ITRC
ICT Research Institute
(Iran Telecommunication Research Center)

Journal of Information and Communication Technology

Iranian Society of Information and Communication Technology

Volume 16, Issue 59-60, Spring and Summer 2024, pp. 125-143

Multi-Level Ternary Quantization for Improving Sparsity and Computation in Embedded Deep Neural Networks

Hosna Manavi Mofrad[1], Seyed Ali Ansarmohammadi[2], Mostafa Ersali Salehi Nasab[3]*

[1] Electrical and Computer Engineering, University of Tehran, Tehran, Iran
[2] Electrical and Computer Engineering, University of Tehran, Tehran, Iran
[3] Electrical and Computer Engineering, University of Tehran, Tehran, Iran

Received: 03 January 2023, Revised: 15 February 2023, Accepted: 25 May 2023
Paper type: Research

Abstract

Deep neural networks (DNNs) have achieved great interest due to their success in various applications. However, the computation complexity and memory size are considered to be the main obstacles for implementing such models on embedded devices with limited memory and computational resources. Network compression techniques can overcome these challenges. Quantization and pruning methods are the most important compression techniques among them. One of the famous quantization methods in DNNs is the multi-level binary quantization, which not only exploits simple bit-wise logical operations, but also reduces the accuracy gap between binary neural networks and full precision DNNs. Since, multi-level binary can't represent the zero value, this quantization does not take advantage of sparsity. On the other hand, it has been shown that DNNs are sparse, and by pruning the parameters of the DNNs, the amount of data storage in memory is reduced while computation speedup is also achieved. In this paper, we propose a pruning and quantization-aware training method for multi-level ternary quantization that takes advantage of both multi-level quantization and data sparsity. In addition to increasing the accuracy of the network compared to the binary multi-level networks, it gives the network the ability to be sparse. To save memory size and computation complexity, we increase the sparsity in the quantized network by pruning until the accuracy loss is negligible. The results show that the potential speedup of computation for our model at the bit and word-level sparsity can be increased by 15x and 45x compared to the basic multi-level binary networks.

Keywords: Deep Neural Networks, Multi-Level Ternary Quantization, Sparse Neural Network, Pruning, Embedded Devices.

* Corresponding Author's email: mersali@ut.ac.ir

دوفصلنامه
فناوری اطلاعات و ارتباطات ایران

سال شانزدهم، شماره های ۵۹ و ۶۰، بهار و تابستان ۱۴۰۳، صفحه ۱۲۵ الی ۱۴۳

چندی‌سازی غیریکنواخت سه حالتی جهت بهبود تنکی و محاسبات شبکه‌های عصبی عمیق در کاربردهای نهفته

حسنا معنوی مفرد[۱]، سید علی انصارمحمدی [۲]، مصطفی ارسالی صالحی نسب [۳*]

[۱] دانشجو مقطع ارشد، دانشکده مهندسی برق و کامپیوتر، دانشکدگان فنی، دانشگاه تهران، تهران، ایران

[۲] دانشجو مقطع دکتری، دانشکده مهندسی برق و کامپیوتر، دانشکدگان فنی، دانشگاه تهران، تهران، ایران

[۳] استادیار، دانشکده مهندسی برق و کامپیوتر، دانشکدگان فنی، دانشگاه تهران، تهران، ایران

تاریخ دریافت: ۱۴۰۱/۱۰/۱۳ تاریخ بازبینی: ۱۴۰۱/۱۱/۲۶ تاریخ پذیرش: ۱۴۰۲/۰۳/۰۴

نوع مقاله: پژوهشی

چکیده

شبکه‌های عصبی عمیق به دلیل موفقیت در کاربردهای مختلف، به جذابیت فوق‌العاده‌ای دست یافته‌اند. اما پیچیدگی محاسبات و حجم حافظه از موانع اصلی برای پیاده‌سازی آن‌ها در بسیاری از دستگاه‌های نهفته تلقی می‌شود. از مهم‌ترین روش‌های بهینه‌سازی که در سال‌های اخیر برای برطرف نمودن این موانع ارائه شده، می‌توان به چندی‌سازی و هرس کردن اشاره کرد. یکی از روش‌های معروف چندی‌سازی، استفاده از نمایش اعداد غیریکنواخت دو حالتی است که علاوه بر بهره‌بردن از محاسبات بیتی، افت صحت شبکه‌های دو حالتی را در مقایسه با شبکه‌های دقت کامل کاهش می‌دهد. اما به دلیل نداشتن قابلیت نمایش عدد صفر در آن‌ها، مزایای تنکی داده‌ها را از دست می‌دهند. از طرفی، شبکه‌های عصبی عمیق به صورت ذاتی تنک هستند و با تنک کردن پارامترهای شبکه عصبی عمیق، حجم داده‌ها در حافظه کاهش می‌یابد و همچنین به کمک روش‌هایی می‌توان انجام محاسبات را تسریع کرد. در این مقاله می‌خواهیم هم از مزایای چندی‌سازی غیریکنواخت و هم از تنکی داده‌ها بهره ببریم. برای این منظور چندی‌سازی غیریکنواخت سه حالتی برای نمایش اعداد ارائه می‌دهیم که علاوه بر افزایش صحت شبکه نسبت به شبکه غیریکنواخت دو حالتی، قابلیت هرس کردن را به شبکه می‌دهد. سپس میزان تنکی در شبکه چندی شده را با استفاده از هرس کردن افزایش می‌دهیم. نتایج نشان می‌دهد که تسریع بالقوه شبکه ما در سطح بیت و کلمه می‌تواند به ترتیب ۱۵ و ۴۵ برابر نسبت به شبکه غیریکنواخت دو حالتی پایه افزایش یابد.

کلیدواژگان: شبکه‌های عصبی عمیق، چندی‌سازی غیریکنواخت سه حالتی، شبکه عصبی تنک، هرس کردن، دستگاه‌های نهفته.

* رایانامهٔ نویسنده مسؤول: mersali@ut.ac.ir

۱- مقدمه

در سال‌های اخیر تحقیقات درزمینه هوش مصنوعی، به‌ویژه یادگیری ماشین، به‌طور چشم‌گیری افزایش پیداکرده است.یادگیری ماشین[1] از زیرمجموعه‌های هوش مصنوعی[2] است که هدف آن هوشمند کردن رایانه‌هاست بدون اینکه مستقیماً به آن‌ها یاد بدهیم چطور رفتار کنند. یادگیری ماشین شامل الگوریتم‌های متعددی است که پرکاربردترین آن‌ها الگوریتم‌های برگرفته از مغز انسان هستند؛ یادگیری عمیق، در این دسته قرار دارند، در بیشتر روش‌های یادگیری عمیق از معماری شبکه عصبی مصنوعی[3] استفاده می‌شود، به همین دلیل مدل‌های یادگیری عمیق اغلب تحت عنوان شبکه‌های عصبی عمیق شناخته می‌شوند. اصطلاح عمیق معمولاً به تعداد لایه‌های پنهان در شبکه عصبی اشاره دارد. شبکه‌های عصبی سنتی فقط شامل ۲-۳ لایه مخفی هستند، درحالی‌که شبکه‌های عمیق می‌توانند تا ۱۵۰ لایه داشته باشند، یکی از رایج‌ترین انواع شبکه‌های عصبی عمیق، تحت عنوان شبکه‌های عصبی کانولوشنی[4] شناخته می‌شود. این شبکه‌ها توانایی‌های خارق‌العاده‌ای در کاربردهای پیچیده مانند طبقه‌بندی تصویر[5]، تشخیص اشیا[6] و قطعه‌بندی معنایی[7] تصاویر نشان داده‌اند [۱].

اگرچه طراحی‌های اخیر شبکه‌های عصبی کانولوشنی با میلیاردها پارامتر قابلیت‌هایی در سطح فرا انسانی دارند، بااین‌حال، اندازه بزرگ مدل و هزینه بالای محاسبات برای بسیاری از برنامه‌ها کاربردی به‌ویژه با هدف پیاده‌سازی بر روی دستگاه‌های نهفته که محدودیت حافظه، منابع محاسباتی و توان دارند، همچنان مانع بزرگی محسوب می‌شود [۲]. به‌طور مثال شبکه vgg16 با مجموعه داده ImageNet به ۵۵۲ مگابایت پارامتر و ۳۰,۸ گیگافلاپس محاسبات برای هر تصویر نیاز دارد. این میزان از حافظه و محاسبات برای گسترش دادن شبکه‌های عصبی عمیق روی دستگاه‌های با محدودیت منابع دشوار است [۳]. در حال حاضر، توجه به سمت گسترش دادن شبکه‌های عصبی عمیق روی دستگاه‌های نهفته[8] جلب شده است [۴].

با توجه به محدودیت‌های دستگاه‌های نهفته، لازم است که شبکه‌های عصبی از نظر حافظه‌ی مصرفی و حجم محاسبات فشرده شوند تا بتوانند روی این دستگاه‌ها گسترش یابند [۵]. سطوح بهینه‌سازی شبکه‌های عصبی عمیق جهت کاهش حافظه و حجم محاسبات شامل موارد زیر است: ۱) طراحی شبکه‌های بهینه مانند MobileNets [۶]، SqueezeNet [۷]، ۲) استفاده از روش‌های فشرده‌سازی شامل چندی‌سازی[9] [۸-۱۰] و هرس کردن[10] [۱۱-۱۳]، ۳) بهینه‌سازی در سطح الگوریتم برای سرعت بخشیدن به محاسبات مانند ضرب عمومی ماتریس[11] یا روش winogard [۱۴] و ۴) استفاده از سخت‌افزار اختصاصی برای تسریع جریان داده‌ها و انجام محاسبات به‌صورت موازی [۱۵].

چندی‌سازی و هرس کردن دو روش مؤثر در فشرده‌سازی شبکه‌های عصبی عمیق است. درروش چندی‌سازی سعی بر این است که پارامترها را (وزن‌ها[12] و فعال‌سازها[13]) که در حالت عادی به‌صورت ممیز شناور ۳۲ بیتی هستند، با یک سیستم نمایش اعداد با تعداد بیت کمتر نشان دهند تا منجر به هزینه محاسباتی و مصرف حافظه کمتر شود. به عنوان اولین تلاش‌ها در این زمینه، سیستم اعداد ممیز ثابت چندی شده [۱۶-۱۸] مطرح شد که سعی می‌کرد شبکه را با تعداد بیت کمتر طراحی کند و برای کاهش صحت از دست رفته، چندی‌سازی همراه با یادگیری شبکه را ارایه داده اند. چالشی که در این سیستم اعداد حتی بعد از چندی‌سازی وجود دارد این است که هزینه محاسبات و میزان حافظه مصرفی هنوز قابل‌توجه است.

برای برطرف کردن این مسئله سعی شد که دقت اعداد با شدت بیشتری کاهش پیدا کند به این صورت که وزن‌ها و فعال‌سازها با حتی یک بیت نمایش داده شوند. به این مدل چندی‌سازی، دو حالتی[14] [۹, ۱۹] می‌گویند که شبکه‌های عصبی دو حالتی را ایجاد می‌کنند. در چندی‌سازی دو حالتی پارامترهای شبکه با یک مقدار ۱ یا ۱- تخمین زده می‌شوند که با یک بیت قابل نمایش است. درنتیجه حافظه لازم برای ذخیره‌سازی پارامترهای مدل و مقادیر میانی به‌شدت کاهش می‌یابد. از طرفی در شبکه‌های عصبی دو حالتی، یک عملیات کانولوشن را می‌توان با روش Xnor_PopCount که شامل عملیات بیتی Xnor و عملیات ساده شمارش بیت است، انجام داد. به دلیل استفاده از عملیات ساده بیتی، در شبکه‌های عصبی دو حالتی پیچیدگی محاسبات نیز کاهش خواهد یافت. اما

[1] Machine learning
[2] Artificial Intelligence
[3] Artificial neural network
[4] Convolutional neural network
[5] Image classification
[6] Object detection
[7] Semantic segmentation
[8] Embedded device
[9] Quantization
[10] Pruning
[11] General matrix multiplication (GEMM)
[12] Weights
[13] Activations
[14] Binary

چالشی که در این شبکه‌های دو حالتی وجود دارد افت صحت[1] است [۲۰]. همین امر باعث ظهور روش دیگری از چندی‌سازی به نام سه‌حالتی[2] [۱۰, ۲۱, ۲۲] شد که در این حالت هر پارامتر را با (۱و۰و۱-) نشان می‌دهیم. در مقایسه با نمایش دو حالتی که برای نمایش هر پارامتر یک بیت کافی بود، در این نمایش برای نمایش هر یک از مقادیر (۱و۰و۱-) به ۲ بیت نیاز داریم. دلیل اصلی پیدایش این نوع چندی‌سازی پر کردن فاصله صحت بین مدل دو حالتی و ۳۲ بیت ممیز شناور بود. هرچند صحت شبکه‌های عصبی عمیق سه حالتی از دو حالتی بهتر شد ولی با وجود دو برابر شدن حجم پارامترهای شبکه هنوز با صحت شبکه‌های ۳۲ بیتی ممیز شناور فاصله‌ی قابل توجه‌ای داشتند.

