مطالعه بهینه‌سازی کارایی سخت‌افزار برای شبکه‌های عصبی با حافظه طولانی‌مدت و کوتاه‌مدت

امیر ارسلان ساختیانچی[۱] ، کوروش منوچهری کلانتری[۲] ، مهرشاد خسرویانی.[۳]
[۱] گروه مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)، پردیس گرمسار،ایران،
[۲] گروه مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)، پردیس گرمسار،ایران،
[۳] گروه مهندسی کامپیوتر و فناوری اطلاعات، واحد پرند، دانشگاه آزاد اسلامی، پرند، تهران،

چکیده :

شبکه‌های عصبی ، که با الهام از ساختار مغز انسان ساخته شده‌اند، به‌طور گسترده در هوش مصنوعی و یادگیری ماشین برای وظایفی مختلف از جمله استخراج ویژگی‌ها، طبقه‌بندی داده و کنترل سیستم‌ها مورد استفاده قرار می‌گیرند. دلیل این کاربرد گسترده، به خاطر داشتن ویژگی‌های غیرخطی و توانایی خود-انطباقی این شبکه‌ها است. این مقاله به بررسی پیاده‌سازی سخت‌افزاری شبکه‌های عصبی حافظه طولانی‌مدت و کوتاه‌مدت می‌پردازد، که نوعی پیشرفته از شبکه‌های عصبی بازگشتیهستند که به دلیل توانایی‌شان در پردازش داده‌های ترتیبی به خوبی شناخته شده‌اند. با این حال، تقاضای محاسباتی بالای شبکه‌های LSTM[1] باعث ایجاد چالش‌های بزرگی در پیاده‌سازی سخت‌افزاری به ویژه در محیط‌هایی با منابع محدود، مانند اینترنت اشیا شده است. در این مقاله، پیشرفت‌های اخیر در خصوص طراحی سخت‌افزار برای شبکه‌های LSTM مورد بررسی قرار می‌گیرند، با تأکید بر روش‌هایی که از محاسبات غیر دقیق بهره می‌برند تا پیچیدگی سخت‌افزاری و مصرف انرژی را کاهش دهند.همچنین، این مقاله به بررسی سایر تکنیک‌های مختلف بهینه سازی مانند الگوریتم‌های فشرده سازی و هرس کردن شبکه های عصبی در جهت بهبود کارایی پردازش LSTM می‌پردازد و ظرفیت این روش‌ها را ارزیابی می‌کند. علاوه بر این، مقاله به بررسی موازنه‌های موجود بین دقت و مصرف منابع در زمینه استفاده این روش‌های نوظهور سخت‌افزاری می‌پردازد.

واژه‌های کلیدی:شبکه های عصبی ، حافظه طولانی‌مدت و کوتاه‌مدت، محاسبات تصادفی ، محاسبات تقریبی

[1] Long Short-Term memory

بخش اول. مقدمه

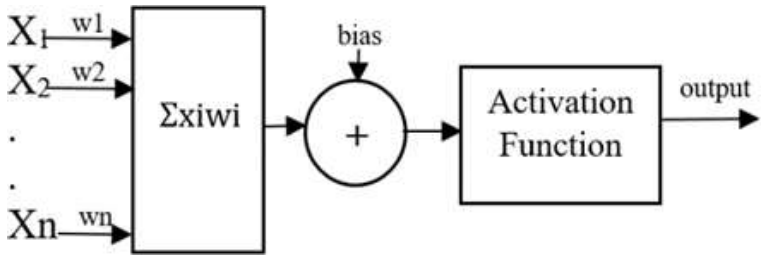
شبکه‌های عصبی از اجزای اصلی هوش مصنوعی و یادگیری ماشین هستند و برای وظایفی مانند استخراج ویژگی‌ها (Mao and Jain,1995)، طبقه‌بندی (Krizhevsky and Sutskever,2012)، و کنترل سیستم‌ها (Fierro and Lewis,1998) استفاده می‌شوند. این کاربردها به دلیل ویژگی‌های غیرخطی و قابلیت انطباق این شبکه‌ها می‌باشد (Zhang, 2000). شبکه‌های عصبی با الگوبرداری از مغز انسان، از طریق آموزش یاد می‌گیرند و دانش خود را در اتصالات نورونی ذخیره می‌کنند (Hopfield, 1982). انواع مختلفی از شبکه‌های عصبی وجود دارند، از جمله پرسپترون‌های چندلایه که برای طبقه‌بندی الگوهای پیچیده استفاده می‌شوند. (Haykin, 2009)، شبکه‌های عصبی عمیق با چندین لایه غیرخطی (Hikawa, 2003) ، شبکه‌های عصبی کانولوشنی برای طبقه‌بندی تصاویر (Schmidhuber, 2015) ، و شبکه‌های حافظه طولانی‌مدت و کوتاه‌مدت که برای داده‌های ترتیبی مانند گفتار و ویدیو استفاده می‌شوند(Graves et al, 2013) . پیاده‌سازی سخت‌افزاری شبکه‌های عصبی به دلیل توانایی بالا در پردازش موازی داده‌ها مزایای بسیاری را ارائه می‌دهد، اما این کار به دلیل وجود تعداد زیاد پارامترها ی محاسباتی پیچیده است. (Y. Liu et al, 2019) برای بهبود کارایی، تکنیک‌هایی مانند هرس آگاهانه به تعادل (Han et al,2017) و تبدیل سریع فوریه (Li et al 2018) توسعه داده شده‌اند. با این حال، پیاده سازی سخت افزاری شبکه‌های LSTM همچنان از نظر هزینه سخت‌افزاری و مصرف انرژی، به ویژه در محیط‌هایی با منابع محدود مانند اینترنت اشیا ، با چالش‌هایی همراه بوده که برای مقابله با آنها، روش‌های محاسبات غیر دقیق مانند محاسبات تصادفی و محاسبات تقریبی به کار گرفته شده‌اند تا با ساده‌سازی مدارها، در حالی که تنها اندکی از دقت کاسته می‌شود پیچیدگی سخت‌افزاری کاهش یابد، (Tina et al, 2024). محاسبات تصادفی [2] که به سخت‌افزار کمتری نیاز دارند از رشته های بیتی برای عملیات‌های حسابی استفاده میکند، پیشرفت‌های اخیر در این زمینه، سبب رقابت شبکه‌های تصادفی با طرح‌های سنتی شده است (Y.Liu et al, 2021)، همچنین رویکردهای دیگری نیز وجود دارند که بر روی شتاب‌دهنده‌های بهینه‌سازی‌شده برای سلول‌های LSTMتمرکز دارند. انتخاب روش مناسب به دقت و عملکرد مورد نیاز در برنامه بستگی دارد . در این مقاله ما به بررسی پیشرفت‌های اخیر در پیاده‌سازی سخت‌افزاری شبکه‌های عصبی LSTM بر رویASIC ها وFPGA ها می‌پردازیم، که شامل شبکه‌های محاسبات تصادفی و تکنیک‌های تقریبی می‌شود، همچنین در آخر کارایی و عملکرد آن‌ها را با هم مقایسه می‌کنیم.

