](#footnote-12)، یک لایه کاملاً متصل و یک لایه طبقه‌بندی است.

[**The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches**](https://www.semanticscholar.org/paper/The-History-Began-from-AlexNet%3A-A-Comprehensive-on-Alom-Taha/b57e6468740d9320f3f14c6079168b8e21366416)

**کاربردهای شبکه عصبی AlexNet در صنعت:**

شبکه عصبی کانولوشن یک موضوع تحقیقاتی داغ در زمینه تشخیص تصویر است. آخرین تحقیقات نشان می‌دهد که مدل Deep CNN در استخراج ویژگی‌ها و نمایش تصاویر خوب است. این ظرفیت برای بازیابی تصویر در این مقاله اعمال می‌شود. ما اهمیت هر لایه را مطالعه می‌کنیم و آزمایش‌های بازیابی تصویر را بر روی ویژگی‌های همجوشی انجام می‌دهیم. از چارچوب Caffe و مدل AlexNet برای استخراج اطلاعات ویژگی در مورد تصاویر استفاده شد[18].

Feature extraction and image retrieval based on AlexNet



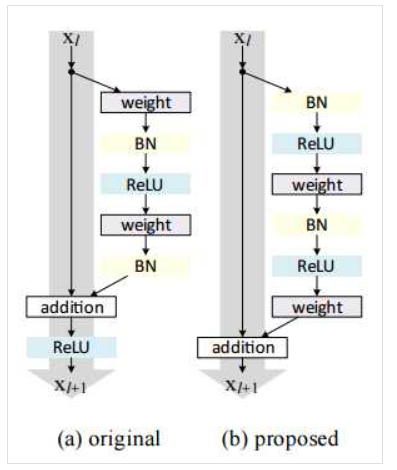
شکل ‏2‑25جزئیات پیاده‌سازی شبکه‌های عصبی کانولوشنی AlexNet

آموزش انتقال یک روش کارآمد برای حل مسئله طبقه‌بندی با مقدار کمی داده است. آموزش یادگیری را به شبکه عصبیAlexNet CNN برای تشخیص انسان بر اساس تصاویر گوش اعمال کردیم.AlexNet CNN را متناسب با حوزه مشکل خود انتخاب و تنظیم کردیم. آخرین لایه کاملاً متصل شده با یک لایه کاملاً متصل جایگزین می‌شود تا 10 کلاس را به جای 1000 کلاس تشخیص دهد. یک لایه دیگر واحد تصحیح خطی ReLU نیز برای بهبود توانایی حل مسئله غیر‌خطی شبکه اضافه شده است[19].

Utilizing AlexNet Deep Transfer Learning for Ear Recognition

##### شبکه‌های عصبی کانولوشنی ResNet

ResNet یکی از معماری‌های بزرگ است که نشان می‌دهد یک معماری یادگیری عمیق تا چه حد می‌تواند عمیق باشد. ResNet (که مخفف Residual Networks است) شامل چندین ماژول رسوبی است که برروی هم سوار شده‌اند، که درواقع ساختمان اصلی معماری ResNet را تشکیل می‌دهند. تصویر زیر یک نمای کلی از ماژول رسوبی را نشان می‌دهد[9]



شکل 27. میانگین و حداکثر عملیات جمع آوری.

Introduction to Resnet or Residual Network

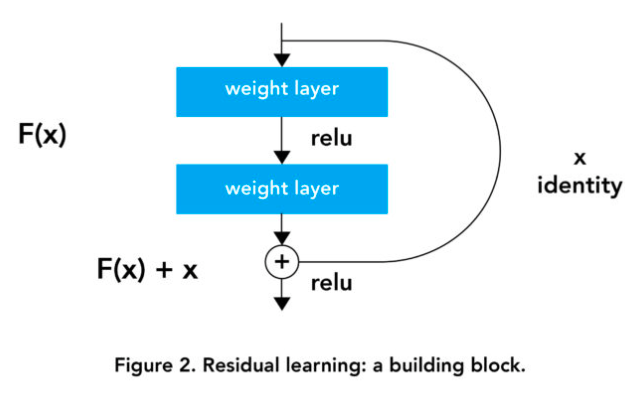
ResNet[[13]](#footnote-13)، نوع خاصی از شبکه عصبی که در سال 2015 توسط Kaiming He ،Xiangyu Zhang ،Shaoqing Ren و Jian Sun در مقاله "یادگیری عمیق باقیمانده برای تشخیص تصویر" معرفی شد. مدلهای ResNet بسیار موفق بودند که شامل موارد زیر هستند[20]:

* Won 1st place in the ILSVRC 2015 classification competition with a top-5 error rate of 3.57% (An ensemble model)
* Won the 1st place in ILSVRC and COCO 2015 competition in ImageNet Detection, ImageNet localization, Coco detection and Coco segmentation.
* Replacing VGG-16 layers in Faster R-CNN with ResNet-101. They observed relative improvements of 28%
* Efficiently trained networks with 100 layers and 1000 layers also.

**نیاز برای ResNet**

عمدتاً برای حل یک مشکل پیچیده، برخی از لایه‌های اضافی را در شبکه‌های عصبی عمیق قرار می‌دهیم که منجر به بهبود دقت و عملکرد می‌شود. شهود پشت افزودن لایه‌های بیشتر این است که این لایه‌ها به تدریج ویژگی‌های پیچیده‌تری را می‌آموزند. به عنوان مثال، در صورت تشخیص تصاویر، لایه اول ممکن است تشخیص لبه‌ها را یاد بگیرد، لایه دوم تشخیص بافت‌ها و به طور مشابه لایه سوم می‌تواند تشخیص اشیاء و غیره را بیاموزد. اما مشخص شده که با مدل شبکه عصبی Convolutional سنتی حداکثر آستانه عمق وجود دارد. در اینجا یک نمودار است که درصد خطا را در آموزش و آزمایش داده‌های 20 لایه شبکه و 56 لایه شبکه توصیف می‌کند.

**بلوک Residual**



اولین چیزی که متوجه تفاوت آن می‌شویم این است که ارتباط مستقیمی وجود دارد که برخی از لایه‌ها (ممکن است در مدلهای مختلف متفاوت باشد) را در بین آنها رد می‌کند. این اتصال 'اتصال پرش' نامیده می‌شود و هسته بلوک‌های باقیمانده است. با توجه به این اتصال پرش‌، خروجی لایه در حال حاضر یکسان نیست. بدون استفاده از این اتصال رد شده، ورودی 'x' در وزن لایه ضرب می‌شود و سپس یک عبارت تعصب اضافه می‌شود.

<https://www.mygreatlearning.com/blog/resnet/>

شبکه‌های باقیماندهResNet در کار طبقه‌بندیILSVRC 2015 دارای پیشرفته ترین عملکرد هستند (دنگ و همکاران ، 2009) و امکان آموزش شبکه‌های بسیار عمیق تا بیش از 1000 لایه را فراهم می‌کند معماری های شبکه باقیمانده فعلی دارای چندین محدودیت بالقوه هستند: ارتباطات هویتی که در ResNet فعلی پیاده‌سازی شده است منجر به تجمع ترکیبی از سطوح نمایش ویژگی‌ها در هر لایه می‌شود ، حتی اگر در یک شبکه عمیق برخی ویژگی‌ها قبلاً آموخته شده باشد لایه‌ها ممکن است دیگر اطلاعات مفیدی در لایه‌های بعدی ارائه ندهند[21]

Zagoruyko, S. and N. Komodakis Wide Residual Networks. 2016. arXiv:1605.07146.

**معماریResNet**

شبکه ResNet از معماری شبکه ساده 34 لایه با الهام از VGG-19 استفاده می‌کند که در آن اتصال میانبر اضافه می‌شود. این اتصالات میانبر سپس معماری را به شبکه باقیمانده مانند شکل زیر تبدیل می‌کند:

<https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/>

<https://d2l.ai/chapter_convolutional-modern/resnet.html>

با طراحی شبکه‌های عمیق تر ، درک اینکه چگونه افزودن لایه‌ها می‌تواند پیچیدگی و بیان شبکه را افزایش دهد ضروری است. حتی مهمتر توانایی طراحی شبکه‌هایی است که در آن افزودن لایه‌ها شبکه ها را کاملاً رسا می‌کند.

Deep Limits of Residual Neural Networks

در سالهای اخیر کاربرد شبکه‌های عصبی عمیق به سرعت در حال افزایش است. دو عامل مهم تعیین کننده کارآیی آموزش سیستم بینایی رایانه‌ای با استفاده از شبکه‌های عصبی عمیق است. اولین عامل دشواری آموزش یک شبکه عصبی عمیق تعداد زیادی پارامتر است. عامل دوم کارایی شبکه آموزش‌دیده برای کاهش هزینه محاسباتی است. در این مقاله یک شبکه عصبی عمیق کارآمد که از کاهش اندازه شبکه، عوامل و تنظیم پارامترهای فوق استفاده می‌کند ، پیشنهاد شده است. برای مقابله با تعداد زیادی لایه از واحدهای باقیمانده[[14]](#footnote-14) استفاده می‌شود[22].

A deep residual neural network for low altitude remote sensing image classification

در حوزه طبقه‌بندی تصویر، میزان تشخیص و عملکرد استنتاج[[15]](#footnote-15) به چالش های اصلی در برنامه واقعی تبدیل می‌شود. اخیراً، شبکه باقیمانده (ResNet) دقت رقابتی و رفتارهای همگرایی خوب در شبکه‌های عصبی عمیق را نشان داده است. در این مقاله، ما به تسریع این چارچوب امیدوار‌کننده در برنامه استنتاج در FPGA[[16]](#footnote-16) با استفاده از زبان برنامه‌نویسی OpenCL اختصاص داده ایم. در مرحله اول، یک شتاب‌دهنده یادگیری عمیق تکمیل شده برای درک تابع باقیمانده در ResNet ساخته شده است. ثانیاً، ساختار ما سه بافر روی تراشه را مجدداً برنامه ریزی می‌کند تا داده‌های ویژگی را ذخیره کرده و به طور متناوب به عناصر پردازنده منتقل کند. علاوه بر این، ما داده‌های موازی و اجرای خط لوله را نیز پیاده‌سازی می‌کنیم تا پارامترهای فیلتر بتوانند همزمان با داده‌های تصویر در FPGA پردازش شوند. علاوه بر این، ما از یک مبدل برای تبدیل هر ResNet در چارچوب CAFFE به پلت فرم FPGA استفاده می‌کنیم. این می‌تواند فایل‌های هد FPGA را با استفاده از فایل‌های اصلی اولیه خود از طریق تابع فرهنگ لغت پایتون تولید کند. تجزیه و تحلیل نتایج آزمایش نشان می‌دهد که شتابدهنده ما در عین حفظ نرخ دقت بالا عملکرد رقابتی دارد. در نهایت، یک راه‌حل برای تسریع در ساخت ResNet با استفاده از زبان برنامه‌نویسی OpenCL در FPGA ارائه می‌دهیم[23].

FPGA accelerates deep residual learning for image recognition

آموزش شبکه‌های عصبی عمیق تر دشوارتر است. ما یک چارچوب یادگیری باقیمانده را برای سهولت آموزش شبکه‌هایی که عمیقاً عمیق تر از شبکه‌های قبلی هستند ارائه می‌دهیم. ما به صراحت لایه‌ها را به عنوان یادگیری توابع باقیمانده با اشاره به ورودی‌های لایه، به جای یادگیری توابع نامرتبط، دوباره فرموله می‌کنیم. ما شواهد تجربی جامعی را ارائه می‌دهیم که نشان می‌دهد بهینه‌سازی این شبکه‌های باقیمانده آسان‌تر است و می‌توانند از عمق قابل‌ملاحظه ای دقت را به دست آورند. در مجموعه داده ImageNet ما شبکه‌های باقیمانده با عمق حداکثر 152 لایه را ارزیابی می‌کنیم، 8 برابر عمیق تر از شبکه‌هایVGG اما هنوز پیچیدگی کمتری دارند. مجموعه ای از این شبکه‌های باقیمانده به 3.57٪ خطا در مجموعه آزمایش ImageNet دست می‌یابد. این نتیجه مقام اول را در کار طبقه‌بندی ILSVRC 2015 کسب کرد. ما همچنین تجزیه و تحلیل روی CIFAR-10 با 100 و 1000 لایه ارائه می‌دهیم. عمق بازنمایی ها برای بسیاری از کارهای تشخیص بصری اهمیت محوری دارد. صرفاً به دلیل نمایش های بسیار عمیق ما ، 28 درصد بهبود نسبی در مجموعه داده تشخیص شیء COCO بدست می آوریم. شبکه‌های باقیمانده عمیق پایه و اساس ارسال های ما به مسابقات ILSVRC & COCO 2015 است ، جایی که ما همچنین در کارهای تشخیص ImageNet ، محلی سازی ImageNet ، تشخیص COCO و تقسیم بندی COCO مقام های اول را کسب کردیم[23].

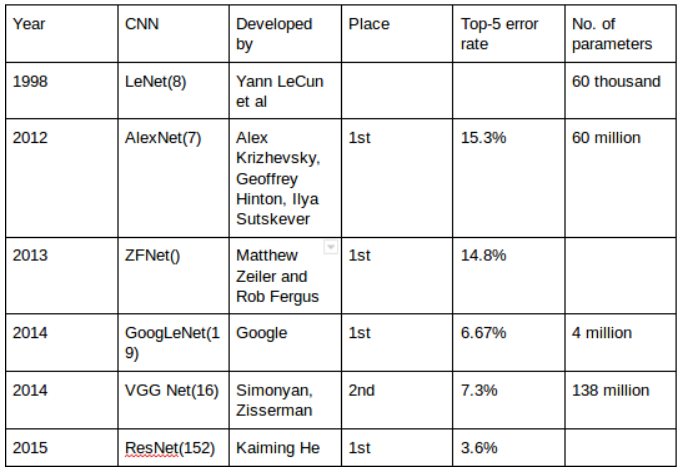
FPGA accelerates deep residual learning for image recognition

این مقاله یک شبکه عصبی باقیمانده عمیق ResNet را برای رگرسیون توابع غیر‌خطی توسعه می‌دهد. لایه‌های کانولوشن و ترکیبی[[17]](#footnote-17) با لایه‌های کاملاً متصل در بلوک باقیمانده جایگزین می‌شوند. برای ارزیابی مدل رگرسیون جدید، شبکه‌های عصبی با عمق و عرض مختلف را بر روی داده‌های شبیه‌سازی شده آموزش می‌دهیم و آزمایش می‌کنیم و پارامترهای بهینه را پیدا می‌کنیم. چندین آزمایش عددی[[18]](#footnote-18) از مدل رگرسیون بهینه را بر روی چندین داده شبیه‌سازی شده انجام می‌دهیم و نتایج نشان می‌دهد که مدل رگرسیون جدید بر روی داده‌های شبیه‌سازی شده به خوبی رفتار می‌کند. همچنین مقایسه‌هایی بین رگرسیون باقیمانده بهینه و سایر تکنیک‌های تقریبی خطی و غیر خطی مانند رگرسیون لاسو[[19]](#footnote-19)، درخت تصمیم‌گیری[[20]](#footnote-20) و ماشین بردار پشتیبان[[21]](#footnote-21) انجام می‌شود. مدل رگرسیون پسماند بهینه نسبت به سایر مدلها دارای ظرفیت تقریبی بهتری است. در نهایت، رگرسیون باقیمانده در پیش‌بینی یک سری رطوبت نسبی در دنیای واقعی اعمال می‌شود. مطالعه ما نشان می‌دهد که مدل رگرسیون باقیمانده پایدار و در عمل قابل اجرا است[24].

Deep Residual Learning for Nonlinear Regression

در جدول زیر خلاصه‌ای از مقایسه این شبکه ها قابل مشاهده است.

جدول ‏2‑2مقایسه شبکه‌های مختلف کانولوشنی



طی چند سال گذشته، مجموعه‌ای از پیشرفت‌ها در زمینه بینایی رایانه‌ای رخ داده است. به‌ویژه با معرفی شبکه‌های عصبی عمیق کانولوشنال، در زمینه مشکلاتی مانند طبقه‌بندی تصویر و تشخیص تصویر به نتایج روز دست یافته‌ایم. بنابراین، در طول سالها، محققان تمایل به ایجاد شبکه‌های عصبی عمیق (افزودن لایه‌های بیشتر) برای حل چنین وظایف پیچیده و همچنین بهبود طبقه بندی/تشخیص داشتند و دارند. اما، مشاهده شده است که هرچه لایه‌های بیشتری به شبکه عصبی اضافه می‌کنیم، آموزش آنها دشوارتر می‌شود و دقت شروع به اشباع و سپس تنزل می‌کند. شبکه‌های عصبی عمیق و اثربخش به منابع محاسباتی قابل‌ملاحظه‌ای نیاز دارند. از این‌رو تحقیقات زیادی برای بهبود شبکه‌های عصبی عمیق انجام شده است.

در نتیجه با توجه به اینکه هنوز پژوهشگران به آن سرعت و زمان اجرایی که ایده‌آل باشد، نرسیده‌اند، ما در این پژوهش تصمیم گرفتیم که با استفاده از محاسبات تقریبی سرعت اجرای این الگوریتم را بهبود دهیم.

## مروری بر رایانش تقریبی approximate computing

محاسبات تقریبی یک الگوی طراحی است که پیاده‌سازی سخت‌افزاری و نرم‌افزاری بسیار کارآمد را با بهره‌گیری از قابلیت انعطاف‌پذیری ذاتی برنامه‌ها به دقت در محاسبات آنها امکان‌پذیر می‌کند. کارهای قبلی در این زمینه پتانسیل بهبود قابل‌توجه انرژی و عملکرد را نشان داده است. برای محاسبه تقریبی پذیرش جریان اصلی زیر نیاز است: (i) درک عمیق‌تری از قابلیت انعطاف‌پذیری برنامه‌های کاربردی در طیف وسیعتری از کاربردها (ii) ابزارهایی که می‌توانند به طور کمی تاب‌آوری ذاتی[[22]](#footnote-22) یک برنامه را تعیین کنند و (iii) روش‌هایی برای ارزیابی سریع پتانسیل برنامه‌ها، تکنیک‌های مختلف محاسبه تقریبی برای یک برنامه معین. برای تسهیل تجزیه و تحلیل، یک چارچوب سیستماتیک برای ویژگی‌های انعطاف‌پذیری برنامه[[23]](#footnote-23) (ARC) پیشنهاد کرده که (الف) یک برنامه را به قسمتهای مقاوم و حساس تقسیم می‌کند و (ب) با استفاده از مدل‌های تقریبی که طیف وسیعی از تکنیک‌های محاسبه تقریبی را انتزاع می‌کند، قطعات مقاوم را مشخص می‌کند. مفاهیم کلیدی که ارائه شده می‌تواند به شکل‌گیری تحقیقات بیشتر در زمینه محاسبات تقریبی کمک کند، در حالی که چارچوب‌های مشخصه قابلیت انعطاف‌پذیری خودکار مانند ARC می‌تواند به طراحان در محاسبه تقریبی کمک کند[25].