مقاله‌های زیادی جهت جبران صحت از دست رفته‌ی شبکه‌های عصبی عمیق دو حالتی چاپ شده‌است، از جمله: روش‌های مبتنی بر افزایش ابعاد عرضی شبکه (افزایش تعداد فیلترهای هر لایه) [۲۳]. هرچند این روش‌ها نتایج قابل ملاحظه‌ای در جهت بهبود صحت شبکه داشته‌اند ولی در این روش‌ها کماکان حجم بالای پارامترها آزاردهنده هست و همچنین نیاز است تا شبکه با فراپارامترهای جدید طراحی و آموزش داده شود. برای برطرف کردن فاصله صحت بین شبکه دو حالتی و شبکه اصلی (۳۲بیتی ممیز شناور) روش غیریکنواخت دو حالتی [۲, ۲۴] ارائه گردید که در این حالت به‌جای نمایش یک عدد ۳۲ بیتی ممیز شناور، از چند رقم دو حالتی (دو حالتی ۲، ۳ و ... رقمی) برای نمایش استفاده می‌شود. با توجه به غیریکنواخت بودن سطوح چندی‌ساز در این روش و همچنین استفاده از چند رقم دو حالتی، می‌توان از این روش به عنوان چندی‌ساز غیریکنواخت چندرقمی دو حالتی نام برد ولی به اختصار در این مقاله از چندی‌ساز غیریکنواخت دو حالتی استفاده شده است. مقاله‌های [۲, ۲۴] نشان داده‌اند که با دو حالتی کردن وزن‌ها و فعال‌سازها به ۳ یا ۴ رقم، صحت شبکه قابل قبول و نزدیک به صحت شبکه در حالت دقت کامل است. لذا با این نوع چندی‌سازی هم از خواص عملیات بیتی در محاسبات دو حالتی استفاده شده‌است و هم افت صحت شبکه قابل چشم پوشی هست.

روش موثر دیگر در فشرده‌سازی شبکه‌های عصبی کانولوشنی، روش هرس کردن پارامترهای صفر شبکه است. درواقع با هرس کردن، پارامترهای نزدیک صفر که تأثیر خیلی کمی در صحت شبکه دارند، حذف می‌شوند. در این حالت تعداد پارامترهای شبکه موردنظر کمتر می‌شود، بنابراین میزان حافظه مورد نیاز کاهش پیدا می‌کند. از طرفی به این دلیل که ضرب صفر در هر عدد، صفر می‌شود بنابراین می‌توان از این نوع محاسبات در شبکه کانولوشنی جلوگیری کرد که این روش باعث کم شدن پیچیدگی محاسبات نیز خواهد شد [۲۵-۲۸]. برای فشرده‌سازی حداکثری شبکه از آموزش شبکه همگام با چندی‌سازی و هرس کردن بهره برده اند.

یکی از مزایای تنکی در وزن‌ها، شانس استفاده از کدگذاری جهت فشرده‌سازی وزن‌ها در حافظه‌ی خارجی تراشه هست که هر چه تنکی بیشتر باشد تاثیر عملکرد فشرده‌سازی بیشتر می‌شود و از هزینه‌های دسترسی بالای انتقال داده از حافظه‌های خارجی به داخل تراشه می‌کاهد. الگوریتم‌های مختلفی برای فشرده‌سازی وزن‌های غیرصفر وجود دارد که چهار مورد از مهمترین آن‌ها به نام فشرده‌سازی سطری[3] [۲۹]، فشرده‌سازی ستونی[4] [۳۰, ۳۱]، فشرده‌سازی با توجه به ابعاد ورودی[5] [۳۲-۳۴] و کدبندی طول اجرا[6][۳۵] هستند.

در این مقاله قصد داریم روش چندی‌سازیی ارایه دهیم که از مزایای تنکی بهره ببرد و هم‌زمان بتوانیم از تنکی ذاتی شبکه‌های عصبی جهت فشرده‌سازی حافظه و جلوگیری از محاسبات بی‌تاثیر استفاده کنیم و تا حد امکان از کاهش صحت شبکه جلوگیری کنیم. لذا در مرحله‌ی اول، با استفاده از چندی‌سازی غیریکنواخت سه حالتی امکان بهبود صحت شبکه و نمایش مقادیر صفر را ارایه می‌کنیم و سپس با استفاده از اعمال تنکی در پارامترهای وزن‌های شبکه همگام با چندی‌سازی و عملیات آموزش شبکه، حجم پارامترها و حجم محاسبات در اجرای لایه‌های شبکه را کاهش می‌دهیم. در پایان با ارائه یک معماری پایه جهت مقایسه عملیات چندی‌سازی غیریکنواخت دو حالتی با سه حالتی و با امکان حذف عملیات غیرموثر ضرب، حداکثر تسریع بالقوه محاسبات با استفاده از سیستم نمایش اعداد غیریکنواخت سه حالتی پیشنهادی را در سطح کلمه و بیت نسبت به دو حالتی نشان خواهیم داد.

نوآوری‌های این مقاله به شرح زیر است:

- استفاده از چندی‌سازی غیریکنواخت سه حالتی برای رفع مشکل تنکی در شبکه‌های عصبی عمیق دو حالتی
- آموزش شبکه غیریکنواخت سه حالتی با آگاهی همزمان از چندی‌سازی و هرس
- افزایش تنکی وزن‌ها در سطح کلمه به کمک هرس کردن با

[1] Accuracy
[2] Ternary
[3] Compressed Sparse Row (CSR)
[4] Compressed Sparse Column (CSC)
[5] Compressed Image Siz (CIS)
[6] Run Length Coding (RLC)

حفظ صحت در شبکه غیریکنواخت سه حالتی

- جبران سربار حافظه ناشی از روش چندی‌سازی غیریکنواخت سه حالتی به کمک فشرده‌سازی وزن‌های تنک‌شده
- ارائه معماری مناسب برای حذف عملیات غیر موثر ضرب در مقدار صفر جهت کاهش حجم محاسبات شبکه‌های غیریکنواخت سه حالتی

ساختار این مقاله به شرح زیر سازماندهی‌شده است: در بخش دو به انگیزه‌ای که باعث شد این تحقیق انجام شود خواهیم پرداخت و در بخش سوم به بیان کارهای که در پژوهش‌های پیشین در این حوزه انجام شده می‌پردازیم. ایده مقاله در بخش چهارم که شامل نحوه‌ی چندی‌سازی شبکه و تنک کردن شبکه است ارایه می‌شود. همچنین نحوه‌ی فشرده‌سازی حافظه و کاهش محاسبات با استفاده از واحدهای محاسباتی پایه معرفی می‌شود. در نهایت نتایج را در بخش پنجم بررسی و تحلیل می‌کنیم و در انتها نیز نتیجه‌گیری و مراجع به ترتیب در بخش‌های ششم و هفتم قرارگرفته است.

۲- انگیزه پژوهش

یکی از روش‌های رایج برای کاهش محاسبات شبکه‌های عصبی عمیق در راستای قابل پیاده‌سازی شدن آن‌ها برای یک دستگاه نهفته، استفاده از تنکی پارامترهای شبکه است. به این معنی که مقادیر خیلی کوچک را به صفر تقریب می‌زنند و به این ترتیب سعی می‌کنند تعداد پارامترهای با مقدار صفر را افزایش دهند. انگیزه‌ی اصلی از ایجاد تنکی در شبکه‌ها، درواقع امکان حذف عملیات ضرب در صفر در محاسبات شبکه است. این مقادیر صفر می‌توانند برای وزن‌ها یا ورودی‌ها و یا خروجی‌های یک نرون باشند که معمولاً به آن تنکی در سطح کلمه می‌گویند. علاوه بر سطح کلمه، در سطح بیت، حذف عملیات بی اثر برای بیت‌های صفر هم می‌تواند سبب کاهش عملیات محاسباتی شود که به آن تنکی در سطح بیت گفته می‌شود. مقاله‌های [۳۶, ۳۷] برای تنکی در سطح بیت ۵ الی ۹ برابر بهبود در سرعت و کارایی انرژی را گزارش کرده‌اند. درواقع با توجه به این شواهد، امید داریم که با تنک کردن شبکه به محدودیت‌های ناشی از پیاده‌سازی شبکه‌های عصبی کانولوشنی روی دستگاه‌های نهفته غلبه کنیم.

در کارهای پیشین که در زمینه تنکی شبکه‌های عصبی انجام‌شده، نتایج به‌دست‌آمده حاکی از آن است که شبکه‌های عصبی چه در حالت ممیز شناور چه در حالتی که به‌صورت ممیز ثابت چندی شده اند، حاوی تعداد زیادی پارامتر صفر و یا نزدیک به صفر در سطح کلمه و بیت (هم در وزن‌ها و هم در فعال‌سازها) هستند [۲۹, ۳۶, ۳۷]. درواقع شبکه‌های عصبی کانولوشنی به‌صورت ذاتی شبکه‌هایی تنک هستند. در چندی‌سازی ممیز ثابت با کمتر از ۸ بیت که به‌عنوان یکی از مرسوم‌ترین و محبوب‌ترین چندی‌سازی‌ها مطرح‌شده، مقاله‌های [۳۶-۳۸] نشان داده‌اند که با از دست دادن مقداری قابل‌چشم‌پوشی از صحت، وزن‌های شبکه‌های عصبی عمیق می‌توانند، تا ۷۰ درصد در سطح کلمه و تا ۹۲ درصد در سطح بیت صفر داشته باشند. این خود انگیزه‌ای شده است تا از این تنکی شبکه ها جهت کاهش حجم پارامترها در حافظه‌های خارجی تراشه و همچنین کاهش محاسبات استفاده کنیم.

از طرف دیگر روش‌های چندی‌سازی دو حالتی مطرح شدند تا پارامترها را فقط با دو مقدار ۱- و یا ۱ با تنها یک بیت نمایش دهند و باعث کاهش هرچه بیشتر حجم داده‌ها در حافظه و ساده کردن عملیات کانولوشنی با استفاده از عملیات بیتی و شمارش شوند. ولی به دلیل اینکه فاصله بین صحت شبکه‌های دو حالتی نسبت به شبکه‌های ممیز ثابت زیاد است شبکه‌های غیریکنواخت دو حالتی مطرح شدند. این شبکه‌ها علاوه بر در نظر گرفتن سطوح چندی‌سازی غیریکنواخت، با استفاده از افزایش سطوح چندی‌سازی باعث افزایش صحت شبکه‌ی چندی شده‌اند. به‌عنوان‌مثال برای مجموعه داده‌ی ImageNet مقاله‌ی [۲] با استفاده از ۸ یا ۱۶ سطح چندی غیریکنواخت به صحت نزدیک به شبکه با دقت کامل رسیده است که برای نمایش این ۸ یا ۱۶ سطح چندی به ۳ یا ۴ بیت در سخت‌افزار نیاز هست.

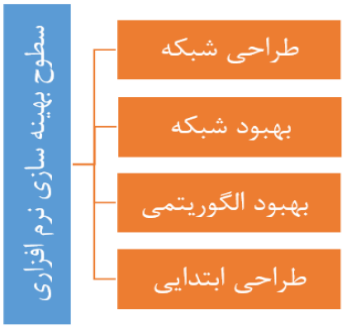
همان‌طور که پیش‌تر اشاره شد شبکه‌های کانولوشنی به‌طور ذاتی قابلیت تنکی زیادی در پارامترهای وزن دارند. در مقابل شبکه‌های چندی دو حالتی غیریکنواخت در کنار بهره بردن از محاسبات سبک بیتی و همچنین جبران صحت ازدست‌رفته و نزدیک شدن به صحت شبکه با دقت کامل، شبکه‌های محبوب و مناسبی برای دستگاه‌های نهفته می‌توانند باشند. اما در شبکه‌های چندی دو حالتی غیریکنواخت چون برای نمایش هر رقم تنها از (۱و۱-) استفاده می شود، عدد صفر قابل‌نمایش نیست. بنابراین در شبکه‌های غیریکنواخت دو حالتی نمی‌توان از مزایای تنکی در سطح کلمه برای کاهش محاسبات و حافظه بهره برد.

ایده این مقاله از همین نکته به ذهن ما رسید، که روش چندی‌سازیی ارائه کنیم که علاوه بر بهره بردن از محاسن محاسبات دو حالتی جهت کاهش حافظه و انرژی، امکان نشان دادن اعداد صفر را داشته باشد تا بتوانیم از مزایای تنکی شبکه‌های عصبی استفاده کنیم و تا جایی که صحت شبکه آسیب نبیند بتوانیم شبکه را هرس کنیم. بنابراین روش استفاده از سیستم اعداد غیریکنواخت سه حالتی

جهت چندی‌سازی شبکه را ارائه خواهیم داد که در بخش ایده مقاله به تفضیل درباره آن صحبت خواهیم کرد.

۳- کارهای پیشین

با توجه به محدودیت‌های دستگاه‌های نهفته، یعنی قدرت محاسباتی محدود، ظرفیت کم‌حافظه و محدودیت مصرف توان، چندین هدف برای بهینه‌سازی وجود دارد. برای مثال، ممکن است کسی تصمیم بگیرد که صحت شبکه عصبی را قربانی کند تا مصرف حافظه، یا میزان پیچیدگی محاسبات را کاهش دهد. بسته به هدف بهینه‌سازی، شبکه‌های عصبی طیف وسیعی از فرصت‌های بهینه‌سازی را ارائه می‌دهند. همان‌طور که در شکل ۱ نشان داده‌شده است [۵]، این فرصت‌ها را به چند دسته کلی تقسیم می‌کنیم:



شکل ۱. سطوح بهینه‌سازی شبکه‌های عصبی

سطح طراحی شبکه: بهینه‌سازی در سطح طراحی شبکه مجموعه‌ای از روش‌هایی است که ساختار شبکه را قبل یا در حین آموزش برای بهبود تأخیر یا هزینه استنتاج شبکه تنظیم می‌کند. نمونه‌هایی از این موارد MobileNet-V1/V2 [۳۹]، SqueezeNet [۷] و ShuffleNet [۴۰] هستند که به صورت دستی طراحی شده‌اند. مجموعه‌ای از کارهای جدیدتر، جستجوی معماری عصبی[1] را به‌عنوان روشی برای کاهش دخالت انسان برای طراحی چنین معماری‌های پیچیده و زمان‌بری معرفی کردند. نمونه‌هایی از این موارد MNASnet [۴۱]، FbNet [۴۲] و Lemonade [۴۳] هستند که از NAS آگاه از سخت‌افزار از طریق یادگیری تقویتی، الگوریتم های تکاملی یا روش‌های مبتنی بر گرادیان برای کشف ساختارهای شبکه عصبی با دقت خوب و عملکرد بالا استفاده کردند.