[2] Stochastic

بخش دوم. پیش زمینه

الف. شبکه های عصبی

شبکه‌های عصبی از واحدهای اساسی به نام نورون تشکیل شده‌اند که الهام‌گرفته از نورون‌های موجود در مغز انسان هستند. مغز انسان حدود ده میلیارد نورون دارد (Haykin, 2009). در نورون‌های بیولوژیکی، بدن سلول ورودی‌هایی را از دندریت‌ها [3]دریافت می‌کند که به ترمینال‌های سیناپسی [4]متصل هستند و تحت تأثیر نورون‌های دیگر قرار می‌گیرند. این سیگنال‌ها به پالس‌های ولتاژ تیز، یا همان اسپایک‌ها[5]، تبدیل می‌شوند که از طریق آکسون [6]به نورون‌های دیگر منتقل می‌شوند. مغز با ایجاد اتصالات سیناپسی جدید و تغییر در اتصالات موجود به محیط خود سازگار می‌شود، که این امر منجر به تغییر در ساختار و پارامترهای شبکه عصبی می‌گردد. به طور مشابه، نورون‌های مصنوعی در شبکه‌های عصبی به عنوان واحدهای پردازش اطلاعات عمل می‌کنند و اجزای اصلی شبکه را تشکیل می‌دهند. در شکل ۱ عملکرد یک نرون مصنوعی نشان داده شده است.



شکل ۱. نرون مصنوعی(Haykin, 2009)

[3] Dendrite
[4] Synapses
[5] Spike
[6] axons

یک نورون مصنوعی یک یا چند ورودی دریافت می‌کند که هر کدام در یک وزن خاص ضرب می‌شوند. این ورودی‌های وزندار جمع می‌شوند و نتیجه توسط یک تابع فعال‌ساز پردازش می‌شود. این تابع خروجی نورون را تولید می‌کند. ورودی‌ها با $\{x_i\}$،که $i = 1,2,...,n$ است، و پارامترهای نورون (که به عنوان وزن لایه داده شده است) با $\{w_i\}, i = 1, 2, \dots, n$ مشخص میشوند. خروجی نورون طبق رابطه زیر تولید می‌شود:

$$y = \varphi(\sum_{i=1}^{n} w_i x_i) \qquad (۱)$$

در این رابطه، w وزن‌های نورون و ϕ تابع فعال‌ساز است. در شبکه‌های عصبی ، انواع مختلفی از توابع فعال‌سازی به‌طور گسترده مورد استفاده قرار می‌گیرند، از جمله تابع تانژانت هیپربولیک، تابع سیگموئید، و تابع خطی ساز، اگر فرض کنیم v ورودی تابع فعال‌سازی باشد، این توابع فعال‌سازی به صورت زیر تعریف می‌شوند:

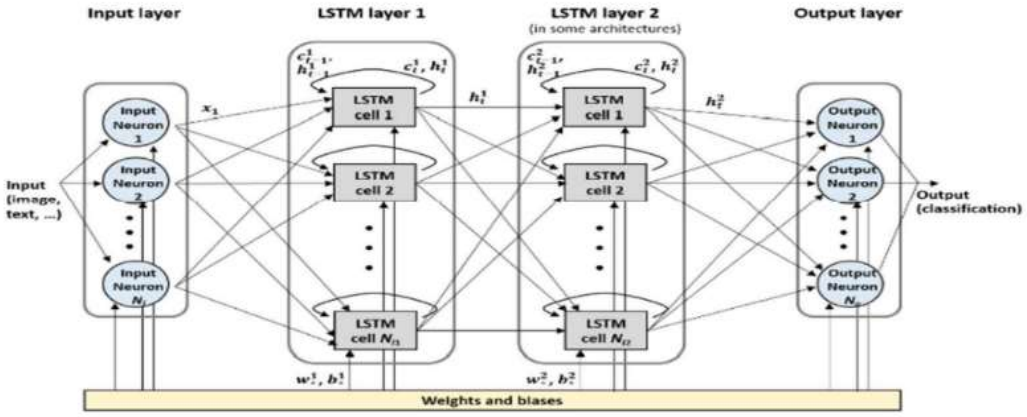
$$\text{given by} = \begin{cases} \tanh(2v) & \tanh \\ \frac{1}{1+\exp(-v)} & \text{sigmoid} \\ \max(0.v) & \text{ReLu} \end{cases} \qquad (۲)$$

شبکه‌های عصبی بازگشتی نوعی خاص از شبکه‌های عصبی هستند که برای پردازش داده‌های ترتیبی طراحی شده‌اند. برخلاف شبکه‌های عصبی پیش‌خور سنتی، شبکه‌های عصبی بازگشتی خروجی مرحله زمانی قبلی را به‌عنوان بخشی از ورودی برای مرحله زمانی جاری در نظر می‌گیرند. به‌عنوان مثال، در مرحله زمانی t، با توجه به داده ورودی x_t و خروجی مرحله زمانی قبلی H_{t-1}، همان‌طور که در معادله زیر توصیف شده است مدل f() پیش‌بینی را برای مرحله زمانی جاری H_t انجام می‌دهد،(Sengupta,et al, 2022):

$$h_t = f([x_f, h_{t-1}]) \qquad (۳)$$

ب. شبکه‌های حافظه طولانی‌مدت و کوتاه‌مدت(LSTM)

شبکه‌های حافظه طولانی‌مدت و کوتاه‌مدت (LSTM) نوعی از شبکه‌های عصبی بازگشتی (RNN) هستند که برای پردازش داده‌های ترتیبی پیچیده طراحی شده‌اند.LSTM ها به طور مؤثر در وظایفی مانند تشخیص گفتار و ترجمه زبان مورد استفاده قرار گرفته‌اند. شبکه‌های عصبی بازگشتی ی پایه با چالش‌هایی مانند مشکلات ناپدید شدن و انفجار گرادیان مواجه هستند. مشکل ناپدید شدن گرادیان زمانی رخ می‌دهد که تغییرات در لایه‌های اولیه شبکه تأثیر کمی بر خروجی داشته باشند، در حالی که مشکل انفجار گرادیان زمانی رخ می‌دهد که تغییرات کوچک در لایه‌های اولیه به‌طور بیش از حد تقویت شده و در طول شبکه منتشر شوند.یک شبکه LSTM شامل حداقل یک لایه از سلول‌هایLSTM است که همراه با لایه‌های ورودی و خروجی قرار دارند (به شکل ۲ مراجعه کنید) (Sengupta et al, 2022).



شکل ۲. تصویرکلی شبکه LSTM(Sengupta, et al, 2022)

هر سلولLSTM ، همان‌طور که در شکل۳ نشان داده شده است، یک حالت سلولی c و یک حالت پنهان h را حفظ می‌کند و این حالت‌ها را به صورت ترتیبی به‌روزرسانی می‌کند. در هر مرحله زمانی t، سلول، حالت سلولی c_t و حالت پنهان h_t خود را بر اساس ورودی فعلی x_t، وزن‌ها و بایاس‌های یادگرفته‌شده، و حالت‌های قبلی c_{t-1} و h_{t-1} به‌روزرسانی می‌کند. وضعیت‌های تمامی سلول‌های LSTM به‌طور جمعی حافظه شبکه را تشکیل می‌دهند. شبکه‌های LSTM با کنترل صریح رشته اطلاعات درون هر سلول، مشکل ناپدید شدن گرادیان را برطرف می‌کنند. این کار از طریق سه دروازه انجام می‌شود: دروازه فراموشی ، که تعیین می‌کند چه اطلاعاتی از گذشته حفظ شود؛ دروازه ورودیکه تصمیم می‌گیرد چه اطلاعات جدیدی اضافه شود؛ و در نهایت دروازه خروجی ، که حالت پنهان فعلی را محاسبه می‌کند. معادلات زیر عملکردهایی مطابق شکل ۳. را نشان می‌دهند که برای محاسبه حالت پنهان بعدی h_t استفاده می‌شوند:

$$F = \sigma(w_f.[h_{t-1}, x_t] + B_f) \quad (۴)$$

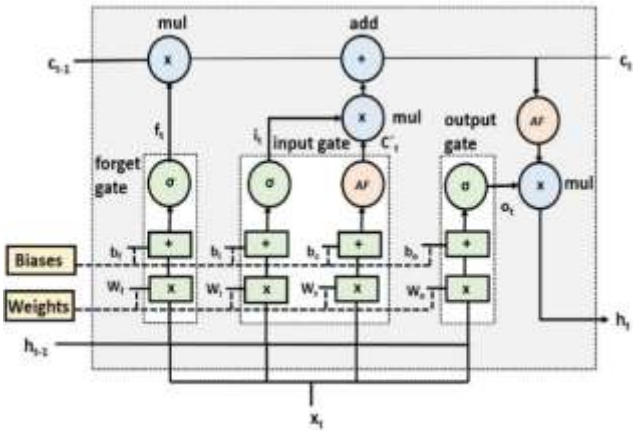
$$I = \sigma(w_I.[h_{t-1}, x_t] + B_I) \quad (۵)$$

$$\dot{C} = tanh(w_C.[h_{t-1}, x_t] + B_c) \quad (۶)$$

$$O = \sigma(w_o.[h_{t-1}, x_t] + B_o) \quad (۷)$$

$$C_t = C_{t-1} * F + I * \dot{C} \quad (۸)$$

$$h_t = tanh(C).O \quad (۹)$$



شکل ۳. تصویر سلولLSTM(Sengupta, et al, 2022)

ج. محاسبات تصادفی:**(Stochastic Computing)**

در محاسبات مرسوم، اعداد با استفاده از نمایش در مبنای ۲ نمایش داده می‌شوند، به طوری که هر بیت یا ۰ است یا ۱، و این نمایش در طول زمان ثابت می‌ماند. در مقابل، در محاسبات تصادفی ، اعداد به صورت احتمالات نمایش داده می‌شوند. یک مقدار تصادفی v با یک بیت تکی که ممکن است در هر سیکل ساعت تغییر کند نمایش داده می‌شود، به طوری که با احتمالی p برابر با ۱ و با احتمال $P-۱$ برابر با ۰ است. این احتمال مقدار ذخیره شده را تعریف می‌کند. برای مثال، اگر یک مقدار تصادفی در طول ۱۰۰ سیکل ساعت خوانده شود و بیت در ۵۰ مورد از این سیکل‌ها ۱ باشد، احتمال p برابر با ۰.۵ است. دو روش برای تفسیر این مقدار تصادفی وجود دارد. روش اول، غیرقطبی است. در این روش، احتمال p به طور مستقیم به مقدار منطقی v نگاشت می‌شود. اگر p برابر با ۰.۵ باشد، مقدار منطقی v نیز ۰.۵ است. مشکل نگاشت غیرقطبی این است که نمی‌توانیم اعداد منفی را نمایش دهیم. روش دوم تفسیر مقدار تصادفی، قطبی است. در این روش، مقدار p از دامنه [۰ ,۱] به مقدار منطقی v در دامنه [-۱ ,۱] نگاشت می‌شود. به عبارت دیگر، اگر p برابر با ۰.۵ باشد، مقدار نگاشت شده v برابر با ۰ است در مقایسه با روش غیرقطبی، روش قطبی امکان نمایش مقادیر به صورت اعداد منفی را فراهم می‌کند. (Y. Liu et al, 2021)

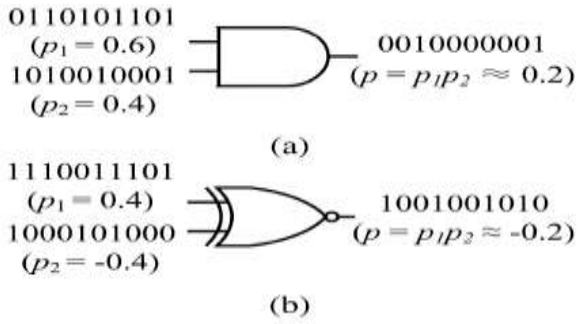
معادلات زیر نگاشت‌های خاص بین احتمال p و مقدار منطقی v را نشان می‌دهند:

$$p = (v + 1)/2 \qquad (۱۰)$$

$$v = 2 * p - 1 \qquad (۱۱)$$

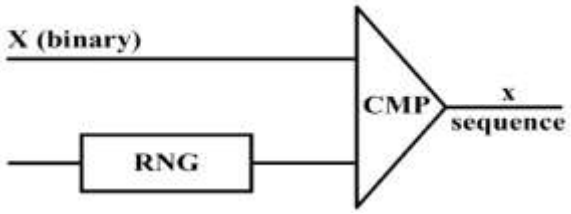
در محاسبات تصادفی ، ضرب‌کننده که یکی از مدارهای اصلی حسابی در یک نورون است، می‌تواند با استفاده از یک گیتAND برای نمایش یونی‌پلاریته یا یک دروازه XNOR برای نمایش بای‌پلاریته پیاده‌سازی شود (به شکل ۴ مراجعه کنید) (Z. Li et al, 2018). اگرچه ضرب‌کننده‌های تصادفی به فضای بسیار کمتری نسبت به ضرب‌کننده‌های باینری سنتی نیاز دارند، دقت

آن‌ها با افزایش تعداد ورودی‌ها کاهش می‌یابد. برای دستیابی به دقت قابل قبول، باید تعداد ورودی‌ها کم نگه داشته شود و دوره‌های ارزیابی طولانی‌تر باشد.



شکل ۴. (a) ضرب تصادفی برای نمایش یونی‌پلاریته(unipolar) ، و (b) برای نمایش بای پلاریته (bipolar) (Y. Liu et al, 2021)

تولیدکننده‌های اعداد تصادفی اعداد باینری را به رشته های بیتی تصادفی تبدیل می‌کنند. یک تولیدکننده‌های اعداد تصادفی معمولاً از یک رجیستر شیفت بازخوردی [7]خطی با k بیت و یک مقایسه‌کننده تشکیل شده است (به شکل ۵ مراجعه کنید).



شکل ۵. تولیدکننده‌های اعداد تصادفی (Stochastic Number Generator)

در هر سیکل ساعت، رجیستر شیفت بازخوردی خطی یک عدد تصادفیk بیتی تولید می‌کند که با ورودی باینریk بیتی مقایسه می‌شود. اگر ورودی بزرگ‌تر باشد، مقایسه‌کننده ۰ را خروجی می‌دهد، در غیر این صورت ۱ را خروجی می‌دهد. این ۰ها و ۱ها رشته بیتهای تصادفی را تشکیل می‌دهند. برای تبدیل رشته بیتهای تصادفی به اعداد باینری، از دیکودر عدد تصادفی استفاده می‌شود، که یک شمارنده باینری است و تعداد ۱ها را دریک رشته بیت شمارش می‌کند.یکی از چالش‌های محاسبات تصادفی همبستگی بین رشته‌های بیت تصادفی است که می‌تواند منجر به کاهش دقت شود. به عنوان مثال، اگر یک رجیستر شیفت بازخوردی خطی یکسان چندین رشته بیت تولید کند، این همبستگی می‌تواند دقت عملیات‌هایی مانند ضرب را تحت تأثیر قرار دهد. برای جلوگیری از این مشکل، می‌توان از رجیستر شیفت بازخوردی خطی های جداگانه برای ورودی‌های مختلف استفاده کرد، همان‌طور که در شکل۶(a) نشان داده شده است، تا خروجی‌های دقیقی حاصل شود . (Sen and Raghunathan,2018) با پیاده‌سازی حالتهای مختلف انتقال میتوان چندین تابع فعال‌سازی را پیاده‌سازی کرد. به عنوان مثال، مدار Btanh سرعت محاسبات تابع فعال‌ساز در تصادفی