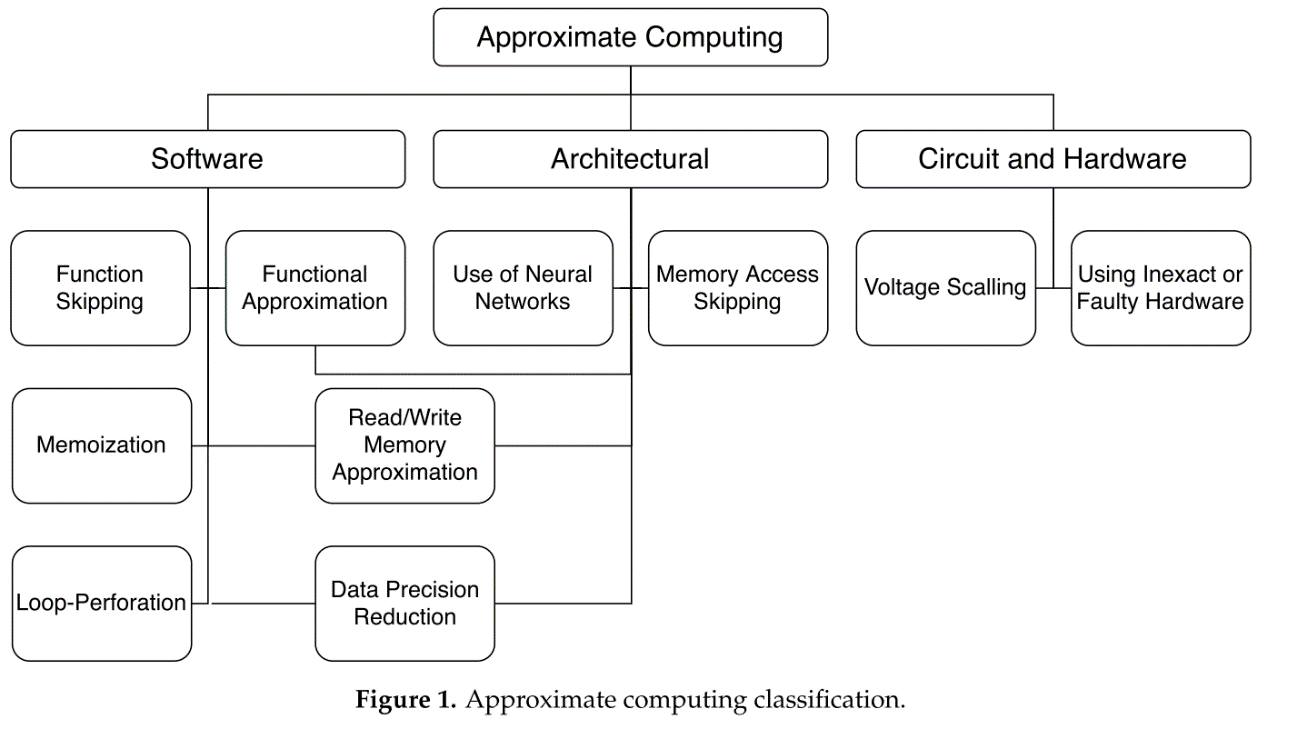
Analysis and characterization of inherent application resilience for approximate computing

کاهش مزایای مقیاس‌بندی تکنولوژی، طراحان را بر آن داشته تا به دنبال منابع جدید کارایی محاسبات باشند. برای افزایش عملکرد پلتفرم های محاسباتی در سطح توان مشابه یا کمتر لازم است از معماری‌های چند‌هسته‌ای و ناهمگن مبتنی بر شتاب‌دهنده استفاده شود. اصول اصلی محاسبه تقریبی- محاسبه کارآمد[[24]](#footnote-24) با تولید نتایج به اندازه کافی خوب یا با کیفیت کافی- جدید نیست و در بسیاری از زمینه‌ها از طراحی الگوریتم گرفته تا شبکه ها و سیستم‌های توزیع شده[[25]](#footnote-25) به اشتراک گذاشته شده است، اما اخیراً رشد استفاده ازمحاسبات تقریبی بیشتر شده است. اصول مربوط به تمام لایه‌های محاسبات، از جمله مدارها، معماری و نرم افزار. تکنیک‌های محاسبات تقریبی نیز از کاربردهای خاص و کاربردی[[26]](#footnote-26) به کاربردهای وسیعتری تبدیل شده‌اند که با روش‌های طراحی سیستماتیک پشتیبانی می‌شوند. سرانجام، ظهور بارهای کاری مانند تشخیص[[27]](#footnote-27)، استخراج[[28]](#footnote-28)، جستجو، تجزیه و تحلیل داده ها[[29]](#footnote-29)، استنباط[[30]](#footnote-30) و بینایی[[31]](#footnote-31) فرصت‌های محاسبات تقریبی را به میزان زیادی افزایش می‌دهد. اصول کلیدی که کار ما را در این زمینه هدایت کرده‌اند، توصیف می‌کنیم و یک چارچوب جامع چند لایه‌ای[[32]](#footnote-32) برای محاسبات تقریبی ترسیم می‌کنیم[26].

Approximate computing and the quest for computing efficiency

این کار یک نظرسنجی در مورد محاسبات تقریبی و تأثیر آن بر تحمل خطا[[33]](#footnote-33)، به ویژه در برنامه‌های ایمنی[[34]](#footnote-34) مهم است. این روش، روشهای تقریبی زیادی را ارائه می‌دهد که معمولاً در سطح نرم‌افزار، معماری و مدار کاربرد دارد. این روش‌ها در تمام سطوح ممکن پیاده‌سازی مورد بحث و مقایسه قرار می‌گیرند (برخی از تکنیک‌ها در بیش از یک سطح اعمال می‌شوند). تقریب نیز به عنوان ابزاری برای تحمل خطا و قابلیت اطمینان بالا ارائه شده است: تکنیک‌های سنتی پوشاندن خطا[[35]](#footnote-35)، مانند افزونگی مدولار سه گانه[[36]](#footnote-36)، می‌تواند تقریب زده شود و بنابراین هزینه پیاده‌سازی و اجرای آنها در مقایسه با تکنولوژی روز کاهش می‌یابد.

AC مبتنی بر مشاهدات بصری است؛ در حالی که انجام محاسبات دقیق به منابع زیادی نیاز دارد، اجازه تقریب انتخابی مشخصات می‌تواند باعث افزایش کارایی (یعنی مصرف توان کمتر، مساحت کمتر، عملکرد بیشتر تولید) بدون افزایش قابل‌توجه شود و کیفیت خروجی را تحت تأثیر قرار می‌دهد[27].



Survey on Approximate Computing and Its Intrinsic Fault Tolerance

محاسبات تقریبی به عنوان رویکردی امیدوارکننده در طراحی کم‌مصرف انرژی سیستم‌های دیجیتالی مطرح شده است. محاسبه تقریبی متکی بر توانایی بسیاری از سیستم‌ها و برنامه‌های کاربردی است که برخی از افت کیفیت یا بهینه را در نتیجه محاسبه شده تحمل می‌کنند. با کاهش نیاز به عملیات کاملاً دقیق یا کاملاً قطعی، تکنیک‌های محاسباتی تقریبی به طور قابل‌توجهی باعث افزایش بهره‌وری انرژی می‌شود. این مقاله پیشرفت‌های اخیر در مساحت، از جمله طراحی بلوک‌های تقریبی حساب، اندازه‌گیری خطاهای مربوط و کیفیت و تکنیک‌های سطح الگوریتم برای محاسبه تقریبی را مرور می‌کند[28].

Approximate computing: An emerging paradigm for energy-efficient design

## مروری بر حساب تقریبی approximate arithmetic

حساب تقریبی[[37]](#footnote-37) یا محاسبات تقریبی یکی از روش‌های بهبود عملکرد و کاهش مصرف انرژی در اجرای الگوریتم‌های شبکه‌عصبی است. برنامه‌های رسانه‌ای از محدودیت‌های حواس انسانی بهره‌مند می‌شوند، به این ترتیب اجازه می‌دهد تفاسیر صوتی و تصویری مقادیر کمی خطا داشته باشند بدون اینکه توسط بیننده قابل‌توجه باشد. تقریب بخش‌های رسانه می‌تواند باعث صرفه‌جویی عمده در مصرف انرژی در تصاویر یا ویدئوها شود. به نظر می‌رسد محاسبات تقریبی یک روش امیدوار‌کننده برای بهبود عملکرد است. طبق مقالاتی که تا‌کنون مطالعه کرده‌ام دو نوع تقریب نرم‌افزاری و سخت‌افزاری داریم؛ تقریب‌های نرم‌افزاری از قبیل: Quantization، هرس کردن و... و تقریب‌های سخت‌افزاری که در آن به این صورت است که روی یکی اعمال محاسبات ریاضی از قبیل ضرب، تقسیم، جمع، تفریق و ... در حین اجرای شبکه‌های عصبی به کار می‌رود، الگوریتم‌های مختلف را به کار می‌بریم تا بررسی کنیم با کدام روش حساب کامپیوتری و رایانش تقریبی به بهترین زمان زمان اجرا می‌رسیم و می‌توانیم بالاترین سرعت را داشته باشیم.

اغلب به عنوان مهمترین ماژول های حسابی در یک پردازنده ، جمع‌کننده‌ها ، ضرب‌کننده‌ها و تقسیم‌کننده‌ها عملکرد و بازده انرژی بسیاری از کارهای محاسباتی را تعیین می‌کنند. تقاضای بهره وری سرعت و توان بیشتر ، و همچنین ویژگی انعطاف پذیری خطا در بسیاری از برنامه ها (به عنوان مثال ، چند رسانه ای ، تشخیص و تجزیه و تحلیل داده ها) ، توسعه طراحی تقریبی محاسباتی را به دنبال داشته است. در این مقاله ، مرور و طبقه‌بندی طرح های فعلی مدارهای تقریبی محاسباتی شامل جمع ، ضرب کننده و تقسیم کننده ارائه شده است. یک ارزیابی جامع و مقایسه ای از ویژگی‌های خطا و مدار آنها برای درک ویژگی‌های طرح های مختلف انجام می‌شود. با استفاده از ضرب کننده ها و جمع‌کننده های تقریبی ، مدار برنامه کاربردی پردازش تصویر تنها 47 درصد از توان و 36 درصد از ضرب تأخیر توان را در یک طراحی دقیق مصرف می‌کند و در عین حال به کیفیت پردازش تصویر مشابهی دست می‌یابد. بهبود در تأخیر، توان و مساحت برای تشخیص تفاوت تصاویر با استفاده از تقسیم کننده‌های تقریبی به دست می‌آید[29].

A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits

تقاضای بهره وری سرعت و توان بالاتر ، و همچنین ویژگی انعطاف پذیری خطا در بسیاری از برنامه ها (به عنوان مثال ، چند رسانه ای ، تشخیص و تجزیه و تحلیل داده ها) ، توسعه مدارهای محاسباتی تقریبی را به دنبال داشته است. اغلب به عنوان مهمترین ماژول های حسابی در پردازنده ، جمع‌کننده ها ، ضرب کننده ها و تقسیم کننده ها عملکرد و بازده انرژی بسیاری از کارهای محاسباتی را تعیین می‌کنند[30].

Approximate arithmetic circuits and their applications

حسابی تقریبی[[38]](#footnote-38) در سیستم های محاسباتی با استفاده از عملیات نقطه ثابت (غیر صحیح) و ممیز نقطه شناور به طور عملی وجود داشته است. هدف اصلی بدست آوردن نتایج دقیق یا نزدیک به آن با استفاده از تکنیک‌های کاهش خطا از گرد کردن به طور پیچیده تا حساب فاصله ای بوده است. خطاهای انجام شده بد تلقی شده اند. با این حال ، اخیراً ایده هایی در مورد چگونگی ایجاد اشتباهات در کاهش تاخیر در عملیات ، اتلاف توان[[39]](#footnote-39) و مصرف انرژی مطرح شده است. خطاها ، که امروزه اغلب مفید به نظر می رسند ، به دلیل پرداخت های ملموس و تحمل خطای مبتنی بر برنامه ، به یک ویژگی عمدی در طراحی تبدیل می‌شوند[31].

On approximate arithmetic

مدارهای حسابی ماژول های محاسباتی مهمی در پردازنده هستند. آنها نقش کلیدی در عملکرد و مصرف انرژی بسیاری از برنامه های پردازش تصویر دارند. طبقه‌بندی برای طرح های فعلی مدارهای حساب تقریبی شامل جمع‌کننده، ضرب کننده و تقسیم کننده ارائه شده است. برای درک ویژگی‌های طرح های مختلف ، ارزیابی مقایسه ای خطاها و ویژگی‌های مدار آنها انجام می‌شود. صحت مدارهای حسابی تقریبی با انجام شبیه‌سازی های مونت کارلو[[40]](#footnote-40) ارزیابی می‌شود. اندازه گیری مدار با سنتز طرح های تقریبی در یک فرایند STM CMOS 28 nm ارزیابی می‌شود. نتایج شبیه‌سازی و سنتز، مبادله بین مدارهای حسابی تقریبی بین دقت و کارایی سخت افزار را نشان می‌دهد[32].

Approximate Arithmetic Circuits: Design and Evaluation

حوزه محاسبات تقریبی[[41]](#footnote-41) در چند سال گذشته مورد‌توجه جامعه تحقیقاتی به‌ویژه در زمینه برنامه‌های مختلف پردازش سیگنال قرار گرفته است، الگوریتم‌های فشرده‌سازی تصویر و ویدئو، مانند JPEG، MPEG و غیره، کاندیدهای بسیار جذابی برای محاسبات تقریبی هستند، زیرا در محاسبه عدم دقت به دلیل نامحسوس بودن انسان تحمل می‌کنند، که می‌تواند برای تحقق پیاده‌سازیهای بسیار کارآمد[[42]](#footnote-42) این الگوریتم‌ها مورد سوء‌استفاده قرار گیرد. با این حال، معماریهای تقریبی[[43]](#footnote-43) موجود معمولاً سطح تقریب سخت‌افزار را به صورت ایستا ثابت می‌کنند و با داده‌های ورودی سازگار نیستند. به عنوان مثال، اگر از یک پیکربندی سخت‌افزاری تقریبی ثابت برای رمزگذار MPEG (به عنوان مثال، سطح تقریبی ثابت) استفاده شود، کیفیت خروجی برای فیلم‌های ورودی مختلف بسیار متفاوت است. این مقاله با ارائه معماری تقریبی قابل‌تنظیم مجدد[[44]](#footnote-44) برای رمزگذارهای MPEG که توان مصرفی را با هدف حفظ آستانه پیک سیگنال به صدا[[45]](#footnote-45) (PSNR) خاص برای هر ویدئو بهینه می‌کند، به این موضوع می پردازد. در این راستا، ما بلوکهای جمع‌کننده و تفریق‌کننده[[46]](#footnote-46) (RAB) قابل‌تنظیم مجدد را طراحی می‌کنیم که توانایی تعدیل درجه تقریب خود را دارند و متعاقباً این بلوکها را در برآورد حرکت ادغام کرده و ماژولهای تبدیل کسینوس گسسته رمزگذار MPEG را ایجاد می‌کنند. ما دو روش اکتشافی برای تنظیم خودکار درجه تقریبی RAB ها در این دو ماژول در طول زمان اجرا بر اساس ویژگی‌های هر ویدئوی جداگانه پیشنهاد می‌کنیم[33].

Input-Based Dynamic Reconfiguration of Approximate Arithmetic Units for Video Encoding

کوتاه‌کردن یکی از رایج‌ترین روش‌ها برای محاسبه تقریبی سطح مدار است. این مقاله طرحی را برای جبران خطای مدارهای حسابی پیشنهاد می‌کند که در آن از یک پد[[47]](#footnote-47) به اصطلاح برای کوتاه‌کردن[[48]](#footnote-48) بیت خروجی عملوندهای ورودی در خروجی استفاده می‌شود. جبران خسارت به تعدیل نتایج خروجی یک مدار حسابی بستگی دارد.padding بااستفاده از اطلاعات آماری بر اساس پروفایل یک مدار حسابی مقداری را تعیین می‌کند تا میانگین تفاوت علامت‌دار بین مقادیر نادقیق و دقیق و میانگین خطای مربع[[49]](#footnote-49) را کاهش دهد. تجزیه و تحلیل گسترده و ارزیابی مبتنی بر شبیه‌سازی از معیارهای خطا بر روی جمع[[50]](#footnote-50)، ضرب و تقسیم کننده علامت‌دار انجام می‌شود؛ یک توافق عالی پیدا می‌شود. معیارهای طراحی اضافی نیز مانند توان مصرفی و پیچیدگی مدار ارزیابی می‌شود. کاربردهای مختلف مدارهای حسابی تقریبی با طرح جبران خطای خروجی پیشنهادی ارائه شده است. ضرب ماتریس و پردازش تصویر (تغییر تشخیص) برای نشان دادن اثربخشی طرح ارائه شده در این مقاله مورد بررسی قرار گرفته است[34].

Profile-Based Output Error Compensation for Approximate Arithmetic Circuits

تقریب می‌تواند عملکرد را افزایش داده یا توان مصرفی را با یک مدار ساده یا نادقیق در زمینه‌های کاربردی که نیازمندی‌های سخت برطرف شده، کاهش دهد. برای کاربردهای مربوط به حواس انسان، از حساب تقریبی می‌توان برای تولید نتایج کافی و نه نتایج دقیق استفاده کرد. طراحی تقریبی از یک مبادله دقت در محاسبه در مقابل عملکرد و توان استفاده می‌کند. با این‌حال، دقت مورد‌نیاز با توجه به برنامه‌های کاربردی متفاوت است، و نتایج دقیق هنوز 100٪ در برخی شرایط مورد‌نیاز است. در این مقاله، یک جمع‌کننده تقریبی قابل‌تنظیم با دقت[[51]](#footnote-51) ACA را پیشنهاد می‌کند که برای آن دقت نتایج در زمان اجرا قابل‌تنظیم است. به دلیل قابلیت تنظیم، ACA adder می‌تواند به صورت تطبیقی در دو حالت تقریبی (نادقیق) و حالت دقیق عمل کند. حالت تقریبی جمع‌کننده پیشنهادی می‌تواند به بهبود قابل‌توجهی در گذرگاه[[52]](#footnote-52) و کاهش توان کلی نسبت به طرح‌های جمع‌کننده معمولی دست یابد. می‌توان از آن در برنامه‌های قابل‌تنظیم بادقت استفاده کرد و مبادله قابل‌دستیابی بین عملکرد/توان و کیفیت را بهبود می‌بخشد. جمع‌کننده ACA تقریباً 30٪ کاهش توان را در مقایسه با جمع‌کننده خط‌لوله معمولی بادقت لازم انجام می‌دهد[35].

