سامانه یادگیری ماشین توزیع شده (زمستان ۱۴۰۱ – دکتر دوستی)

پروژه‌ی بررسی کارایی مدل‌های توزیع شده

نگارنده: پدرام رستمی

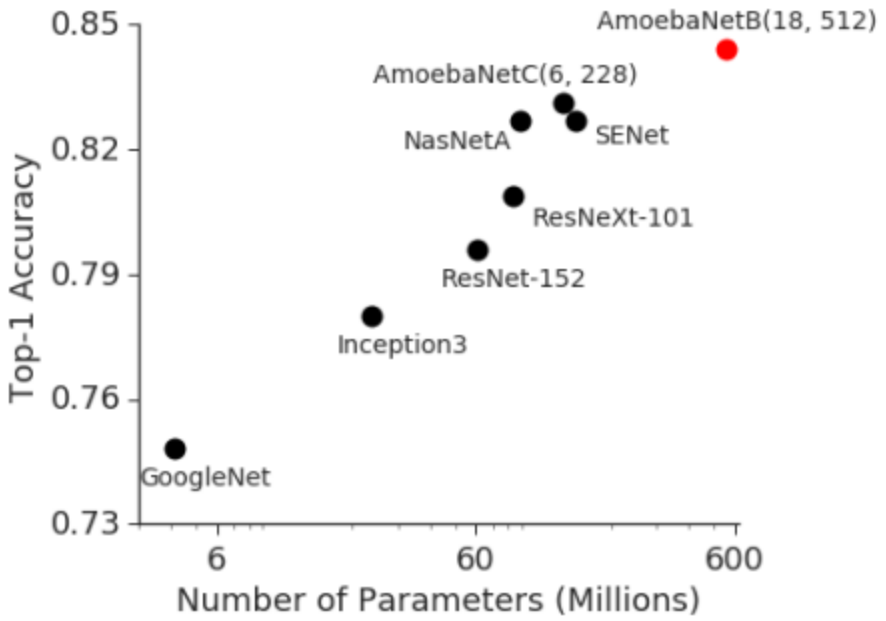
---

۱. چکیده

در سال ۲۰۱۲ شرکت گوگل با معرفی فریمورک DistBelief گام بزرگی در راستای آموزش مدل‌های مقیاس بزرگ توزیع شده برداشت. چارچوب DistBelief با بهره‌مندی از تکنیک‌های DownpourSGD و Sandblaster L-BFGS توانست مدل‌هایی در ابعاد ۱.۷ میلیارد پارامتر را بر روی ده‌ها هزار هسته‌ی CPU آموزش دهد. در حالی که در سال ۲۰۱۲، مدلی در ابعاد ۱.۷ میلیارد پارامتر نزدیک به ۳۰ برابر بزرگتر از مدل‌های گزارش شده‌ی بزرگ دیگر بود، امروزه مدل‌هایی با ابعاد ۱.۷ تریلیارد پارامتر (مدل WuDao) هم به کمک چارچوب‌های یادگیری ماشین توزیع شده آموزش می‌بینند. مدل‌هایی که چند صد یا حتی هزار برابر نسبت به بزرگترین مدلی که DistBelief قادر به آموزش آن بود، بزرگترند [1]. در این پروژه به بررسی ابزارها و ایده‌هایی خواهیم پرداخت که به ما قدرت آموزش چنین مدل‌های بزرگی را دادند. ابزارها و ایده‌هایی که افزایش سرعت نزدیک به خطی را با افزایش منابع حتی در ابعاد بسیار بزرگ به وجود آوردند.

۲. مقدمه

در سال‌های اخیر، یادگیری عمیق پیشرفت چشمگیری داشته است و بخش بزرگی از این پیشرفت مدیون مقیاس پذیری بسیار بالای مدل‌های آن بوده است. طبق تحقیقات مختلف، نشان داده شده است که با بزرگ شدن اندازه‌ی مدل‌ها، دقت و عملکرد آن‌ها نیز افزایش یافته است. در شکل ۱، نموداری از دقت مدل‌های مختلف بر اساس تعداد پارامترهایشان برای مسئله‌ی دسته بندی مجموعه داده‌ی ImageNet رسم شده است. در این نمودار مشخص است که با افزایش تعداد پارامترهای مدل، دقت حل این مسئله افزایش می‌یابد. این اتفاق تنها مختص به مسائل حوزه‌ی تصویر نبوده و در مورد حوزه‌های دیگر مانند پردازش زبان طبیعی هم صدق می‌کند [2].