[7] Linear Feedback Shift Register) LFSR)

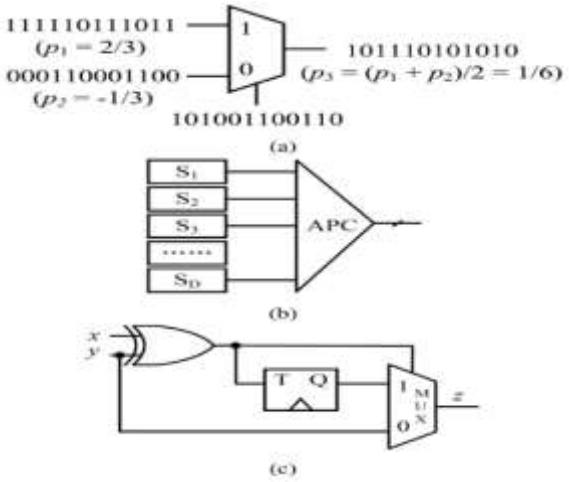
را با جمع‌آوری بیت‌های ورودی با استفاده از شمارنده موازی تقریبی و به‌روزرسانی حالت FSM در چندین مرحله در هر سیکل بهبود بخشیده است ، این طراحی دقت را حتی زمانی که احتمالات ورودی از دامنه معمول [+۱, -۱] فراتر رود، حفظ می‌کند. بهبودهای بیشتر در مدارBtanh ، مانند جایگزینی شمارنده بالا/پایین با واحد تقریب خطی ، کارایی آن را افزایش داده و توانایی آن در پردازش گسترده‌تر ورودی‌ها بدون از دست دادن دقت بهبود بخشیده است(Kim et al, 2016) . معادله LAU :

$$\psi(x) = \min\left(1, \mathrm{Max}\left(p, \frac{1}{r}x + s\right)\right) \qquad (۱۳)$$

پیکربندی واحد تقریب خطی می‌تواند با تنظیم پارامترهای p ، r ، و s توابع فعال‌سازی مختلفی مانند سیگموئید یا ReLU را تقریب بزند (Z.Li et al, 2019) . این طراحی نسبت به همبستگی بین توالی‌های ورودی مصون است و امکان اشتراک مولد اعداد تصادفی بین نورون‌ها را فراهم می‌کند که منجر به کاهش هزینه‌های سخت‌افزاری می‌شود. همچنین یک مدار تصادفی برای ReLU پیشنهاد شده است که در آن ورودی انباشته شده و با یک مقدار مرجع مقایسه می‌شود و خروجی بر اساس آن انتخاب می‌گردد. این طراحی تضمین می‌کند که توالی خروجی در نمایش یونی‌پلاریته کمتر از ۰.۵ و در نمایش بای‌پلاریته کمتر از ۰ نباشد:

$$y = \max(0, \tanh(2x)) \qquad (۱۴)$$

مدارهای حسابی چندجمله‌ای در محاسبات تصادفی می‌توانند با بسط توابع غیرخطی با استفاده از سری تیلور یا چندجمله‌ای‌های برنشتاین، توابع فعال‌سازی را نیز پیاده‌سازی کنند . اگرچه این مدارها نسبت به روش‌های مبتنی بر FSM بازپیکربندی آسان‌تری دارند، اما به فضای بیشتری نیاز دارند (B. Li et al, 2019). به طور کلی، تصادفی برای محاسبات مبتنی بر توالی به تعداد زیادی سیکل ساعت نیاز دارد، و همچنین دامنه محاسباتی جمع‌کننده تصادفی دقیق یک چالش بزرگ محسوب می‌شود (S. Lu et al, 2024). جمع‌کننده اولیه در محاسبات تصادفی از یک مالتی‌پلکسر دو ورودی استفاده می‌کند که احتمال سیگنال انتخاب آن روی ۰.۵ تنظیم شده است به شکل ۷ (a)مراجعه کنید . (Y. Liu et al,2021)



شکل ۷ (a) جمع‌کننده SC اولیه (b) .جمع‌کننده SC مبتنی بر (c) .APC
جمع‌کننده SC مبتنی برTFF .

این طراحی تحت تأثیر همبستگی ورودی‌ها قرار نمی‌گیرد و به این دلیل امکان استفاده از مولدهای اعداد تصادفی مشترک را فراهم می‌کند، که به کاهش هزینه سخت‌افزار و مصرف انرژی منجر می‌شود. با این حال، سیگنال انتخاب باید بدون همبستگی با ورودی‌ها باشد، که نیاز بهRNG های اضافی دارد. برای ورودی‌های متعدد، می‌توان از یک درخت جمع‌کننده تصادفی استفاده کرد که خروجی آن در ۰.۵ ضرب می‌شود:

$$p_3 = \frac{p_1 + p_2}{2} \qquad (۱۵)$$

برای اجتناب از مشکلات مقیاس‌بندی روش قبل، یک جمع‌کننده مبتنی بر شمارنده موازی تجمعی [8]در محاسبات تصادفی معرفی شده است (به شکل ۷ (b)مراجعه کنید.)این جمع‌کننده، تعداد ۱ها در توالی‌های ورودی را به‌طور مستقیم جمع می‌کند و احتمال خروجی را از مجموع احتمالات ورودی تعیین می‌کند. این طراحی به تولیدکننده‌های اعداد تصادفی های اضافی نیاز ندارد، که باعث کاهش نیاز به سخت‌افزار و افزایش سرعت پردازش در مقایسه با جمع‌کننده‌های تصادفی سنتی می‌شود . خروجی جمع‌کننده مبتنی بر APC به صورت باینری است و می‌توان با استفاده از شمارنده موازی تقریبی که جمع‌کننده‌های کامل را با گیت‌های منطقی ساده‌تر جایگزین می‌کند، مساحت طراحی را بیشتر کاهش داد، اگرچه این روش ممکن است برای پیاده‌سازی‌های FPGA بهینه نباشد (Y. Lee et al, 2024). علاوه بر این، یک جمع‌کننده تصادفی مبتنی بر فلیپ‌فلاپ تغییر وضعیت نیز پیشنهاد شده است (به شکل ۷ (c) مراجعه کنید) (V.T. Lee et al, 2017). این جمع‌کننده مقداری را بر اساس میانگین احتمالات ورودی تولید می‌کند، مشابه جمع‌کننده تصادفی مرسوم، اما بدون نیاز به توالی‌های تصادفی اضافی برای سیگنال انتخابMUX ، که منجر به کاهش مساحت طراحی می‌شود.

توابع فعال‌ساز در شبکه‌های عصبی معمولاً یا با ماشین‌های حالت متناهی پیاده سازی میشوندو یا با مدارهای حسابی چندجمله‌ای تصادفی(Tina et al, 2024) ، (Y. Liu et al, 2021) مدارهای مبتنی بر مدارهای حسابی چندجمله‌ای آسانتر پیاده سازی می شوند اما نیازمند مساحت بیشتر هستند.