Accuracy-configurable adder for approximate arithmetic designs

## جمع‌بندی

مسئله حل مشکل زمان اجرای الگوریتم‌های شبکه عصبی به‌خصوص در زمینه پردازش تصویر یکی از مسائل مهم و پرکاربرد است، که امیدواریم با استفاده از تکنیک‌های موجود در حساب کامپیوتری و معماری کامپیوتر از جمله حساب تقریبی و ابزار سنتز سطح بالا این مشکل زمان اجرای طولانی آنها حل شود و استفاده از منابع موردنیاز کاهش یابد. در این پژوهش، قصد داریم روی عمل ضرب با‌استفاده از حساب تقریبی کار کنیم و روی الگوریتم‌های مختلف ضرب که تا‌کنون کار شده است، با اعمال تغییراتی در بخش‌های مختلف زمان اجرا را کاهش دهیم.

علی‌رغم پژوهش‌های مختلفی که در این زمینه انجام شده است؛ هنوز هم چالش‌های اساسی، یعنی سرعت کم پردازش و توان مصرفی بالا، مانع کاربرد وسیع‌تر و کارایی شبکه‌های عصبی در حوزه‌های مختلف است. در این پژوهش برای مقابله با این چالش‌ها از حساب تقریبی و سنتز سطح بالا استفاده خواهد شد و به دنبال آن خواهیم بود تا با‌استفاده از روش‌های حساب کامپیوتری و حساب تقریبی، مثلا با‌استفاده از ضرب‌کننده‌های هرس‌شده[[53]](#footnote-53)، مساحت و تاخیر مدار را کاهش دهیم. بدین ترتیب کارایی سخت‌افزار مدارهای شبکه عصبی و بهره‌وری انرژی آن افزایش خواهد یافت.

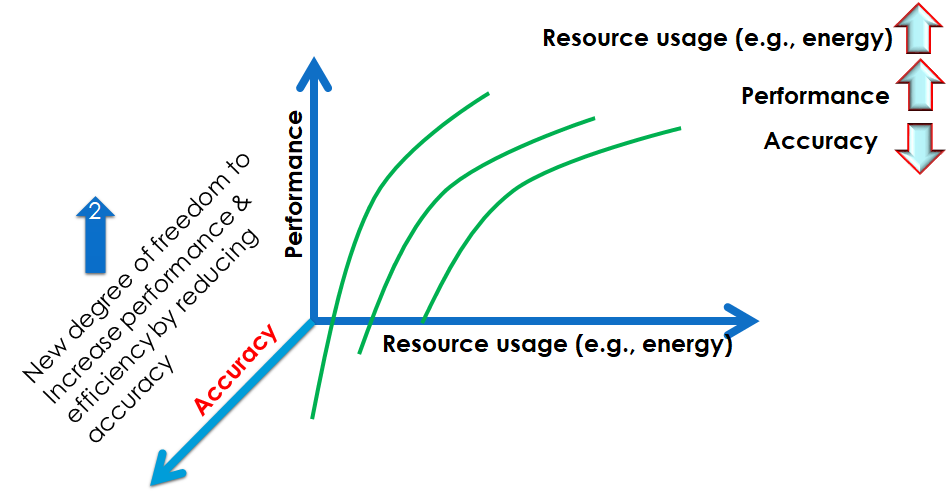
در راستای این کار تحقیقاتی قصد داریم از ابزار سنتز سطح بالایی همچون Vivado و زبان برنامه‌نویسی پایتون استفاده کنیم.

در شکل زیر مزایای استفاده از محاسبات تقریبی را در قالب کلی یک شکل نمایش داده شده است.



شکل ‏2‑28 نمایش مزایا در قالب خلاصه

نمودار زیر هم نمایانگر رابطه بین این سه مولفه در هنگام استفاده از محاسبات تقریبی است:

****

شکل ‏2‑29 نمودار کارایی روش محاسبات تقریبی

# فصل سوم: پیشینه تحقیق

## مروری بر تئوری‌ها و مطالعات پیشین

در این فصل قصد دارم که به بررسی پژوهشهای مختلفی که تاکنون در زمینه بهبود شبکه‌های عصبی پرداخته‌اند، بپردازم.

### بهبود شبکه‌های عصبی مصنوعی عمیق با استفاده از ابزارهای سنتز سطح بالا

ابزار سنتز سطح بالا ابزاری است که برنامه‌نویسانی که آشنایی چندانی با کدهای HDL و زبانهای توصیف سخت‌افزار ندارند، این قابلیت را می‌دهد که کدهای خود را با زبانهای سطح بالا از قبیل پایتون، متلب، C یا C++ بنویسند و خود ابزارهایی از قبیل abc، Vivado، LeFlow و ... آنها را به HDL تبدیل کنند. در این راستا پژوهش‌های فراوانی انجام شده است که در ادامه به پژوهشهای انجام شده پرداختم:

-کیم و همکاران[2]. در این مقاله، شتاب‌دهنده شبکه عصبی کانولوشن عمومی[[54]](#footnote-54) CNNA برای سیستم روی تراشه[[55]](#footnote-55) SoC ارائه شده است. هدف این بود که استنباط از شبکه‌های مختلف یادگیری عمیق در یک سیستم‌عامل SoC تعبیه‌شده تسریع شود. CNNA ارائه شده دارای معماری مقیاس‌پذیر است که از سنتز سطح بالا[[56]](#footnote-56) HLS و SystemC برای شتاب‌دهنده سخت‌افزاری استفاده می‌کند. این دستگاه قادر است هر CNN صادر شده از پایتون را تسریع کند و ترکیبی از لایه‌های کانولوشن، حداکثر تجمع و کاملاً متصل را پشتیبانی می‌کند. یک روش آموزش با استفاده از وزنهای کوانتیزه‌شده با نقطه ثابت پیشنهاد شده و در مقاله ارائه شده است. CNNA مبتنی بر الگو است و آن را قادر می سازد تا اهداف مختلف بستر Xilinx ZYNQ را مقیاس‌بندی کند. این رویکرد امکان طراحی فضای کاوش را فراهم می‌کند، که امکان کشف چندین شکل از CNNA در حین شبیه‌سازی C و RTL را فراهم می‌کند، و آن را متناسب با سیستم‌عامل و مدل مورد نظر قرار می‌دهد. از شبکه عصبی کانولوشن VGG16 برای آزمایش محلول بر روی صفحه[[57]](#footnote-57) Xilinx Ultra96 استفاده شد. نتیجه در آموزش با یک قالب Q2.14 با نقطه ثابت با مقیاس خودکار در مقایسه با مدل مشابه شناور، از دقت بالایی برخوردار است. این دستگاه در حالیکه میانگین توان مصرفی W 2.63 وات را داشت، می‌تواند استنباط را در 2.0 ثانیه انجام دهد که مربوط به بازدهی توان 6.0 GOPS / W برای شتاب‌دهنده CNN است[2]. به طور کلی یک فریم‌ورک یادگیری عمیق End-to-End برای کشف سریع شبکه‌های عصبی کوانتیزهQNN ارائه کرده که روی شتاب‌دهنده FINN ساخته شده است. FINN-R شامل آبشار شتاب‌دهنده‌های چند لایه است که برای معماری خط لوله بهینه شده است. این طراحی باعث کاهش انتقال‌داده‌ها بین حافظه اصلی و شتاب‌دهنده‌ها می‌شود. با این وجود فریم‌ورک، مشکل تولید متفاوت برای هر لایه را حل نمی‌کند.

همچنان که CNN ها همچنان برای مشکلات پیچیده به کار گرفته می شوند، توان کم، تأخیر و بهره‌وری انرژی در دستگاه‌های تعبیه شده با واحد پردازش مرکزی CPU یا واحد پردازش گرافیکی GPU چالش‌هایی را ایجاد می‌کند. به طور کلی، FPGA ها عملکرد بالاتری نسبت به CPU ها دارند و بازده انرژی بهتری نسبت به CPU ها و GPU ها دارند ولی با این وجود، زمان طراحی طولانی و نیاز به متخصصان HW استفاده از FPGA را محدود کرده است. در اینجا، ابزارهای سنتز سطح بالا HLS باعث گردآوری خودکار از برنامه‌های ضروری سطح بالا به مشخصات سطح پایین در زبان توصیف سخت افزار HDL شده است. با این حال، تسریع CNN های مقیاس بزرگ بر روی FPGA هنوز مشکل است، زیرا پارامترهای مدل معمولاً به حافظه بسیار بیشتری نسبت به ظرفیت روی تراشه FPGA نیاز دارند. چالش دیگر یافتن پیکربندی بهینه برای یک طرح شتابدهنده HW با توجه به زمان طولانی طراحی است. دامنه کار این مقاله توسعه معماری عمومی و انعطاف‌پذیر است که می‌تواند استنتاج شبکه‌های CNN را در سیستم چند پردازنده در طراحی تراشه MPSoC تسریع کند.

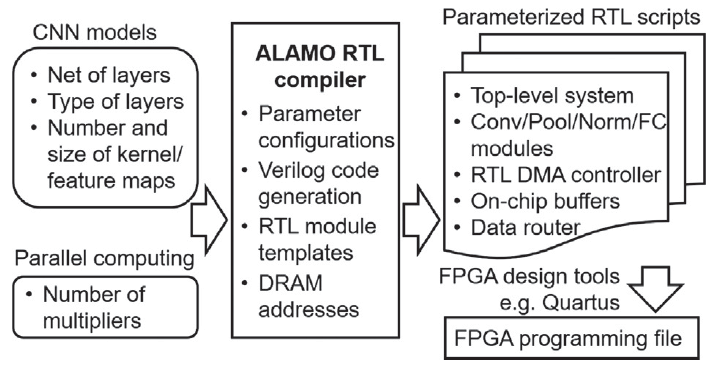
این طرح معماری HW/SW را ارائه می‌دهد، یعنی منطق قابل برنامه‌ریزی که در بافت FPGA و طراحی نرم‌افزار قرار دارد. معماری کلی طوری است که می‌تواند CNN های اصلی مانند AlexNetو VGG16 را که می‌توانند از یک چارچوب یادگیری عمیق مانند Keras صادر شوند، بپذیرد. این برنامه در چارچوب PYNQ با استفاده از Python و SystemC به منظور ایجاد یک الگوی عمومی بر اساس شتاب‌دهنده HW توسعه یافته است. برای یافتن طراحی بهینه، از شبیه‌سازی مبتنی بر SystemC برای کاوش در فضای طراحی پارامترهای پیکربندی بهینه شتاب‌دهنده CNN استفاده می‌شود. مدل طراحی با استفاده از HLS به مشخصات HDL ترجمه شده است. این مقاله در مورد دقت، سرعت و توان مصرفی شتاب‌دهنده و همچنین بازآموزی نقطه ثابت CNN بحث می‌کند.

در این بخش، به طور مختصر روشها و مفاهیم طراحی مورد استفاده در طراحی و پیاده‌سازی معماری شتاب‌دهنده شبکه عصبی کانولوشن CNNA را شرح می‌دهد. هنگام کار با FPGA ، روشهای مختلفی برای توسعه هسته خصوصیت ذهنی[[58]](#footnote-58) (IP) وجود دارد. یک روش پرکاربرد نوشتن کد در سطح رفتاری یا RTL به زبان HDL مانند Verilog یا VHDL است. این زبانها معمولاً وقت‌گیر هستند و به مهندس ماهر HDL نیاز دارند. امکان دیگر استفاده از HLSبرای توسعه IP است. HLS روشی برای نوشتن مدل‌های رفتاری سخت‌افزار به زبان انتزاعی بالاتر، معمولاً C است. این مدل رفتاری را می‌توان با استفاده از یک ابزار سنتز مانند Xilinx Vivado HLS به یک مدل RTL ترسیم و برنامه‌ریزی کرد. توسعه IP با استفاده از HLS به دلیل سطح انتزاعی بالاتر و همچنین جریان طراحی ذاتی می تواند بسیار کارآمد باشد.

در این کار، SystemC با جریان طراحی توصیف‌شده در یک روش کارآمد است که در آن می‌توان IP را با استفاده از HLS نوشت و تأیید کرد. SystemC می‌تواند ساختار سخت‌افزاری و همزمانی را به نحو مطلوبتری نسبت به C مدلسازی کند.

نویسندگان مقاله حاضر استدلال می‌کنند که انتخاب پایتون بهره‌وری را افزایش می‌دهد، زیرا پایتون یک زبان مفسری سطح بالا است، که می‌تواند به سرعت توسعه داده شود و به صورت تکه‌تکه اجرا شود، بنابراین امکان اشکال‌زدایی سریع را فراهم می‌کند.

- روش پیشنهادی ALAMO [4]. استقرار شبکه‌های عصبی Convolutional (CNN) بر روی یک سیستم قابل حمل به دلیل حجم زیاد داده ها، میزان گسترده محاسبات و دسترسی های مکرر حافظه هنوز هم چالش برانگیز است. اگرچه ابزارهای سنتز سطح بالا (به عنوان مثالHLS ، OpenCL) برای FPGA به طرز چشمگیری زمان طراحی را کاهش می‌دهد، اما پیاده‌سازی های حاصل با توجه به تخصیص منابع برای به حداکثر رساندن موازی سازی و توان تولید هنوز ناکارآمد هستند. طراحی دستی در سطح سخت افزار (به عنوان مثال، RTL) می‌تواند کارایی را بهبود بخشد و به شتاب بیشتری دست یابد، اما این امر به درک عمیقی از ساختار الگوریتم و معماری سیستم FPGA نیاز دارد. این کار یک راه حل مقیاس پذیر را ارائه می‌دهد که به انعطاف پذیری و زمان طراحی سنتز سطح بالا و نزدیکترین حد بهینه اجرای RTL می رسد. راه حل پیشنهادی کامپایلری است که ساختار و پارامترهای الگوریتم را تجزیه و تحلیل می‌کند و به طور خودکار مجموعه ای از محاسبات مدولار و مقیاس پذیر را برای تسریع در کارکرد الگوریتم های یادگیری عمیق مختلف بر روی FPGA به طور خودکار ادغام می‌کند. با ادغام این ماژول ها در کنار هم برای اجرای نهایی CNN ، این کار کمی استراتژی طراحی کامپایلر را برای بهینه‌سازی توان تولید مدل CNN داده شده تحت محدودیت های منابع FPGA ، کمی تجزیه و تحلیل می‌کند. کامپایلر RTL پیشنهادی، به نام ALAMO ، در Altera Stratix-V GXA7 FPGA برای کارهای استنباطی مدل های AlexNet و NiN CNN نشان داده شده است. نتایج راه حل کامپایلر خودکار برای شتاب‌دهنده سخت افزاری مدولار و مقیاس پذیر یادگیری عمیق را نشان می‌دهد.



شکل ‎1‑4 The compilation flow with the proposed ALAMO RTL compiler

### بهبود شبکه‌های عصبی مصنوعی عمیق با اجرای موازی آنها

حوزه‌های کاربرد GPU ها عبارتند از: ترمیم تصویر[[59]](#footnote-59)، تقسیم‌بندی تصویر[[60]](#footnote-60)، کاهش نویز تصویر[[61]](#footnote-61)، بازسازی تصویر[[62]](#footnote-62) و فیلتر کردن[[63]](#footnote-63).

چند تا شرکت معروف هستند که کارت گرافیک تولید می‌کنند. دو تا از معروفترین فریم‌ورک‌ها برای این نوع برنامه‌نویسی عبارتند از:

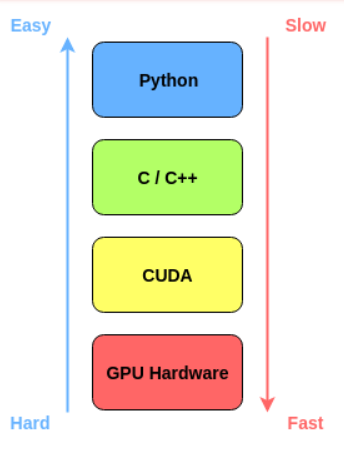
**فریم‌ورک CUDA:** مخفف Compute Unified Device Architecture مختص شرکت Nvidia

**فریم‌ورک OpenCL:** در کارت‌های گرافیک دیگر شرکت‌ها مانند AMD استفاده می‌شود

همانطور که اشاره شد، CUDA را فقط در کارت‌های گرافیک شرکت Nvidia میتوان استفاده کرد در حالیکه OpenCL یک استاندارد برای محاسبات موازی روی دستگاه‌های مختلف مثل GPU، CPU و FPGA ها است.

فریم‌ورک CUDA به زبان C نوشته شده است اما در پایتون می‌توان با استفاده از کتابخانه PyCuda و در جاوا از JCuda از آن استفاده کرد. اما یادگیری و برنامه‌نویسی در آنها کمی سخت است. همچنین باید بدانید که CUDA و واحد‌های سازنده آن مانند cuDNN و cuBLAS چگونه عمل می‌کنند.

نمونه‌های آسانتری هم مانند Deeplearning4J و ND4J وجود دارند که از GPU استفاده می‌کنند. ND4J را می‌توان معادل کتابخانه NumPy در پایتون، برای جاوا در نظر گرفت.



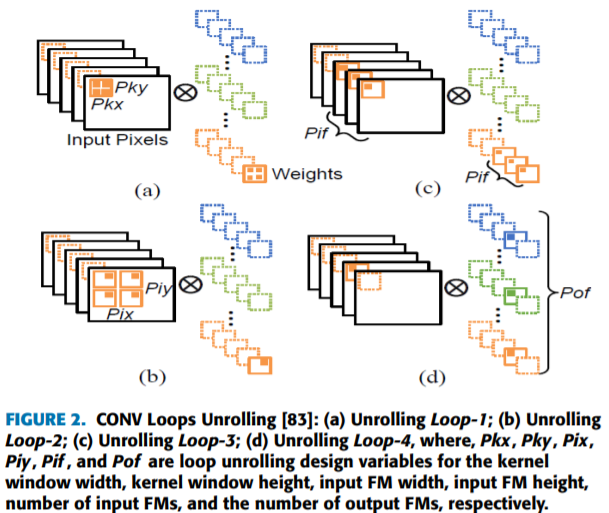
در این قسمت قصد دارم پژوهش‌هایی را توضیح دهم که با استفاده از اجرای موازی برنامه‌ها سرعت اجرای این الگوریتمها را بهتر کرده‌اند:

- شاوانا و همکاران[36]. یک مدل roofline برای تسریع شبکه‌های عصبی کانولوشن در FPGA پیشنهاد کردند، مدل roofline مدلی با عملکرد بصری است که برای ارتباط عملکرد قابل دستیابی با حداکثر عملکرد قابل استفاده، از بستر سخت‌افزاری و ارتباط حافظه خارج از تراشه استفاده می‌کند. تمرکز در کار آنها در درجه اول سرعت بخشیدن به لایه‌های کانولوشن است، زیرا بیش از 90% از زمان محاسبات را در طی فرآیند پیش‌بینی مصرف می‌کند.

