بررسی مقاله :

Utilizing Deep Learning Models for Image Analysis at Scale: Comparison of Deployment Solutions:

گزارش جامع: بررسی و مقایسه پلتفرم‌های ارائه مدل‌های بینایی ماشین

مقدمه و هدف تحقیق:

این گزارش، خلاصه‌ای جامع و قابل فهم از یک تحقیق علمی است که به بررسی و مقایسه عملکرد سه پلتفرم محبوب و متن‌باز برای **ارائه (Serving)** مدل‌های یادگیری ماشین، با تمرکز ویژه بر **مدل‌های بینایی ماشین**، پرداخته است. هدف اصلی این مطالعه، سنجش و مقایسه کارایی (شامل **سرعت استنتاج** و **میزان مصرف منابع سخت‌افزاری** مانند CPU و GPU) پلتفرم‌های **TensorFlow Serving (TF Serving)**، **TorchServe** و **Nvidia Triton Inference Server** بوده است. برای این منظور، از مدل‌های شناخته‌شده طبقه‌بندی تصویر مانند **EfficientNet-B7** و **MobileNetV3-Large** به همراه نسخه **کوانتیزه شده** آن) که بر روی مجموعه داده ImageNet آموزش دیده‌اند، استفاده شده است.

چالش استقرار مدل و نقش Docker:

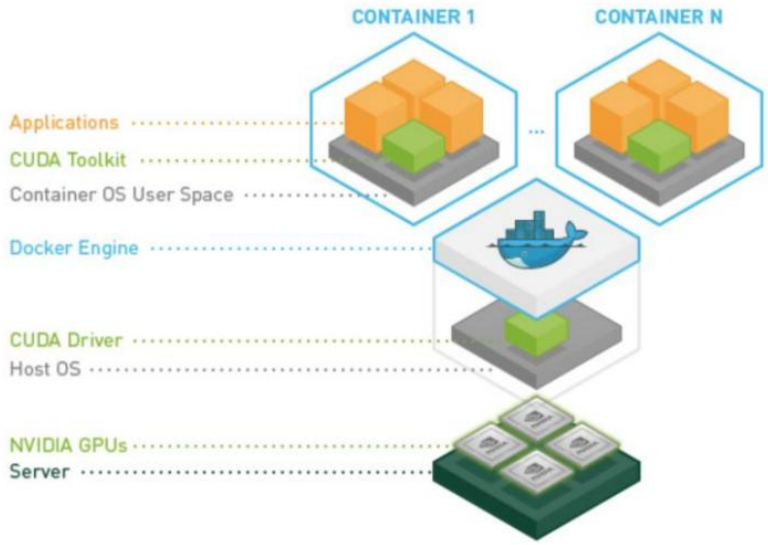

Figure 26. Docker architecture, enhanced with Nvidia GPU components

استقرار مدل‌های یادگیری ماشین در محیط‌های عملیاتی (Production) با چالش‌هایی نظیر تنوع سخت‌افزارها، نرم‌افزارها، کتابخانه‌های بهینه‌سازی و مشکلات سازگاری بین آن‌ها روبرو است. برای غلبه بر این مشکلات و ایجاد یک محیط آزمایش استاندارد و ایزوله، محققان از فناوری **Docker** بهره برده‌اند. داکر امکان بسته‌بندی برنامه‌ها و تمام وابستگی‌هایشان را در

واحدهای مستقلی به نام **کانتینر** فراهم می‌کند. این امر باعث می‌شود هر پلتفرم ارائه مدل، در محیطی جداگانه و بدون تداخل با سایرین اجرا شود و شرایط یکسانی برای مقایسه فراهم گردد.