شکل ۱- دقت top-1 مدل‌های مختلف برای مسئله‌ی دسته بندی مجموعه داده‌ی ImageNet نسبت به تعداد پارامترهایشان

در حالی که مدل‌های بزرگتر می‌توانند دقت و عملکرد بهتری نسبت به رقبای کوچک‌ترشان داشته باشند، فرآیند آموزش آن‌ها با چالش‌های فراوانی همراه است. محدودیت‌های سخت افزاری مانند محدودیت در حافظه یا توان محاسباتی در نهایت محققان را مجبور به تقسیم و آموزش مدلشان بر روی چندین واحد پردازش گرافیکی (GPU) یا تنسوری(TPU) می‌کند. مهم‌ترین چالش این مسئله، موازی‌سازی مناسب مدل است به طوری که آموزش مدل همچنان بهینه باشد [2].

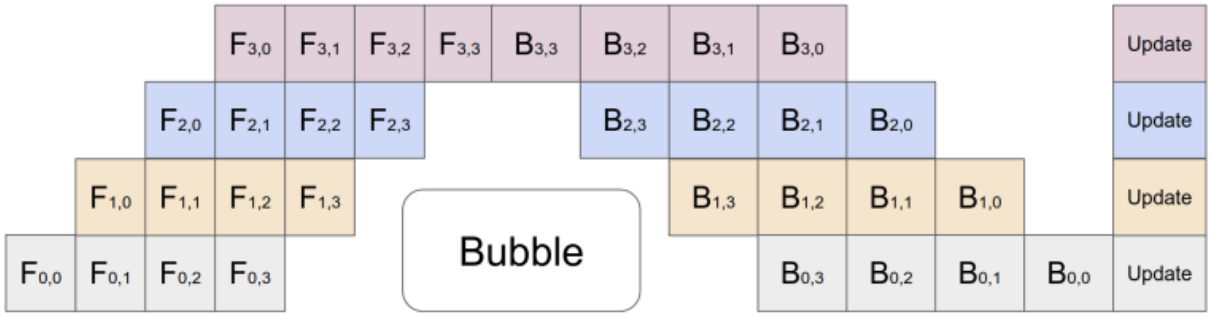
برای حل بهینگی موازی‌سازی مدل‌های مقیاس بزرگ و حل مسئله‌ی آموزش آن‌ها، چارچوب‌ها و ایده‌های فراوانی مطرح شده‌اند ولی بسیاری از آن‌ها به دلیل در اختیار نداشتن منابع پردازشی فراوان، نتایجی برای بهینگی عملکرد ایده‌ها و محصولاتشان در مقیاس بزرگ نداشتند. تنها سازمان‌ها و شرکت‌هایی که قادر به فراهم کردن منابع پردازشی فراوان جهت آزمایش ایده‌هایشان بودند، شرکت‌های بزرگ حوزه‌ی تکنولوژی مانند گوگل، ماکروسافت، انویدیا و غیره بودند. به همین علت، تقریبا تمامی کتابخانه‌ها و چارچوب‌هایی که برای حل این مشکل در سال‌های اخیر طراحی شده، توسط این شرکت‌ها انجام شده است. در ادامه‌ی این پروژه به بررسی راه حل‌های مطرح شده توسط این شرکت‌ها پرداخته می‌شود.

۳. راه حل‌ها

۱.۳. GPipe

در سال ۲۰۱۹، شرکت گوگل کتابخانه‌ی GPipe را برای بهبود موازی سازی مدل‌های یادگیری ماشین ارائه داد. این کتابخانه شامل راه حل‌هایی بود که در زیر به آن‌ها به صورت مختصر اشاره می‌شود [2].