د. تکنیک‌های محاسباتی تقریبی(Ax)

در سیستم‌های دیجیتال، محاسبات تقریبی به عنوان رویکردی نویدبخش برای بهبود عملکرد و کاهش مصرف توان معرفی شده است. این رویکرد به ویژه در کاربردهایی مانند پردازش چندرسانه‌ای و یادگیری ماشین که به میزان مشخصی از خطا تحمل دارند، بسیار کارآمد است. محاسبات تقریبی با تغییر یا حذف بخش‌هایی از عملیات محاسباتی دقیق، منابع سخت‌افزاری را کاهش می‌دهد و به این ترتیب بهره‌وری انرژی را افزایش می‌دهد، در حالی که تنها میزان کمی از دقت فدا می‌شود.جمع‌کننده‌های تقریبی به دو دسته اصلی تقسیم می‌شوند: استاتیک و دینامیک. جمع‌کننده‌های استاتیک دارای سطح ثابتی از تقریب هستند و پارامترهای طراحی مانند تأخیر، توان و مساحت را در مقایسه با جمع‌کننده‌های دقیق کاهش می‌دهند. هر چه میزان تقریب بیشتر باشد، بهینه‌سازی بیشتری

[8] Accumulative parallel counter

حاصل می‌شود. با این حال، از آنجا که سطح تقریب در این جمع‌کننده‌ها ثابت است، ممکن است برای همه برنامه‌ها مناسب نباشند و نیاز به تنظیم اولیه دارند.در مقابل، جمع‌کننده‌های دینامیک قادر به تولید نتایج دقیق یا تقریبی بسته به نیاز هستند، که این ویژگی آن‌ها را برای کاربردهای مختلف بدون نیاز به دانش قبلی از هدف مناسب می‌سازد. این جمع‌کننده‌ها معمولاً از منطق تشخیص و تصحیح خطا استفاده می‌کنند که ممکن است نیاز به چندین سیکل محاسباتی برای دستیابی به دقت مورد نظر داشته باشد، اما در مقابل انعطاف‌پذیری بیشتری فراهم می‌کنند.همچنین، ضرب‌کننده‌های تقریبی نیز برای کاهش مصرف توان و بهبود سرعت در کنار کاهش اندکی دقت مورد توجه قرار گرفته‌اند. تکنیک‌هایی مانند کوتاه کردن و جبران خطا، به طور قابل توجهی مصرف توان را کاهش داده و در عین حال سطح دقت قابل قبولی را حفظ می‌کنند. این تکنیک‌ها به خصوص در برنامه‌هایی که دقت مطلق نیاز ندارند، بسیار مؤثر هستند.

طراحی‌های بهینه‌سازی شده LSTM-NNs :

تلاش‌ها برای بهبود کارایی اجرای LSTM ها را می‌توان به دو رویکرد اصلی تقسیم کرد:

دسته اول طراحی شتاب‌دهنده‌های تخصصی :این شتاب‌دهنده‌ها برای بهبود اجرای LSTM با بهره‌گیری از الگوهای دسترسی به داده و محاسباتی منحصر به فرد مدل طراحی شده‌اند. آن‌ها شامل واحدهای محاسباتی اختصاصی برای عملیات‌های LSTM ، ماژول‌های ارتباطی تخصصی برای جریان کارآمد داده‌ها، و معماری‌های پایپلاین متعادل که محاسبات و ارتباطات را هماهنگ می‌کنند، هستند. همچنین تکنیک‌هایی مانند فشرده سازی و هرس که نوی محاسبات باینری تقریبی ساده شده هستند را به کار می‌گیرند که سبب کاهش اندازه مدل و در نهایت منجر به صرفه‌جویی قابل توجهی در زمان و انرژی می‌شود. این رویکرد LSTM ها را برای استفاده در دستگاه‌هایی با منابع محدود قابل تحقق‌تر می‌کند(S.Li et al, 2019) . (Gong et al, 2019) یک شتاب‌دهنده LSTM با قابلیت پیکربندی مجدد با قابلیت پردازش تقریبی نزدیک به داده را پیشنهاد می‌کند تا بهره‌وری انرژی و پردازش موازی را افزایش دهد. این رویکرد یک نوع پیکربندی مجدد را معرفی می‌کند که با استفاده از یک مدل تقسیم شبکه با دانه‌بندی ترکیبی، بهره‌وری انرژی و پردازش موازی را افزایش می‌دهد. این مدل با بهینه‌سازی پردازش LSTM ، تعادلی بین موازی‌سازی دانه‌ریز و دانه‌درشت ایجاد می‌کند. طراحی این شتاب‌دهنده، مصرف انرژی بالا در وظایف ضرب ماتریس-برداری را با به کارگیری یک ضرب‌کننده لگاریتمی تکراری، که از ضرب‌کننده‌های لگاریتمی Babic الهام گرفته شده است، بهبود می‌بخشد (Babi et al, 2011). این ضرب‌کننده دقت را به صورت پویا بر اساس وزن شبکه تنظیم می‌کند و از یک کنترل‌کننده تکرار و بلوک‌های محاسباتی پایه برای پردازش کارآمد استفاده می‌کند. این معماری دارای یک آرایه محاسباتی 4x8 برای عملیات‌های VMM و یک زنجیره جمع‌کننده 1x8 چندمنظوره قابل پیکربندی مجدد برای اجرای انعطاف‌پذیر سلول LSTM است. پیکربندی این معماری امکان تطبیق با وظایف مختلف را فراهم می‌کند و عملکرد و بهره‌وری انرژی را بهینه می‌سازد. استفاده از ضرب‌کننده‌های تقریبی منجر به کاهش ۵۲.۳ درصدی مصرف انرژی شده، در حالی که دقت شبکه به میزان ۹.۲ درصد کاهش یافته. معماری (Perumal, ,and Rajendiran, 2022) برای مدیریت انرژی شبکه‌های هوشمند طراحی شده است. سلول‌های SRAM سنتی با سلول‌های الحاقی مبتنی بر LUT [9]جایگزین شده‌اند و از ضرب‌کننده‌های بهینه برای کاهش توان و مساحت استفاده شده است. روش‌های تقریبی