مروری بر پلتفرم‌های مورد بررسی:

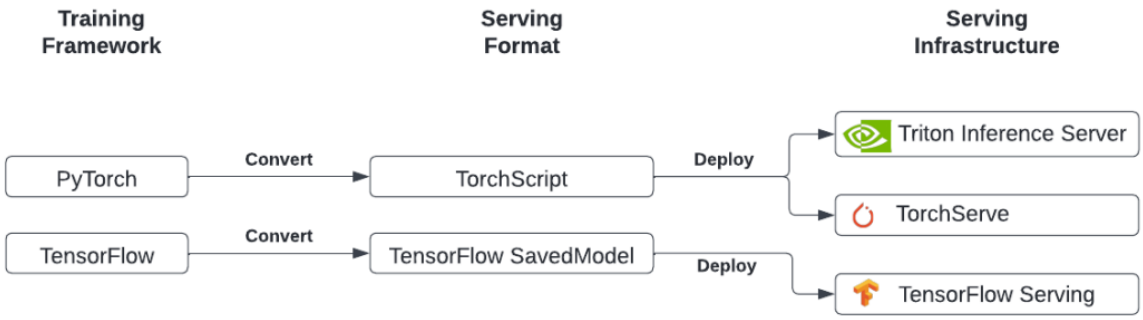

Figure 36. Three serving software infrastructures under test

۱. **TensorFlow Serving (TF Serving):**

- **تمرکز :**عمدتاً برای ارائه مدل‌های توسعه‌یافته با فریم‌ورک TensorFlow طراحی شده است) فرمت SavedModel).
- **راه‌اندازی :**فرآیند نصب و راه‌اندازی آن نسبت به دو پلتفرم دیگر ساده‌تر است، به خصوص اگر از قبل با اکوسیستم TensorFlow آشنا باشید.
- **ویژگی‌ها و محدودیت‌ها :**قابلیت‌های اصلی آن بر روی خودِ عملِ ارائه مدل متمرکز است. مراحل **پیش‌پردازش** (مانند تغییر اندازه یا نرمال‌سازی تصویر) و **پس‌پردازش** (مانند تفسیر خروجی مدل) به صورت داخلی تعبیه نشده‌اند و باید توسط کلاینت (برنامه‌ای که درخواست استنتاج را ارسال می‌کند) یا با استفاده از ابزارهای دیگر TensorFlow پیاده‌سازی شوند. این موضوع می‌تواند پیاده‌سازی سمت کلاینت را کمی پیچیده‌تر کند.
- **عملکرد :**در آزمایش‌ها، هنگام کار با تصاویر با ابعاد کوچک) مانند ۲۲۴ x224پیکسل(، عملکرد سریع‌تری نسبت به TorchServe داشت.

۲. **TorchServe:**

- **تمرکز :**به طور خاص برای ارائه مدل‌های توسعه‌یافته با فریم‌ورک PyTorch طراحی شده است.

- **راه‌اندازی :**نیازمند آماده‌سازی مدل در یک فرمت آرشیو خاص (.mar) است که شامل خود مدل (معمولاً در فرمت TorchScript) و **فایل‌های Handler** می‌شودHandler .ها کدهای پایتونی هستند که منطق بارگذاری مدل، پیش‌پردازش ورودی و پس‌پردازش خروجی را تعریف می‌کنند.

- **ویژگی‌ها و محدودیت‌ها :**امکان گنجاندن منطق پیش/پس‌پردازش درHandler ، کار را برای کلاینت ساده‌تر می‌کند، زیرا کلاینت می‌تواند داده خام (مانند فایل تصویر) را ارسال کند و نتیجه نهایی را دریافت نماید. مستندات مربوط به خطایابی در هنگام بروز مشکل درHandler ها می‌توانست کامل‌تر باشد.