سطح بهبود شبکه: روش‌های بهبود در سطح شبکه‌های عصبی بر اصلاح نحوه نمایش شبکه با بهره بردن از انواع داده‌های با دقت پایین‌تر (چندی‌سازی) و یا استفاده از تنکی وزن‌ها و فعال‌سازها (هرس‌کردن) مبتنی هستند. در بهبود شبکه، اغلب برای اینکه به کاهش صحت ناشی از اعمال این روش‌ها غلبه کنند نیاز هست که شبکه را دوباره آموزش دهند یا از هم‌زمانی آموزش شبکه با توجه به چندی‌سازی یا تنکی شبکه استفاده شود.

چندی‌سازی پارامترهای شبکه یک روش برای فشرده‌سازی حافظه شبکه‌های عصبی عمیق، کاهش هزینه سخت‌افزار و هزینه محاسبات با استفاده از عملیات منطقی با پهنای بیت کم است. وقتی‌که وزن‌ها و فعال‌سازها به‌درستی کوانتیزه شوند، می‌توان عملیات کانولوشن را به‌طور مؤثر توسط عملگرهای بیتی محاسبه کرد [۲, ۴۴]. این روش با رویکردهایی مانند شبکه‌های عصبی دو حالتی [۲]، شبکه‌های سه حالتی [۱۰] معرفی‌شد که بین صحت شبکه‌های عصبی عمیق چندی شده و نمونه‌های کاملاً دقیق آن‌ها فاصله وجود داشت، خصوصاً وقتی شبکه با پهنای بیت بسیار کم مانند یک بیت یا دو بیت چندی شود. برای رفع این شکاف Lqnet [۲] و RebNet [۲۴] چندی‌سازی غیریکنواخت دو حالتی برای وزن‌ها و فعال‌سازها پیشنهاد دادند که صحت شبکه آن‌ها با صحت شبکه ممیز شناور تقریباً برابر است.

علاوه بر روش‌های چندی‌سازی، بهره بردن از تنکی بیت‌های وزن و فعال‌ساز و جلوگیری از عملیات بی‌اثر، یک روش کارآمد برای کاهش هزینه محاسبات و سربار حافظه است. اساساً، در شبکه‌های عصبی عمیق دو سطح تنکی وجود دارد: سطح کلمه و سطح بیت. اکثر شتاب‌دهنده‌های مبتنی بر تنکی که اخیراً پیشنهادشده‌اند، تنکی سطح کلمه را هدف قرار داده‌اند که مبتنی به از بین بردن ضریب‌های صفر عملوند و ذخیره وزن‌ها به‌صورت فشرده هستند. چندین معماری برای بهره بردن از تنک بودن بیت‌ها در داخل هر کلمه پیشنهادشده است، ازجمله صرفه‌جویی در انرژی لازم برای محاسبات، برای مقادیر صفر فعال‌سازها [۲۹, ۳۰]، یا وزن‌ها [۳۲] یا هر دو [۳۱, ۳۸]. لاکونیک [۳۷] تنکی سطح بیت را هدف قرار داده و یک شتاب‌دهنده سخت‌افزاری برای بهره بردن از تنکی در وزن و فعال‌ساز پیشنهاد داده است. برای این منظور، Laconic هر ورودی را به فهرستی از توان‌های دو علامت‌دار یا عباراتی با رمزگذاری بوث تبدیل می‌کند و سپس عبارات را به‌صورت سریال ضرب و جمع می‌کند. هرچند هر عملیات ضرب وزن در فعال‌ساز به سیکل‌های متعددی نیاز دارد، اما عملکرد کلی این روش بهتر است. درحالی‌که معماری‌های قبلی عمدتاً بر بهره بردن از تنکی در پارامترهای با دقت بالا (ممیز شناور) یا پهنای بیت بالا متمرکز بودند، این کار به دنبال یک شتاب‌دهنده است که به‌طور مؤثر هم ازنظر تنکی در سطح کلمه و هم در سطح بیت، هر دو پارامتر، وزن

[1] Neural architecture search (NAS)

و فعال‌ساز را کنترل کند که منجر به عملیات منطقی ساده با پهنای بیت پایین شده است.

سطح بهبود الگوریتم: در این بخش روی بهینه‌سازی عملیات هر لایه تمرکز می‌شود به‌عنوان‌مثال روش‌های مختلفی برای انجام کانولوشن وجود دارد یکی از آن‌ها روش مستقیم[1] است در این روش کانولوشن را با ۶ حلقه تودرتو پیاده‌سازی می‌کنند [۴۵]. روش winogard [۴۶] یک روش دیگر برای بهبود الگوریتمی به‌منظور کاهش هزینه های محاسباتی عملیات کانولوشن در شبکه‌های عصبی کانولوشنی است. این الگوریتم با تبدیل عملیات سنگین و پیچیده مانند عملیات ضرب به چندین عملیات ساده مانند جمع باعث کاهش پیچیدگی کلی عملیات کانولوشن می‌شود. پیاده‌سازی این الگوریتم‌ها برای سیستم‌های نهفته با توان کم مناسب است، زیرا منابع و بودجه توان بسیار محدود است. در مقابل، آن‌ها هزینه بیشتری در مصرف حافظه و دقت دارند [۱۴].

سطح طراحی اولیه: در این قسمت به پایین‌ترین سطح بهبود نرم‌افزاری یک شبکه عصبی خواهیم پرداخت. بهینه‌سازی در این سطح می‌تواند از تغییر چیدمان داده در حافظه [۴۷] ، تغییر نحوه استفاده از دستورالعمل‌های برداری برای پردازش لایه‌ها [۴۸] تا نوشتن پیاده‌سازی اسمبلی هسته‌ها برای رسیدن به عملکرد بهتر پردازنده‌های خاص متفاوت باشد.

با هدف بهره بردن از مزایای تنکی و چندی‌کردن در شبکه‌های عصبی عمیق، در این مقاله می‌خواهیم برای اولین بار یک روش چندی‌سازی غیریکنواخت سه حالتی همراه با تنکی را ارائه دهیم. با استفاده از این روش علاوه بر حفظ صحت شبکه از محاسبات سبک بیتی به همراه حذف عملیات ضرب بی تاثیر حداکثر بهره را جهت فشرده‌سازی شبکه می‌بریم. انتظار داریم کارایی این شبکه نسبت به حالت‌هایی که تنها از روش چندی‌کردن و یا فقط روش تنکی استفاده می‌کنند بهتر باشد. درواقع انتظار می‌رود مصرف حافظه، پیچیدگی عملیات و انرژی و توان مصرفی نسبت به حالتی که تنها از یکی از روش‌ها استفاده می‌کند بیشتر کاهش یابد و شبکه ایجادشده به محدودیت‌های موجود در دستگاه‌های نهفته غلبه کند.

۴- چندی‌سازی غیریکنواخت سه حالتی

در این بخش از مقاله در ابتدا مقدمه‌ای از ساختار لایه‌های کانولوشنی دو حالتی و مبانی عملکرد چندی‌سازی جهت کاهش تعداد بیت موردنیاز برای پارامترهای شبکه بیان می‌شود. در ادامه نمایش اعداد دو حالتی و غیریکنواخت دو حالتی و نحوه‌ی محاسبات ضرب در آن بررسی می‌شود. سپس در بخش بعدی سیستم نمایش اعداد غیریکنواخت سه حالتی پیشنهادی ارائه می‌شود و در مرحله بعد عملیات چندی‌سازی و تنکی درحین یادگیری شبکه انجام می شود تا با استفاده از حداقل بیت و حداکثر تنکی در پارامترهای وزن، حداقل افت را در صحت شبکه داشته باشیم. در گام بعدی جهت کاهش حجم موردنیاز برای ذخیره‌سازی پارامترهای تنک شده‌ی شبکه در حافظه‌ی خارجی تراشه، الگوریتم‌های فشرده‌سازی مورد بررسی قرار می‌گیرد. در پایان، معماری واحدهای محاسباتی پایه‌ای برای نمایش اعداد سه حالتی و دو حالتی ارائه‌شده است تا بتوانیم نتایج حاصل از روش‌های چندی‌سازی به همراه تنکی در سیستم نمایش اعداد دو حالتی و سه حالتی غیریکنواخت مختلف را در سطح نتایج سخت‌افزاری موردبررسی قرار دهیم.

۴-۱- ادبیات و مفاهیم پایه‌ای

۴-۱-۱- شبکه‌های عصبی عمیق دو حالتی

همان‌طور که در بخش مقدمه نیز بیان شد، یکی از شبکه‌های عصبی عمیق معروف، شبکه‌های عصبی کانولوشنی است، که از آن‌ها به‌طور گسترده در تشخیص اشیاء، کلاس‌بندی تصاویر و ... استفاده می‌شود. این شبکه‌ها از چندین لایه شامل کانولوشن، تجمیع[2] و تمام‌اتصال[3] تشکیل‌شده‌اند. در این شبکه‌ها به‌منظور استخراج ویژگی، از لایه‌های کانولوشن و تجمیع استفاده می‌شود و لایه تمام‌اتصال در لایه آخر شبکه برای کلاس‌بندی خروجی‌ها طراحی شده‌است.

درواقع در این مدل شبکه‌های عمیق، بیشترین بار محاسباتی در لایه‌های کانولوشنی و تمام‌اتصال است، علت آن این است که این شبکه‌ها حاوی میلیون‌ها وزن هستند و ضرب این میزان وزن در فعال‌ساز نظیر آن‌ها منجر به هزینه بالا در حافظه و محاسبات پیچیده و سنگین می‌شود. این هزینه‌های بالای حافظه و محاسبات مانع بزرگی برای پیاده‌سازی شبکه‌های عصبی کانولوشنی بر روی دستگاه‌های نهفته با محدودیت منابع تلقی می‌شود.

برای غلبه به موانع موجود برای پیاده‌سازی CNNها بر روی دستگاه‌های نهفته، روش‌هایی برای فشرده‌سازی این شبکه‌ها وجود دارد که یکی از این روش‌ها، چندی‌سازی است. درواقع هدف اصلی چندی‌سازی این است که نوع نمایش وزن‌ها و فعال‌سازها را به‌جای ۳۲ بیت ممیز شناور، با چند بیت محدود نشان دهد. مقاله‌های

[1] Direct
[2] Pooling
[3] Fully connected

زیادی از نمایش اعداد ممیز ثابت، با عرض ۱۶، ۸ و حتی ۴ بیت جهت چندی‌سازی شبکه‌های عصبی عمیق استفاده کرده‌اند و حداکثر حدود یک درصد افت در صحت شبکه داشته‌اند. از طرفی شبکه‌های عصبی عمیق دو حالتی جهت کاهش حداکثری حجم پارامترها و حجم محاسبات معرفی‌شده‌اند. در چندی‌سازی شبکه های عصبی عمیق دو حالتی هر پارامتر از وزن‌ها و فعال‌سازها تنها با یک بیت نمایش داده می‌شود. یکی از ساده‌ترین روش‌های چندی‌سازی دو حالتی بر روی وزن‌ها، با استفاده از تابع Sign است (معادله ۱)، که در آن w و w^b نشان‌دهنده وزن‌های با دقت کامل و وزن‌های دو حالتی می‌باشند.

$$w^b = Sign(w) = \begin{cases} +1 & w \geq 0, \\ -1 & w \leq 0. \end{cases} \quad (۱)$$

البته برای جبران قسمتی از افت صحت شبکه ناشی از چندی‌سازی دو حالتی، معمولا در نمایش اعداد دو حالتی از یک ضریب مقیاس استفاده می‌شود که این ضریب مقیاس می‌تواند برای گروهی از پارامترها مشترک باشد [۲, ۲۴]. به‌عنوان‌مثال برای وزن‌های هر فیلتر در یک لایه‌ی کانولوشنی، از یک ضریب مقیاس مستقل و یکسان استفاده می‌شود. لذا مقدار عددی پارامترهای چندی شده w^q به‌صورت معادله ۲ نمایش داده می‌شود.

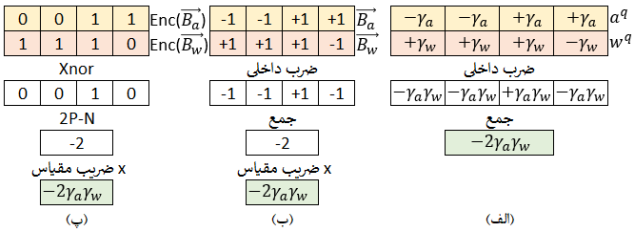
$$w^q = \gamma * Sign(w) = \begin{cases} +\gamma & w \geq 0, \\ -\gamma & w \leq 0. \end{cases} \quad (۲)$$

به دلیل آن که تعداد وزن‌ها در شبکه‌های کانولوشنی بسیار زیاد است، با چندی‌کردن وزن‌ها حجم داده‌های ذخیره‌شده در حافظه به‌شدت کاهش پیدا می‌کند. درواقع اگر از n بیت برای چندی‌سازی استفاده کنیم، میزان فشرده‌سازی شبکه در مقایسه با شبکه ممیز شناور ۳۲ بیتی، $\frac{32}{n}$ خواهد شد. به‌طور مثال زمانی که چندی‌سازی به کمک معادله ۲ باشد، یعنی هر عدد ۳۲ بیتی ممیز شناور به یک عدد یک بیتی دو حالتی چندی شود، در این صورت مقدار n، برابر با یک خواهد بود و حجم وزن‌های شبکه چندی شده ۳۲ برابر نسبت به شبکه اصلی فشرده شده‌است. چندی‌سازی علاوه بر اینکه حجم حافظه را کم می‌کند عملیات محاسباتی را نیز ساده‌تر خواهد کرد. یعنی به‌جای آن که وزن‌ها و فعال‌سازها را با عملیات ممیز شناور در هم ضرب کنیم، در صورتی‌که چندی‌سازی از نوع ممیز ثابت باشد، این عملیات با پیچیدگی به‌مراتب کمتر انجام خواهد شد. به‌علاوه اگر عملیات چندی‌سازی دو حالتی بر روی وزن‌ها و فعال‌سازها پیاده‌سازی شود، به‌جای عملیات سنگین ضرب، از عملیات بیتی که شامل عملیات Xnor و PopCount هست استفاده می‌شود که با این روش نیز می‌توان سنگینی و پیچیدگی محاسبات لایه کانولوشن را به‌طور چشمگیری کاهش داد.