- این کتابخانه با گرفتن تعداد partition ها، تعداد لایه‌های شبکه و تابع هزینه‌ای که هزینه‌ی محاسباتی هر لایه را مشخص می‌کند، لایه‌های مدل را طوری به partition ها تقسیم می‌کند که هم ارتباطات کمینه باشد و هم لود محاسباتی partition ها تقریبا مساوی باشند [2].
- در GPipe هر mini batch به M قسمت مساوی تقسیم می‌شود (micro-batch) و به ترتیب به partition ها داده می‌شوند. این کار کمک می‌کند تا زمانی که برای مثال micro-batch دوم وارد partition اول می‌شود، partition دوم شروع به پردازش بر روی micro-batch اول کند. این تکنیک هم در forward path و هم در backward path باعث می‌شود تا partition ها همزمان بر روی یک mini batch کار کرده و محاسبات آن را سریع‌تر انجام دهند. در شکل ۲ این تکنیک قابل مشاهده است [2].



شکل ۲- تقسیم هر mini batch به M قسمت و اجرای موازی آن‌ها بر روی partition ها

- همچنین این کتابخانه برای کاهش حداکثر حافظه‌ی مصرفی، در هر لایه تنها خروجی توابع فعالساز را نگه می‌دارد و در backward path متغیرهای اضافی را دوباره محاسبه می‌کند [2].

نتایج بهینه سازی‌های حافظه‌ی مصرفی GPipe در جدول ۱ قابل مشاهده است. مدل AmoebaNet برای دسته بندی عکس‌های ImageNet و مدل Transformer برای ترجمه‌ی ماشینی چندزبانه است. در هر آزمایش مدل‌ها تا جای ممکن بزرگ در نظر گرفته شده‌اند. همانطور که قابل مشاهده است، GPipe می‌تواند زمانی که حتی بر روی یک ماشین اجرا می‌شود، مدل‌هایی ۳ تا ۴ برابر بزرگتر از حالت عادی آموزش دهد. برای

مدل AmoebaNet این کتابخانه توانسته است بر روی ۸ GPU مدلی با ۱.۸ میلیارد پارامتر آموزش دهد و برای مدل Transformer این کتابخانه توانسته است بر روی ۱۲۸ TPU مدلی با ۸۳.۹ میلیارد پارامتر آموزش دهد [2].

جدول 1 - نتایج بهینه سازی استفاده از حافظه‌ی کتابخانه‌ی GPipe

NVIDIA GPUs (8GB each)	Naive-1	Pipeline-1	Pipeline-2	Pipeline-4	Pipeline-8
AmoebaNet-D (L, D)	(18, 208)	(18, 416)	(18, 544)	(36, 544)	(72, 512)
# of Model Parameters	82M	318M	542M	1.05B	1.8B
Total Model Parameter Memory	1.05GB	3.8GB	6.45GB	12.53GB	24.62GB
Peak Activation Memory	6.26GB	3.46GB	8.11GB	15.21GB	26.24GB
Cloud TPUv3 (16GB each)	**Naive-1**	**Pipeline-1**	**Pipeline-8**	**Pipeline-32**	**Pipeline-128**
Transformer-L	3	13	103	415	1663
# of Model Parameters	282.2M	785.8M	5.3B	21.0B	83.9B
Total Model Parameter Memory	11.7G	8.8G	59.5G	235.1G	937.9G
Peak Activation Memory	3.15G	6.4G	50.9G	199.9G	796.1G

همچنین در جدول ۲ افزایش سرعت آموزش مدل‌ها توسط GPipe برای تعداد partition های مختلف (K) و تعداد micro-batch های مختلف (M) قابل مشاهده است. طبق نتایج این جدول، زمانی که M=32 فرض شده باشد، افزایش سرعت مدل Transformer تقریبا به حالت خطی در می‌آید. در حالی که برای مدل AmoebaNet افزایش سرعت کمتر از حالت خطی می‌شود [2].