- **عملکرد :**در آزمایش‌ها، هنگام کار با تصاویر با ابعاد بزرگتر) مانند ۶۰۰ x600پیکسل(، عملکرد بهتری نسبت به TF Serving نشان داد. نکته قابل توجه، مصرف منابع سخت‌افزاری CPU) و (GPU آن به طور متوسط کمتر از دو پلتفرم دیگر بود، اما این به قیمت سرعت استنتاج پایین‌تر) نسبت به (Triton تمام می‌شد. همچنین مشاهده شد که اگر به درستی برای استفاده از GPU پیکربندی نشود، می‌تواند منجر به مصرف بسیار بالای RAM شود) به دلیل ایجادWorker های زیاد بر اساس تعداد هسته‌های.(CPU

۳. **Nvidia Triton Inference Server:**

- **تمرکز :**یک پلتفرم بسیار قدرتمند و **چند-فریم‌ورکی** است که از مدل‌های) TensorFlow, PyTorch به صورت TorchScript), TensorRT, ONNX و غیره پشتیبانی می‌کند.

- **راه‌اندازی :**پیچیده‌ترین فرآیند نصب و پیکربندی اولیه را در میان سه پلتفرم دارد. نیازمند نصب پیش‌نیازهایی مانند درایورهای به‌روزNvidia ، Dockerو) Nvidia Container Toolkit برای فعال‌سازی استفاده از GPUدر کانتینر (است. آماده‌سازی مدل) مثلاً تبدیل مدل PyTorch به (TorchScript و ایجاد فایل پیکربندی (config.pbtxt) برای تعریف جزئیات ورودی/خروجی مدل ضروری است.

- **ویژگی‌ها و محدودیت‌ها :**قابلیت‌های پیشرفته‌ای مانند بهینه‌سازی خودکار برای استفاده حداکثری ازGPU ، مدیریت هوشمند **بچینگ** (گروه‌بندی درخواست‌ها برای پردازش بهینه‌تر) و **استریمینگ** ورودی‌ها را ارائه می‌دهد. مستندات آن برای برخی موارد خاص) مانند نحوه دقیق فراخوانی API استنتاج از طریق کلاینت پایتون یا خطایابی مشکلات ناسازگاری (CUDA می‌توانست شفاف‌تر باشد.

- **عملکرد :**به طور **چشمگیری سریع‌ترین** پلتفرم در انجام استنتاج روی GPU بود (پس از طی شدن مرحله گرم شدن اولیه). در برخی سناریوها تا **۱۶ برابر** سریع‌تر از پلتفرم‌های دیگر عمل کرد. این سرعت بالا با مصرف بیشتر منابع GPU همراه بود که نشان‌دهنده بهره‌برداری کارآمدتر از سخت‌افزار گرافیکی است. مرحله **گرم شدن (Warmup)** اولیه آن (زمان لازم برای رسیدن به حداکثر سرعت پس از اولین درخواست‌ها) طولانی‌تر از دو پلتفرم دیگر بود.

جزئیات آزمایش‌ها و یافته‌های کلیدی:

- **محیط آزمایش :**از سیستم‌عامل Ubuntu 20.04 ، پردازنده AMD Ryzen 5 ، کارت گرافیک Nvidia GeForce RTXو ابزار **Wandb (Weights & Biases)**برای مانیتورینگ دقیق عملکرد و مصرف منابع در طول آزمایش‌ها استفاده شد.

- **سناریوهای بنچمارک :**آزمایش‌ها در شرایط مختلفی انجام گرفتند: استنتاج با و بدون گرم کردن مدل، استفاده از یک تصویر تکراری یا مجموعه‌ای از تصاویر متنوع، و استفاده از تصاویر با ابعاد ثابت از پیش تعیین‌شده یا تصاویر با ابعاد متغیر که در لحظه استنتاج تغییر اندازه می‌یافتند.

- **تأثیر گرم کردن :(Warmup)** تمام پلتفرم‌ها در چند درخواست اولیه کندتر عمل می‌کنند و سپس به سرعت پایدار می‌رسند. این پدیده به دلیل نیاز به بارگذاری اولیه مدل در حافظه و بهینه‌سازی‌های اولیه است. گرم کردن مدل قبل از بنچمارک اصلی برای دستیابی به نتایج دقیق ضروری است Triton .زمان گرم‌شدن بیشتری نیاز داشت.