عملیات Xnor_PopCount در معادله ۳ نشان داده شده است، در این معادله $\vec{w}$ و $\vec{a}$ به ترتیب نشان‌دهنده بردار ممیز شناور وزن و فعال‌ساز ، $\overrightarrow{B_w}$ و $\overrightarrow{B_a}$ بیانگر بردار علامت وزن و فعال‌ساز، γ_w و γ_a نیز ضریب مقیاس برای وزن‌ها و فعال‌سازها و بردارهای $Enc(\overrightarrow{B_w})$ و $Enc(\overrightarrow{B_a})$ نیز بردارهای کدگذاری شده از روی بردار علامت هستند. درواقع همان‌طور که در معادله ۳ نشان داده‌شده است، برای ضرب وزن‌ها و فعال‌سازها کافی است ضریب مقیاس وزن‌ها و فعال‌سازها از آن بردارها جدا شود و آن‌ها به‌تنهایی ضرب شوند و سپس بردارهای علامت وزن‌ها و فعال‌سازها کدگذاری شود ($-1 \rightarrow 0, +1 \rightarrow 1$) و درنهایت بین بردارهای کدگذاری شده، عملیات بیتی ساده انجام پذیرد.

$$dot(\vec{w}, \vec{a}) = \gamma_w \gamma_a dot(\overrightarrow{B_w}, \overrightarrow{B_a}) = \gamma_w \gamma_a \, XnorPopcount(Enc(\overrightarrow{B_w}), Enc(\overrightarrow{B_a})) \quad (۳)$$

عملیات Xnor_PopCount که در معادله ۳ به آن اشاره شد، در شکل ۲ با ذکر مثال توضیح داده شده‌است. در این مثال قرار است یک بردار وزن در بردار فعالساز نظیرش که هر دو شامل چهار المان تک‌بیتی هستند ضرب شوند. در شکل ۲ (الف) بردار وزن‌ها و فعالسازها، بردارهای چندی شده دو حالتی به همراه ضریب مقیاس هستند که برای انجام عملیات کانولوشن بر روی آنها نیاز است که هرکدام از وزن‌ها در فعالساز نظیرش ضرب شود و درنهایت با یکدیگر جمع شوند. اما به دلیل اینکه ضریب مقیاس وزن‌ها و فعال‌سازها یکسان هستند، روش ساده‌تر، جدا کردن ضریب مقیاس وزن‌ها و فعالسازها همانند شکل ۲ (ب) است، که درنهایت خروجیِ جمع در ضرایب مقیاس ضرب می‌شوند. در مدل (ب) به دلیل اینکه نیازی نیست در هر بار ضرب وزن در فعال‌ساز نظیرش، ضرایب مقیاس ۳۲ بیتی نیز ضرب شوند و بعد از ضرب مقادیر دو حالتی، در خروجی حاصله ضرایب مقیاس را اعمال می‌کنیم، پس بنابراین عملیات ضرب مدل (ب) نسبت به مدل (الف) بهتر است، اما هنوز عملیات ضرب بین وزن‌ها و فعالسازها پابرجاست. روشی که پیچیدگی محاسبات ضرب در شبکه‌های دو حالتی را کاهش می‌دهند روش Xnor_PopCount است (شکل ۲ (پ))، به این صورت که بردارهای علامت که حاوی ضرایب مقیاس نیستند را کد می‌کند و بیتهای وزن و فعال‌ساز نظیرش را Xnor کرده و سپس عملیات PopCount را به کمک فرمول 2P-N روی خروجی Xnor اعمال خواهیم کرد. به اینصورت که دو برابر تعداد یکها (p) در بردار خروجی را منهای N یا اندازه بردار خروجی (تعداد بیت‌های بردار خروجی) می‌کنیم و در نهایت ضرایت مقیاس وزن‌ها و فعال‌سازها را در خروجی ضرب خواهیم کرد.



شکل ۲. نمایش عملیات ضرب کانولوشنی بین دو بردار وزن و فعال‌ساز با استفاده از: (الف) بردارهای چندی شده‌ی وزن‌ها و فعال‌سازها. (ب) بردارهای علامت دو حالتی وزن‌ها و فعال‌سازها. (پ) بردارهای کدگذاری شده وزن‌ها و فعال‌سازها و عملیات Xnor_PopCount

۴-۱-۲- چندی‌سازی با سیستم نمایش اعداد غیریکنواخت دو حالتی

هر چند شبکه‌های عصبی عمیق دو حالتی سنتی از نظر حافظه و حجم محاسبات به حداقل میزان ممکن کاهش پیدا کرده بودند ولی اختلاف صحت شبکه در مقایسه با شبکه‌ی دقت کامل باعث شده بود تا در بسیاری از کاربردها که میزان صحت شبکه از اهمیت بالایی برخوردار هست نتوانند مورداستفاده قرار بگیرد. تحقیق‌های زیادی انجام شد و جهت بهبود این مشکل راه‌های متفاوتی ارائه شده‌است. استفاده از وزن‌های سه حالتی (سه‌گانه) به‌جای دو حالتی امکان نمایش وزن‌های صفر را برای یادگیری شبکه عصبی عمیق فراهم می‌کند که خود سبب بهبود صحت شبکه در مقایسه با شبکه‌های عصبی عمیق دو حالتی می‌شود. جهت کدگذاری هر پارامتر سه حالتی که درواقع سه مقدار متفاوت را شامل می‌شود، در سخت‌افزار نیاز به استفاده از دو بیت خواهد بود، لذا علاوه بر سربار محاسباتی بیشتر نسبت به مقادیر دو حالتی، نیاز به حافظه‌ی دو برابری نسبت به وزن‌های دو حالتی خواهد داشت. در تحقیقات دیگری، جهت جبران صحت ازدست‌رفته‌ی شبکه‌های عصبی عمیق دو حالتی، استفاده از افزایش ابعاد عرضی شبکه (افزایش تعداد فیلترهای هر لایه) پیشنهاد شده است [۲۳]. هرچند این روش‌ها نتایج قابل‌ملاحظه‌ای در جهت بهبود صحت شبکه داشته‌اند، ولی در این روش‌ها نیاز است تا شبکه با فرا پارامترهای جدید طراحی و یادگیری شود و علاوه بر آن هنوز قابلیت نمایش پارامترهای صفر را ندارد. لذا سربارها و محدودیت‌های مخصوص به خود را دارد. نهایتاً برای اینکه دقت شبکه‌های دو حالتی افزایش یابد و همچنان از مزایای محاسبات سبک بیتی دو حالتی بهره برده شود، روش چندی‌سازی غیریکنواخت دو حالتی در مقاله‌های [۲, ۲۴, ۴۹] بیان‌شده است که در ادامه جزئیات بیشتری در رابطه با این سیستم نمایش اعداد ذکرشده است.

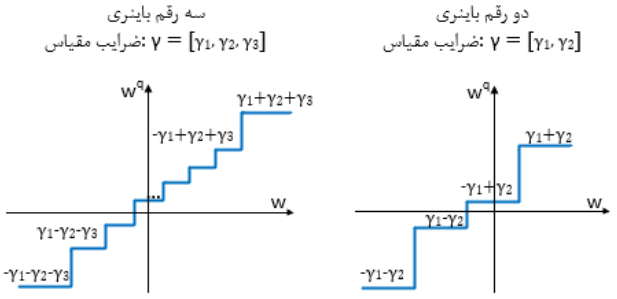
سیستم نمایش اعداد غیریکنواخت دو حالتی: ابتدا نحوه‌ی نمایش برای اعداد صحیح بدون علامت نشان داده می‌شود. فرض می‌شود مقدار اسکالر q با عرض k رقم وجود دارد این دو مقدار را با استفاده از دو بردار ضریب مقیاس(γ) و دو حالتی (b) همانند معادله ۴ می‌توان نمایش داد.

$$\gamma = \begin{bmatrix} 1 \\ 2 \\ . \\ . \\ 2^{k-1} \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_k \end{bmatrix}, b_i \in \{0,1\} \quad (۴)$$
$$==> q = \gamma^T b$$

مقادیر قابل‌نمایش در این حالت به‌صورت یکنواخت و با فاصله‌ی یکسان از یکدیگر قرار می‌گیرند. در نمایش اعداد غیریکنواخت دو حالتی بردار ضریب مقیاس لزوماً مقادیر توان دو نیست و می‌تواند هر مقداری داشته باشد. لذا برای مقدار اسکالر q که به‌صورت k رقم دو حالتی هستند نحوه‌ی نمایش اعداد به‌صورت معادله ۵ می‌شود.

$$\gamma = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ . \\ . \\ \gamma_k \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_k \end{bmatrix}, b_i \in \{-1,1\}, \quad (۵)$$
$$\gamma i \in Z \rightarrow q = \gamma T b = \gamma_1 b_1 + \gamma_2 b_2 + \cdots + \gamma_k b_k$$

این سبب می‌شود تا سطوح مقادیر قابل‌نمایش با k رقم به‌صورت یکنواخت نباشد. شکل ۳ مقادیر قابل‌نمایش در این سیستم اعداد به ازای ۲ و ۳ رقم را نشان می‌دهد.



شکل ۳. سطوح چندی‌سازی غیریکنواخت دوحالتی به ازای ۲ و ۳ رقم

در مقاله‌های [۲, ۲۴] نشان داده شده که مقادیر وزن‌ها و فعال‌سازها یک توزیع غیریکنواخت دارند، لذا برای نمایش دادن این پارامترها از سیستم نمایش اعدادی باید استفاده کرد که سطوح نمایش آن به‌صورت غیریکنواخت باشد، تا بتوانیم حداکثر بهره‌وری را از سیستم نمایش اعداد داشته باشیم. لذا سیستم نمایش اعداد غیریکنواخت دو حالتی می‌تواند گزینه‌ی مناسب‌تری برای نمایش پارامترهای یک شبکه عصبی عمیق با توزیع غیریکنواخت باشد [۵۰].

حال بدون از دست دادن کلیت مسئله عملیات ضرب داخلی دو بردار w^q و a^q را در نظر می‌گیریم که به روش مذکور چندی‌شده و شامل N پارامتر k رقمی دو حالتی است. ضرب داخلی بردار به‌عنوان سنگین‌ترین محاسبات لایه‌های کانولوشنی و تمام‌اتصال مورد بررسی قرار می‌گیرد. در معادلات بیان‌شده، w نشان‌دهنده وزن‌ها و a نیز بیانگر فعال‌سازها هستند. ماتریس $B^w = [b_1^w, b_2^w, \dots, b_N^w]$ و $B^a = [b_1^a, b_2^a, \dots, b_N^a]$ را به صورتی که $B^w\{-1,1\}^{KxN}$ و $B^a \in \{0,1\}^{KxN}$ در نظر بگیرید (توجه شود که مقادیر ماتریس وزن‌ها (۱و۱-) و فعال‌سازها (۱و۰) هستند)، مقدار ماتریس γ برای تمام مقادیر وزن‌ها و فعال‌سازها یکسان و به ترتیب با $\gamma^w \in R^k$ و $\gamma^a \in R^k$ در نظر گرفته می‌شود. حال مقدار حاصل‌ضرب داخلی برای یک مجموعه با N زوج عدد به‌صورت معادله ۶ و ۷ به دست می‌آید.

$$DotProduct(w^q, a^q) = \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K}\left(b_{ni}^w * \gamma_i^w * b_{nj}^a * \gamma_j^a\right) \quad (۶)$$

$$DotProduct(w^q, a^q) = \sum_{i=1}^{K}\sum_{j=1}^{K}(\gamma_i^w * \gamma_j^a \sum_{n=1}^{N} b_{ni}^w * b_{nj}^a) \quad (۷)$$

با فاکتورگیری از معادله ۶ با توجه به یکسان بودن مقادیر γ_(i ,j) در N زوج عدد مختلف، میتوان به معادله ۷ رسید که $\sum_{n=1}^{N} b_{ni}^w * b_{nj}^a$ قسمت عملیات بیتی دو حالتی را نشان میدهد که میتواند مطابق قسمت قبل با استفاده از عملیات Xnor_PopCount انجام شود. در ادامه روشهای مختلف چندی‌سازی به سیستم نمایش اعداد غیریکنواخت دو حالتی و نحوهی به دست آوردن مقادیر ماتریس B و γ که هم‌زمان با عملیات یادگیری شبکه به دست میآید توضیح داده‌شده است.

چندی‌سازی با استفاده از سیستم نمایش اعداد غیریکنواخت دو حالتی: معادله ۸ مسئله بهینه‌سازی به حداقل رساندن خطای چندی‌سازی جهت به دست آوردن مقدار چندی Q(x) به ازای ورودی x را نشان می‌دهد.

$$Q^*(x) = arg_Q \min \int (Q(x) - x)^2 dx \quad (۸)$$

الگوریتم‌های متفاوتی جهت به دست آوردن مقدار چندی Q(x) در سیستم نمایش اعداد غیریکنواخت دو حالتی معرفی‌شده است. مقاله [۲۴] از یک الگوریتم پیشنهادی residual مطابق الگوریتم۱ که بر مبنای عملیات باقی‌مانده گیری عمل می‌کند استفاده کرده‌است. در این الگوریتم مقادیر ضریب مقیاس جزء پارامترهای باقابلیت آموزش در حین یادگیری شبکه تعریف‌شده‌اند و بردار K رقمی دو حالتی متناظر با هر ورودی مطابق این الگوریتم به دست می‌آید.