در این مقاله برای این منظور به این پژوهش رسیدگی شده، که شبکه‌های عصبی کانولوشنال (CNN) اثربخشی خود را در برنامه‌های تشخیص تصویر نشان داده اند. با این حال، آنها نیاز به عملیات CPU فشرده و پهنای باند حافظه دارند که باعث می شود پردازنده‌های عمومی نتوانند به سطح عملکرد مطلوب برسند. در نتیجه، شتاب دهنده‌های سخت‌افزاری که از مدارهای مجتمع مخصوص برنامه، FPGA و واحدهای پردازش گرافیکی استفاده می کنند برای بهبود توان CNN ها استفاده شده اند. FPGA برای تسریع در اجرای شبکه های یادگیری عمیق به دلیل توانایی آنها در به حداکثر رساندن موازی کاری و بهره وری انرژی آنها ، تصویب شده است. تکنیک های مورد بررسی در این مقاله نشان دهنده روندهای اخیر در شتاب دهنده های مبتنی بر FPGA شبکه های یادگیری عمیق است.

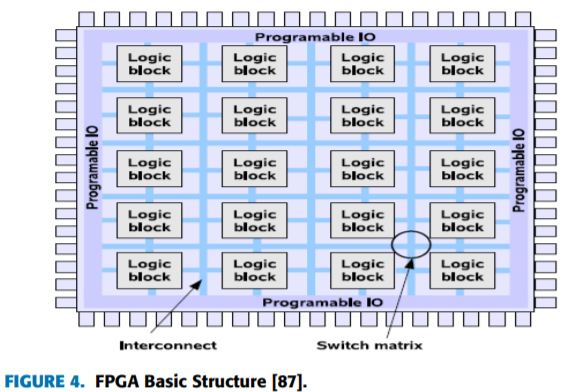
شتاب‌دهنده‌های سخت‌افزاری FPGA و ASIC نسبت به شتاب‌دهنده‌های مبتنی بر GPU دارای حافظه نسبتاً محدود، پهنای باند ورودی/خروجی و منابع محاسباتی هستند. با این حال، آنها می‌توانند حداقل عملکرد متوسط را با مصرف توان کمتر به دست آورند [62]. گذرگاه طراحی ASIC را می‌توان با سفارشی‌سازی سلسله مراتب حافظه و اختصاص منابع اختصاصی بهبود بخشید [63]. با این حال، چرخه توسعه، هزینه و انعطاف‌پذیری در تسریع شبکه‌های یادگیری عمیق مبتنی بر ASIC رضایت‌بخش نیست [64]، [65]. به عنوان یک جایگزین، شتاب‌دهنده‌های مبتنی بر FPGA در حال حاضر برای ارائه گذرگاه بالا با قیمت مناسب با مصرف توان کم و تنظیم مجدد مورد استفاده قرار می گیرند [66] ، [67]. در دسترس بودن ابزارهای سنتز سطح بالا (HLS) ، با استفاده از C++ یا C ، از فروشندگان FPGA ، مانع برنامه نویسی را کاهش داده و زمان توسعه شتاب‌دهنده‌های سخت‌افزاری مبتنی بر FPGA را کوتاه می‌کند [68]-[70].

به طور خلاصه، عملیات کانولوشن شامل چهار سطح حلقه است. حلقه FM خروجی (4Loop-)، حلقه FM های ورودی (3Loop-) ، حلقه در ابعاد یک FM ورودی واحد (عملیات اسکن ، 2Loop-) و حلقه اندازه پنجره هسته (multiply and-accumulate (MAC) Loop-1). لایه های CONV در الگوریتم‌های CNN غالب هستند؛ زیرا اغلب بیش از 90٪ از کل عملیات CNN را تشکیل می‌دهند [28] ، [29] ، [49] ، [74] ، [77] ، [78]. بنابراین، تلاشهای زیادی برای سرعت بخشیدن به عملیات CONV با استفاده از تکنیک باز کردن[[64]](#footnote-64) حلقه انجام شده است [55] ، [79]. باز کردن حلقه موازی محاسبه CONV MAC را به حداکثر می‌رساند که نیاز به توجه ویژه‌ای به عناصر پردازش[[65]](#footnote-65) (PE) و رجیسترهای معماری آرایه‌ها دارد. شکل 2 باز شدن حلقه از سطوح حلقه های CONV را نشان می‌دهد.



داخل این مقاله مختصری راجع‌به شبکه‌های عصبی عمیق یک مقداری توضیح داده شده است.

FPGA ها دستگاه های قابل برنامه‌ریزی هستند که بستری انعطاف‌پذیر برای اجرای عملکرد سخت‌افزاری سفارشی با هزینه توسعه کم ارائه می دهند. آنها عمدتا از مجموعه ای از سلول های منطقی قابل برنامه ریزی ، به نام بلوک‌های منطقی قابل تنظیم[[66]](#footnote-66) (CLB) ، یک شبکه اتصال قابل برنامه‌ریزی و مجموعه‌ای از سلول‌های ورودی و خروجی قابل برنامه‌ریزی در اطراف دستگاه تشکیل شده‌اند [87]. علاوه بر این، آنها دارای مجموعه‌ای غنی از اجزای تعبیه شده مانند بلوک های پردازش سیگنال دیجیتال (DSP) هستند که برای انجام عملیات حساس مانند ضرب و انباشت ، بلوک RAM (BRAM) ، جداول جستجو (LUT) استفاده می‌شود. ، فلیپ فلاپ (FF)، واحد مدیریت ساعت ، پیوندهای ورودی/خروجی با سرعت بالا و سایر موارد تشکیل شده‌اند. شکل 4 ساختار اساسی یک FPGA را نشان می‌دهد.



FPGA از مزیت حداکثر عملکرد در هر وات مصرف توان و کاهش هزینه برای عملیات در مقیاس بزرگ برخوردار است [91]. این باعث می‌شود، به عنوان شتاب‌دهنده برای دستگاه‌های باتری و سرویس‌های ابری در سرورهای بزرگ انتخابی عالی باشند. FPGA با توجه به انعطاف‌پذیری در اجرای معماری‌ها با درجه موازی کاری زیاد و سرعت اجرای بالا، به طور گسترده برای تسریع یادگیری عمیق استفاده می‌شود [92].

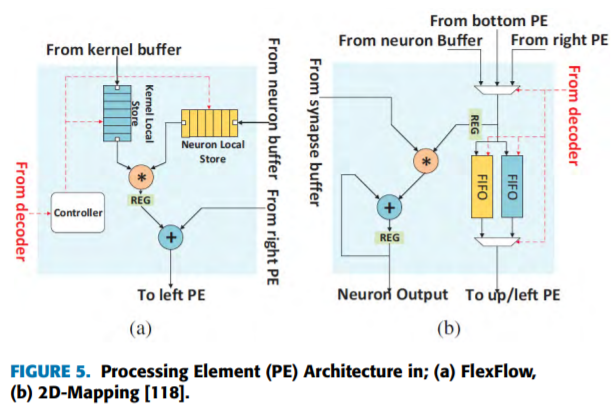
پیاده سازی شبکه های یادگیری عمیق و به ویژه CNN ها در FPGA دارای چند چالش از جمله نیاز به مقدار قابل توجهی از حافظه، پهنای باند حافظه خارجی و منابع محاسباتی به ترتیب میلیاردها عملیات در ثانیه است [97]. بدون پیاده سازی دقیق شبکه های یادگیری عمیق و به حداکثر رساندن اشتراک منابع ، ممکن است پیاده سازی به دلیل محدودیت منابع منطقی در FPGA ها مناسب نباشد.

در اثری دیگر ، هان و همکاران. [104] از تکنیک های هرس شبکه [105]-[107] برای کاهش برازش و پیچیدگی مدل های شبکه عصبی استفاده کرد. نتایج آنها نشان داد که هرس اتصالات اضافی و اتصالات کمتر تأثیرگذار به ترتیب برای مدلهای AlexNet و VGG-16 فشرده سازی 9 و 13برابر را به دست آوردند ، در حالی که از دست دادن دقت صفر برای هر دو به دست آمد.

در اکثر تکنیک های قبلی راه حل هایی را ارائه می دهند که در یکی از سه معماری نماینده قرار می‌گیرد: (i) سیستولیک ، (ii) نگاشت دو بعدی و (iii) کاشی کاری.

به دلیل محدودیت جریان داده در هر یک از سه معماری فوق ، اکثر شتاب دهنده های موجود تنها از یک موازی کاری خاص پشتیبانی می کنند. معماری سیستولیک از موازی کاری سیناپس[[67]](#footnote-67) (SP) ، معماری های نقشه برداری دوبعدی از موازی سازی نورون[[68]](#footnote-68) (NP) و معماری های کاشی کاری از موازی کاری نقشه[[69]](#footnote-69) (FP) بهره می برند. با این حال ، در یک CNN عملی ، نوع موازی غالب به تعداد FM های ورودی ، تعداد FM های خروجی ، اندازه FM های خروجی و اندازه هسته بستگی دارد.

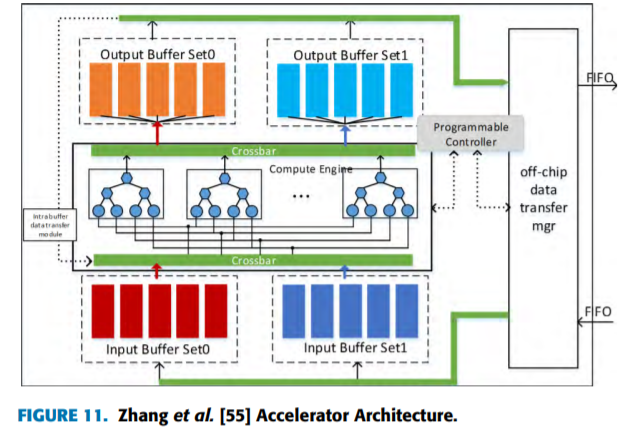
معماری FlexFlow برای محاسبه استفاده از منابع ، عملکرد ، قدرت ، انرژی و مساحت مورد ارزیابی قرار گرفت که شکل ساختار آن در زیر آمده است:



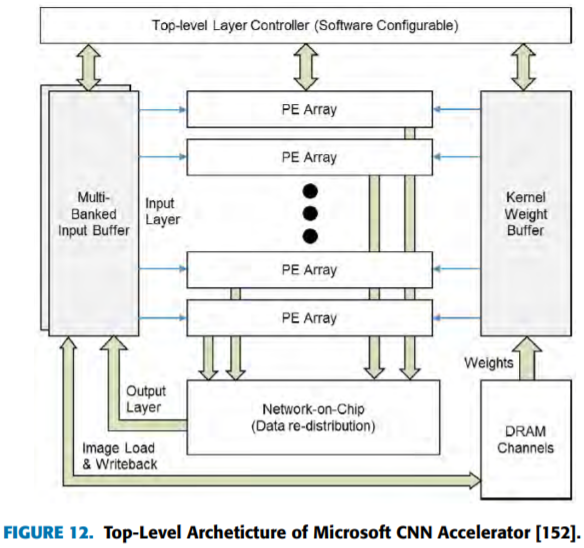
مقایسه با سه معماری معمولی (یعنی سیستولیک ، نقشه برداری دو بعدی و کاشی کاری) با استفاده از شش بار عملی، از جمله AlexNet و VGG انجام شد. آنها همچنین مقیاس پذیری FlexFlow را از نظر استفاده از منابع، توان و مساحت با مقیاس های رو به رشد موتور محاسباتی مورد بررسی قرار دادند.

از مدل عملکرد سقف برای شناسایی طرح بهینه از تمام راه حل های ممکن در فضای طراحی استفاده می شود. به طور خاص ، نویسندگان تمام طرح های قانونی احتمالی را که از تجزیه و تحلیل چند وجهی در خط سقف ارائه شده اند ، مدل سازی می کنند تا فاکتور باز شدن مطلوب برای هر لایه کانولوشن ، که در آن Tm و Tn به ترتیب اندازه کاشی برای FM های خروجی و FM های ورودی هستند. با این حال ، طراحی یک شتاب دهنده CNN با عوامل بازکننده متفاوت برای هر لایه کانولوشن چالش برانگیز است. بنابراین ، معماری پیشنهادی تمام طرح های معتبر احتمالی را برای یافتن عوامل باز شدن یکنواخت لایه ای متقابل برشمرده است. پس از آن ، شتاب دهنده سخت افزاری بر اساس فاکتورهای باز شدن مطلوب لایه ای اجرا می شود.

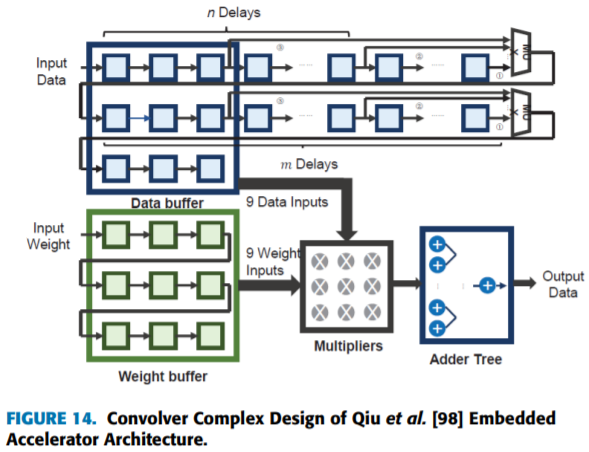
شتاب دهنده با استفاده از ابزار سنتز سطح بالا 2013.4 Vivado طراحی شده و بر روی برد Xilinx VC FPGA 707 با سرعت 100 مگاهرتز اجرا شده است. نتایج تجربی نشان می دهد که پیاده سازی پیشنهادی به حداکثر عملکرد GFLOPS 61.62 و همچنین سرعت 17.42 برابر در اجرای نرم افزار بر روی پردازنده Intel Xeon CPU E 5-2430 در 2.20 گیگاهرتز با 15 مگابایت حافظه پنهان و 16 نخ دست می یابد. علاوه بر این ، نتایج نشان می دهد که معماری FPGA پیشنهادی 24.6 برابر نسبت به اجرای نرم افزار دارای مصرف انرژی کمتری است زیرا کل مصرف توان تنها 18.6 وات است. پیاده سازی پیشنهادی دارای محدودیت هایی مانند طراحی شتاب دهنده با فاکتورهای کانولوشن جدید لایه ای برای معماری های مختلف CNN است. علاوه بر این ، استفاده از شتاب دهنده CNN با عوامل باز شدن یکنواخت ممکن است برای برخی از لایه های CONV غیر مطلوب باشد ، که بر عملکرد کلی تأثیر می‌گذارد.



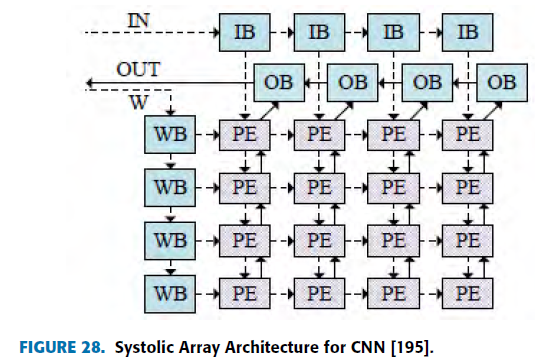
معماری سطح بالای شتاب دهنده پیشنهادی CNN در شکل 12 نشان داده شده است. بافر ورودی چند بانکی و بافر وزن هسته به ترتیب برای ارائه یک طرح بافر کارآمد FM و وزن استفاده می شود. برای به حداقل رساندن ترافیک حافظه خارج از تراشه ، یک شبکه تخصصی روی تراشه طراحی شده است تا FM های خروجی را به جای انتقال به حافظه خارجی ، دوباره روی بافر ورودی چند بانکی توزیع کند.



معماری پیشنهادی شامل یک سیستم پردازش (CPU) و منطق قابل برنامه ریزی (FPGA) است. محاسبات CNN از طریق طراحی ویژه ماژول های عنصر پردازش در FPGA انجام می شود. ماژول های اصلی در عنصر پردازش عبارتند از convolver complex، max-pooling، non-linearity، data shift، bias shift و adder tree، همانطور که در شکل 13 نشان داده شده است. همانطور که در شکل 14 نشان داده شده است، برای دستیابی به عملیات کانولوشن و همچنین محاسبه ضرب لایه FC بردار ماتریسی.



در این مقاله از معماری سیستولیک استفاده کرده است:



این مقاله تأکید ویژه ای بر CNN ها دارد، زیرا آنها کاربردهای وسیعی در زمینه تشخیص تصویر دارند و نیاز به پردازنده و عملیات فشرده حافظه دارند که می تواند با استفاده از توانایی ذاتی FPGA برای به حداکثر رساندن موازی کاری ، به طور موثری تسریع شود.

این مقاله به تکنیکهای شتاب دادن به الگوریتم های یادگیری عمیق و CNN ها از نظر نرم افزاری و سخت افزاری می پردازد، هسته اصلی این مقاله مرور تکنیک های اخیر مورد استفاده در تسریع CNN ها در FPGA ها است. یک بررسی کامل و به روز ارائه شده که استفاده از امکانات و تکنیک های مختلف مانند بهره برداری از موازی سازی با استفاده از حلقه کاشی و باز کردن حلقه ، استفاده موثر از حافظه داخلی برای حداکثر استفاده مجدد از داده ها ، خط کشی عملیات و استفاده موثر از اندازه داده ها را برای بهینه سازی استفاده از منابع FPGA نشان می دهد.

این مقاله همچنین استفاده از ابزارهایی برای ایجاد اسکریپت های رجیستر سطح انتقال[[70]](#footnote-70) (RTL) را ارائه می دهد که نه تنها به خودکارسازی فرایند طراحی کمک می کند ، بلکه به کاوش در فضای طراحی و پیشنهاد سخت افزار کارآمد نیز کمک می کند. این مقاله در مورد استفاده از تجزیه و تحلیلهایی مانند: (i) تجزیه و تحلیل بار کار در تعیین محاسبات قابل موازی سازی ، (ii) عوامل unrolling حلقه بهینه، (iii) تعیین الگوهای دسترسی برای بهبود موقعیت داده ها و غیره. علاوه بر این ، مروری کوتاه بر استفاده از روشهای اکتشافی غیر قطعی در حل مشکلات بهینه سازی ترکیبی NP در طراحی و اجرای CNN ها ارائه شده است. در نهایت ، مقاله ویژگی های کلیدی مورد استفاده در تکنیک های مختلف شتاب‌دهنده CNN مبتنی بر FPGA را خلاصه می کند و توصیه هایی را برای افزایش اثربخشی استفاده از FPGA در شتاب CNN ارائه می دهد.