[9] Lookup Table

کلیدی به کار گرفته شده شامل:کوتاه‌سازی[10] که با حذف برخی بیت‌ها سبب کاهش اندازه و پیچیدگی مدارهای ضرب‌کننده میشود. هرس کردن بیت[11] که شامل جایگزینی منطق ترتیبی با یک واحد هرس کوچک میشود، که باعث کاهش تعداد فلیپ‌فلاپ‌ها و صرفه‌جویی در انرژی می‌شود.ضرب‌کننده مبتنی برLUT که نتایج ضرب از پیش محاسبه شده در جدولی ذخیره می‌شوند، که نیاز به محاسبات لحظه‌ای را کاهش می‌دهد.قابلیت تطبیق‌پذیری دقت به این صورت که دقت به صورت پویا بر اساس نیازهای برنامه تنظیم میشود، که سبب تعادل بین دقت و مصرف توان را ایجاد می‌کند.این معماری که با استفاده از تکنیک‌های طراحی ASIC پیاده‌سازی شده است،و توانسته به کاهش ۴.۱ درصدی مصرف توان کل و ۵۰ درصدی اندازه LUT با حداقل کاهش دقت دست پیدا کند. (Kouris et al, 2020) یک طرح محاسباتی استنتاجی تدریجی جدید برای شبکه‌های LSTM معرفی می‌کنند که برای ارائه بهترین تقریب ممکن در یک محدوده زمانی خاص طراحی شده است که آن را به‌ویژه برای سیستم‌های حساس به تأخیر مانند خودروهای خودران مناسب می‌کند. این رویکرد با ترکیب هرس مدل و بازسازی محاسبات، باعث می‌شود که شبکه LSTM بتواند با کاهش قابل توجهی در نیازهای محاسباتی و حافظه‌ای، تصمیمات سریع و آگاهانه بگیرد .این روش ماتریس‌های وزن را با استفاده از تجزیه مقدار منفرد (SVD[12]) با رتبه پایین تقریب می‌زند که باعث کاهش بار محاسباتی و حافظه‌ای مورد نیاز برای استنتاج LSTMمی‌شود و سرعت و بهره‌وری را افزایش می‌دهد. ماتریس‌های وزن بر اساس اهمیت عناصرشان که با یک معیار مبتنی بر اندازه تعیین می‌شود، هرس می‌شوند. این کار باعث ایجاد پراکندگی در ماتریس‌های وزن می‌شود و در عین حال اطلاعات حیاتی را حفظ می‌کندSVD یک روش ریاضی است که در آن یک ماتریس به سه ماتریس دیگر تجزیه می‌شود $W = U\Sigma V^T$ که در آن Wماتریس اصلی، U و V ماتریس‌های متعامد هستند و Σ یک ماتریس قطری است که حاوی مقادیر منفرد است. در زمینه LSTMها، ماتریس وزن کامل W با استفاده از تنها بزرگ‌ترین مقدار منفرد و بردارهای منفرد مربوطه تقریب زده می‌شود که به عنوان یک تقریب با رتبه ۱ شناخته می‌شود $\widetilde{W} \approx \sigma_1 U_1 V_1^T$ ، که در آن σ_1 بزرگ‌ترین مقدار منفرد و U_1 و V_1 بردارهای منفرد مربوطه هستند. هرس کردن بار محاسباتی را کاهش می‌دهد و به LSTM اجازه می‌دهد تا تقریب‌های سریع‌تری انجام دهد و در عین حال اطلاعات کلیدی را حفظ کند. سپس، روش ترکیبی SVD و هرس کردن، خروجی را به صورت تکراری تصحیح می‌کند تا دقت را در چندین مرحله بهبود بخشد. این روش به سیستم اجازه می‌دهد تا با گذشت زمان، کیفیت استنتاج را به تدریج افزایش دهد. این روش تعادلی بین سرعت و دقت ایجاد می‌کند و حتی در شرایط محدودیت‌های زمانی شدید، تقریب‌هایی با کیفیت بالا ارائه دهد. روش پیشنهادی تا ۴۱۵ برابر استنتاج سریع‌تری نسبت به پیاده‌سازی‌های سنتی LSTM به دست می‌آورد و میانگین سرعت افزایش ۱۹۸ برابر را نشان می‌دهد.

دسته دوم واحدهای حسابی بهینه‌شده: این رویکرد شامل بررسی فناوری‌های سخت‌افزاری کم‌مساحت مانند محاسبات تصادفی و محاسبات باینری ساده شده است تا کارایی سخت‌افزار را در حالی که دقت را حفظ می کند از نظر مساحت و انرژی بهبود بخشد. (Joseph and Bindiya, 2021) یک تکنیک مبتنی بر مالتی‌پلکسر برای تسریع ضرب ماتریس-بردار در شبکه‌های حافظه طولانی‌مدت و کوتاه‌مدت را ارائه می‌دهند. که با استفاده از روش ضرب ثابت چندگانه و طراحی جدید برای محاسبات داخلی حاصلضرب ، به کاهش قابل توجه ٪۱۴ در مصرف توان دست یافته است. این روش به خوبی ساختار یافته و بر بهینه‌سازی دو جزء

[10] Truncation
[11] Bit-Pruning
[12] Singular Value Decomposition

حیاتی فرآیندMVM [13]در شبکه‌های LSTM تمرکز دارد: تولیدکننده حاصلضرب جزئی (PPG[14]) و انتخاب‌گر حاصلضرب جزئی (PPS). طراحیPPS [15]متعارف شامل انتخاب‌گرهای چندورودی-چندخروجی (MIMO[16]) با تعداد زیادی گیت AND و OR است که منجر به افزایش تأخیر و مصرف توان می‌شود و از منطق ترتیبی و عملیات انتقال به راست برای تقسیم بردارهای ورودی استفاده می‌کند که منجر به تأخیر و مصرف توان بالا می‌شود. طراحی PPS پیشنهادی با جایگزینی طراحی پیچیده MIMO-PPS با یک طراحی ساده مبتنی بر مالتی‌پلکسر، تعداد گیت‌ها را به‌طور قابل توجهی کاهش می‌دهد. هر PPS با استفاده از هفت مالتی‌پلکسر ۲:۱ پیاده‌سازی می‌شود. تعداد کل گیت‌ها از ۱۸۹ گیت AND/OR به ۱۶۰ گیت NAND کاهش می‌یابد. طراحی پیشنهادی یک واحد کوتاه‌کننده را معرفی می‌کند تا هرس بیت را انجام دهد، که نیاز به انتقال به راست را از بین می‌برد و اندازه جمع‌کننده‌ها/تفریق‌کننده‌ها را کاهش می‌دهد. و ۳۲ فلیپ‌فلاپ را با یک واحد کوتاه‌کننده ۱ بیتی ساده جایگزین می‌کند. واحد کوتاه‌کننده تقسیم را با استفاده از هرس ۲ بیتی انجام می‌دهد. بهینه‌سازی‌های ترکیبی در PPG و PPS منجر به یک واحد محاسبات حاصلضرب داخلی(IPC[17]) جمع‌وجورتر و کارآمدتر از نظر مصرف توان می‌شود. (S. Le et al, 2019) یک پیاده‌سازی سخت‌افزاری نوین از شبکه‌های عصبیLSTM با استفاده از محاسبات تصادفی برای پیش‌بینی اطلاعات وضعیت کانال ارائه می‌دهند. معماری LSTMبرای انطباق با اصول محاسبات تصادفی دوباره طراحی شده است. معماری شامل واحدهای پردازش ورودی است که داده‌ها را به جریان‌های بیت تصادفی تبدیل می‌کنند، واحدهای گیت (ورودی، فراموشی و خروجی) که عملیات‌های تصادفی انجام می‌دهند، و واحدهای پردازش خروجی پیش‌بینی‌های نهایی را تولید می‌کنند. معماری پیشنهادی بر روی FPGA Xilinx Artix-7 پیاده‌سازی شده است. معماری پیشنهادی هزینه‌های سخت‌افزاری را تقریباً ۷۰% در مقایسه با روش‌های سنتی کاهش می‌دهد. (Maor et al, 2019) به بررسی ادغام محاسبات تصادفی با شبکه‌های LSTM می‌پردازند و هدف آن کاهش مصرف توان و هزینه‌های سخت‌افزاری در پیاده‌سازی‌های FPGA است. آنها توانسته اند نشان دهند که محاسبات تصادفی می‌تواند به‌طور قابل توجهی مصرف توان را (تا ٪۷۳.۲۴) با کمترین کاهش دقت (٪۱.۶۹) در مقایسه با پیاده‌سازی‌های سنتی LSTM باینری کاهش دهد. این پیاده‌سازی با استفاده از مجموعه داده MNIST برروی FPGA Zedboard ارزیابی شده.(Maor et al, 2019) LSTM را با استفاده از یک معماری چند هسته‌ای پیاده‌سازی کرده اند که هر هسته بخشی از عملیات‌های LSTM را پردازش می‌کند. این رویکرد امکان پردازش موازی را فراهم کرده و به توزیع بار محاسباتی کمک کرده است.نورون‌های داخل LSTM با استفاده از یک رویکرد ترکیبی تصادفی-باینری پیاده‌سازی شده اند. ضرب با استفاده از گیت‌های XNOR انجام می‌شود و جمع از طریق یک شمارنده موازی تقریبی (APC[18]) یا یک مالتی‌پلکسر (MUX) انجام می‌شود. داده‌های ورودی باینری با استفاده از رجیسترهای تصادفی و مولدهای عدد تصادفی به مقادیر تصادفی تبدیل می‌شوند. کاهش دقت گزارش‌شده حداقل است که ٪۱.۶۹ برای تصادفی LSTM-مبتنی برMUX است ،اما مقاله به‌طور عمیق عواملی که می‌توانند بر این کاهش تأثیر بگذارند، مانند مجموعه داده‌های بزرگ‌تر، انواع مختلف وظایفLSTM ، یا پردازش دنباله‌های طولانی‌تر، را تحلیل نمی‌کند. طراحی تصادفی LSTM-زمان اجرای را به‌طور قابل توجهی افزایش می‌دهد، که این یک ضعف جدی برای برنامه‌های کاربردی بلادرنگ است. (Y. Liu et al, 2019) یک طراحی ترکیبی را