جدول 2 - افزایش سرعت آموزش مدل‌ها توسط GPipe در حالت‌های مختلف

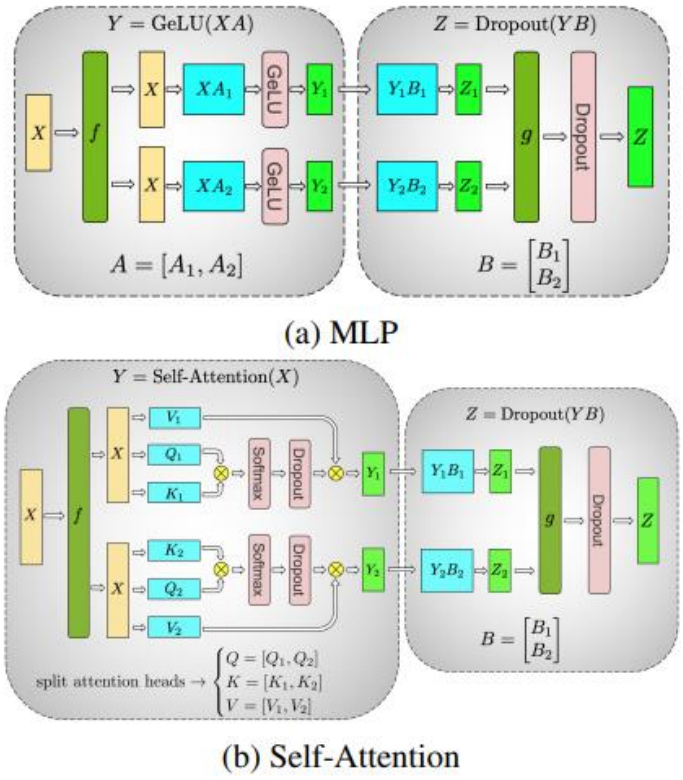
TPU	AmoebaNet			Transformer		
$K =$	2	4	8	2	4	8
$M = 1$	1	1.13	1.38	1	1.07	1.3
$M = 4$	1.07	1.26	1.72	1.7	3.2	4.8
$M = 32$	1.21	1.84	3.48	1.8	3.4	6.3

۲.۳. Megatron-LM

در سال ۲۰۱۹، شرکت انویدیا افزونه[1]ی Megatron-LM را برای کتابخانه‌ی پایتورچ ارائه داد. این افزونه به صورت تخصصی حاوی تکنیک‌هایی برای موازی سازی مدل‌های ترنسفورمر بسیار بزرگ بود.

[1] Extension

Megatron-LM به صورت خاص بر روی موازی سازی بهینه‌ی ماژول ترنسفورمر که امروزه یکی از پر استفاده‌ترین ماژول‌های هوش مصنوعی است، کار کرده است. ماژول ترنسفورمر شامل دو بخش self-attention و MLP است. برای بهینه سازی بخش MLP، Megatron-LM پیشنهاد می‌کند که ماتریس وزن‌های شبکه به صورت ستونی تقسیم شده و بین GPU ها پخش شوند. با این کار GPU ها می‌توانند به صورت مستقل نتایج را محاسبه کرده و در انتهای بخش MLP با تنها یک عملیات AllReduce نتایج کلی نهایی را تولید کنند. در بخش self-attention هم با تقسیم ستونی پارامترهای V، Q و K می‌توان آن‌ها را بین GPU ها تقسیم کرده و مانند بخش MLP هر GPU نتایج را مستقلا حساب کرده و در نهایت با یک عملیات AllReduce نتایج نهایی را تولید کنند. این ایده‌ها کمک می‌کنند تا برای موازی سازی ماژول ترنسفورمر بین GPU های مختلف، تنها نیاز به دو عملیات AllReduce در forward path و دو AllReduce در backward path باشد. در شکل ۳ این تقسیم بندی قابل مشاهده است [3].

