بررسی شتابدهنده های مبتنی بر GPU برای شبکه عصبی ترنسفورمر

دانشجو: محمد فاتح

استاد راهنما: دکتر خونساری

دانشکده مهندسی برق و کامپیوتر، دانشگاه تهران

مقدمه / خلاصه

شبکه عصبی انواع گوناگونی دارد. و یکی این انواع، **ترنسفورمر**[1] نام دارد. این مدل در سال ۲۰۱۷ توسط گوگل ابداع شد[1] و در همه زمینه ها پیشرفت چشمگیری را رقم زد. ناگفته نماند که ChatGPT[2] نیز بر پایه مدل ترنسفورمر ساخته شده است.

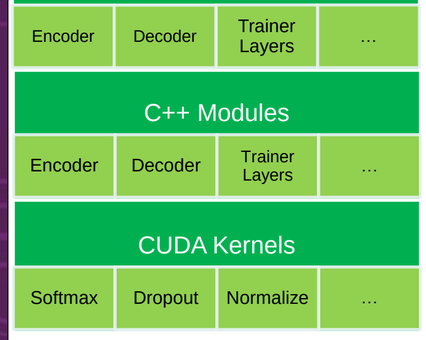
اما استفاده از مدل ترنسفورمر در صنعت با یک چالش اساسی رو به روست. وآن حجم سنگین محاسبات آن است. مثلا یادگیری مدل GPT-3 با استفاده از تنها یک GPU، چیزی حدود ۱.۳میلیون ساعت زمان نیاز دارد![2] بنابراین استفاده از این مدل در صنعت، بدون شتاب دهنده سخت افزاری موثر نخواهد بود.

ما در این پروژه چند مورد از شتابدهنده های مهم مدل ترنسفورمر که مبتنی بر GPU هستند را بررسی کرده ایم. و با مطالعه مستندات و کد منبع آن ها، با ایده های تسریع سازی آشنا شده ایم. همچنین هر کدام از این شتابدهنده ها را راه اندازی کرده و عملکردشان را سنجیده ایم. نتایج نشان میدهد که استفاده از این شتابدهنده ها میتواند مدل را به صورت تخمینی ۲۵٪ سریع تر کند. [2]

1: Transformer Model, 2: GPT: Generative Pre-trained Transformer

روش/ساختار/مدل پیشنهادی

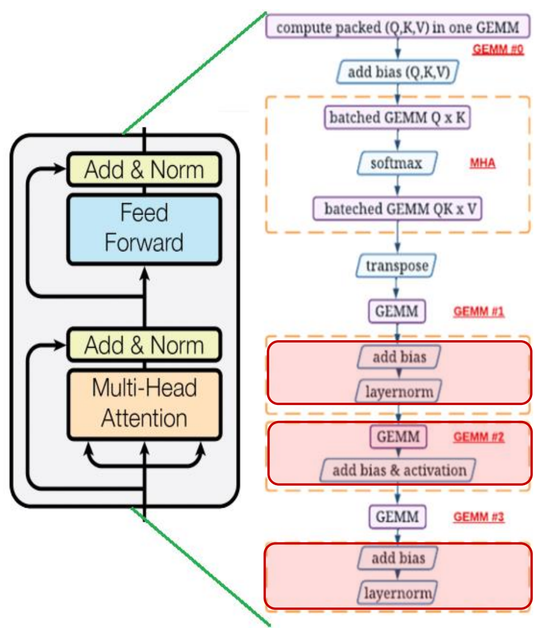
شکل زیر معماری نرم افزاری شتابدهنده ها را نشان میدهد. هر بخش را جداگانه شرح میدهیم.