### بهبود شبکه‌های عصبی مصنوعی عمیق با استفاده از محاسبات تقریبی

پارا و همکاران[37][14]. محاسبات تقریبی در کاهش مصرف انرژی شبکه‌های عصبی عمیق (DNN) که در سیستم های تعبیه‌شده اجرا شده، موفقیت آمیز بوده است. برای تقریب کارآمد DNN در سطح نرم افزار و سخت افزار، برای کاهش زمان اجرا و بهینه‌سازی و همچنین برای به حداکثر رساندن صرفه جویی در انرژی، یک محیط شبیه‌سازی تخصصی و روش بهینه‌سازی مورد نیاز است. فریم ورک های سنتی برای محاسبه تقریبی لایه‌های متقابل DNN ها عموماً فقط برای شبیه‌سازی لایه‌های کانولوشن و کاملاً متصل ساخته می‌شوند و انواع DNN را برای تقریب بهینه‌سازی می‌کنند. در این کار، یک محیط شبیه‌سازی ویژه برای DNN های تقریبی ارائه می‌دهیم، که امکان بهینه‌سازی چندین ساختار DNN ساخته شده با لایه‌های پیچیده تر DNN مانند کانولوشن های عمیق و واحدهای عصبی بازگشتی (RNN ها) برای پردازش سری های زمانی را فراهم می‌کند[37] [14].

سربار زمان اجرای کم از طریق شتاب کارآمد GPU به دست می آید. علاوه بر این ، ما یک تجزیه و تحلیل از استحکام تقریبی DNN و RNN در برابر سر و صدای کمی و سطوح تقریبی مختلف ارائه می دهیم. سرانجام ، از طریق بازآموزی تقریبی تخصصی ، به صرفه جویی در انرژی و ضایعات ناچیز دقت با DNN های بسیار پیچیده برای طبقه بندی تصویر با ImageNet ، مانند MobileNet و RNNs برای تشخیص لغات کلیدی با مجموعه داده های فرمان گفتار ، می رسیم.

یک رویکرد امیدوار کننده محاسبه تقریبی چند لایه است که شامل ترکیب تکنیک های تقریب در سطح نرم افزاری و سخت افزاری برای کاهش منابع محاسباتی در برنامه های تعبیه شده است.

تقریب در سطح نرم افزاری ، مانند هرس یا کوانتیزه ، حتی برای DNN های بسیار پیچیده ای که به کارهای واقعی اختصاص داده شده اند ، مانند تشخیص تصویر در مقیاس بزرگ، به طور کامل مورد بررسی قرار گرفته است [1]. این امر به لطف چارچوب های تخصصی یادگیری ماشین منبع باز مانند Tensorflow Lite [2] یا Ristretto [3] امکان پذیر شده است ، که امکان کاوش سریع تکنیک های تقریب نرم افزاری را فراهم می آورد.

از سوی دیگر ، تقریب در سطح سخت افزار برای کشف ، ارزیابی و بهینه سازی دشوارتر است ، عمدتا به این دلیل که شبیه سازی واحدهای محاسباتی تقریبی[[71]](#footnote-71) (ACU) زمان اجرا را در چارچوبهای سنتی برای محاسبات تقریبی افزایش می دهد. علاوه بر این ، چارچوب های شبیه سازی چند لایه تخصصی برای DNN های تقریبی به طور کلی برای CPU اجرا می شود ، که زمان آموزش و اعتبارسنجی[[72]](#footnote-72) را در مقایسه با پیاده سازی دقیق GPU نیز افزایش می دهد [4]. در کارهای اخیر ، چارچوب های شبیه‌سازی مبتنی بر GPU برای تقریب لایه ای ، مانند Concrete [5] و ProxSim [6] ، برای تسریع در بهینه سازی معماری تقریبی DNN با لایه های عصبی مشترک مانند لایه های کانولوشن و کاملاً متصل به یکدیگر پیشنهاد شده است.

در این بخش ، ما یک بررسی کوتاه از روش های تقریبی مربوط به نرم افزار و سخت افزار برای DNN ها گزارش می دهیم.

\* **کوانتیزیشن پهنای بیت پایین DNN ها**[[73]](#footnote-73). به عنوان مقیاس بندی دقیق وزن ها و ورودی های DNN شناخته می شود ، کوانتیزه منجر به کاهش حافظه و منطق محاسباتی[[74]](#footnote-74) می شود. از طریق رویکردهای مختلف ، پارامترها و فعالسازی های DNN را می توان به طور کلی تا عدد صحیح 8 بیتی بدون از دست دادن دقت افزایش داد [1]. یک تابع کوانتیزاسیون رایج ، کوانتیزاسیون خطی است.

\* **هرس فیلتر**[[75]](#footnote-75). برای کاهش تعداد هسته های DNN بدون از دست دادن دقت ، چندین روش برای هرس هسته یا فیلتر بر اساس خواص مختلف هسته پیشنهاد شده است. محبوب ترین روشهای هرس در [17،18] ارائه شده است. در [17] ، نویسندگان پیشنهاد می کنند که بر اساس مجموع وزن هر هسته ، حداقل فیلترهای مهم را هرس کنند. در [18] ، پیشنهاد می شود که فیلترهایی شبیه به فیلترهای مهم تر ، که توسط خوشه بندی k-means انتخاب شده اند ، هرس شوند تا زمانی که دقت فراتر از یک آستانه از پیش تعیین شده کاهش یابد.

**\* ACU ها در لایه های DNN.** همراه با کمی سازی پهنای بیت پایین ، استفاده از ACU ها در محاسبه DNN منجر به صرفه جویی در انرژی بیشتر در مقایسه با نمونه دقیق آن می شود. روشهای تقریبی جزئی برای کاهش مصرف انرژی در DNN های پیچیده برای طبقه بندی تصویر اثبات شده است. نمونه هایی از تقریب نسبی DNN در [19] و [20] ارائه شده است ، جایی که ضریب تقریبی در نورونهای مقاوم در برابر خطا یا با تجزیه و تحلیل انتشار مجدد خطا یا محاسبه مشتق خطای تقریب انتخاب شده است.

روش تقریبی جزئی دیگر بر اساس جستجوی ژنتیکی [4] ، برای تقریب نسبی کارآمد DNN بدون بازآموزی پیشنهاد شده است. با این وجود ، با تقریب کامل ، که شامل استفاده از سخت افزار تقریبی در همه نورونهای DNN است ، می توان صرفه جویی بهتری در انرژی به دست آورد. از طریق تکنیک های بهینه سازی مناسب ، تقریب کامل DNN نیز می تواند منجر به کاهش دقت کمتر شود. قابلیت استفاده از تقریب کامل به DNN با پیچیدگی متفاوت اخیراً در [6،21] ثابت شده است و تمرکز ما در این کار تحقیقاتی است. مروری بر ویژگیها و نمایش گرافیکی تکنیکهای تقریب جزئی و کامل در شکل 1 ارائه شده است.



شکل ‎1‑5 محاسبات تقریبی برای DNN ها

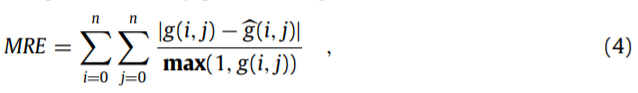
اکتشافات اخیر تکنیک های تقریب لایه ای در DNN ها منجر به معرفی چارچوب های شبیه سازی مختلف برای محاسبه تقریبی DNN شده است. نویسندگان [22] AxDNN را معرفی کردند ، یک چارچوب شبیه سازی پیش از RTL جدید برای مقایسه استراتژی های تقریبی مختلف مانند مقیاس بندی دقیق ولتاژ ، ضرب تقریبی و هرس فعال سازی. چارچوب های تخصصی جهت بهینه سازی DNN های تقریبی ، مانند ALWANN [4] ، بهینه سازی کارآمد تقریب جزئی DNN را ارائه می دهند. با این حال ، یک نقطه ضعف قابل توجه این چارچوب ها زمان اجرای زیاد در مقایسه با محاسبه دقیق DNN است ، زیرا این موارد فقط برای CPU اجرا می شوند. یکی دیگر از چارچوبهای تقریبی چند لایه ای برای شبیه سازی GPU ، Concrete [5] است که بر اساس Caffe [23] است ، که شامل بلوکهای تخصصی برای شبیه سازی ACU ها ، مانند ضربات تقریبی ، در لایه‌های مشترک DNN (متحرک و کاملاً متصل) با سربار کمی نسبت به همتای دقیق در نهایت ، ProxSim [6] ، بر اساس Tensorflow [2] ، نه تنها به سمت شبیه سازی ACU با سرعت GPU برای تقریب نسبی و کامل DNN ، بلکه به سمت بهینه سازی تقریبی کارآمد DNN جهت دستیابی به صرفه جویی بیشتر در انرژی با ضررهای ناچیز دقت است.

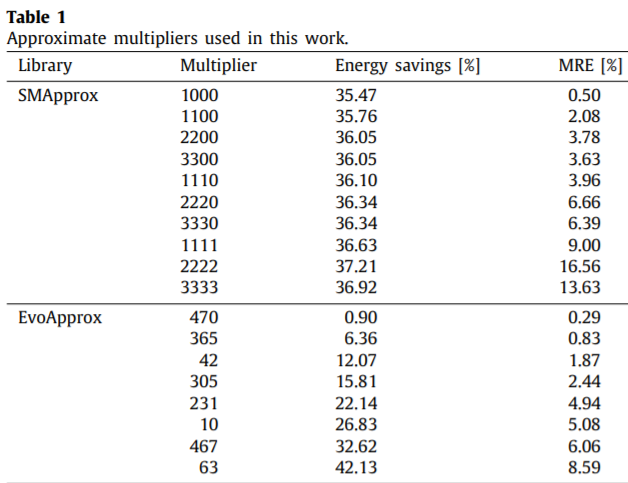
برای بررسی استفاده از ACU در محاسبه DNN ، ما 18 ضرب کننده متفاوت تقریبی 8 بیتی مختلف را از منابع باز زیر پیاده سازی می کنیم:

* SMApprox [24]. Library with more than 200 approximate multipliers optimized for FPGA.
* EvoApprox [25]. Library with 471 approximate multipliers designed through cartesian genetic programming.

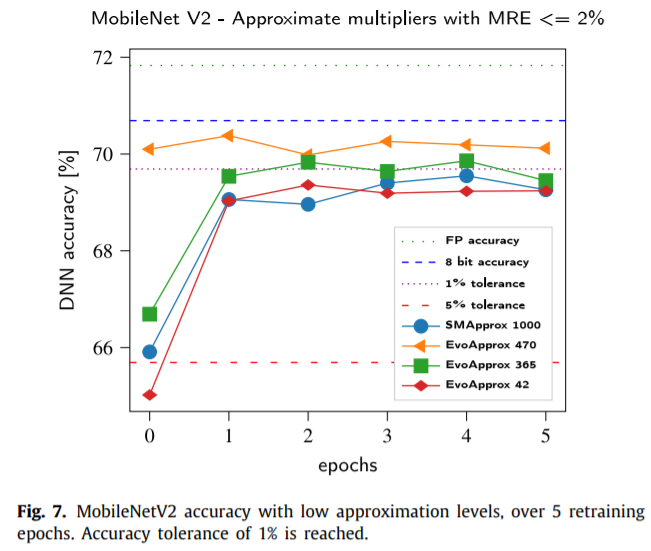
همه ضرب های اجرا شده به طور تصادفی از Pareto front در MRE و صرفه جویی در انرژی کتابخانه های مربوطه انتخاب شدند. در اینجا MRE را به عنوان معیار اصلی انتخاب می کند زیرا مستقیماً با تخریب دقت نهایی DNN متناسب است.

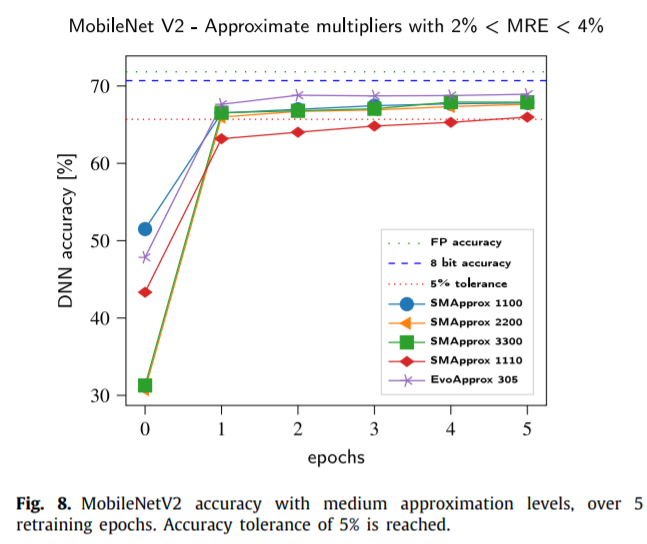
در جدول 1 ، صرفه جویی نسبی انرژی همه ضرب های انتخاب شده در بسترهای مربوطه را گزارش می کند، همانطور که در [24،25] گزارش شده است ، و همچنین خطای نسبی متوسط[[76]](#footnote-76) آنها (MRE)، که به طور رسمی برای هر خروجی احتمالی به شرح زیر محاسبه می شود:

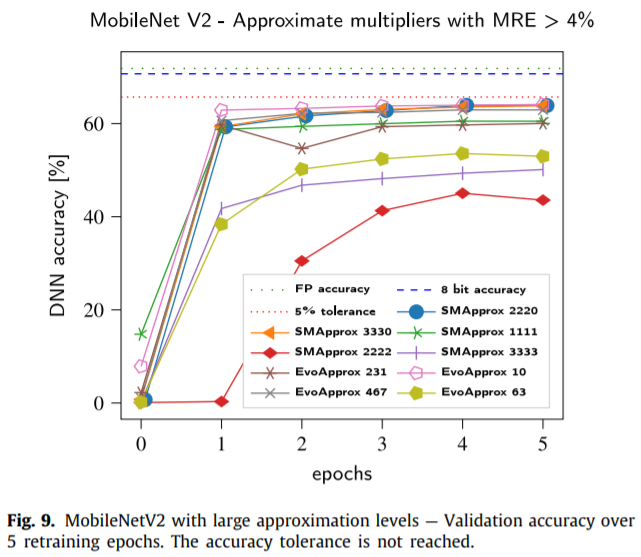




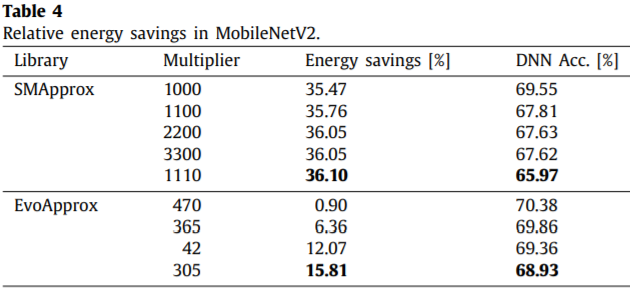
با توجه به پیچیدگی MobileNetV2 ، دو تحمل دقت را پیشنهاد می کند: برای سطوح تقریبی پایین (ضرب با MRE کوچکتر از 2)) ، افت دقت 1 w w.r.t. دقت کمی 8 بیتی مجاز است ، و برای سطوح تقریبی بزرگتر ، حداکثر تخریب دقت 5% پیشنهاد شده است. MobileNetV2 را برای 5 دوره با همه 18 ضرب کننده تقریبی مختلف از جدول 1 ، با استفاده از روش محاسبه گرادیان، مجددا آموزش می دهد. نتایج در شکل 7 ارائه شده است (برای ضرب با MRE کوچکتر از 2)) ، شکل 8 (برای ضرب با MRE بین 2 تا 4 درصد) ، و شکل 9 (برای ضرب با MRE بزرگتر از 4 درصد)







در مورد بهینه سازی دقت DNN ، مشاهده می کنیم که پس از اولین دوره ، بزرگترین بهبود دقت حاصل می شود. پس از آن ، دقت DNN به آرامی به حداکثر مقدار ممکن همگرا می شود ، که به شدت به خطای تقریب بستگی دارد. علاوه بر این ، ما مشاهده می کنیم که محدودیت تخریب دقت پیشنهادی ما با استفاده از ضرب تقریبی با MRE بزرگتر از 4 قابل دستیابی نیست.

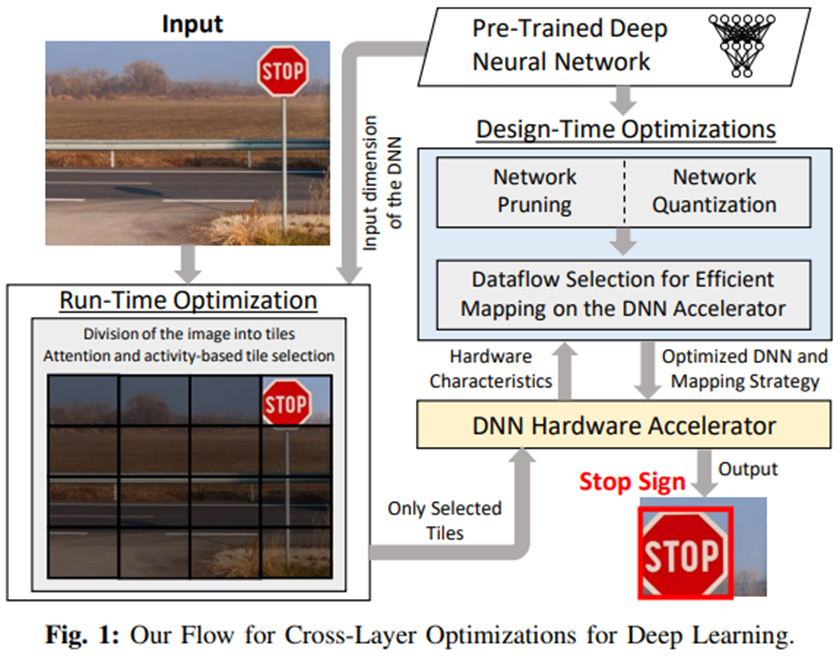


برای این ارزیابی ، صرفه جویی نسبی انرژی را در ضرب های انجام شده در کانولوشن های عمیق ، در کانولوشن های سنتی و در لایه های FC با استفاده از صرفه جویی در انرژی یک ضرب تقریبی واحد برآورد می کنیم. در جدول 4 ، تمام ضرب کننده هایی که دارای تحمل دقت DNN 5٪ هستند ، ارائه شده است. ما دقت نهایی پس از بازآموزی و صرفه جویی نسبی انرژی را نیز شامل می شویم. با استفاده از کتابخانه EvoApprox (ضرب 305) و 36٪ با استفاده از کتابخانه SMApprox (ضرب 1110) ، می توانیم 15 درصد صرفه جویی انرژی نسبی داشته باشیم.