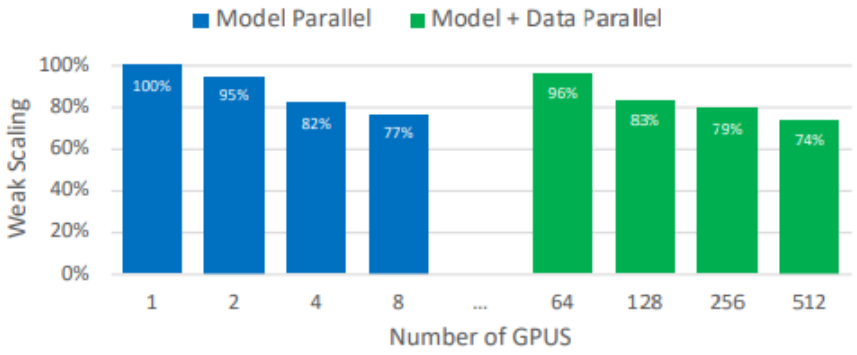


شکل ۳- نحوه‌ی تقسیم بخش‌های MLP و Self-Attention در ماژول ترنسفورمر

در شکل ۴ نتایج آزمایشات بر روی تعداد مختلف GPU نشان داده شده است. در این آزمایشات سعی شده است تا افزایش تعداد پارامترهای مدل به صورت خطی با تعداد GPU ها افزایش یابد تا پردازش هر GPU تقریبا ثابت بماند. سپس تعداد عملیات‌های اعشاری انجام شده در واحد زمان بر روی یک GPU در حالت‌های مختلف به

دست آمده و نمودار آن کشیده شده است. طبیعتا با افزایش تعداد GPU ها، سربارهایی مانند ارتباطات و غیره افزوده می‌شود که باعث کاهش تعداد عملیات انجام شده می‌شود. در حالت موازی سازی همزمان داده‌ها و مدل، داده‌ها به ۶۴ بخش تقسیم شده است. در نتیجه در ترکیب با موازی سازی مدل، با ۶۴، ۱۲۸، ۲۵۶ و ۵۱۲ GPU تست شده اند. همانطور که مشخص است، این تکنیک توانسته است با کاهش سربار ارتباطات، با افزایش GPU ها عملکرد خوبی همچنان داشته باشد [3].

Hidden Size	Attention heads	Number of layers	Number of parameters (billions)	Model parallel GPUs	Model +data parallel GPUs
1536	16	40	1.2	1	64
1920	20	54	2.5	2	128
2304	24	64	4.2	4	256
3072	32	72	8.3	8	512



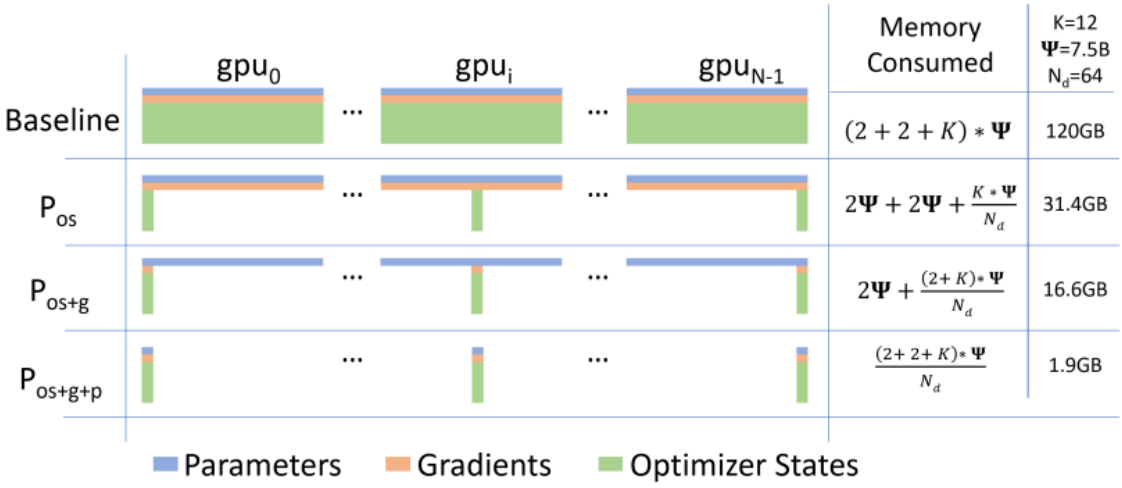
شکل ۴- نتایج آزمایشات انجام شده بر روی Megatron-LM

۳.۳. DeepSpeed

در سال ۲۰۱۹ ماکروسافت مقاله‌ی ZeRO را منتشر کرد و در سال بعد، ایده‌های آن را در کتابخانه‌ی DeepSpeed که مبتنی بر پایتورچ است، پیاده سازی کرد.