- CUDA Kernels : کد های سطح پایینی که مستقیما بر روی GPU اجرا میشوند.
- C++ Modules : در این قسمت با استفاده از kernel های لایه قبل، ماژول های شبکه عصبی را پیاده سازی میکنیم.
- Python APIs : واسط هایی که میتوانند کد های C++ را درون پایتون قابل استفاده سازند.
- Prepared Models : برخی مدل های استاندارد ترنسفورمر برای راحتی کار پیاده سازی شده اند. برای ساخت این مدل های تسریع یافته آماده، کافی است پارامتر های آن مدل تنظیم شود.

ایده های تسریع سازی

هر کدام از شتابدهنده های مطالعه شده، ایده های مختلفی را انجام داده بودند. ما در این قسمت مهم ترینشان را بیان میکنیم.



✓ **ادغام مراحل محاسباتی** : هر kernel درون CUDA میتواند عملیات های ضرب ماتریکسی و ضرب اسکالر و اضافه کردن بایاس را در یک گام انجام دهد. و ما میتوانیم درون محاسباتمان این گام ها را به جای تقسیم در چند گام، در یک گام انجام دهیم. شکل رو به رو اعمال این ایده را نشان میدهد. قسمت های قرمز شده بیانگر ادغام دو مرحله می باشند. به این کار به اصطلاح kernel fusion میگویند.

✓ **مدیریت حافظه آگاهانه** : بدون استفاده از شتابدهنده ها، در حین محاسبات، حافظه لازم از GPU گرفته شده و پس از آن آزاد می شود. این درخواست و آزاد سازی مکرر حافظه، سربار زیادی را تحمیل میکند. اما شتابدهنده ها با آگاهی از الگوی استفاده از حافظه در مدل ترنسفورمر، خودشان در ابتدا تمام حافظه مورد نیازشان را در اختیار میگیرند. سپس این حافظه را تبدیل به تکه هایی با اندازه مشخص میکنند. و در طول روند اجرا از همین تکه ها استفاده میکنند. در این صورت زمان کمتری صرف درخواست حافظه میشود.

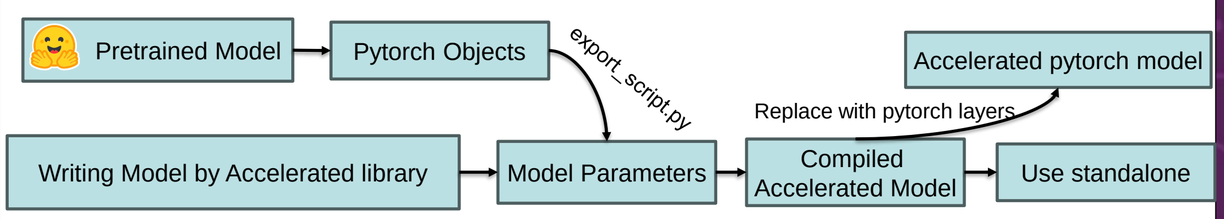
✓ **بسط فرمول ها** : با بسط دادن فرمول ها میتوان مقادیری را که لازم است در چند مرحله محاسبه شوند را در یک مرحله محاسبه کرد. مثلا برای واریانس هر دو رابطه زیر وجود دارند. اما رابطه بسط یافته (سمت راست) در یک گام کمتر میتواند محاسبه شود.

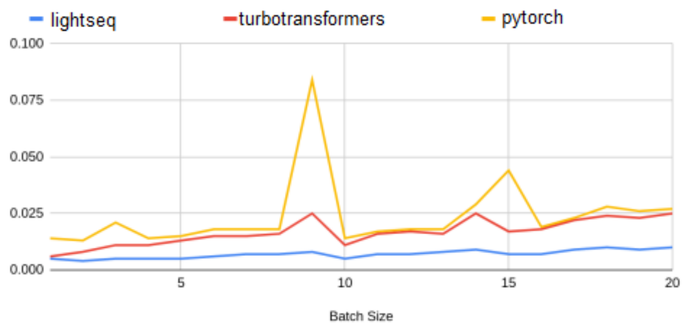
$$\sigma^2 = \overline{(x^2)} - \bar{x}^2 = \frac{\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2 / N}{N}.$$