[13] Matrix Vector Multiplication
[14] Partial product Generator
[15] Partial Product Selector
[16] Multi In Multi Out
[17] Iner Product computation
[18] Approximate Parallel Counter

معرفی می‌کند که محاسبات تصادفی را با مدارهای باینری برای پیاده‌سازیLSTM ادغام می‌کند. این طراحی با استفاده از مجموعه داده‌های مختلف ارزیابی شده و بهبودهایی در حدود ٪۱.۶ تا ٪۲.۳ در مساحت و ٪۶.۵ تا ٪۱۱.۲ در مصرف انرژی نسبت به پیاده‌سازی ۳۲ بیتی با نقطه شناور گزارش داده است. در عین حال دقتش قابل مقایسه با آن است. اجزای تصادفی عملیات‌هایی را که کارایی سخت‌افزاری در آن‌ها حیاتی است، مانند فعال‌سازی گیت‌ها را مدیریت می‌کنند، در حالی که مدارهای باینری عملیات‌های پیچیده‌تری را که نیاز به دقت بالاتری دارند، مانند به‌روزرسانی حالت‌ها و خروجی‌های نهایی را مدیریت می‌کنند. نوآوری کلیدی استفاده از مدارهای تصادفی تقریبی برای واحدهای گیت است که هزینه‌های سخت‌افزاری را کاهش و در عین حال دقت را حفظ می‌کند. با بهره‌گیری از محاسبات تصادفی، طراحی پیشنهادی بهبود در تحمل نویز را به دست آورده و دقت بالاتری را تحت شرایط نویزی در مقایسه با پیاده‌سازی‌های باینری حفظ کرده است.(Sengupta, et al, 2022) بر ایجاد توازن میان کاهش مساحت و مصرف انرژی و احتمال افت دقت در طبقه‌بندی تمرکز کرده اند، همچنین طرح آنها مبتنی بر طراحی (Y. Liu et al, 2019) است. این مطالعه به بررسی مقایسه معماری‌های کاملاً تصادفی، کاملاً باینری و ترکیبی تصادفی-باینری LSTM با استفاده از معیارهای طبقه‌بندی استاندارد مانندMNIST ، TIMIT، و IMDB می‌پردازد. نتایج کاهش قابل توجهی در مساحت (تا ٪۴۷) و مصرف انرژی (تا ٪۸۶) را با تأثیرات جزئی بر دقت نشان می‌دهد و قابلیت استفاده از تصادفی برای سیستم‌های با منابع محدود را اثبات می‌کند. مقاله به بررسی عملکرد توابع فعال‌سازی tanh و ReLU در داخل سلول‌های تصادفی LSTM-می‌پردازد مه نشان داده شده است که ReLUدر محیط‌های تصادفی مؤثرتر است و احتمال ناپدید شدن گرادیان را کاهش می‌دهد و دقت کلی را در معماری‌های LSTM تصادفی بهبود می‌بخشد. مقاله همچنین به تعادل بین طول رشته تصادفی (SN[19]) و دقت اشاره می‌کند و تأکید دارد کهSN های بلندتر (۵۱۲ بیت) دقت بهتری ارائه می‌دهند، اما با هزینه‌ی تأخیر بیشتر همراه هستند. بهبود حدود ٪۳ با افزایش طولSN از ۲۵۶ بیت به ۵۱۲ بیت و ٪۰.۵ بهبود اضافی با افزایش از ۵۱۲ بیت به ۱۰۲۴ بیت گزارش شده است.

نتیجه‌گیری:

با افزایش تقاضا برای شبکه‌های عصبی کارآمد و قدرتمند، به‌ویژه در محیط‌های محدود به منابع مانند اینترنت اشیا و محاسبات موبایل، پیاده‌سازی سخت‌افزاری شبکه‌های LSTM چالش‌ها و فرصت‌های قابل توجهی را به همراه دارد. این مقاله به بررسی پیشرفت‌های اخیر در بهینه‌سازی سخت‌افزار LSTM از طریق تکنیک‌های محاسبات تصادفی و تقریبی پرداخته است، و بر پتانسیل آن‌ها برای کاهش چشمگیر مصرف توان و پیچیدگی سخت‌افزار در حالی که سطوح قابل قبولی از دقت را حفظ می‌کنند، تأکید می‌کند. مرور روش‌های اخیر، از جمله تأثیر الگوریتم‌های فشرده‌سازی و هرس، بهینه‌سازی‌های ضرب ماتریس-بردار، محاسبات تصادفی و ضرب‌کننده‌های کم‌مصرف، اهمیت یک رویکرد متعادل را برجسته می‌کند که به‌طور دقیق موازنه بین کارایی محاسباتی و دقت است .این نوآوری‌ها نه تنها برای گسترش دامنه برنامه‌های یادگیری عمیق به حوزه‌های جدید حیاتی هستند، بلکه برای اطمینان از اینکه این برنامه‌ها می‌توانند به‌طور مؤثر در محدودیت‌های سخت‌افزاری مدرن عمل کنند نیز ضروری هستند. با نگاهی به آینده، تکامل مداوم این تکنیک‌ها همراه با پیشرفت‌های طراحی و ساخت سخت‌افزار، نوید بهبود عملکرد و افزایش دسترسی به شبکه‌هایLSTM را می‌دهد. با بهبود و اصلاح این رویکردها توسط پژوهشگران و مهندسان، ادغام یادگیری ماشین در دستگاه‌های روزمره

[19] Stochastic Number

روان‌تر شده و راه را برای فناوری‌های هوشمندتر و پاسخگوتر که می‌توانند به طور کارآمدتر در لبه شبکه عمل کنند، هموار کند. این پیشرفت برای بهره‌برداری کامل از پتانسیل هوش مصنوعی در دنیای به‌طور فزاینده‌ای متصل بسیار حیاتی خواهد بود.

منابع

J. Mao and A. K. Jain. (1995) Artificial neural networks for feature extraction and multivariate data projection, IEEE Transaction on Neural Network, vol. 6, no. 2, pp. 296–317.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. (2012) ImageNet classification with deep convolutional neural networks, Advances in Neural Information Processing System, vol. 1, pp. 1097–1105.

R. Fierro and F. L. Lewis, (1998) Control of a nonholonomic mobile robot using neural networks, IEEE Transaction on Neural Network, vol. 9, no. 4, pp. 589–600.