در مقاله‌ی ZeRO تکنیک‌های مختلفی برای بهینه سازی حافظه‌ی مصرفی در حین آموزش مدل، ارائه شده است. آموزش مدل در حالت موازی سازی داده‌ها، افزونگی داده‌ی (data redundancy) فراوانی دارد زیرا optimizer state های مدل به همراه گرادیان‌ها و پارامترها در تمام ماشین‌ها در حافظه قرار می‌گیرند. در مقاله‌ی ZeRO پیشنهاد شده است که هر ۳ مورد ذکر شده بین partition ها تقسیم شده و هر partition تنها قسمت مربوط به خود را بگیرد. در شکل ۵ تاثیر کاهش حافظه در صورت partition کردن هر کدام از موارد

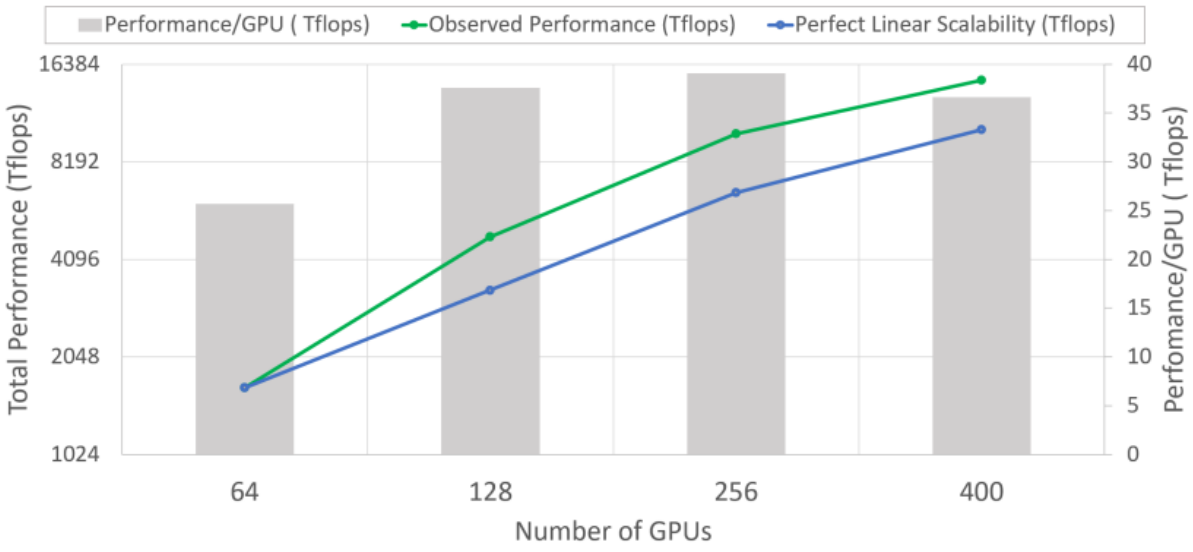
گفته شده بر روی حافظه‌ی هر دستگاه آورده شده است. در این حالت تعداد پارامترهای مدل ۷.۵ میلیارد و موازی سازی داده بین ۶۴ دستگاه فرض شده است [4].



شکل ۵- تاثیر partition کردن پارامترها، گرادیان‌ها و optimizer state ها بر خافظه‌ی مصرفی در هر دستگاه

در آموزش مدل‌های یادگیری عمیق، همواره بخشی از حافظه به خروجی‌های توابع فعالساز و متغیرهای موقت اختصاص داده می‌شود. همچنین همیشه بخشی از حافظه به دلیل memory fragmentation بی استفاده باقی می‌ماند. در واقع حافظه‌های ذکر شده هم بخش‌هایی هستند که هدر می‌روند. ZeRO سعی کرده است حافظه‌ی هدر رفته توسط این بخش‌ها را هم کمتر کند [4].