محاسبه تقریبی در معماری DNN برای کارهای مختلف ادراک می تواند منجر به صرفه جویی در محاسبات امیدوار کننده شود. در این کار ProxSimV2 ، یک چارچوب شبیه سازی تخصصی برای DNN های تقریبی با لایه های کانولوشن مانند کانولوشن های عمیق و واحدهای بازگشتی مانند GRU ها را ارائه می دهد. علاوه بر این ، ترکیبی کارآمد از تکنیک های تقریب لایه ای را پیشنهاد می کند مانند کمی سازی پهنای بیت پایین و استفاده از ضرب کننده های تقریبی. این امر با پیاده سازی انواع لایه های تقریبی DNN در چارچوب شبیه سازی و همچنین یک روش بازآموزی بهینه برای محاسبه تقریبی DNN به دست می آید. از طریق ارزیابی گسترده DNN های پیشرفته مانند MobileNetV2 و دو RNN برای تشخیص کلمات کلیدی ، تطبیق پذیری ProxSimV2 و اثربخشی روش بهینه سازی پیشنهادی برای تقریب DNN چند لایه کارآمد را نشان می دهیم و حداقل ضرر دقت و حداکثر صرفه جویی در انرژی را ارائه می دهد.

در دوران یادگیری ماشین، شبکه‌های عصبی عمیق (DNN) به دلیل عملکرد بی‌نظیر آنها در چندین برنامه از جمله پردازش تصویر، بینایی کامپیوتر و پردازش زبان طبیعی، مورد توجه قرار گرفته‌اند. با این حال، با افزایش پیچیدگی DNN ها، مصرف انرژی مرتبط با آنها به یک مشکل چالش برانگیز تبدیل می‌شود. چنین چالشی برای محاسبات لبه‌ای افزایش می‌یابد، جایی که دستگاه های محاسباتی هنگام کار با بودجه محدود انرژی، محدودیت منابع دارند. بنابراین، بهینه‌سازی‌های تخصصی برای یادگیری عمیق باید در دو سطح نرم‌افزاری و سخت‌افزاری انجام شود. در این مقاله، به طور جامع روند فعلی این بهینه‌سازی ها را بررسی می‌کنیم[38].

در شکل زیر روند کلی کار این مقاله شرح داده شده است:



شکل ‎1‑6 نمای بهینه‌سازی بین لایه ای برای یادگیری عمیق

در این مقاله، از دو روش تقریب نرم افزاری و سخت افزاری از هر دو مورد استفاده شده است؛ روش های تقریب نرم افزاری که در این مقاله به کار برده شده است، هرس کردن و کوانتیزاسیون هستند و روش های تقریب سخت افزاری هم در آن به کار برده شده است که ما در این جا با روش های سخت افزاری آن بیشتر کار داریم.

نکات کلی پیاده‌سازی این مقاله به قرار زیر است که در هنگام کار باید به آنها توجه کرد:

\* Low Power & Memory Budget. انجام استنتاج DNN در دستگاه های لبه، که معمولاً از نظر منابع و توان محدود هستند، یک کار چالش برانگیز است. برای مثال، ResNet-50 [23] به بیش از 95 مگابایت حافظه برای ذخیره وزن و بیش از 3.8 میلیارد ضرب برای پردازش یک تصویر نیاز دارد. چنین نتیجه ای از پردازش غیرممکن است که در دستگاه های لبه استفاده شود تا نتایج را در زمان واقعی نشان دهد.

\* Latency. در حالی که برنامه‌های کاربردی تشخیص صدای تلفن همراه مانندApple Siri ، Amazon Alexa و Google Assistant پردازش را بر اساس cloud انجام می‌دهند، اما برای سایر برنامه‌های مهم (برای مثال، وسایل نقلیه خودکار، هواپیماهای بدون سرنشین و دستگاه‌های بهداشتی پوشیدنی) پردازش سنسور برای پاسخ سریع DNN، و همچنین به دلایل حفظ حریم خصوصی و امنیت لازم است. علاوه بر این، نه تنها تأخیر، بلکه مسائل امنیتی و حریم خصوصی نیز باعث ایجاد انگیزه در پردازش نزدیک به سنسور می‌شود. بنابراین، شتاب‌دهنده های سخت‌افزاری تخصصی برای انجام موثر استنباط DNN در لبه مورد نیاز هستند تا شرایط تأخیر، امنیت و حریم خصوصی را برآورده کنند.

\* Accuracy vs. Speed and Efficiency. DNN های با دقت بالا بسیار محاسباتی و حافظه زیادی دارند. حتی اگر برخی از روندهای اخیر به طراحی DNN با حافظه کوچک اشاره می‌کنند [26] [25]، امیدوارکننده‌ترین روش فشرده‌سازی DNN با هرس پارامتر، به اشتراک‌گذاری و کمی‌سازی است. چندین شتاب‌دهنده متراکم DNN پیشنهاد شده است، اما برای تسهیل بهینه‌سازی فشرده‌سازی مانند هرس، شتاب‌دهنده‌های پراکنده DNN می‌توانند از نظر کارآیی به نتایج بهتری برسند.

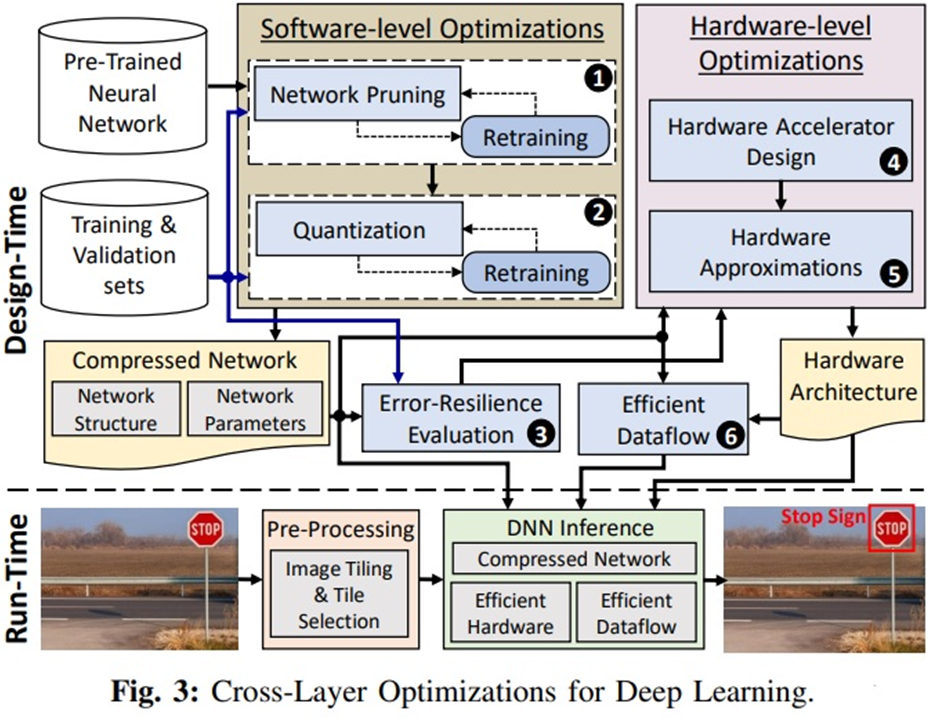
\* Redundant Operations. DNN ها معمولاً شامل چندین عملیات اضافی، مانند ضرب‌های صفر و ورودی‌های مرتبط در برنامه‌های پخش جریانی هستند. بنابراین، لزوماً نیازی به پردازش مجموعه کاملی از ورودی‌ها در هر مرحله نداریم. یک کار چالش برانگیز یافتن این افزونگی‌ها و حذف موثر آنهاست.

\* Memory. چرخه‌ها و انرژی قابل‌توجهی برای انتقال داده‌های حافظه به/از آرایه محاسباتی موردنیاز است، که این امر مستلزم معماری‌های کارآمد حافظه و استراتژی‌های سازماندهی داده‌ها برای سخت‌افزار DNN است. علاوه بر این، با توجه به روند محاسبات در حافظه ، دستگاه های ممریستور اجازه می دهند از حافظه های مقاومتی برای محاسبه آنالوگ ، با هزینه اضافی سربار ADC/DAC استفاده کنند.

\* Security. به دلیل برون سپاری آموزش و وابستگی داده ، DNN دارای چندین آسیب پذیری امنیتی است که می تواند برای انجام حملات امنیتی مورد استفاده قرار گیرد ، به عنوان مثال ، مثالهای خصمانه، درهای پشتی[[77]](#footnote-77) و مسمومیت داده ها[[78]](#footnote-78) ، برای کاهش اطمینان (ابهام در طبقه بندی)، طبقه‌بندی اشتباه و مدل تصادفی یا هدفمند سرقت این آسیب پذیری های امنیتی چالش های اساسی مانند حریم خصوصی مدل و اجرای امن DNN ها را در مورد اطمینان از استحکام سیستم های مبتنی بر DNN ایجاد می کند. به طور سنتی ، از پیش پردازش ، رمزگذاری داده ها و علامت گذاری استفاده می شود، با این حال ، همه این دفاع ها را می توان با سرقت مدل پیچیده یا حملات جعبه سیاه خنثی کرد. بنابراین ، نیاز مبرم به توسعه سیستم های دفاعی پیچیده و کارآمد برای اطمینان از حریم خصوصی مدل و اجرای امن DNN ها وجود دارد.

بهینه سازی های سطح نرم افزار: بهینه سازی های سطح نرم افزار عمدتا شامل هرس شبکه (مرحله 1 در شکل 3) و کمی سازی (مرحله 2 در شکل 3) پارامترها می باشد. هرس شبکه معمولاً به صورت تکراری انجام می شود ، جایی که در هر تکرار ، تعداد کمی از پارامترها حذف می شوند و شبکه حاصله با بازآموزی برای تعداد محدودی از دوره ها به طور دقیق تنظیم می شود.

بهینه سازی در سطح سخت افزار: یک معماری سخت افزاری کارآمد (مرحله 4 در شکل 3) می تواند شبکه فشرده را به شیوه ای بسیار کارآمد و با کارایی بالا پردازش کند. این مرحله مستلزم آگاهی کامل از جریان داده و استفاده مجدد از پارامترهای مختلف شبکه است. پس از نهایی شدن معماری سخت افزار ، انواع مختلف تقریبی را می توان برای بهبود بیشتر انرژی آن اعمال کرد (مرحله 5 در شکل 3)



شکل ‎1‑7 نمای بهینه‌سازی بین لایه ای برای یادگیری عمیق به صورت جزیی

تقریب ها را می توان هم در سطح سخت افزاری و هم در سطح نرم افزاری به کار برد. با این حال ، در اینجا عمدتا در مورد تقریب سطح سخت افزار صحبت می کنیم ، زیرا هرس و تکنیک های کمی سازی ، نمونه های کامل بهینه سازی/تقریب در سطح نرم افزار هستند، اما در اینجا قصد داریم درباره تقریب های سطح سخت افزار شامل ساده سازی های معماری و مدار صحبت کنیم. این انواع را می توان بیشتر به داده ها و تقریب های عملکردی [63] طبقه بندی کرد ، جایی که تقریب داده ها به تقریب در ذخیره سازی داده ها [61] (یعنی حافظه) و تقریب عملکردی به تقریب در عملکرد واحدهای پردازشی [17] [[ 62] [57] اشاره می کند.

در زیر، برخی از چالش های کلیدی تحقیقاتی را که می توانند تأثیر بسزایی در بهبود کارایی الگوریتم‌های یادگیری عمیق برای محاسبات پیشرفته داشته باشند ، لیست می کنیم.

Hardware Software Co-Design: یک روند متداول بهینه سازی DNN برای دستیابی به دقت بالا است ، بدون توجه زیاد به پیچیدگی سخت افزاری و مصرف انرژی دستگاه محاسباتی. از سوی دیگر ، طراحان سخت افزار باید از معماری های پسینی[[79]](#footnote-79) استفاده کنند تا از بهینه سازی های سطح نرم افزار استفاده کنند. با این حال ، بهینه سازی های سخت افزاری در سطح نرم افزار ، مانند کاوش معماری DNN [69] یا فشرده سازی [43] امیدوار کننده است و برای موفقیت نیاز به تلاش های بیشتری دارد.

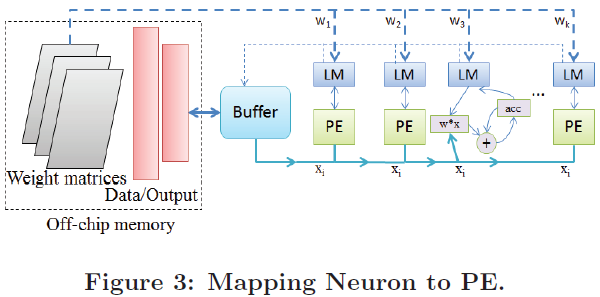
In-Memory Computing: یک الگوی امیدوار کننده برای توسعه شتاب دهنده هایی است که می توانند در مقایسه با سیستم های معمولی مبتنی بر CPU و GPU ، مقادیر بزرگی از بهره وری انرژی را ارائه دهند. با این حال ، ویژگیهای تنوع زیاد مرتبط با ReRAM و سایر حافظه‌های غیر فرار ، شتابدهنده هایی را که بر اساس آنها تنظیم شده، محدود می کند تا عملکرد دقیقی را ارائه دهد. در این راستا ، فناوری ReRAM سلول چند سطحی (MLC) باید به اندازه کافی بالغ باشد تا بتواند ضمن ارائه تراکم داده بالا ، دقت معقولی را ارائه دهد. همچنین ، مقدار قابل توجهی کار برای توسعه روش هایی مورد استفاده قرار می گیرد که بتوانند برای آموزش شبکه ها به گونه ای استفاده شوند که بتوانند دقت بالایی را حتی در هنگام کار بر روی دستگاه های محاسباتی درون حافظه ای مبتنی بر NVM ، ارائه دهند.

Hardware-Aware Hyperparameter Tuning and DNN Architectural Exploration: چندین تکنیک بهینه سازی در سطح نرم افزار پیشنهاد شده که نشان می‌دهد DNN های پراکنده ، یعنی دارای تعداد پارامترهای کمتر، می توانند تقریباً میزان دقت خروجی را با DNN های متراکم ارائه دهند. روش‌های سیستماتیک مورد نیاز است که با وجود آگاهی از معماری سخت‌افزاری و سیستم، می‌تواند شبکه را به گونه ای تنظیم کند که در عین حفظ دقت پایه، انرژی و بازدهی تقریبا مطلوبی را ارائه دهد.

Event-based Spiking Neural Networks: آنها در مقایسه با DNN های مبتنی بر دیجیتال دارای پتانسیل بسیار کارآمدتری برای صرفه‌جویی در مصرف انرژی هستند، زیرا انرژی فقط در هنگام شلیک موج مصرف می‌شود. چنین پردازش رویدادی محور امیدوار‌کننده است. بنابراین، شرکت‌هایی مانند IBM و Intel روی تراشه‌های معماری نورومورفیک و شتاب‌دهنده های آن سرمایه گذاری می‌کنند [48] [9].

- ژانگ و همکاران[1]. شبکه‌های عصبی مصنوعیANN دامنه وسیعی از برنامه‌‌ها مانند RMS[[80]](#footnote-80) را دارند. از آنجا که بسیاری از این برنامه‌ها ذاتاً متحمل خطا[[81]](#footnote-81) هستند، در این مقاله، یک فریم‌ورک محاسباتی تقریبی جدید برای ANN، یعنی ApproxANN را پیشنهاد می‌کند. هنگامی که با راه‌حل‌های موجود مقایسه می‌شود، ApproxANN تقریب را برای محاسبات و دسترسی به حافظه در نظر می‌گیرد، بنابراین صرفه جویی در انرژی بیشتری به دست می‌آورد. به طور خاص، ApproxANN تأثیر نورون‌ها بر کیفیت خروجی را به روش موثری مشخص می‌کند و با آگاهی تعیین می‌کند که چگونه محاسبات و دسترسی‌های حافظه به سلول‌های عصبی خاص چندان حساس نباشد تا حداکثر بهره‌وری انرژی را در محدودیت کیفیت مشخص بدست آورد. نتایج تجربی در برنامه‌های مختلف شبکه عصبی با مجموعه داده‌های مختلف، تأثیر راه‌حل پیشنهادی را نشان می‌دهد و با این کار با کاهش نیاز به دقت محاسباتی برای سلول‌های عصبی در شبکه‌های عصبی، می‌توانیم در عین‌حال که صرفه‌جویی قابل‌توجهی در مصرف انرژی داریم، به کاهش دقت جزئی در سطح کاربرد دست یابیم[1].

در این پیاده‌سازی، از معماری سخت‌افزاری[39] ([17]) در شکل 3 استفاده کرده، که در آن هر عنصر پردازش[[82]](#footnote-82)PE ، که به عنوان یک نورون کار می‌کند، شامل یکسری واحدهای حسابی و یک حافظه محلی[[83]](#footnote-83) LM است و فقط بردار وزن مربوطه را از حافظه خارج از تراشه به حافظه محلی خود می‌خواند. با جریان بردار-از طریق هر نورون در همان لایه، ضرب بردار ماتریسی به طور طبیعی انجام می‌شود (با هر نورون محاسبه ضرب نقطه‌ای دو بردار)



در این کار، از سه تکنیک زیر برای نشان دادن روش طراحی پیشنهادی خود استفاده کرده است:

1- Memory Access Skipping. از آنجایی که PE های جداگانه می‌توانند تصمیم بگیرند که آیا می‌توان عملیات را رد کرد یا خیر، برای ذخیره انرژی مصرفی در دسترسی به حافظه و بهبود عملکرد کلی، می‌توان به راحتی از چندین نورون (به عنوان مثال، فقط خواندن سطرهای خاص در ماتریس وزن) صرف نظر کرد.

2- Precision Scaling. ساده‌ترین روش برای مبادله دقت انرژی در ANN ، کنترل عرض‌بیت داده‌ها است. هنگامی که به‌سادگی تعداد خاصی از کمترین بیت[[84]](#footnote-84) (LSB) داده ها را کنار می‌گذارید، کیفیت محاسبات کمی کاهش می‌یابد، اما ذخیره انرژی و عملکرد را می‌توان به میزان قابل‌توجهی بهبود بخشید.

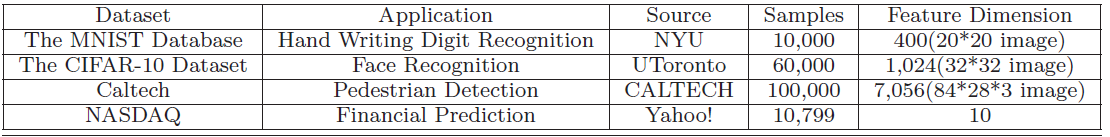
3- Approximate Arithmetic Building Blocks. روش دیگر جایگزینی واحدهای حسابی دقیق در یک نورون با روش‌های تقریبی است. این ضرب دارای خروجی قابل تنظیم[[85]](#footnote-85) (𝑛 + 𝑘) بیت است، که در آن 𝑛 عرض‌بیت داده‌های ورودی را نشان می‌دهد. با استفاده از مقادیر مختلف پارامتر accuracy ، می‌توانیم دقت محاسبه و مصرف انرژی را جابجا کنیم.