الگوریتم۱ – نحوه‌ی چندی‌سازی غیریکنواخت دو حالتی به روش Residual

Inputs: $\gamma_1, \gamma_2, \dots, \gamma_M, x$
Outputs: $b_1, b_2, \dots, b_K$

1: $r \leftarrow x$
2: **for** i = 1 ... K **do**
3: $b_i \leftarrow Binarize(Sign(r))$
4: $r \leftarrow r - Sign(r) * \gamma_i$
5: **end for**

مقاله‌ی [۲] با استفاده از روش چندی‌سازی جدیدی با استفاده از نمایش اعداد غیریکنواخت دو حالتی صحت‌های قابل‌مقایسه باحالت دقت کامل به دست آورده است. [۲] برای به دست آوردن مقادیر بهینه‌ی ضریب مقیاس‌ها و مقادیر بردار دو حالتی متناظر معادله ۸ را به‌صورت معادله ۹ جایگزین کرده است.

$$\gamma^*, B^* = arg\ Min\|B^T\gamma - x\|_2^2 \quad (۹)$$

در این فرمول $x = [x_1, \dots, x_N] \in R^N$ مقادیر ورودی‌های (وزن‌ها یا فعال‌سازها) ممیز شناور، K نشان‌دهنده تعداد ارقام چندی شده است، همان‌طور که گفته شد هدف پیدا کردن بردار $\gamma \in R^K$ بهینه به همراه $B = [b_1, \dots, b_N] \in \{-1, 1\}^{K*N}$ است. درواقع هدف به دست آوردن بردار ضریب مقیاس و بردار دو حالتی است که مقدار خطای چندی‌سازی را به حداقل برساند. برای بهینه‌سازی کوانتایزر دو روش موردبررسی قرار گرفته‌است. درروش اول از روش سنتی پس‌انتشار[1] استفاده‌شده است به این صورت که مقادیر γ ها همراه با پارامترهای شبکه آموزش می‌بینند و مقادیر B به صورتی انتخاب می‌شود که مقدار چندی شده‌ی Q(x) حداقل فاصله را از مقدار واقعی x داشته باشد. در روش دوم از الگوریتم حداقل خطای کوانتایزر یا QEM استفاده کرده است. برای حل معادله ۹ در حلقه اصلی آموزش شبکه، ابتدا γ را ثابت در نظر گرفته و B را به صورتی انتخاب می‌کنیم که Q(x) به سطح چندی‌ای نگاشت شود که نزدیک‌ترین فاصله را از خود x داشته باشد. سپس برای به دست آوردن γ بهینه از فرمول بسته‌ی معادله ۱۰ استفاده می‌شود. نتایج صحت شبکه برای این دو روش در مقاله [۲] نشان داده است که روش QEM بهتر عمل کرده و شبکه می‌تواند به صحتِ نزدیک به

[1] Back propagation

شبکه با دقت کامل برسد.

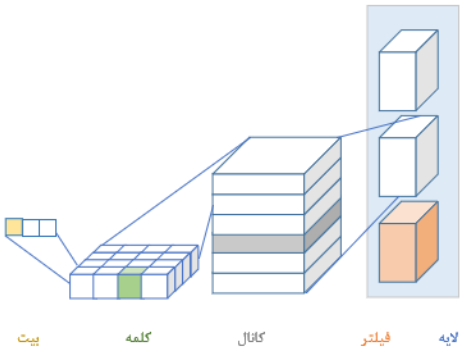
$$\gamma^* = (BB^T)^{-1}Bx \quad (۱۰)$$

این نوع چندی‌سازی غیریکنواخت دو حالتی برای وزن‌ها به‌صورت آگاه به کانال و برای فعال‌سازها به‌صورت آگاه به لایه انجام می‌گیرد. مبنای الگوریتم کوانتایزر استفاده شده در این مقاله، مطابق روش QEM در [۲] هست.

۴-۱-۳- انواع دانه‌بندی پراکندگی و نحوه‌ی تنک کردن شبکه‌های عصبی عمیق

انواع دانه‌بندی پراکندگی:

از روش‌های دیگری که باعث بهبود فشرده‌سازی شبکه‌های عصبی می‌شوند، روش هرس کردن است. هرس کردن شبکه‌های عصبی ابعاد گسترده‌ای دارد که در شکل ۴ به‌طور کامل دانه‌بندی تنکی در شبکه‌های عصبی عمیق نشان داده‌شده است. در این شکل از سمت راست به چپ، در ابتدا تنکی در سطح لایه است که در هر لایه یک سری فیلتر قرار دارد. تنکی می‌تواند در سطح فیلتر اعمال شود و داخل هر فیلتر، می‌توان تنکی در سطح کانال داشته باشیم و داخل هر کانال نیز در سطح کلمه می‌توان تنکی اعمال کرد و برای هر کلمه نیز در سطح بیت می‌توان عملیات هرس کردن را اعمال کرد.



شکل ۴. انواع دانه‌بندی پراکندگی در شبکه‌های عصبی عمیق

نحوه‌ی اعمال تنکی به شبکه: نحوه اعمال تنکی به شبکه به دو صورت افزایشی و به یکباره انجام می‌گیرد. روش افزایشی به اینصورت است که درصد تنکی مورد نظر که می‌خواهیم به شبکه اعمال کنیم را طی چند مرحله با درصدهای کمتر به شبکه تحمیل کنیم تا به درصد مورد نظر دست یابیم. به طور مثال، اگر بخواهیم به شبکه ۷۰ درصد تنکی اعمال کنیم، در روش افزایشی در ابتدا ۱۰ درصد وزن‌ها را تنک کرده سپس از میان وزن‌های باقی‌مانده ۱۰ درصد بعدی را تنک کرده، این کار را تا زمانی که به ۷۰ درصد تنکی در وزن‌ها برسیم ادامه خواهیم داد. اما روش به یکباره به اینصورت است که در ابتدا و به یکباره ۷۰ درصد وزن‌ها را هرس کنیم. نتایج صحت در این دو حالت با یکدیگر تفاوتی ندارد [۵۱].

به علاوه زمان اعمال تنکی به شبکه نیز در دو حالت همزمان با یادگیری و بعد از یادگیری می‌باشد، که در روش همزمان با یادگیری میزان افت صحت نسبت به زمان بعد از یادگیری کمتر است، لذا در این مقاله روش هرس کردن ما در هنگام یادگیری و به یکباره می‌باشد.

انواع فشرده‌سازی حافظه با استفاده از پراکندگی وزن‌ها: پس از تنک کردن شبکه، به دلیل اینکه مقادیر غیرصفر وزن‌ها از ۲۰ تا ۸۰ درصد و برای فعال‌سازها از ۵۰ تا ۷۰ درصد در سطح کلمه کاهش می‌یابد، می‌توان از مزایای تنکی ایجاد شده در وزن‌ها و فعال‌سازها بهره برد و علاوه بر کاهش محاسبات، میزان حافظه لازم برای ذخیره‌سازی داده‌ها را در حافظه‌ی ثانویه کاهش داد که این خود سبب کاهش انرژی دسترسی به حافظه‌ی ثانویه می‌شود.

۴-۲- ارائه مدل چندی‌سازی آگاه از توزیع تراکم صفر

در این بخش از مقاله ایده چندی‌سازی آگاه از تراکم صفر با استفاده از نمایش اعداد غیریکنواخت را موردبررسی قرار می‌دهیم.

۴-۲-۱- چندی‌سازی غیریکنواخت سه حالتی

در این بخش به توضیح نحوه نمایش اعداد استفاده‌شده در این مقاله، که اعداد غیریکنواخت سه حالتی هستند می‌پردازیم. همان‌طور که در بخش‌های گذشته بیان شد برای پر کردن فضای خالی بین صحت شبکه‌های دو حالتی و شبکه‌های ممیز شناور، شبکه‌های غیریکنواخت دو حالتی معرفی شدند که تا حدودی توانستند مسئله مذکور را برطرف نمایند. اما مشکلی که در شبکه‌های غیریکنواخت دو حالتی وجود دارد این است که این شبکه‌ها قدرت نمایش دادن عدد صفر را ندارند درصورتی‌که در بسیاری از مقاله‌ها نشان داده‌شده است که شبکه‌های عصبی عمیق از پراکندگی بالای وزن‌های صفر بهره می‌برند. لذا در شبکه‌های چندی شده با استفاده از نمایش اعداد غیریکنواخت دو حالتی نمی‌توانیم از مزایای تنکی استفاده کنیم. این چالش انگیزه‌ای شد تا چندی‌سازی غیریکنواخت سه حالتی را معرفی کنیم، که علاوه بر بهبود صحت شبکه و استفاده از محاسبات سبک بیتی، علاوه بر ۱ و ۱- می‌توانیم عدد صفر را نیز در نمایش هر سطح دخیل کنیم تا کلمه‌های کامل صفر قابل نمایش باشند. در ادامه به بررسی روند چندی‌سازی مطرح‌شده در این مقاله خواهیم پرداخت.

اعداد غیریکنواخت سه‌حالتی علاوه بر اینکه تعداد سطوح بیشتری

را می‌توانند مهیا کنند، قدرت نمایش عدد با مقدار صفر را هم دارند. جدول ۱ تمام اعداد قابل‌نمایش با دو رقم سه‌حالتی را نمایش می‌دهد (3^2 یا ۹ حالت). در نمایش اعداد غیریکنواخت سه‌حالتی، برای هر رقم سه‌حالتی نیاز به دو بیت برای نمایش در سخت‌افزار داریم.

جدول ۱. نمایش سطوح چندی‌سازی دورقمی سه حالتی

	۱	۲	۳	۴	۵	۶	۷	۸	۹
B_w	{0, 0}	{0, +1}	{0, -1}	{+1, 0}	{+1, +1}	{+1, -1}	{-1, 0}	{-1, +1}	{-1, -1}
w^q	0	$+\gamma_2$	$-\gamma_2$	$-\gamma_1$	$+\gamma_1 +\gamma_2$	$+\gamma_1 -\gamma_2$	$-\gamma_1$	$-\gamma_1 +\gamma_2$	$-\gamma_1 -\gamma_2$

به‌منظور ارائه چندی‌سازی بهینه و بهبود صحت شبکه‌های چندی شده دو حالتی، می‌بایست کوانتایزر به همراه شبکه آموزش داده شود. یکی از موثرترین روش‌ها برای یادگیری کوانتایزرها این است که سطوح چندی‌سازی را در طول یادگیری بهینه کنیم. یعنی اینکه سطوح آموزش ببینند، مشکلی که در این روش وجود دارد این است که تابع چندی‌سازی با عملیات بیتی سازگار نیست. برای حل این چالش، باید تابع چندی‌سازی را محدود به فضایی کنیم که با عملیات بیتی‌سازگار باشند. به همین دلیل در این مقاله با الهام گیری از یک نوع چندی‌سازی غیرهمسان که اعداد را به‌صورت ترکیب ضریب مقیاس و دنباله‌ای از (۱و۰) نشان می‌دهند (معادله ۴)، ما نیز می‌خواهیم اعداد سه حالتی را به‌صورت ترکیبی از بردار ضریب مقیاس که نشان‌دهنده ارزش‌های هر بیت و بردار سه حالتی که حاوی مقادیر (۱و۰و۱-) است نشان دهیم (معادله ۵). درواقع برای اینکه هم کوانتایزرها آموزش ببینند و هم از عملیات بیتی استفاده کنیم، می‌توانیم به‌راحتی بردار γ که حاوی K مقدار ممیز شناور است را در طول برنامه مطابق الگوریتم QEM که در قسمت ۱،۲،۴ ارایه شد، آموزش بدهیم. بنابراین معادله آموزش چندی‌سازی ما به‌صورت معادله ۹ است.

در این مقاله ما چندی‌سازی آگاه از لایه برای فعال‌سازها (یک کوانتایزر به ازای لایه) و چندی‌سازی آگاه از کانال برای وزن‌ها (یک کوانتایزر برای هر فیلتر کانولوشنی) اعمال کردیم و در ضمن میزان ضرایب مقیاس برای اعمال چندی‌سازی در مقابل حجم زیاد وزن‌ها قابل‌چشم‌پوشی بود.

برای اعمال چندی‌سازی ابتدا فعال‌سازها را با دقت ممیز شناور در نظر گرفتیم و چندی‌سازی را فقط روی وزن‌ها پیاده‌سازی کردیم، مشاهده شد که با تعداد محدودی سطح برای مقادیر وزن‌ها به صحت با دقت کامل رسیدیم. اما این کار مطلوب ما نبود زیرا هدف ما کاهش هزینه محاسبات و کم کردن میزان حافظه مصرفی بود که لازمه‌ی آن چندی کردن هر دو پارامتر وزن‌ها و فعال‌سازها هست. پس علاوه بر چندی کردن وزن‌ها، فعال‌سازها را نیز یک‌بار به‌صورت غیریکنواخت دو حالتی و بار دیگر به‌صورت غیریکنواخت سه حالتی چندی کردیم و بعد از چندی کردن به دو روش غیریکنواخت دو حالتی (با استفاده از بیت‌های ۰و۱) و غیریکنواخت سه حالتی متوجه شدیم به دلیل اینکه فعال‌سازها از تابع Relu عبور می‌کنند و درنهایت به صفر و یک نگاشت می‌شوند بهتر است فعال‌سازها به‌صورت دو حالتی با ۰و۱ چندی شود. پس وزن‌ها را به کمک غیریکنواخت سه حالتی و فعال‌سازها را به‌صورت غیریکنواخت دو حالتی چندی کردیم.