همچنین برای بررسی عملکرد ZeRO، مدلی با ۶۰ میلیارد پارامتر بین تعداد مختلف GPU آموزش داده شده است. نمودار عملکرد ZeRO هم در حالت کلی و هم برای هر دستگاه در شکل ۶ رسم شده است. همانطور که در نمودار مشخص است، عملکرد کلی ZeRO حتی از عملکرد خطی هم بهتر است [4].



شکل ۶- بررسی عملکرد ZeRO برای آموزش مدلی با ۶۰ میلیارد پارامتر در تعداد GPU های مختلف

۴.۳. سایر

به جز کتابخانه‌های GPipe، Megatron-LM و DeepSpeed در سال‌های اخیر کتابخانه‌های دیگری نیز توسعه داده شده‌اند. در سال ۲۰۲۲ متا کتابخانه‌ی fairScale را به عنوان افزونه‌ای برای کتابخانه‌ی پایتورچ منتشر کرد. همچنین در همین سال Hugging Face کتابخانه‌ی Accelerate را برای آموزش مدل‌های توزیع شده منتشر کرد. به کمک این کتابخانه می‌توان از کتابخانه‌های DeepSpeed و Megatron-LM هم استفاده کرد [6,5].

۴. نتیجه گیری

در سال‌های اخیر، مدل‌های یادگیری عمیق بسیار بزرگ شده‌اند و فرایند آموزش آن‌ها تبدیل به چالش سختی شده است. بسیاری از سیستم‌های یادگیری ماشین توزیع شده بهره وری مناسبی برای استفاده در این مقیاس را ندارند و بسیاری از شرکت‌ها تنها از کتابخانه‌هایی که قوی‌تر هستند و پذیرش بیشتری دارند، استفاده می‌کنند مانند پایتورچ یا تنسورفلو. در نتیجه بسیاری از ایده‌هایی که برای بهبود آموزش مدل‌های در مقیاس بزرگ مطرح شده اند، به عنوان افزونه‌هایی برای این کتابخانه‌ها عرضه شده‌اند. همچنین به دلیل نیاز به انجام تست‌هایی با تعداد GPU یا TPU های بالا، تنها شرکت‌های بزرگ حوزه‌ی تکنولوژی قادر به اجرای این کار بودند.

در سال ۲۰۱۹ شرکت‌های گوگل و انویدیا کتابخانه‌های GPipe و Megatron-LM را برای آموزش مدل‌های مقیاس بزرگ منتشر کردند. هر چند کتابخانه‌ی GPipe به دلیل ارائه بر روی تنسورفلو از استقبال کمتری برخوردار شد. در سال ۲۰۲۰ ماکروسافت کتابخانه‌ی DeepSpeed را منتشر کرد. در حال حاضر این کتابخانه

بزرگترین جامعه‌ی کاربران را نسبت به تمام رقبیان دارد. در سال ۲۰۲۲ هم شرکت‌های متا و Hugging Face کتابخانه‌های fairScale و Accelerate را منتشر کردند. هر دو کتابخانه‌های fairScale و Accelerate دارای مستندات بسیار خوب و کاملی هستند. همچنین کتابخانه‌ی Accelerate از آموزش به روش کتابخانه‌های DeepSpeed و Megatron-LM هم پشتیبانی می‌کند.

۵. مراجع

[1] Dean, Jeffrey, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato et al. "Large scale distributed deep networks." *Advances in neural information processing systems* 25 (2012).

[2] Huang, Yanping, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Yonghui Wu. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." *Advances in neural information processing systems* 32 (2019).

[3] Shoeybi, Mohammad, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. "Megatron-lm: Training multi-billion parameter language models using model parallelism." *arXiv preprint arXiv:1909.08053* (2019).

[4] Rajbhandari, Samyam, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. "Zero: Memory optimizations toward training trillion parameter models." In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1-16. IEEE, 2020.

[5] https://github.com/facebookresearch/fairscale

[6] https://huggingface.co/docs/accelerate