E. J. King and E. Swartzlander, “Data-dependent truncation scheme for parallel multipliers,” in Proc. Asilomar Conference on Signals, Systems & Computers, vol. 2, pp. 1178–1182, 1997.

برای ارزیابی اثربخشی و کارایی پیشنهادی، شبیه‌سازی را بر اساس کتابخانه سلول استاندارد 45𝑛𝑚 با ولتاژ تأمین 1V توسط ابزارهای تجاری Synopsys EDA انجام دادیم. این انرژی شامل هر دو توان دینامیکی و نشتی است. مقادیر توان اجزای تقریبی مختلف با ‌استفاده از Synopsys Prime Time بدست آمد و مصرف انرژی روی خواندن / نوشتن حافظه توسط CACTI بدست آمد[18].

سپس آزمایشات را در چهار شبکه نماینده با مجموعه داده‌های مختلف انجام می‌دهیم. جدول 2 اطلاعات دقیق این برنامه‌ها و مجموعه داده‌ها را توصیف می‌کند، در حالی که اولین الگوریتم یادگیری معمولی برای شبکه‌های عصبی در شناسایی رقم نوشتن دستی است و به دنبال آن دو شبکه عصبی کانولوشن در شناسایی چهره و تشخیص عابر پیاده وجود دارد، در حالی که آخرین شبکه عصبی روش مبتنی بر برنامه‌های کاربردی تجزیه و تحلیل مالی است.

جدول ‎1‑1تنظیمات مجموعه داده‌ها و پارامترها



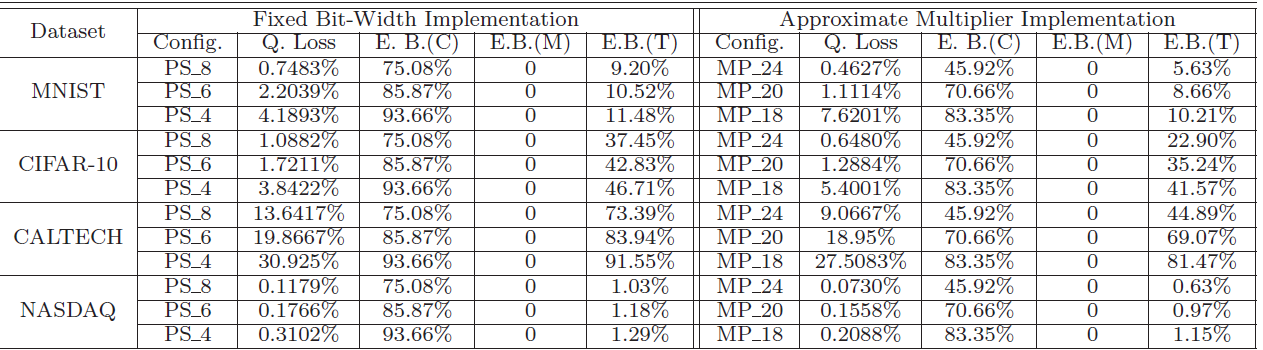
آزمایش برای مقایسه مصرف انرژی اصلی در حافظه و محاسبه و ارزیابی مزایای انرژی ApproxANN در برابر اجرای دقیق انجام شده است[[86]](#footnote-86).

جدول ‎1‑2 Experimental Statistics

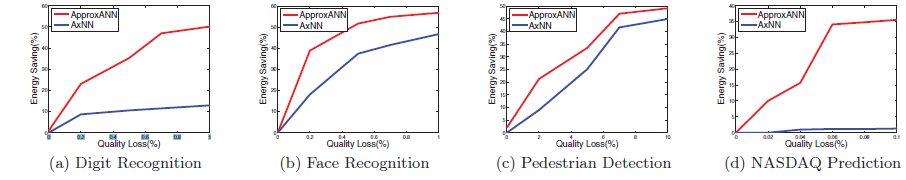
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| E.B.(M) | E.B.(C) | Q.Loss[[87]](#footnote-87) | Constraint | 𝑀/𝐶 | Application |
| 28.57% | 83.32% | 0.4891% | 0.5% | 7.160 | MNIST Hand Writing Recognition |
| 32.67% | 70.85% | 0.5000% | 0.5% | 1.005 | CIFAR-10 Face Recognition |
| 19.78% | 52.89% | 4.977% | 5% | 0.152 | Pedestrian Detection |
| 33.33% | 89.96% | 0.0499% | 0.06% | 71.630 | Nasdaq Prediction |

جدول ‎1‑3نتایج را با استفاده از روش متداول از نظر ذخیره انرژی و کاهش کیفیت نشان می‌دهد. در چنین استراتژی معمولی، همه نورونها با نسخه‌های تقریبی یکسان (یعنی همه نورونها با تقریب حالت واحد) بدون در نظر گرفتن تأثیر حافظه جایگزین می‌شوند. برای مقایسه عادلانه، از توپولوژی شبکه و مقداردهی اولیه پارامترها استفاده می‌شود و 6 نسخه تقریبی مختلف در هر یک از 4 برنامه مقایسه می‌شود. این جدول مقایسه‌های بین پیاده‌سازی ضرب تقریبی را با حالت نقطه ثابت در مجموعه‌داده‌های MNISTT,CIFAR10,CALTECH و NASDAQ نشان می‌دهد.

جدول ‎1‑3نتایج: روش تقریب حالت کانولوشن تکی



در این شکل هم نتایج شبیه‌سازی را با استفاده از رویکردی که در مقاله آورده شده است، با هر چهار شبکه نشان داده است.

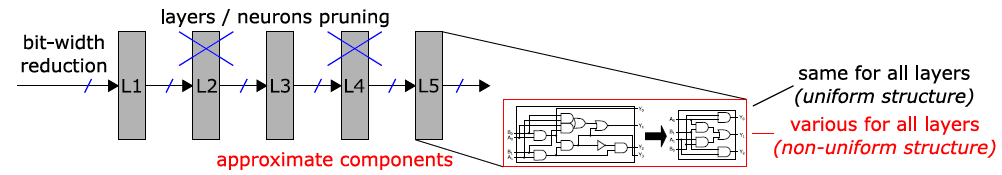


شکل ‎1‑1 مقایسه بین ApproxANN و AxNN

-روش پیشنهادی ALWANN [40]. رویکردهای پیشرفته برای کاهش مصرف انرژی سخت افزار DNN از محاسبات تقریبی استفاده می‌کنند. سپس DNN های تقریبی برای بازیابی از افت دقت ناشی از استفاده از عملیات تقریبی، نیاز به آموزش مجدد زیادی دارند. با این حال، آموزش مجدد DNN های پیچیده مقیاس خوبی ندارد. در این مقاله، نشان داده شده که تقریب موثر می‌تواند به مسیر محاسباتی شتاب‌دهنده های DNN وارد شود؛ در حالی که به طور کامل می توان از آموزش مجدد جلوگیری کرد. ALWANN پیاده‌سازی های بسیار بهینه شده DNN ها را برای شتاب‌دهنده های کم مصرف سفارشی ارائه می‌دهد که در آنها تعداد واحدهای محاسباتی کمتر از تعداد لایه‌های DNN است. ابتدا یک DNN کاملاً آموزش دیده (به عنوان مثال، درTensorFlow ) تبدیل می‌شود تا با 8-bit weights و 8-bit multipliers در لایه‌های کانولوشن کار کند. سپس برای هر عنصر محاسباتی از یک کتابخانه approximate multipliers، ضرب تقریبی مناسب انتخاب می‌شود؛ به گونه ای که اولا: یک ضرب تقریبی چند لایه را سرویس دهد و ثانیا: خطای کلی دسته بندی و مصرف انرژی به حداقل برسد. بهینه‌سازی‌ها از جمله مسئله انتخابmultiplier با الگوریتم بهینه‌سازی چند هدفه NSGA-II حل می‌شود. به منظور جلوگیری از آموزش مجدد گران محاسباتی DNN ، که معمولاً برای بهبود دقت دسته بندی استفاده می‌شود، یک طرح ساده به روزرسانی وزن ارائه می دهند که عدم صحت معرفی شده را با به کارگیری ضرب های تقریبی جبران می‌کند.

رویکرد پیشنهادی در این مقاله برای دو معماری شتاب‌دهنده DNN با ضرب تقریبی از کتابخانه منبع باز ”EvoApprox” ارزیابی می‌شود، در حالی که سه نسخه ResNet در CIFAR-10 اجرا می‌شود و موجب صرفه جویی در 30٪ انرژی مورد نیاز برای ضرب در لایه‌های کانولوشن ResNet-50 می‌شود؛ در حالی که دقت تنها 0.6% (0.9% for the ResNet-14) تخریب می‌شود.

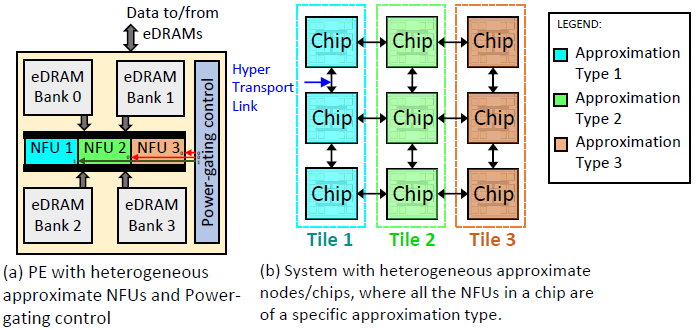
به طور کلی در این جا، به کاربر اجازه می‌دهد تا کارکرد شتاب­دهنده های سخت افزاری شبکه‌های عصبی کانولوشن را تخمین بزند و به این صورت مصرف انرژی DNN ها را بهینه کند. کاربر با همان دقت سایر الگوریتم هایی که از بازآموزی استفاده می‌کنند، به صرفه جویی بهتر در انرژی می رسد. معمولا بازآموزی منجر به شبکه­های خیلی کوچکتر یا مجموعه محدودی با اجزای تقریبی می­شود. این الگوریتم نگاشت سریع به کاربر امکان می‌دهد تا بدون هیچ گونه پردازشی روی داده‌های ورودی، شبکه را با خطاهای تقریب تطبیق دهد. این روش می‌تواند بر اساس تخمین آغازین کاربر، بهره وری انرژی سخت افزار DNN را بهبود دهد.



شکل ‎1‑2Illustration of typical approximations of computational path of DNNs.The methods targeted in this work are marked.

این شکل نمایانگر مداراتی است که در اثر این مولفه های محاسبات تقریبی دستخوش تغییرات می‌شوند.

در این مقاله، در کل هدف ایجاد چارچوب ALWANN است که یک NN آموزش دیده و مجموعه‌ای از ضرب‌های تقریبی را به عنوان ورودی در نظر گرفته و مجموعه‌ای از شبکه‌های عصبی تقریبیPareto (AxNNs) را تولید می‌کند که دقت و انرژی را مبادله می‌کند. روش پیشنهادی از الگوریتم‌های تنظیم پارامترها و الگوریتم‌های ژنتیک الهام گرفته شده است. این دو معیار اصلی طراحی برای DNN ها را بهینه می‌کند - دقت کلی DNN و انرژی مصرف شده توسط لایه‌های تقریبی، تمام محدودیت‌های ناشی از شتاب‌دهنده DNN خاص را برآورده می‌کند. برای اجتناب از فرایند بازآموزی بحرانی یا در‌دسترس، الگوریتم تنظیم سریع وزن را پیشنهاد کردیم که وزن لایه‌ها را با ضرب‌های به کار گرفته شده تطبیق می‌دهد و باعث افزایش دقت NN به میزان %4 در میانگین می شود. رویکرد پیشنهادی DNN ها را با نتایج قابل مقایسه با نتایج بدست آمده از سایر روشهای تقریب خودکار، اما برای NN های پیچیده‌تر و بدون بازآموزی، تقریبی می‌کند. علاوه بر این، روش پیشنهادی قادر به ساخت NN های کم مصرف‌تر از رویکرد مبتنی بر برداشتن لایه به دنبال آموزش از ابتدا است[12] .



شکل ‎1‑3 سناریوهای مختلف محاسبات تقریبی

شکل 4. (الف) سناریو 1: همه تراشه ها یکدست هستند، با این حال، در هر [[88]](#footnote-88)PE چندین نسخه تقریبی NFU در آن نصب شده که یکی از آنها انتخاب شده و بقیه دارای توان هستند. (ب) سناریو 2: هر PE شامل یک NFU تقریبی است و تمام NFU های یک تراشه از نوع تقریبی خاص هستند، با این حال، نوع تقریب می‌تواند در تراشه ها متفاوت باشد.

روش پیشنهادی ALWANN به کاربر اجازه می‌دهد تا شتاب‌دهنده های سخت افزاری شبکه‌های عصبی کانولوشن را تخمین زده و در نتیجه مصرف انرژی مسیر استنباط DNN ها را بهینه کند و با همان دقت سایر الگوریتم هایی که از بازآموزی استفاده می‌کنند، به ذخیره انرژی بهتر رسیدند. بازآموزی معمولاً منجر به تقریب شبکه‌های به طور قابل توجهی کوچکتر (مقیاس پذیری محدود) یا مجموعه محدودی از اجزای تقریبی در نظر گرفته شده می‌شود. الگوریتم پیشنهاد سریع نگاشت به کاربر امکان می‌دهد تا بدون هیچ گونه پردازش داده‌های ورودی، شبکه را با خطاهای تقریب تطبیق دهد، این روش می‌تواند بهره وری انرژی سخت افزار DNN در زمان واقعی بهبود دهد[40].

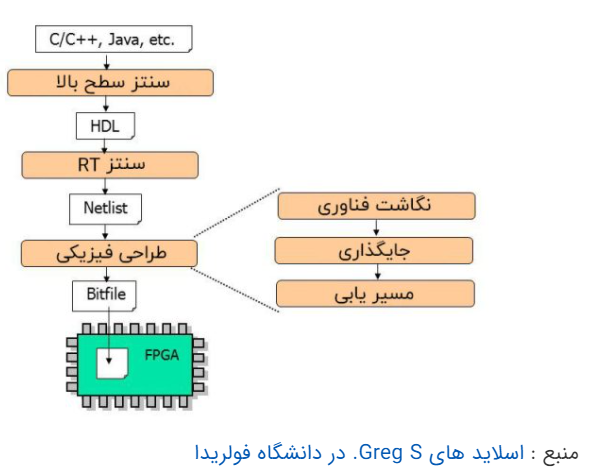
- وانگ و همکاران[41] [7] از تجزیه و تحلیل بین نسبت سیگنال به نویز کوانتیزه (SQNR) و دقت، جستجوی کاملی برای دقت بهینه در LSTM را پیشنهاد داده­اند. پیچیدگی زمانی چنین جستجوهایی برای اجرا شدن یا عملی بودن بسیار زیاد است که با استفاده از محاسبات تقریبی پیچیدگی زمانی را کاهش می دهند.

- سهرابی زاده و همکاران[42] [8] بیان کرده­اند که تکامل سریع شبکه‌های CNN مشکلات جدیدی را برای شتاب‌دهنده های FPGA ایجاده کرده است. یک مشکل اختلاف عملکرد لایه‌های مختلف CNN و مشکل دیگر هزینه های بالای یکپارچه سازی FPGA در قالب ML است. برای مقابله با این دو چالش، یک شتاب‌دهنده به نام FlexCNN ارائه داده اند که با تکه­تکه کردن دینامیک و نیز بهینه­سازی طرح داده، کارایی سخت­افزار در بین لایه‌ها را بهبود داده­اند. علاوه بر این، با پیشنهاد یک خط لوله ادغام دوسطحی، هزینه های اضافی ادغام را کاهش داده و به سرعت 5 برابری رسیده‌اند.

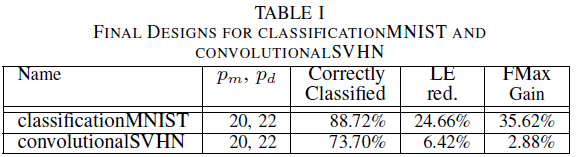
- فرولیچ و همکاران[43] [[9](#ASNet)] مشخصات اولیه شبکه‌های عصبی را با استفاده از زبان های سطح بالا مانند پایتون انجام می­دهند. سپس خروجی آن در اختیار یک ابزار سنتز قرار می گیرد که این مرحله بسیار طولانی است و عملکرد FPGA را محدود می‌کند. برای این منظور راهکاری با استفاده از حساب تقریبی ارائه شده است. در این پژوهش از کامپایلری به نام XLA Google استفاده شده که کد LLVM را به طور مستقیم از مشخصات Tensorflow تولید می‌کند. سپس می توان از این کد LLVM با یک ابزار HLS برای تولید اتوماتیک سخت افزار استفاده کرد. LeFlow ابزاری است که بر اساس حساب تقریبی ارائه شده و شبکه‌های عصبی عمیق را با استفاده از کد پایتون تولید می‌کند.

- نرونها و همکاران[44] [10] مشخصات اولیه برنامه های یادگیری ماشین را با استفاده از یک فریم ورک سطح بالا مبتنی بر پایتون و سپس ترجمه آن به RTL انجام داده­اند. مرحله ترجمه دستی زمان بر است و نیاز به یک جریان دارد که نقشه مدل های محاسباتی عددی نوشته شده در Tensorflow را برای سخت افزارهای قابل تنظیم ترسیم می­کند. این جریان از کامپایلر XLA Google استفاده می‌کند که کد LLVM را مستقیماً از مشخصات Tensorflow منتشر می‌کند. سپس می توان از این کد LLVM با یک ابزار HLSبرای تولید خودکار سخت افزار استفاده کرد و به کاربران امکان می‌دهد شبکه­های عصبی عمیق را با تعداد خطوط کمی از کد پایتون تولید کنند.

ابزارهای HLS به کدپایتون یا C به عنوان یک ورودی نیاز دارند که بهLLVM IR نمایش میانی، نگاشت دهد. از این ابزارها برای تبدیل توضیحات روش به یک پیاده‌سازی سخت‌افزاری استفاده می‌شود.



در این کار نویسنده در نهایت روشی را که ارائه داده است را دو شبکه MNIST و SVHN مورد ارزیابی قرار داده است که در آن نتایج را برای ضرب و تقسیم به دست آورده است. در این بررسی ها فرکانس، دقت و کارایی performance مورد بررسی قرار گرفته است.