۴-۲-۲- چندی‌سازی به همراه هرس کردن

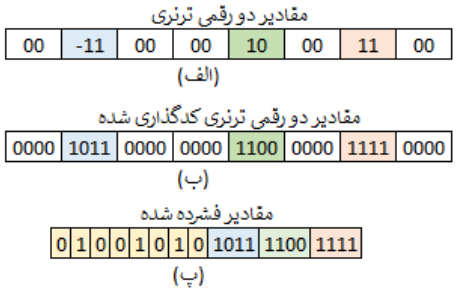
مزایای مطرح‌شده درباره تنکی، انگیزه‌ای شد، تا شبکه غیریکنواخت سه حالتی ارائه شده را هرس کنیم. بنابراین در شبکه برای هر لایه، تمامی وزن‌ها در سطح کلمه را که نزدیک به صفر بودند، با درصدهای تنکی مختلف هرس کردیم. به این صورت که تمامی وزن‌ها در سطح کلمه برای هر لایه را مرتب کردیم و درصدی از آنها که نزدیک صفر بودند و با حذف شدنشان، صحت شبکه تغییر زیادی نمی‌کرد هرس شدند. با این کار مقدار بیشتری وزن صفر در سطح کلمه و بیت به شبکه تحمیل شد، که باعث کم شدن هزینه حافظه و پیچیدگی محاسبات شد و درعین‌حال میزان کاهش صحت نیز بسیار ناچیز بود. نکته‌ی قابل ذکر این هست که در این جا عملیات هرس کردن همراه با یادگیری شبکه و به یکباره انجام شده است تا حداکثر صحت شبکه به همراه حداکثر میزان تنکی بدست آید.

۴-۳- فشرده‌سازی وزن‌های تنک شده

همان‌طور که در قسمت‌های گذشته اشاره شد، برای نمایش اعداد سه حالتی برخلاف شبکه دو حالتی به دو بیت نیاز داریم بنابراین سربار حافظه برای ذخیره‌سازی داده‌ها در شبکه‌های سه حالتی دو برابر شبکه‌های دو حالتی است. اما به دلیل اینکه شبکه‌های سه حالتی، شبکه‌های تنکی هستند و علاوه بر آن وزن‌های شبکه را هرس کردیم، بنابراین وزن‌های موجود دارای تعداد زیادی صفر هستند و با استفاده از این تنکی با فشرده‌سازی وزن‌ها، می‌توان حجم حافظه و متناظرا دسترسی به حافظه خارج از تراشه را کاهش داد.

الگوریتم فشرده‌سازی که در این مقاله استفاده شده‌است به این صورت است که وزن‌ها را به گروه‌های nتایی تقسیم می‌کند و برای فشرده‌سازی آن، در ابتدا n بیت برای نشان دادن اینکه هر پارامتر صفر است یا نه تخصیص داده می‌شود، در صورتی که وزن صفر بود مقدار صفر و در غیر این صورت مقدار ۱ قرار می‌گیرد و در ادامه

زمانی که n بیت ابتدایی پر شد با توجه به تعداد یک‌هایی که داخل n بیت اول قرار دارد و با توجه به اینکه هرکدام از کلمه‌های وزن‌ها با چند رقم چندی شده‌اند، به تعداد حاصل‌ضرب یک‌ها در تعداد رقم‌های چندی شده، بیت‌هایی بعد از n بیت اولیه قرار می‌گیرد. به‌طور مثال در شکل ۵ بخش (الف) یک گروه ۸تایی از مقادیر دو رقمی سه حالتی در نظر گرفته‌شده که حاوی مقادیر صفر و غیر صفر است و در شکل ۵ (پ) نیز الگوریتم فشرده‌سازی مفروض اعمال‌شده است. در واقع میزان حافظه لازم برای نشان دادن ۸ عدد دو رقمی سه حالتی، همان طور که در شکل ۵ (ب) نشان داده شده‌است ۳۲ بیت است، اما میزان حافظه لازم به کمک روش فشرده‌سازی ۲۰ بیت است، به اینصورت که در ابتدا ۸ بیت برای نشان دادن صفر بودن یا نبودن مقادیر تخصیص داده‌شده و سپس به دلیل اینکه ۳ تا از کلمات این گروه ۸تایی غیر صفر بود بعد از ۸ بیت اولیه، سه بخش چهار بیتی (۱۲ بیت) برای نشان دادن مقادیر غیر صفر اضافه شده‌است.

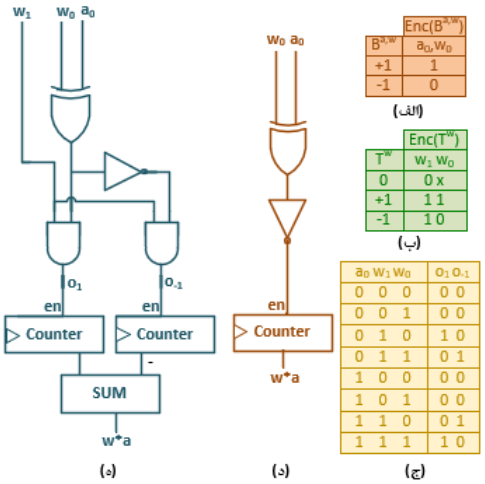


شکل ۵. الگوریتم فشرده‌سازی برای مقادیر چندی‌شده دو رقمی تنک

۴-۴- معماری واحدهای محاسباتی دوحالتی و سه‌حالتی

در این بخش از مقاله به بررسی معماری واحدهای محاسباتی برای عملیات ضرب وزن‌ها در فعال‌سازها در شبکه‌های دو حالتی و سه‌حالتی می‌پردازیم. در ابتدا، برای بررسی معماری واحدهای ضرب، به نحوه کدگذاری مقادیر دو حالتی و سه حالتی خواهیم پرداخت. در شکل ۶ (الف) به کمک یک بیت w_0 یا a_0 یک رقم وزن یا فعال‌ساز دو حالتی (B^w یا B^a) کدگذاری شده‌اند و در شکل ۶ (ب) نیز یک رقم وزن سه حالتی T^w به کمک دو بیتِ w_0 و w_1 کدگذاری شده‌است. در ادامه در شکل ۶ (ج) نیز جدول درستی برای ضرب یک رقم دو حالتی در سه حالتی ذکر شده است.

در شکل ۶ (د) معماری ضرب یک رقم وزن دو حالتی در یک رقم فعال‌ساز سه حالتی ترسیم شده است.که در ابتدا عملیات ساده بیتی Xnor بین وزن و فعال‌ساز دو حالتی اعمال می‌شود و سپس عملیات شمارش با استفاده از Counter روی آن‌ها اعمال می‌شود. اما در این مقاله به دلیل اینکه وزن‌ها سه حالتی و فعال‌سازها دو حالتی در نظر گرفته‌شده‌اند می‌بایست یک معماری برای انجام محاسبات ضرب برای آن‌ها طراحی کرد که مدار متناظر با این عملیات در شکل ۶ (ه) نشان داده‌شده است. بنابراین ضرب ارقام وزن سه حالتی در فعال‌ساز دو حالتی، با دو بیت وزن w_0 و w_1 در یک بیت فعال‌ساز a_0 همانند شکل ۱ (ه) با استفاده از گیت‌های Xor، Not و And انجام‌شده است. در مدار سه حالتی از دو عملیات Counter جهت به دست آوردن مجموع ۱ ها و مجموع ۱- ها استفاده شده که نهایتا این دو مقدار با هم جمع شده‌اند.



شکل ۶. (الف) کدگذاری ارقام دو حالتی فعال‌سازها و وزن‌ها، (ب) کدگذاری ارقام سه‌حالتی وزن‌ها، (ج) جدول درستی ضرب یک رقم فعال‌ساز دوحالتی کدگذاری شده در یک رقم وزن سه‌حالتی کدگذاری شده متناظر با آن، (د) مدار محاسبات ضرب یک رقم دوحالتی وزن در یک رقم دوحالتی فعال‌ساز متناظرش، (ه) مدار محاسبات ضرب یک رقم وزن سه‌حالتی در یک رقم فعال‌ساز دوحالتی

۵- نتایج

در این بخش از این مقاله، به بررسی نتایج و بهبود شبکه به کمک چندی‌سازی آگاه از تراکم صفر خواهیم پرداخت.

۵-۱- راه‌اندازی

در این مقاله برای به دست آوردن نتایج از مجموعه داده CIFAR10 و شبکه VGG_Small به عنوان یک مجموعه داده و شبکه طبقه‌بند تصویر که محبوبیت زیادی در حوزه ی پردازش تصویر در سیستم‌های نهفته دارد استفاده شده است. در ادامه درباره آن‌ها توضیح مختصری خواهیم داد.

این مجموعه داده شامل نمونه‌هایی از ده کلاس: هواپیما، خودرو،

پرنده، گربه، آهو، سگ، قورباغه، اسب، کشتی و کامیون است. مجموعه آموزشی شامل ۵۰۰۰۰ نمونه RGB و مجموعه آزمون شامل ۱۰۰۰۰ نمونه است. هر نمونه دارای وضوح ۳۲x۳۲ پیکسل است.

شبکه‌ی کانولوشنی VGG-Small که در واقع نسخه‌ی ساده شده‌ی شبکه‌ی VGG-Net استفاده شده در [۴۴, ۵۲] هست، شامل ۵ لایه کانولوشن، تجمیع بیشینه[1]، نرمالسازی[2]، تابع یکسوساز خطی[3] و در انتها یک لایه‌ی تمام‌اتصال جهت کلاس‌بندی نتایج است.

نتایج به‌دست‌آمده در بخش بعدی برای ۴۰۰ دوره[4] است که در بررسی نتایج از حروف A (بیانگر فعال‌سازها) و W (بیانگر وزن‌ها) و همراه با آن‌ها اعدادی (بیانگر تعداد سطوح) استفاده شده است. همچنین از (b) ، (t) و (f) به ترتیب برای رقم‌های دو حالتی و سه حالتی و ممیز شناور استفاده شده است.

۵-۲- نتایج صحت چندی‌سازی غیریکنواخت سه‌حالتی

برای اندازه‌گیری و بررسی صحت چندی‌سازی ارائه شده در این مقاله، چهار حالت در نظر گرفته شد، که نتایج آنها بر روی شبکه VGG_Small با مجموعه داده CIFAR10 در نمودار ۱ نشان داده شده است. اولین حالتی که برای چندی‌سازی در نظر گرفته شد، چندی‌سازی غیریکنواخت دو حالتی است که این نوع چندی‌سازی هم برای وزن‌ها و هم برای فعال‌سازها اعمال شده و در نمودار ۱ آن را با A(b)W(b) نمایش داده‌ایم. در صورتی که تعداد رقم را یک در نظر بگیریم، این حالت نماینده ای از مقاله‌ی [۴۴] و با افزایش تعداد رقم مشابه مقاله‌ی [۲] خواهد بود. در نوع بعدی، فعال‌سازها به صورت دقت کامل (ممیز شناور) در نظر گرفته شده و فقط وزن‌ها را به صورت غیریکنواخت سه حالتی چندی کردیم که با A(f)W(t) در نمودار مشخص شده‌است. روش بعدی، همان ایده ما در این مقاله است به اینصورت که وزن‌ها به صورت غیریکنواخت سه حالتی و فعال‌سازها به صورت غیریکنواخت دو حالتی چندی شدند که در نمودار ۱ به اختصار با A(b)W(t) نشان داده شده است. در این حالت نیز در صورتی که تعداد رقم را یک فرض کنیم نماینده‌ی مقاله‌ی [۲۲] خواهد شد. برای سنجش صحت، مدل‌های مختلف چندی‌سازی‌های ذکر شده را با شبکه ممیزشناور با عنوان A(f)W(f) مقایسه کرده‌ایم. همانطور که در نمودار ۱ مشخص است، در محور عمودی صحت و در محور افقی شاهد تعداد ارقام چندی‌سازی برای وزن‌ها و فعال‌سازها هستیم به اینصورت که مثلا A3W2 به معنای فعال‌ساز سه رقمی و وزن دو رقمی است.

با توجه به نمودار ۱ در حالت A(f)W(t) که فعال‌سازها به صورت ممیزشناور و وزن‌ها به صورت غیریکنواخت سه حالتی چندی شده‌اند، صحت شبکه در حالتی که فعال‌سازها ۳۲ بیت و وزن‌ها به صورت یک رقمی سه حالتی چندی شده باشند، تقریبا ۰,۲ درصد از صحت شبکه ممیزشناور کمتر است. اما در صورتی‌که چندی‌سازی وزن‌ها را از حالت یک رقم سه حالتی به چند رقم سه حالتی تغییر دهیم، مثل ۲ یا ۳ رقم سه حالتی، صحت شبکه به صحت شبکه ممیز شناور می‌رسد. در واقع در صورتی که فعال‌سازها دقت کامل باشند، با چندی کردن وزن‌ها تا ۲ رقم سه حالتی به صحت شبکه ممیز شناور خواهیم رسید.

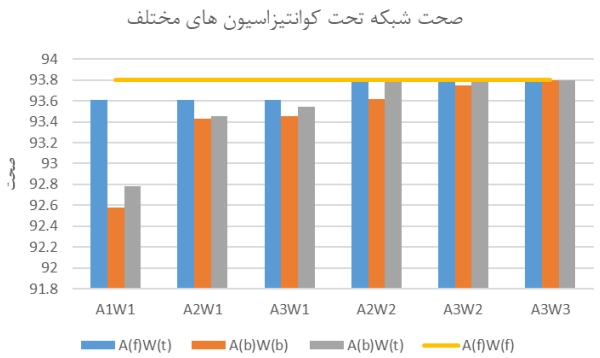
اما در نمودار ۱ در حالتی که هم وزن‌ها و هم فعال‌سازها به صورت دو حالتی A(b)W(b) چندی شده‌اند، زمانی که وزن‌ها به صورت یک رقمی دو حالتی باشند و فعال‌سازها از یک رقم دو حالتی به دو الی سه رقم افزایش پیدا کنند (A1W1 و A2W1 و A3W1) شاهد افزایش صحت در شبکه هستیم. همانطور که در نمودار ۱ نیز مشخص است در حالتی که وزن‌ها را به صورت یک رقمی دو حالتی و فعال‌سازها را به صورت غیریکنواخت دو حالتی چندی کردیم در حالت A3W1 به بیشترین صحت خواهیم رسید. به همین منظور در ادامه فعال‌سازها را سه رقمی دو حالتی در نظر می‌گریم. همانطور که ملاحظه می‌شود در حالت A3W2 صحت شبکه ۰,۰۵ درصد از صحت شبکه ممیز شناور کمتر است، اما در حالت A3W3 صحت شبکه به اندازه صحت شبکه ممیز شناور شده است. بنابراین در حالتی که فعال‌سازها و وزن‌ها به صورت غیریکنواخت دو حالتی چندی می‌شوند، اگر هر کدام از وزن‌ها و فعال‌سازها را با سه رقم دو حالتی چندی کنیم صحت شبکه چندی شده ما به صحت شبکه ممیز شناور خواهد رسید.