- **تأثیر ابعاد تصویر و پیش‌پردازش :**نتایج نشان داد که TF Serving برای تصاویر کوچک و TorchServe برای تصاویر بزرگ (نسبت به یکدیگر) کارایی بهتری دارند Triton .در هر دو حالت سریع‌تر بود. زمانی که تغییر اندازه تصاویر در لحظه استنتاج انجام می‌شد (سناریو ۴)، TF Servingکندتر از TorchServe عمل کرد که احتمالاً به دلیل سربار اضافی پیش‌پردازش در سمت کلاینت یا خود پلتفرم است.

- **تأثیر کوانتیزه‌سازی مدل :**کوانتیزه‌سازی (کاهش دقت محاسبات مدل برای کاهش حجم و افزایش سرعت) نتایج متفاوتی داشت. در Triton ، استفاده از مدل MobileNetV3 کوانتیزه شده، بهبود سرعت **جزئی و از نظر آماری غیرمعناداری** را نشان داد. اما در TorchServe ، مدل کوانتیزه شده به طور **معناداری کمی کندتر** از مدل اصلی عمل کرد. این نشان می‌دهد که بهره‌مندی از کوانتیزه‌سازی ممکن است به پلتفرم ارائه و نحوه پیاده‌سازی آن بستگی داشته باشد.

- **مصرف منابع و ملاحظات هزینه Triton :**با وجود مصرف بالاتر GPU ، به دلیل سرعت بسیار زیادش، می‌تواند در کاربردهای با حجم پردازش بالا (مانند تحلیل فریم‌های یک ویدئوی طولانی) در محیط‌های **ابری(Cloud)** ، گزینه **مقرون‌به‌صرفه‌تری** باشد. محاسبه انجام شده در تحقیق نشان داد که هزینه پردازش ۲۴ ساعت ویدئو با استفاده از Tritonبه مراتب کمتر از TF Serving خواهد بود (حدود ۱۸ یورو در مقابل ۲۹۵ یورو)، زیرا زمان پردازش کل به

شدت کاهش می‌یابد.

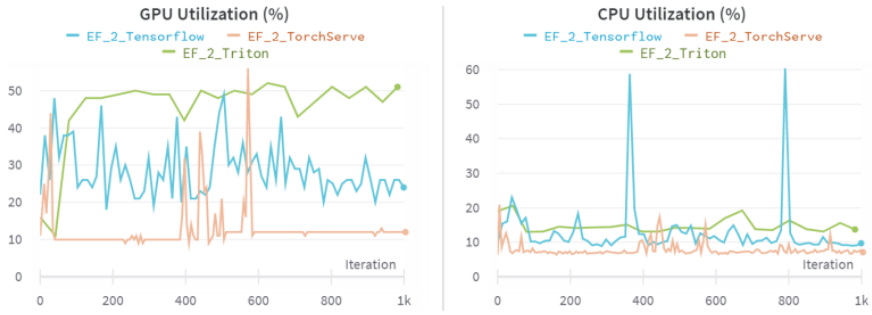

Figure 43. Scenario #2: GPU (left) and CPU (right) utilization during the experiment

-

چالش‌های مشاهده شده در طول تحقیق:

پیکربندی اولیه و نصب پیش‌نیازهای Triton زمان‌بر بود. یافتن مستندات کامل و مثال‌های کاربردی برای برخی موارد خاص)مانند فرمت دقیق ورودی مورد انتظار TF Serving یا تفسیر پیام‌های خطا در TorchServe و (Triton چالش‌برانگیز بود. مدیریت وابستگی‌ها، به خصوص نسخه‌های سازگار درایور Nvidia و کتابخانهCUDA ، نیز نیاز به دقت داشت.