به طور کلی نتایج حاصل از مطالعاتی که در این مقالات داشتم را در جدول زیر خلاصه کرده ام:

جدول 3 خلاصه مرور ادبیات پایان نامه

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **نویسندگان(سال)** | **DNN** | **محاسبات تقریبی** | **FPGA** | **HLS** | **توضیحات** |
|
| Zhang و همکاران | **√** | √ |  | √ | فریم ورک محاسباتی تقریبی جدید برای ANN که همانApproxANN است. |
| کیم و همکاران | **√** | **√** | **√** |  | فریم ورک یادگیری عمیق End-to-End برای کشف سریع QNN ارائه کرده است. |
| ALWANN | **√** | **√** |  |  | به کاربر اجازه می‌دهد تا کارکرد شتاب¬دهنده های سخت افزاری CNN را تخمین بزند |
| شاوانا و همکاران | **√** | **√** | **√** |  | مدل roofline برای تسریع شبکه‌های عصبی کانولوشن در FPGA پیشنهاد کردند |
| سهرابی زاده و هم | **√** |  | **√** |  | شتاب دهندهFlexCNN ارائه داده اند که با تکه¬تکه کردن دینامیک و نیز بهینه¬سازی طرح داده، کارایی سخت¬افزار در بین لایه‌ها را بهبود داده-اند. |
| فرولیچ و همکاران | **√** | **√** |  | **√** | از کامپایلر XLA Google استفاده می‌کند که LLVM را از Tensorflow منتشر کند. |
| نرونها و همکاران | **√** |  |  | √ | از کد LLVM با یک ابزار HLSبرای تولید خودکار سخت افزار استفاده کرد |
| وانگ و همکاران | **√** | **√** |  |  | از (SQNR) و دقت، جستجوی کاملی برای دقت بهینه در LSTM را پیشنهاد داده¬اند. |

### بیان پژوهش جاری

در این پژوهش قصد دارم تا با استفاده از روش‌های مختلف محاسبات تقریبی الگوریتم‌های شبکه عصبی و یادگیری عمیق را بهبود دهم.

### خلاصه پژوهش

# فصل چهارم: روش پیشنهادی

فصل چهارم: روش پیشنهادی

# فصل پنجم: نتیجه و ارزیابی

1. بحث و تفسیر

2. نتیجه گیری

3. خلاصه پژوهش

4. محدودیت های پژوهش

# فصل ششم: نتیجه گیری

# مراجع

[1] *Zhang, Q., et al.* [ApproxANN](#_تعریف_مسئله_و)*: An approximate computing framework for artificial neural network. in 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE). 2015.*

[2] Bjergea, K., J. Schougaardb, and D.E. Larsenb, *A* [generic](#_تعریف_مسئله_و) *and efficient convolutional neural network accelerator using HLS for a system on chip design.* arXiv e-prints, 2020: p. arXiv: 2004.13075.

[3] Mrazek, V., et al. [ALWANN](#_تعریف_مسئله_و)*: automatic layer-wise approximation of deep neural network accelerators without retraining*. in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2019. IEEE.

[4] Ma, Y., et al., [ALAMO](#_تعریف_مسئله_و)*: FPGA acceleration of deep learning algorithms with a modularized RTL compiler.* Integration, 2018. **62**: p. 14-23.

[5] Blaiech, A.G., et al., *A survey and* [Taxonomy](#_تعریف_مسئله_و) *of FPGA-based deep learning accelerators.* Journal of Systems Architecture, 2019. **98**: p. 331-345.

[6] Shawahna, A., Sait, S., M, and El-Maleh, A., 2019, FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review, IEEE/ACM International Conference on Computer-Aided Design (ICCAD), January 23, 2019, Saudi Arabia, IEEE Access, 2018, 7: 7823-7859.

[6] *Neural Networks and Learning* **[Machines](#_کاربرد_شبکه‌های_عصبی)***, Third Edition, .*

7. Fadaeddini, A., M. Eshghi, and B. Majidi. A deep residual neural network for low altitude remote sensing image classification. in 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS). 2018. IEEE.

8. Korolev, S., et al. Residual and plain convolutional neural networks for 3D brain MRI classification. in 2017 IEEE 14th international symposium on biomedical imaging (ISBI 2017). 2017. IEEE.

9. Yuan, P., et al., HS-ResNet: Hierarchical-split block on convolutional neural network. arXiv preprint arXiv:2010.07621, 2020.

10. Farooq, M. and A. Hafeez, Covid-resnet: A deep learning framework for screening of covid19 from radiographs. arXiv preprint arXiv:2003.14395, 2020.

[7] Wang, T., Wang, C., Zhou, X., and Chen, H., 2018, A Survey of FPGA Based Deep Learning Accelerators: Challenges and Opportunities, arXiv preprint arXiv:1901.04988, 2018, 1-10. ‏

[8] Sohrabizadeh, A., Wang, J., & Cong, J. (2020, February). End-to-End Optimization of Deep Learning Applications. In The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 133-139). ‏

[9] Froehlich, S., Klemmer, L., Große, D., and Drechsler, R., 2020, [ASNet](#_بهبود_شبکه‌های_عصبی): Introducing Approximate Hardware to High-Level Synthesis of Neural Networks, Symposium ISMVL, 22 may.

[10] Noronha, D. H., Salehpour, B., and Wilton, S. J., 2018, LeFlow: Enabling Flexible FPGA High-Level Synthesis of Tensorflow Deep Neural Networks, (2018, August). In FSP Workshop 2018; Fifth International Workshop on FPGAs for Software Programmers (pp. 1-8). VDE, arXiv 2018."arXiv preprint arXiv:1807.05317.

[11] Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., & Chen, M. (2014, December). Medical image classification with convolutional neural network. In 2014 13th international conference on control automation robotics & vision (ICARCV) (pp. 844-848). IEEE. ‏

[12] Lo, S. C. B., Chan, H. P., Lin, J. S., Li, H., Freedman, M. T., & Mun, S. K. (1995). Artificial convolution neural network for medical image pattern recognition. Neural networks, 8(7-8), 1201-1214. ‏

[13] Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M., & Asari, V. K. (2018). Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. arXiv preprint arXiv:1802.06955. ‏

[14] De la Parra, C., Guntoro, A., & Kumar, A. (2020). Improving approximate neural networks for perception tasks through specialized optimization. Future Generation Computer Systems, 113, 597-606. ‏

[15] Marchisio, A., Hanif, M. A., Khalid, F., Plastiras, G., Kyrkou, C., Theocharides, T., & Shafique, M. (2019, July). Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges. In 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) (pp. 553-559). IEEE. ‏

[16] Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440. ‏

[17] Rizakis, M., Venieris, S. I., Kouris, A., & Bouganis, C. S. (2018, May). Approximate FPGA-based LSTMs under computation time constraints. In International Symposium on Applied Reconfigurable Computing (pp. 3-15). Springer, Cham. ‏

[18] E. J. King and E. Swartzlander, “Data-dependent truncation scheme for parallel multipliers,” in Proc. Asilomar Conference on Signals, Systems & Computers, vol. 2, pp. 1178–1182, 1997.

1. Zhang, Q., et al. *ApproxANN: An approximate computing framework for artificial neural network*. in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2015.

2. Bjergea, K., J. Schougaardb, and D.E. Larsenb, *A generic and efficient convolutional neural network accelerator using HLS for a system on chip design.* arXiv e-prints, 2020: p. arXiv: 2004.13075.

3. Mrazek, V., et al. *ALWANN: Automatic Layer-Wise Approximation of Deep Neural Network Accelerators without Retraining*. in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2019.

4. Ma, Y., et al., *ALAMO: FPGA acceleration of deep learning algorithms with a modularized RTL compiler.* Integration, 2018. **62**: p. 14-23.

5. Blaiech, A.G., et al., *A survey and taxonomy of FPGA-based deep learning accelerators.* Journal of Systems Architecture, 2019. **98**: p. 331-345.

6. Hu, Y.H. and J.-N. Hwang, *Handbook of neural network signal processing*. 2002, Acoustical Society of America.

7. Feindt, M. and U. Kerzel, *The NeuroBayes neural network package.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2006. **559**(1): p. 190-194.

8. Korolev, S., et al. *Residual and plain convolutional neural networks for 3D brain MRI classification*. in *2017 IEEE 14th international symposium on biomedical imaging (ISBI 2017)*. 2017. IEEE.

9. Alom, M.Z., et al., *The history began from alexnet: A comprehensive survey on deep learning approaches.* arXiv preprint arXiv:1803.01164, 2018.

10. Shrestha, A. and A. Mahmood, *Review of deep learning algorithms and architectures.* IEEE Access, 2019. **7**: p. 53040-53065.

11. Sun, Y., et al., *Testing deep neural networks.* arXiv preprint arXiv:1803.04792, 2018.

12. Molchanov, P., et al., *Pruning convolutional neural networks for resource efficient inference.* arXiv preprint arXiv:1611.06440, 2016.

13. Albawi, S., T.A. Mohammed, and S. Al-Zawi. *Understanding of a convolutional neural network*. in *2017 International Conference on Engineering and Technology (ICET)*. 2017.

14. Haykin, S., *Neural Networks and Learning Machines,* . Third ed. کتاب. 2009.

15. Kang, X., B. Song, and F. Sun, *A deep similarity metric method based on incomplete data for traffic anomaly detection in IoT.* Applied Sciences, 2019. **9**(1): p. 135.

16. LeCun, Y., et al., *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.

17. Cengil, E., A. Çınar, and E. Özbay. *Image classification with caffe deep learning framework*. in *2017 International Conference on Computer Science and Engineering (UBMK)*. 2017. IEEE.

18. Yuan, Z.-W. and J. Zhang. *Feature extraction and image retrieval based on AlexNet*. in *Eighth International Conference on Digital Image Processing (ICDIP 2016)*. 2016. International Society for Optics and Photonics.

19. Abd Almisreb, A., N. Jamil, and N.M. Din. *Utilizing AlexNet deep transfer learning for ear recognition*. in *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*. 2018. IEEE.

20. Hansen, L.K. and P. Salamon, *Neural network ensembles.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990. **12**(10): p. 993-1001.

21. Zagoruyko, S. and N. Komodakis *Wide Residual Networks*. 2016. arXiv:1605.07146.

22. Fadaeddini, A., M. Eshghi, and B. Majidi. *A deep residual neural network for low altitude remote sensing image classification*. in *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. 2018. IEEE.

23. Li, X., et al. *FPGA accelerates deep residual learning for image recognition*. in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. 2017.

24. Chen, D., et al., *Deep residual learning for nonlinear regression.* Entropy, 2020. **22**(2): p. 193.

25. Chippa, V.K., et al., *Analysis and characterization of inherent application resilience for approximate computing*, in *Proceedings of the 50th Annual Design Automation Conference*. 2013, Association for Computing Machinery: Austin, Texas. p. Article 113.

26. Venkataramani, S., et al. *Approximate computing and the quest for computing efficiency*. in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2015.

27. Rodrigues, G., F. Lima Kastensmidt, and A. Bosio, *Survey on approximate computing and its intrinsic fault tolerance.* Electronics, 2020. **9**(4): p. 557.

28. Han, J. and M. Orshansky. *Approximate computing: An emerging paradigm for energy-efficient design*. in *2013 18th IEEE European Test Symposium (ETS)*. 2013.

29. Jiang, H., et al., *A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits.* J. Emerg. Technol. Comput. Syst., 2017. **13**(4): p. Article 60.

30. Han, J., *Approximate arithmetic circuits and their applications*, in *Proceedings of the Twelfth IEEE/ACM International Symposium on Networks-on-Chip*. 2018, IEEE Press: Torino, Italy. p. Article 22.

31. Ercegovac, M.D. *On approximate arithmetic*. in *2013 Asilomar Conference on Signals, Systems and Computers*. 2013.

32. Jiang, H., et al., *Approximate arithmetic circuits: Design and evaluation*, in *Approximate circuits*. 2019, Springer. p. 67-98.

33. Raha, A., H. Jayakumar, and V. Raghunathan, *Input-Based Dynamic Reconfiguration of Approximate Arithmetic Units for Video Encoding.* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016. **24**(3): p. 846-857.

34. Chen, K., et al., *Profile-Based Output Error Compensation for Approximate Arithmetic Circuits.* IEEE Transactions on Circuits and Systems I: Regular Papers, 2020. **67**(12): p. 4707-4718.

35. Kahng, A.B. and S. Kang. *Accuracy-configurable adder for approximate arithmetic designs*. in *Proceedings of the 49th Annual Design Automation Conference*. 2012.

36. Shawahna, A., S.M. Sait, and A. El-Maleh, *FPGA-based accelerators of deep learning networks for learning and classification: A review.* IEEE Access, 2019. **7**: p. 7823-7859.

37. De la Parra, C., A. Guntoro, and A. Kumar, *Improving approximate neural networks for perception tasks through specialized optimization.* Future Generation Computer Systems, 2020. **113**: p. 597-606.

38. Marchisio, A., et al. *Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges*. in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2019. IEEE.

39. ; Available from: H. Labs. <http://www.hpl.hp.com/research/cacti/>.

40. Mrazek, V., et al. *ALWANN: automatic layer-wise approximation of deep neural network accelerators without retraining*. in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2019. IEEE.

41. Wang, T., et al., *A survey of FPGA based deep learning accelerators: Challenges and opportunities.* arXiv preprint arXiv:1901.04988, 2018: p. 1-10.

42. Sohrabizadeh, A., J. Wang, and J. Cong. *End-to-end optimization of deep learning applications*. in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2020.

43. Froehlich, S., et al. *ASNet: Introducing approximate hardware to high-level synthesis of neural networks*. in *2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL)*. 2020. IEEE.

44. Noronha, D.H., B. Salehpour, and S.J. Wilton. *LeFlow: Enabling flexible FPGA high-level synthesis of tensorflow deep neural networks*. in *FSP Workshop 2018; Fifth International Workshop on FPGAs for Software Programmers*. 2018. VDE.

[42]

1. embedded system [↑](#footnote-ref-1)
2. approximate arithmetic [↑](#footnote-ref-2)
3. feature extraction [↑](#footnote-ref-3)
4. image classification [↑](#footnote-ref-4)
5. classification [↑](#footnote-ref-5)
6. معمولا تابع غیرخطی را همراه با لایه کانولوشنی یک‌جا نشان می‌دهند. [↑](#footnote-ref-6)
7. loss function [↑](#footnote-ref-7)
8. Image classification [↑](#footnote-ref-8)
9. clusters [↑](#footnote-ref-9)
10. تعداد پارامتر‌های هر لایه کانولوشنی برابر است با: طول لایه \* عرض لایه \* عمق ورودی \* تعداد فیلترها (به طور مثال در اینجا برابر است با 11\*11\*3\*96) [↑](#footnote-ref-10)
11. multiple alternating convolution [↑](#footnote-ref-11)
12. Max-pooling [↑](#footnote-ref-12)
13. مخفف Residual Network است. [↑](#footnote-ref-13)
14. residual [↑](#footnote-ref-14)
15. recognition rate and inference performance [↑](#footnote-ref-15)
16. Field Programmable Gate Array [↑](#footnote-ref-16)
17. pooling [↑](#footnote-ref-17)
18. numerical tests [↑](#footnote-ref-18)
19. lasso regression [↑](#footnote-ref-19)
20. decision tree [↑](#footnote-ref-20)
21. support vector machine [↑](#footnote-ref-21)
22. inherent resilience [↑](#footnote-ref-22)
23. Application Resilience Characterization [↑](#footnote-ref-23)
24. approximate computing - computing efficiently [↑](#footnote-ref-24)
25. distributed systems [↑](#footnote-ref-25)
26. ad hoc and application-specific [↑](#footnote-ref-26)
27. recognition [↑](#footnote-ref-27)
28. mining [↑](#footnote-ref-28)
29. data analytics [↑](#footnote-ref-29)
30. inference [↑](#footnote-ref-30)
31. vision [↑](#footnote-ref-31)
32. holistic cross-layer framework [↑](#footnote-ref-32)
33. fault tolerance [↑](#footnote-ref-33)
34. safety-critical [↑](#footnote-ref-34)
35. error masking [↑](#footnote-ref-35)
36. triple modular redundancy [↑](#footnote-ref-36)
37. ریزمعماری [↑](#footnote-ref-37)
38. Approximate arithmetic [↑](#footnote-ref-38)
39. power dissipation [↑](#footnote-ref-39)
40. Monte Carlo simulations [↑](#footnote-ref-40)
41. approximate computing [↑](#footnote-ref-41)
42. power-efficient [↑](#footnote-ref-42)
43. approximate architectures [↑](#footnote-ref-43)
44. reconfigurable approximate architecture [↑](#footnote-ref-44)
45. Peak Signal-to-Noise Ratio (PSNR) [↑](#footnote-ref-45)
46. reconfigurable adder/subtractor blocks (RABs) [↑](#footnote-ref-46)
47. padding [↑](#footnote-ref-47)
48. truncated [↑](#footnote-ref-48)
49. mean square error [↑](#footnote-ref-49)
50. signed truncated adders [↑](#footnote-ref-50)
51. accuracy-configurable approximate (ACA) adder [↑](#footnote-ref-51)
52. throughput [↑](#footnote-ref-52)
53. truncated multiplier [↑](#footnote-ref-53)
54. convolutional neural network accelerator [↑](#footnote-ref-54)
55. system on chip design [↑](#footnote-ref-55)
56. high level synthesis [↑](#footnote-ref-56)
57. board [↑](#footnote-ref-57)
58. developing an Intellectual Property Core (IP) [↑](#footnote-ref-58)
59. Image Restoration [↑](#footnote-ref-59)
60. Image Segmentation [↑](#footnote-ref-60)
61. Image De-noising [↑](#footnote-ref-61)
62. Image Reconstruction [↑](#footnote-ref-62)
63. Image Filtering [↑](#footnote-ref-63)
64. unrolling [↑](#footnote-ref-64)
65. processing elements [↑](#footnote-ref-65)
66. configurable logic blocks [↑](#footnote-ref-66)
67. synapse parallelism [↑](#footnote-ref-67)
68. neuron parallelism [↑](#footnote-ref-68)
69. feature map parallelism [↑](#footnote-ref-69)
70. register transfer level [↑](#footnote-ref-70)
71. Approximate Computational Units [↑](#footnote-ref-71)
72. validation [↑](#footnote-ref-72)
73. Low bit-width quantization of DNNs [↑](#footnote-ref-73)
74. computational logic [↑](#footnote-ref-74)
75. Filter pruning [↑](#footnote-ref-75)
76. Mean Relative Error [↑](#footnote-ref-76)
77. backdoors [↑](#footnote-ref-77)
78. data poisoning [↑](#footnote-ref-78)
79. a-posteriori [↑](#footnote-ref-79)
80. Recognition, Mining and Synthesis [↑](#footnote-ref-80)
81. inherently error-tolerant [↑](#footnote-ref-81)
82. processing element [↑](#footnote-ref-82)
83. local memory [↑](#footnote-ref-83)
84. least significant bits (LSBs) [↑](#footnote-ref-84)
85. tunable [↑](#footnote-ref-85)
86. ratio of energy consumptions between memory and computation without approximation, “Constraint” gives the predefined constraint on final output quality loss, “Q. Loss” is the real quality loss, “*𝐸.𝐵.* (*𝐶*)”, “*𝐸.𝐵.* (*𝑀*)”, and “*𝐸.𝐵.* (*𝑇*)” indicate energy savings on computation part, on memory part, and on total application, respectively. [↑](#footnote-ref-86)
87. quality loss [↑](#footnote-ref-87)
88. Processing Element [↑](#footnote-ref-88)