نتایج

درحقیقت دستاور این پروژه مقایسه چندین شتابدهنده به صورت همزمان با هم است. که تقریبا در پژوهش دیگری مشاهده نشده است. ما در این پروژه، سه شتاب دهنده به نام های LightSeq، TurboTransformers و Byte Transformers را بررسی کرده ایم. این بررسی شامل تحلیل کد و همچنین مطالعه ایده های تسریع سازی است که در بخش **روش/ساختار/مدل پیشنهادی** توضیح داده شده است. سپس هر کدام را به صورت جداگانه راه اندازی کرده و یک نمونه را اجرا گرفتیم .در نهایت یک مدل یکسان را با استفاده از Pytorch، LightSeq و TurboTransformers پیاده سازی کردیم. این مدل بر پایه BERT بود که متن ها را به سه دسته مثبت، خنثی و منفی دسته بندی میکرد.

- **راحتی در استفاده** : به کاربردن شتابدهنده ها در پروژه آزمایشی، دشواری خاصی نداشت. برای پیاده سازی این پروژه، ابتدا یک مدل از پلتفرم Hugging Face را با استفاده از Pytorch آماده سازی کردیم. سپس با استفاده از یک برنامه اسکریپتی، پارامتر های آن را استخراج کرده و به قالب کتابخانه های شتابدهنده در آوردیم. بعد از این که مدل شتابدهی شده ساخته شد، میتوانیم آن را با یکی از لایه های مدل Pytorch جایگزین کرده یا این که کاملا مجزا آن را به کار ببریم. در نهایت با افزودن **۳۰ خط کد پایتون**، مدل های شتابدهی شده آماده شد. البته مدل آزمایشی ما ساده بوده و برای موارد های پیچیده کار دشوارتر است. شکل زیر نحوه راه اندازی را نشان میدهد.





- **کاهش زمان اجرا** : این پروژه در محیط colab و با استفاده از GPU های سری T4 انجام شده است. کاهش زمان اجرا به خوبی در نمودار روبه رو قابل مشاهده است. میتوان گفت زمان اجرا به یک چهارم کاهش یافته است.

جمع بندی

- اهمیت مدل شبکه عصبی ترنسفورمر در صنعت و پژوهش رشد خیره کننده ای کرده است. مثلا ChatGPT برپایه این مدل ساخته شده است و گواهی بر این مدعاست.
- به دلیل ساختار این مدل، با افزایش تعداد پارامتر ها، هزینه زمانی این مدل افزایش چشمگیری میکند. تا جایی که استفاده از این مدل در صنعت بدون شتابدهنده های سخت افزاری موثر نخواهد بود.
- استفاده از از این شتابدهنده ها برای مدل های استاندارد بسیار ساده است و با Pytorch قابل ترکیب هستند.
- شتابدهنده ها با ایده هایی مثل ادغام مراحل محاسباتی و مدیریت هوشمندانه حافظه میتوانند به سرعت بالاتری دست پیدا میکنند.
- در پروژه آزمایشی مان، نتیجه این شد که شتابدهنده Lightseq بهترین عملکرد را در مقایسه با بقیه موارد در زمان اجرا دارد.
- طبق آمار درون مقالات و همچنین آزمایش ما ، میتوان گفت این شتابدهنده ها میتوانند مدل را به صورت تخمینی ۲۵٪ سریع تر کنند.

مراجع اصلی

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need.

[2] Xiaohui Wang, Yang Wei, Ying Xiong, Guyue Huang, Xian Qian, Yufei Ding, Mingxuan Wang, and Lei Li. 2022. LightSeq2: accelerated training for transformer-based models on GPUs. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '22). IEEE Press,Article 38, 1–14.

[3] J. Fang, Y. Yu, C. Zhao, and J. Zhou, “Turbotransformers: an efficient gpu serving system for transformer models,” in Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2021, pp. 389–402.