نتیجه‌گیری نهایی و آموخته‌ها:

این تحقیق به وضوح نشان می‌دهد که انتخاب پلتفرم "بهینه" برای ارائه مدل‌های یادگیری ماشین، یک **بده‌بستان-Trade) off)**بین فاکتورهای مختلف است و هیچ راه‌حل واحدی برای همه سناریوها وجود ندارد. نکات کلیدی آموخته شده عبارتند از:

۱. **اولویت‌بندی نیازها :**انتخاب پلتفرم باید بر اساس اولویت‌های پروژه صورت گیرد: آیا **حداکثر سرعت** مهم‌ترین فاکتور است(Triton) ؟ آیا **سهولت راه‌اندازی اولیه** و کار با مدل‌های TF ارجحیت دارد(TF Serving) ؟ یا **ادغام پیش/پس‌پردازش** با مدل‌های PyTorch و **مصرف کمتر منابع** (با پذیرش سرعت پایین‌تر) مد نظر است ؟(TorchServe)

۲. **Nvidia Tritonبرای سرعت :**اگر هدف اصلی دستیابی به کمترین زمان استنتاج روی GPU است و پیچیدگی اولیه نصب قابل پذیرش است، Tritonبهترین گزینه است. بهینه‌سازی‌های داخلی آن منجر به کارایی فوق‌العاده‌ای می‌شود که می‌تواند هزینه‌های عملیاتی در مقیاس بزرگ را کاهش دهد.

۳. **TF Servingبرای شروع آسان (با محدودیت) :**برای پروژه‌هایی که عمدتاً از TensorFlow استفاده می‌کنند و نیاز به راه‌اندازی سریع دارند، TF Servingگزینه خوبی است، اما باید سربار پیاده‌سازی پیش/پس‌پردازش در جای دیگر (معمولاً کلاینت) را در نظر گرفت.

۴. **TorchServeبرای اکوسیستم PyTorch و سادگی کلاینت :**برای مدل‌هایPyTorch ، به خصوص وقتی می‌خواهیم منطق پردازش داده را در کنار مدل قرار دهیم و یا محدودیت منابع داریم، TorchServeانتخاب مناسبی است.

۵. **اهمیت بنچمارک اختصاصی :**نتایج این تحقیق یک راهنمای کلی ارائه می‌دهد، اما برای تصمیم‌گیری نهایی، انجام بنچمارک با مدل‌ها و داده‌های **واقعی** پروژه خودتان در محیط هدف، امری ضروری است.

۶. **نقش حیاتی ابزارهای :MLOps** استفاده از ابزارهایی مانند Docker برای مدیریت محیط و Wandb برای مانیتورینگ، در توسعه و استقرار موفقیت‌آمیز سیستم‌های یادگیری ماشین بسیار ارزشمند است.

در نهایت، این مطالعه یک مقایسه عملی و کاربردی بین سه ابزار مهم در حوزه عملیات یادگیری ماشین (MLOps) ارائه می‌دهد و به متخصصان کمک می‌کند تا با آگاهی بیشتری، پلتفرم مناسب برای نیازهای خاص خود را انتخاب کنند.

گزارش خلاصه:

گزارش مطالعه مقاله: سرویس‌دهی مدل‌های یادگیری عمیق

استاد گرامی،

با مطالعه مقاله‌ای در مورد سرویس‌دهی مدل‌های یادگیری عمیق، اطلاعات ارزشمندی درباره استقرار این مدل‌ها در محیط‌های عملیاتی کسب کردم. این مقاله سه پلتفرم منبع‌باز، یعنی TensorFlow Serving، TorchServe و Nvidia Triton Inference Server را مقایسه کرده و عملکرد آن‌ها را از نظر سرعت استنتاج، مصرف منابع سخت‌افزاری (GPU و CPU) و تاثیر بهینه‌سازی‌هایی مثل کوانتومی‌سازی مدل‌ها بررسی کرده است. در ادامه، به صورت خلاصه و منسجم، درس‌هایی که آموختم و نکات کلیدی را گزارش می‌دهم.