G. P. Zhang, (2000), Neural networks for classification: A survey, IEEE Transactions on Systems, Man, and Cybernetics, Part C, Application and review. Rev., vol. 30, no. 4, pp. 451–462.

J. J. Hopfield, (1982), Neural networks and physical systems with emergent collective computational abilities, Proc. Nat. Acad. تصادفی i. USA, vol.79, no. 8, pp. 2554–2558.

S. Haykin, (2009), Neural networks and learning machines, Upper Saddle River, NJ, USA: Pearson, vol.3.

H. Hikawa, (2003), A digital hardware pulse-mode neuron with piecewise linear activation function, IEEE Transaction on Neural Network, vol. 14, no. 5, pp. 1028-1037.

J. تصادفی hmidhuber, (2015), Deep learning in neural networks: An overview, IEEE Transaction on Neural Network, vol. 61, pp. 85-117.

A. Graves, A. Mohamed and G. Hinton, (2013), Speech recognition with deep recurrent neural networks, IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, pp. 6645-6649.

T. Masoudi, H. Zhang, A. Alagarsamy, J. Han, and S. Bum Ko, (2024), Stochastic and Approximate Computing for Deep Learning: A Survey. In: W. Liu, J. Han, and F. Lombardi, (eds) Design and Applications of Emerging Computer Systems. Springer, Cham.

Y. Liu, S. Liu, Y. Wang, F. Lombardi and J. Han, "A Survey of Stochastic Computing Neural Networks for Machine Learning Applications," in IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 7, pp. 2809-2824.

S. Liu et al, (2024), From Multipliers to Integrators: A Survey of Stochastic Computing Primitives, in IEEE Transactions on Nanotechnology, vol. 23, pp. 238-249.

S. Han et al, (2017), "ESE: Efficient speech recognition engine with sparse LSTM on FPGA," in Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays, pp. 75–84.

Z. Li, S. Wang, C. Ding, Q. Qiu, Y. Wang, and Y. Liang, (2018), Efficient recurrent neural networks using structured matrices in FPGAs, arXiv:1803.07661, [Online]. Available: http://arxiv.org/abs/1803.07661v2.

G. Maor, X. Zeng, Z. Wang and Y. Hu, (2019), An FPGA Implementation of Stochastic Computing-Based LSTM, IEEE 2019 37th International Conference on Computer Design (ICCD), Abu Dhabi, United Arab Emirates, pp. 38-46.

S. Li, Q. Wang, X. Liu and J. Chen, (2018), Low-Cost LSTM Implementation based on Stochastic Computing for Channel State Information Prediction, 2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Chengdu, China, pp. 231-234.

Y. Liu, L. Liu, F. Lombardi and J. Han, (2019), An energy-efficient and noise-tolerant recurrent neural network using stochastic computing," in IEEE Transactions on Very Large-تصادفی ale Integration (VLSI) Systems, vol. 27, no. 9, pp. 2213-2221.

R. Sengupta, I. Polian and J. P. Hayes, (2022), Stochastic Computing Architectures for Lightweight LSTM Neural Networks," (2022) 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Prague, Czech Republic, pp. 124-129.

Lee. Y, Halim.ZA, Wahab. MNA, Almohamad. TA, (2024), Stochastic Computing Convolutional Neural Network Architecture Reinvented for Highly Efficient Artificial Intelligence Work load on Field-Programmable Gate Array. Research (Wash D C), [Online] Available: https://spj.تصادفی ience.org/doi/10.34133/research.0307.

Y. Gong, B. Liu, W. Ge and L. Shi, (2019), RNA: Reconfigurable LSTM Accelerator with Near Data Approximate Processing, (2019) International Conference on Field-Programmable Technology (ICFPT), Tianjin, China, pp. 311-314.

Perumal, S., Rajendiran, S. (2022), Low power multiplier based long short-term memory hardware architecture for smart grid energy management, International Journal of System Assurance Engineering and Management, vol. 13, pp. 2531–2539.

A. Kouris, S. I. Venieris, M. Rizakis, and C. Bouganis, (2020), Approximate LSTMs for time-constrained inference: enabling fast reaction in self-driving cars," IEEE Consumer Electronics Magazine, vol. 9, no. 4, pp. 11-26, ,2020.

T. Joseph and T. S. Bindiya, (2021) High Speed and Power Efficient Multiplexer based Matrix Vector Multiplication for LSTM Network, (2021) 25th International Symposium on VLSI Design and Test (VDAT), Surat, India, pp. 1-4.

S. Sen and A. Raghunathan, (2018), Approximate Computing for Long Short-Term Memory (LSTM) Neural Networks, in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 11, pp. 2266-2276.

G. Rajput, K. N. Biyani, V. Logashree, and S. K. Vishvakarma, (2022), تصادفی AN: streamlined composite activation function unit for deep neural accelerators," Circuits Syst Signal Process, vol. 41, no. 6, pp. 3465–3486.

J. Li et al, (2017), Towards acceleration of deep convolutional neural networks using stochastic computing," in Proc. 22nd Asia South Pacific Design Automat. Conf. (ASP-DAC), pp. 115-120.

V. T. Lee, A. Alaghi, J. P. Hayes, V. Sathe, and L. Ceze, (2017), Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing, in Proc. Conf. Design, Automat. Test Eur., pp. 13-18.

K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, and K. Choi, (2016), Dynamic energy accuracy trade-off using stochastic computing in deep neural networks, in Proc. 53rd Annu. Design Automat. Conf. (DAC), pp. 1-6, 2016.

Y. Liu, Y. Wang, F. Lombardi, and J. Han, "An energy-efficient stochastic computational deep belief network," in Proc. Design, Automat. Test Eur. Conf., pp. 1175–1178.

Z. Li et al., "HEIF: Highly efficient stochastic computing-based infer Ence framework for deep neural networks," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 38, no. 8, pp. 1543–1556, 2019.

B. Li, Y. Qin, B. Yuan, and D. J. Lilja, (2019), Neural network classifiers using a hardware-based approximate activation function with a hybrid stochastic multiplier," ACM J. Emerg. Technol. Comput. Syst., vol. 15, no. 1, pp. 1-28.

G. Erna, G. Srihari, M. Purna Kishore, Ashok Nayak B., M. Bharathi, (2023), FPGA implementation of high-performance truncated rounding based approximate multiplier with high-level synchronous XOR-MUX Full Adder, WSEAS Transactions on Circuits and Systems, vol. 22, pp. 111-125.

Z. Babi, A. Avramovi, and P. Buli, (2011), An iterative logarithmic multiplier, Microprocessors and Microsystems, vol. 35, no. 1, pp. 23-33.

English Abstract:

Neural networks (NNs), modeled after the human brain, are extensively used in artificial intelligence and machine learning for tasks like feature extraction, classification, and system control, owing to their nonlinear properties and self-adaptive nature. This paper concentrates on the hardware implementation of Long Short-Term Memory (LSTM) neural networks, a specialized form of recurrent neural networks (RNNs) known for their effectiveness in handling sequential data. The significant computational demands of LSTM networks present considerable challenges for hardware implementation, especially in resource-limited environments such as the Internet of Things (IoT). This paper reviews recent advancements in hardware design for LSTM networks, with a focus on strategies that utilize imprecise computing to mitigate hardware complexity and power consumption, it explores various optimization techniques, including quantization and pruning algorithms, assessing their potential to improve LSTM processing efficiency. Additionally, the paper examines the trade-offs between accuracy and resource utilization in the context of these emerging hardware methods.