اما نوع دیگر چندی‌سازی ارائه شده در نمودار ۱ که ایده این مقاله است، حالت A(b)W(t) است که وزن‌ها به صورت سه حالتی و فعال‌سازها به صورت دو حالتی چندی شده‌اند. در این حالت نیز همانند حالت قبل زمانی که وزن‌ها به صورت سه حالتی یک رقمی و فعال‌سازها به صورت دو حالتی از یک رقم تا سه رقم افزایش پیدا می‌کند، صحت شبکه همانند شبکه A(b)W(b) افزایش می‌یابد. در این شبکه در صورتی‌که فعال‌سازها به صورت دو رقمی دو حالتی و وزن‌ها به صورت دو رقمی سه حالتی باشند، صحت شبکه با شبکه

[1] Max-polling
[2] Batch-normalization
[3] Relu
[4] Epoch

ممیز شناور برابر خواهد شد و صحت شبکه‌های A3W2 و A3W3 نیز با شبکه ممیز شناور برابر شد.

در نمودار ۱ صحت شبکه A(b)W(b) در تعداد ارقام مختلف همواره از صحت شبکه A(b)W(t) کمتر است، دلیل آن نیز به این خاطر است که سطوح چندی‌سازی وزن‌ها در مدل ارائه شده در این مقاله، بیشتر از سطوح شبکه دو حالتی است به علاوه صحت شبکه دو حالتی در حالت A3W3 به صحت شبکه ممیز شناور می‌رسد این در حالی است که صحت شبکه سه حالتی در حالت A2W2 به صحت شبکه ممیز شناور خواهد رسید. صحت شبکه A(f)W(t) در اکثر مواقع از مدل ما و شبکه دو حالتی نیز بیشتر است، و دلیل آن هم این است که فعال‌سازها با دقت کامل هستند و فقط وزن‌های آن به صورت غیریکنواخت سه حالتی چندی شده‌اند. اما این شبکه به دلیل آن‌که فعال‌سازهای آن به صورت دقت کامل هستند، مطلوب ما نیست، زیرا این امر به بهبود حافظه و انجام عملیات با پیچیدگی کمتر منجر نخواهد شد.



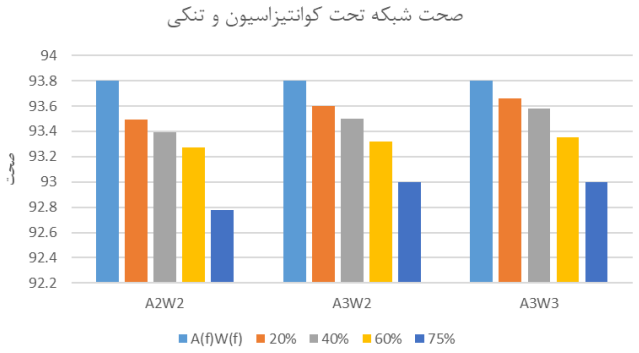
نمودار ۱. مقایسه صحت شبکه vgg-small به همراه مجموعه داده cifar10 برای انواع چندی‌سازی‌های مختلف. A_mW_n در محور افقی به معنی m رقم برای چندی‌سازی فعال‌سازها و n رقم برای چندی‌سازی وزن‌ها هست.

۵-۳- نتایج چندی‌سازی غیریکنواخت سه‌حالتی به همراه هرس کردن

در این بخش ابتدا به بررسی صحت در شبکه‌های چندی شده تنک می‌پردازیم، در ادامه نتایج حاصل از فشرده‌سازی وزن‌های سه حالتی را نشان می‌دهیم و در نهایت تسریع بالقوه حاصل از شبکه چندی شده‌ی تنک پیشنهاد شده در این مقاله را مورد مطالعه قرار خواهیم داد.

۵-۳-۱- بررسی اثر هرس کردن روی صحت شبکه

در بخش ۵-۲ به بررسی صحت شبکه در چندی‌سازی‌های مختلف پرداختیم و نشان دادیم که مدل ما در حالت A2W2 صحتی به اندازه شبکه دقت کامل دارد. حال در ادامه می‌خواهیم مدل چندی‌سازی ارائه شده را هرس کنیم. برای این منظور در شبکه A(b)W(t) برای وزن‌ها ۲۰، ۴۰، ۶۰ و ۷۵ درصد، تنکی اعمال کردیم. در اصل کاری که انجام شده به اینصورت است که از میان وزن‌ها درصد مد نظر (۲۰، ۴۰، ۶۰ و ۷۵ درصد) از آنها که نزدیک صفر هستند را همراه با یادگیری شبکه هرس می‌کنیم. در نمودار ۲ صحت شبکه‌هایی که برای درصدهای مختلف، هرس شده‌اند را گزارش داده‌ایم. همانطور که از نمودار ۲ مشخص است، هر چه درصد هرس کردن را زیاد می‌کنیم، دقت شبکه بیشتر کاهش می‌یابد. به طور مثال در A2W2 در شبکه بدون هرس صحت حدود ۹۳,۸ است اما وقتی ۲۰ درصد وزن‌ها را هرس می‌کنیم دقت شبکه حدود ۰,۳ درصد کاهش می‌یابد و برای ۴۰ درصد حدود ۰,۴ درصد و برای ۶۰ درصد ۰,۵ درصد و برای ۷۵ درصد حدود ۱ درصد صحت شبکه نسبت به شبکه بدون اعمال هرس کاهش می‌یابد.



نمودار ۲. مقایسه صحت شبکه با ترکیب ارقام مختلف وزن و فعال‌ساز تحت درصدهای تنکی مختلف.

در نمودار ۲ در حالت A3W2 و A3W3 زمانی که ۷۵ درصد وزن‌ها را هرس کنیم، صحت شبکه به ۹۳ درصد خواهد رسید که نسبت به شبکه ممیز شناور تنها ۰,۸ درصد افت دقت خواهیم داشت. از آن جا که میزان افت صحت در این حالات از A2W2 کمتر است و بعلاوه در حالت A3W2 تعداد بیت کمتری استفاده شده است، بنابراین در ادامه از شبکه‌ی vgg-small با چندی‌سازی A3W2 استفاده خواهیم کرد.

۵-۳-۲- میزان فشرده‌سازی پارامترها بعد از تنکی شبکه

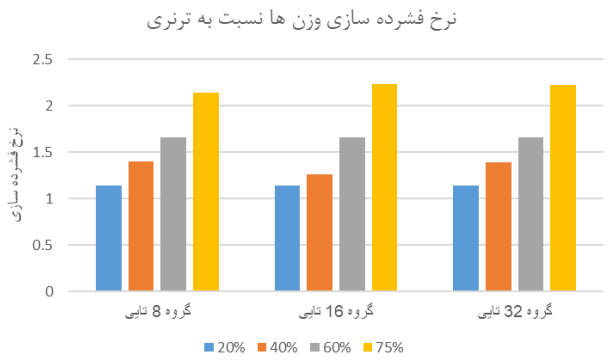
همانطور که در بخش ۳-۴ نیز بیان شد، به دلیل اینکه نشان دادن ارقام سه حالتی نسبت به ارقام دو حالتی، سربار حافظه ای دو برابر دارند، با استفاده از مزایای تنکی یک نوع فشرده‌سازی برای ذخیره‌سازی اعداد سه حالتی در حافظه بیان کردیم. نتایج متناظر

با این عملیات در نمودار ۳ و نمودار ۴ نشان داده شده است. گروه بندی برای ذخیره‌سازی وزن‌ها را ۸، ۱۶ و ۳۲ در نظر گرفتیم، یکبار تعداد بیت سه حالتی را نسبت به سه حالتی فشرده شده (نمودار ۳) و بار دیگر تعداد بیت دو حالتی را نسبت به سه حالتی فشرده شده (نمودار ۴) اندازه گیری کردیم، و این عملیات را برای درصدهای مختلف تنکی در حالت A3W2 بدست آوردیم. انتظار داریم که میزان فشرده‌سازی که صورت گرفته به اندازه‌ای باشد که تعداد بیت‌های فشرده که برای نمایش اعداد سه حالتی نیاز داریم نصف تعداد بیت‌های لازم برای نمایش اعداد سه حالتی، یا هم‌اندازه بیت‌های لازم برای نشان دادن عدد دو حالتی متناظرش باشد، تا بتوانیم ادعا کنیم که فشرده‌سازی که ارائه دادیم مطلوب است و سربار حافظه برای نشان دادن اعداد سه حالتی ندارد. بنابراین حالتی از نظر ما مطلوب است که نرخ فشرده‌سازی آن نسبت به سه حالتی ۲ برابر و نسبت به دو حالتی به اندازه یکسان باشد.

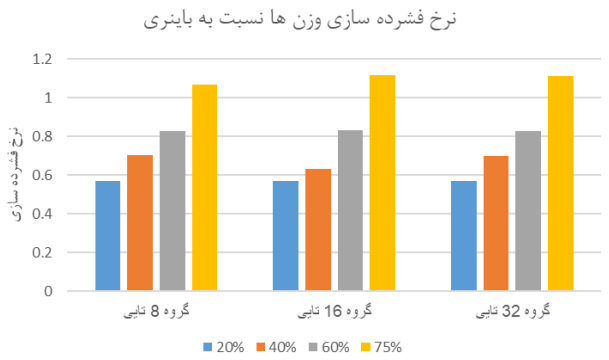
به عنوان مثال در نمودارهای۳ و۴ وزن‌ها را دو رقمی سه حالتی در نظر گرفتیم و در حالتی که هیچ نوع فشرده‌سازی نداشته باشیم و گروه‌ها ۸ تایی باشند، در کل ۳۲=۸×۲×۲ بیت برای نشان دادن اعداد سه حالتی و ۱۶=۸×۲ بیت برای نشان دادن اعداد دو حالتی نیاز خواهیم داشت، اما همانطور که در نمودار ۳ نشان داده شده است برای گروه‌های ۸تایی در صورتیکه ۲۰ درصد وزن‌ها را هرس کنیم، نرخ فشرده‌سازی نسبت به سه حالتی تقریبا ۱ است. در همین حالت در نمودار ۴ نرخ فشرده‌سازی نسبت به دو حالتی تقریبا ۰,۵ خواهد بود. این بدان معناست که تعداد بیت لازم بعد از اعمال عملیات فشرده‌سازی برای وزن‌ها با ۲۰ درصد تنکی تقریبا به اندازه تعداد بیت سه حالتی تنک نشده است.

همانطور که در نمودار ۳ و نمودار ۴ مشاهده می‌شود، هرچه میزان تنکی افزایش می‌یابد، میزان فشرده‌سازی نسبت به سه حالتی به ۲ نزدیک شده و نسبت به دو حالتی به ۱ نزدیک می‌شود، یعنی اگر میزان تنکی را بیشتر کنیم شاهد برابری تعداد بیت فشرده با دو حالتی خواهیم بود.

به طور مثال در حالتی که وزن‌ها را تا ۷۵ درصد هرس کنیم و گروه اعداد را ۸ تایی در نظر گرفتیم، میزان فشرده‌سازی وزن‌ها نسبت به سه حالتی ۲,۱۳ و نسبت به دو حالتی ۱,۰۶ شده این بدان معناست که میزان فشرده‌سازی در ۷۵ درصد تنکی علاوه بر اینکه تعداد بیت مورد نیازاش از دو حالتی بیشتر نیست بلکه توانستیم تعداد بیت مورد نیاز را از دو حالتی نیز کمتر کنیم.



نمودار ۳. نرخ فشرده‌سازی وزن‌ها در حالت A3W2 سه حالتی برای درصدهای تنکی مختلف به نسبت سه حالتی



نمودار ۴. نرخ فشرده‌سازی وزن‌ها در حالت A3W2 سه حالتی برای درصدهای تنکی مختلف به نسبت دو حالتی

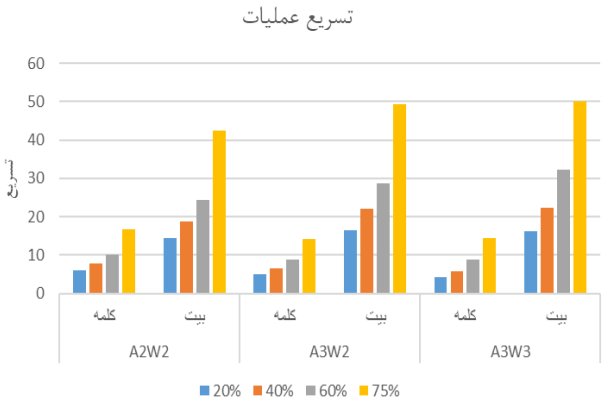
در اصل برای حالت A3W2 با هرس کردن وزن‌ها تا ۷۵ درصد، صحت شبکه برابر با ۹۳ شد، که نسبت به حالت سه حالتی و دو حالتی بدون هرس کردن، که دقت آن ۹۳,۸ بود، حدود ۰,۸ درصد صحت آن کمتر شد و بعلاوه همانطور که در نمودار ۳ مشاهده می‌کنید در ۷۵ درصد تنکی، نسبت تعداد بیت فشرده‌سازی از تعداد بیت دو حالتی کمتر نیز شد. پس در واقع در حالتی که وزن‌های سه حالتی تنک داشته باشیم، تعداد بیت‌های لازم برای نشان دادن ارقام سه حالتی که دو برابر ارقام دو حالتی است را به کمک فشرده‌سازی کاهش داده و نشان دادیم، در ۷۵ درصد تنکی، تعداد بیت فشرده شده حتی از دو حالتی نیز کمتر شد. در واقع میزان حافظه مصرفی شبکه سه حالتی با ۷۵ درصد تنکی از شبکه دو حالتی کمتر است. نکته‌ی پایانی هزینه‌ی سخت افزاری حاصل از فشرده‌سازی وزن‌ها است. هر الگوریتم فشرده‌سازی، هزینه ای را به سخت افزار تحمیل می‌کند ولی از آن جایی که روش مذکور تنها از کدگذاری صفرها به صورت یک ساختار مرتب بهره می‌برد هزینه‌ی خروج از فشرده‌سازی خیلی ناچیز هست و می‌توان با استفاده یک مدار ساده تشکیل شده از یک ثبات تغییر مکان و تسهیم کننده، وزن‌ها را از حالت فشرده شده خارج کرد.