ابتدا متوجه شدم که سرویس‌دهی مدل‌های یادگیری عمیق فراتر از آموزش آن‌هاست و نیازمند ابزارهایی است که مدل‌ها را به‌صورت سریع و کارآمد در محیط‌های واقعی اجرا کنند. هر یک از پلتفرم‌های مورد مطالعه ویژگی‌های خاص خود را دارند:

- **TensorFlow Serving** نصب ساده‌ای دارد و برای مدل‌های TensorFlow مناسب است، اما برای پیش‌پردازش داده‌ها و مدل‌های کوانتومی‌شده نیاز به ابزارهای اضافی دارد.

- **TorchServe** برای مدل‌های PyTorch طراحی شده و قابلیت‌های مفیدی مثل پیش‌پردازش و پس‌پردازش داده‌ها ارائه می‌دهد، ولی استفاده از GPU در آن نیاز به تنظیمات دقیق دارد و مدیریت حافظه RAM گاهی چالش‌برانگیز است.

- **Nvidia Triton Inference Server** از چند چارچوب پشتیبانی می‌کند و برای GPU بسیار بهینه است، اما نصب آن پیچیده‌تر بوده و نیازمند سخت‌افزار و نرم‌افزار خاصی است.

از نظر سرعت استنتاج، آزمایش‌ها نشان داد که Triton پس از مرحله گرم کردن مدل (warm-up) سریع‌تر از دو پلتفرم دیگر عمل می‌کند، هرچند این مرحله در آن طولانی‌تر است. گرم کردن مدل به معنای اجرای چند باره اولیه برای آماده‌سازی آن است که در همه پلتفرم‌ها توصیه می‌شود. همچنین، اندازه تصاویر ورودی تاثیر زیادی بر عملکرد دارد؛ TensorFlow Serving برای تصاویر کوچک‌تر (مانند ۲۲۴x۲۲۴ پیکسل) و TorchServe برای تصاویر بزرگ‌تر (مانند ۶۰۰x۶۰۰ پیکسل) بهتر عمل کردند، اما Triton در هر دو حالت برتر بود.

در مورد مصرف منابع، Triton از GPU بیشترین استفاده را می‌کند (تا ۷۵٪ بیشتر از بقیه) که باعث عملکرد بهترش می‌شود. در مقابل، TorchServe کمترین مصرف منابع را دارد و برای محیط‌هایی با منابع محدود مناسب است، هرچند سرعت کمتری ارائه می‌دهد. درباره کوانتومی‌سازی، که برای سبک‌تر کردن مدل‌ها استفاده می‌شود، نتایج جالب بود: در TorchServe، مدل کوانتومی‌شده کمی کندتر عمل کرد، اما در Triton تفاوت قابل‌توجهی نداشت.

یکی از نکات برجسته، مقایسه هزینه در محیط‌های ابری بود. برای پردازش ۲۴ ساعت فیلم، هزینه با Triton حدود ۱۸ یورو و با TensorFlow Serving حدود ۲۹۵ یورو تخمین زده شد، که نشان‌دهنده برتری اقتصادی Triton است.

در نهایت، این مطالعه به من آموخت که انتخاب پلتفرم مناسب به نیازهای پروژه بستگی دارد. اگر سرعت اولویت باشد، Triton بهترین انتخاب است؛ اگر منابع محدود باشد، TorchServe گزینه بهتری است. همچنین، اهمیت گرم کردن مدل و توجه به اندازه تصاویر ورودی را درک کردم. این تجربه به من نشان داد که آزمایش و مقایسه عملی پلتفرم‌ها برای موفقیت در سرویس‌دهی مدل‌ها حیاتی است.