۵-۳-۳- محاسبه‌ی تسریع بالقوه حاصل از تنکی شبکه

برای بدست آوردن تسریع بالقوه ناشی از تنکی وزن‌ها و فعال‌سازها، می‌خواهیم با تشخیص و شناسایی کلمه‌ها و بیت‌های صفر، بررسی کنیم که در صورت انجام ندادن عملیات ضرب با ورودی صفر در شبکه، به چه میزان تسریع بالقوه در محاسبات دست پیدا خواهیم کرد.

برای این منظور زوج‌هایی از وزن‌ها و فعال‌سازها را در نظر گرفتیم و در صورتی‌که هر کدام از ارقام وزن و فعال‌ساز متناظرش صفر بود، عملیات ضرب بین آن‌ها را نادیده گرفتیم و بدین منظور عملیات ضرب در صفر را در سطح کلمه حذف کردیم. در این حالت تسریع بالقوه را نسبت کل حاصلضرب‌ها به حاصلضرب‌های غیرصفر شده در نظر می‌گیریم (N/Non-zero product). علاوه بر این سعی کردیم عملیات ضرب را در سطح بیتی نیز بررسی کرده و عملیات ضرب بین بیت‌های وزن و فعال‌ساز را در صورتی که یکی از آن‌ها صفر بود در سطح بیت حذف کنیم. نمودار ۵ نشان دهنده میزان تسریع بالقوه عملیات ضرب در شبکه هرس شده، هم در سطح کلمه و هم در سطح بیت می‌باشد. که با توجه به نمودار در تمامی ترکیب‌های ارقام مختلفِ وزن و فعال‌ساز، هرچقدر که درصد تنکی افزایش می‌یابد، به دلیل حذف بیشتر عملیات ضرب در صفر، میزان تسریع بالقوه هم در سطح کلمه و هم در سطح بیت افزایش خواهد یافت.

به عنوان مثال در حالت A3W3 برای عملیات ضرب بین وزن‌ها و فعال‌سازها بدون در نظر گرفتن تنکی (همانند شبکه دو حالتی) به ۹ سیکل کلاک نیاز است، این در حالی است که اگر عملیات صفر در سطح کلمه را نادیده بگیریم، با توجه به نمودار ۵ در ۶۰ درصد تنکی میزان تسریع بالقوه در سطح کلمه ۸ برابر و در سطح بیت ۳۲ برابر بهتر خواهد شد.



نمودار ۵. بررسی میزان تسریع بالقوه در سطح کلمه و بیت برای درصدهای تنکی مختلف

جدول ۲ نتایج توان مصرفی و مساحت برای واحدهای محاسباتی ارایه شده در شکل ۶ را نشان می‌دهد. این نتایج با استفاده از سنتز مدار مورد نظر به کمک ابزار سنتز تجاری و با کتابخانه تکنولوژی ۴۵ نانومتر FreePDK بدست آمده است. توان مصرفی و مساحت مدار محاسباتی ضرب سه حالتی به نسبت ضرب دو حالتی به ترتیب ۲,۲ و ۳,۱۵ برابر بیشتر است.

جدول ۲. نتایج سخت افزاری واحدهای محاسباتی

	دو حالتی	سه حالتی	نسبت سه‌حالتی به دوحالتی
توان‌مصرفی(mW)	0.18	0.40	2.2
مساحت (um^2)	142	448	3.15

با توجه به معماری ارائه شده برای عملیات ضرب بین دو حالتی و سه حالتی، با اینکه مدار ارائه شده در شکل ۶ (ت) تقریبا دو برابر ضرب دو حالتی در دو حالتی سربار محاسباتی دارد، اما با توجه به تسریع‌های بالقوه بیان شده در بالا که همگی از دو برابر خیلی بیشتر هستند، می‌توانیم نشان دهیم که میزان تسریع عملیات ضرب با حذف کردن محاسبات ضرب در صفر می‌تواند بسیار زیاد باشد.

۶- نتیجه‌گیری

چندی‌سازی و هرس کردن دو روش محبوب جهت فشرده‌سازی شبکه‌های عصبی عمیق برای قابل پیاده‌سازی شدن آنها بر روی دستگاه نهفته با محدودیت‌های حافظه و محاسبات هستند. چندی‌سازی غیریکنواخت دو حالتی از محاسبات بیتی بهره می‌برد و افت صحت شبکه‌های دو حالتی پایه را به حداقل می‌رساند. ما در این مقاله چندی‌سازی غیریکنواخت سه حالتی را ارائه دادیم، که صحت آن نزدیک به اندازه صحت شبکه ممیز شناور و بالاتر از صحت شبکه غیریکنواخت دو حالتی است و بر خلاف شبکه غیریکنواخت دو حالتی که وزن‌ها تنک نیستند، در این نوع از چندی‌سازی وزن‌ها قابلیت تنک شدن دارند. البته روشن است که هزینه‌ی حافظه و محاسبات غیریکنواخت سه حالتی برای هر محاسبات در حالتی که هیچ بهینه‌سازی انجام نشود، تقریبا دو برابر غیریکنواخت دو حالتی است. اما غیریکنواخت سه حالتی شانس هرس شدن دارد. لذا وزن‌های شبکه چندی شده با نمایش اعداد غیریکنواخت سه حالتی را با درصدهای مختلف هرس کردیم و بررسی کردیم که در حالتی که وزن‌ها تا ۷۵ درصد هرس شوند، صحت شبکه نسبت به شبکه سه حالتی بدون هرس فقط ۰,۸ درصد کاهش می‌یابد. علاوه بر این در شبکه غیریکنواخت سه حالتی با استفاده از فشرده‌سازی وزن‌ها، نه تنها سربار حافظه نسبت به غیریکنواخت دو حالتی نداشتیم، بلکه توانستیم میزان حافظه مصرفی برای ذخیره‌سازی وزن‌های سه

حالتی را نسبت به شبکه دو حالتی کمتر کنیم. از طرفی برای بهره بردن از تنکی جهت کاهش محاسبات، برای عملیات ضرب وزن سه حالتی در فعال‌ساز دو حالتی یک معماری ارائه دادیم که پیچیدگی محاسبات ضرب، دو برابر شبکه دو حالتی بود، اما به دلیل اینکه شبکه غیریکنواخت سه حالتی ما تنک است با در نظر گرفتن امکان حذف عملیات ضرب غیر موثر، میزان تسریع بالقوه عملیات ضرب در سطح کلمه و بیت به ترتیب ۱۵ و ۴۵ برابر نسبت به شبکه غیریکنواخت دو حالتی بهتر است. و این میزان تسریع به اندازه‌ای بالا است که به سربار تقریبا دو برابری محاسبات غالب می‌شود.

مراجع

[1] LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning*. nature, 2015. **521**(7553): p. 436-444.

[2] Zhang, D., et al. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. in Proceedings of the European conference on computer vision (ECCV). 2018.

[3] Yang, L., Z. He, and D. Fan. Harmonious coexistence of structured weight pruning and ternarization for deep neural networks. in Proceedings of the AAAI Conference on Artificial Intelligence. 2020.

[4] Burrello, A., et al., *Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus.* IEEE Transactions on Computers, 2021. **70**(8): p. 1253-1268.

[5] de Prado, M., et al., *Automated Design Space Exploration for Optimized Deployment of DNN on Arm Cortex-A CPUs.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020. **40**(11): p. 2293-2305.

[6] Howard, A.G., et al., Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[7] Iandola, F.N., et al., SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360, 2016.

[8] Courbariaux, M., Y. Bengio, and J.-P. David, *Binaryconnect: Training deep neural networks with binary weights during propagations.* Advances in neural information processing systems, 2015. **28**.

[9] Cai, Z., et al. Deep learning with low precision by half-wave gaussian quantization. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[10] Li, F., B. Zhang, and B. Liu, *Ternary weight networks.* arXiv preprint arXiv:1605.04711, 2016.

[11] He, Y., X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. in Proceedings of the IEEE international conference on computer vision. 2017.

[12] Han, S., et al., *Learning both weights and connections for efficient neural network.* Advances in neural information processing systems, 2015. **28**.

[13] Luo, J.-H., J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. in Proceedings of the IEEE international conference on computer vision. 2017.

[14] Maji, P., et al. Efficient winograd or cook-toom convolution kernel implementation on widely used mobile cpus. in 2019 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2). 2019. IEEE.

[15] Andri, R., et al. YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights. in 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). 2016. IEEE.

[16] Jacob, B., et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[17] Gysel, P., et al., *Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks.* IEEE transactions on neural networks and learning systems, 2018. **29**(11): p. 5784-5789.

[18] Sharma, H., et al. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. in 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). 2018. IEEE.

[19] Hubara, I., et al., Quantized neural networks: Training neural networks with low precision weights and activations. The Journal of Machine Learning Research, 2017. **18**(1): p. 6869-6898.

[20] Li, Y. and F. Ren. BNN Pruning: Pruning binary neural network guided by weight flipping frequency. in 2020 21st International Symposium on Quality Electronic Design (ISQED). 2020. IEEE.

[21] Jin, C., H. Sun, and S. Kimura. Sparse ternary connect: Convolutional neural networks using ternarized weights with enhanced sparsity. in 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). 2018. IEEE.

[22] Zhu, C., et al., *Trained ternary quantization.* arXiv preprint arXiv:1612.01064, 2016.

[23] Chin, T.-W., et al. One weight bitwidth to rule them all. in European Conference on Computer Vision. 2020. Springer.

[24] Ghasemzadeh, M., M. Samragh, and F. Koushanfar. ReBNet: Residual binarized neural network. in 2018 IEEE 26th annual international symposium on field-programmable custom computing machines (FCCM). 2018. IEEE.

[25] Zhao, Y., et al., *Focused quantization for sparse cnns.* Advances in Neural Information Processing Systems, 2019. **32**.

[26] Long, X., et al., Learning sparse convolutional neural network via quantization with low rank regularization. IEEE Access, 2019. **7**: p. 51866-51876.

[27] Long, X., et al. Low Bit Neural Networks with Channel Sparsity and Sharing. in 2022 7th International Conference on Image, Vision and Computing (ICIVC). 2022. IEEE.

[28] Gadosey, P.K., Y. Li, and P.T. Yamak. On pruned, quantized and compact CNN architectures for vision applications: an empirical study. in Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing. 2019.

[29] Albericio, J., et al., *Cnvlutin: Ineffectual-neuron-free deep neural network computing.* ACM SIGARCH Computer Architecture News, 2016. **44**(3): p. 1-13.

[30] Han, S., et al., *EIE: Efficient inference engine on compressed deep neural network.* ACM SIGARCH Computer Architecture News, 2016. **44**(3): p. 243-254.

[31] Parashar, A., et al., *SCNN: An accelerator for compressed-sparse convolutional neural networks.* ACM SIGARCH computer architecture news, 2017. **45**(2): p. 27-40.

[32] Zhang, S., et al. Cambricon-X: An accelerator for sparse neural networks. in 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). 2016. IEEE.

[33] Delmas, A., et al., DPRed: Making typical activation and weight values matter in deep learning computing. arXiv preprint arXiv:1804.06732, 2018.

[43] Elsken, T., J.H. Metzen, and F. Hutter, *Efficient multi-objective neural architecture search via lamarckian evolution*. arXiv preprint arXiv:1804.09081, 2018.

[44] Rastegari, M., et al. Xnor-net: Imagenet classification using binary convolutional neural networks. in European conference on computer vision. 2016. Springer.

[45] Zhang, J., F. Franchetti, and T.M. Low. High performance zero-memory overhead direct convolutions. in International Conference on Machine Learning. 2018. PMLR.

[46] Lavin, A. and S. Gray. Fast algorithms for convolutional neural networks. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[47] De Prado, M., N. Pazos, and L. Benini. Learning to infer: RL-based search for DNN primitive selection on Heterogeneous Embedded Systems. in 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). 2019. IEEE.

[48] Rovder, S., J. Cano, and M. O'Boyle, Optimising convolutional neural networks inference on low-powered GPUs. 2019.

[49] Lin, X., C. Zhao, and W. Pan, *Towards accurate binary convolutional neural network.* Advances in neural information processing systems, 2017. **30**.

[50] Gholami, A., et al., A survey of quantization methods for efficient neural network inference. arXiv preprint arXiv:2103.13630, 2021.

[51] Hawks, B., et al., Ps and qs: Quantization-aware pruning for efficient low latency neural network inference. Frontiers in Artificial Intelligence, 2021. **4**: p. 676564.

[52] Hubara, I., et al., *Binarized neural networks.* Advances in neural information processing systems, 2016. **29.**

[34] Aimar, A., et al., NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps. IEEE transactions on neural networks and learning systems, 2018. **30**(3): p. 644-656.

[35] Li, J., et al., *SqueezeFlow: A sparse CNN accelerator exploiting concise convolution rules.* IEEE Transactions on Computers, 2019. **68**(11): p. 1663-1677.

[36] Albericio, J., et al. Bit-pragmatic deep neural network computing. in Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. 2017.

[37] Sharify, S., et al. Laconic deep learning inference acceleration. in 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA). 2019. IEEE.

[38] 38. Chen, Y.-H., et al., Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE journal of solid-state circuits, 2016. **52**(1): p. 127-138.

[39] Sandler, M., et al. Mobilenetv2: Inverted residuals and linear bottlenecks. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[40] Zhang, X., et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[41] Tan, M., et al. Mnasnet: Platform-aware neural architecture search for mobile. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

[42] Wu, B., et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.