بسیار خب، در اینجا گزارشی یک صفحه‌ای از درک من از مقاله ارائه شده است که نکات کلیدی و آموخته‌ها را به زبانی ساده و منسجم توضیح می‌دهد:

گزارش درک مقاله: استقرار بهینه مدل YOLOv5 با استفاده از Triton Inference Server

مقاله ارائه شده، روشی کارآمد برای **استقرار (Deployment)** مدل تشخیص اشیاء معروف **YOLOv5** بر روی کارت‌های گرافیک **NVIDIA** با استفاده از ابزارهای پیشرفته شرح می‌دهد. هدف اصلی این مقاله، بهبود چشمگیر **سرعت استنتاج (Inference Speed)** و **توان عملیاتی (Throughput)** مدل YOLOv5 در محیط‌های عملیاتی و واقعی است، جایی که تعداد زیادی درخواست به صورت همزمان باید پردازش شوند.

چه چیزی یاد گرفتم؟

۱. **چالش اصلی :** مدل‌های یادگیری عمیق مانند YOLOv5 ، با وجود دقت بالا، در حالت عادی برای استقرار و اجرای بهینه روی سخت‌افزارهای خاص) مثل GPU های NVIDIA) طراحی نشده‌اند. فریمورک‌های اصلی مانند PyTorch بیشتر روی آموزش تمرکز دارند تا بهینه‌سازی سرعت اجرا (استنتاج).

۲. **راهکار ارائه شده :** این مقاله یک "نقشه راه" یا **طرح استقرار (Deployment Scheme)** ارائه می‌دهد که از دو ابزار کلیدی NVIDIA استفاده می‌کند:

- **Triton Inference Server:** یک فریمورک متن-باز قدرتمند برای *سرویس‌دهی* مدل‌های یادگیری عمیق. این سرور قابلیت‌های مهمی مثل **صف‌بندی درخواست‌ها(Request Queuing)** ، **اجرای همزمان مدل‌ها (Concurrent Model Execution)** و مهم‌تر از همه، **دسته‌بندی پویا (Dynamic Batching)** را فراهم می‌کند.
- **TensorRT:** یک کتابخانه برای *بهینه‌سازی* و *شتاب‌دهی* استنتاج مدل‌های یادگیری عمیق روی GPU های NVIDIA. TensorRT مدل را برای سخت‌افزار خاص بهینه می‌کند) مثلاً با ترکیب لایه‌ها یا انتخاب بهترین هسته‌های (CUDA و می‌تواند از روش‌هایی مانند **کوانتیزاسیون (Quantization)** استفاده کند.

۳. **فرآیند تبدیل مدل :** برای استفاده از TensorRT ، مدل اصلی YOLOv5 که با PyTorch نوشته شده، مستقیماً قابل استفاده نیست. فرآیند تبدیل شامل مراحل زیر است:

- **PyTorch به ONNX:** ابتدا مدل PyTorch به فرمت **ONNX** تبدیل می‌شود ONNX .یک فرمت تبادل استاندارد و میانی برای مدل‌های یادگیری عمیق است که امکان تبدیل بین فریمورک‌های مختلف را می‌دهد.

- **ONNXبه TensorRT:** سپس مدل ONNX به فرمت بهینه شده **TensorRT**تبدیل می‌شود. در این مرحله، TensorRTبهینه‌سازی‌های لازم را انجام می‌دهد.

٤. **بهینه‌سازی‌های کلیدی:**

- **تبدیل به TensorRT:** خود این تبدیل باعث افزایش سرعت می‌شود، زیرا TensorRT مدل را برای معماری خاص GPU بهینه می‌کند.(Kernel Auto-tuning, Layer Fusion)
- **کوانتیزاسیونFP16**

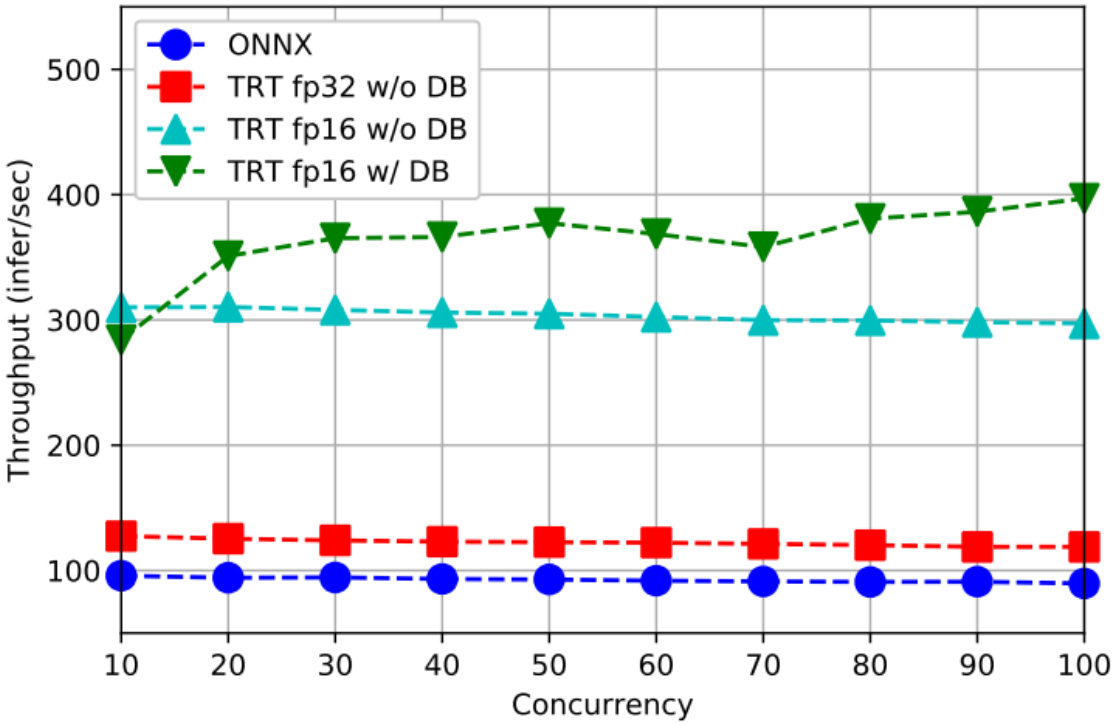

Fig. 5. Throughputs of the ONNX model, TensoRT fp32 model without dynamic batching, TensorRT fp16 model without dynamic batching and TensorRT fp16 model with dynamic batching

- **:**به جای استفاده از اعداد ممیز شناور با دقت کامل(FP32) ، از اعداد با دقت کمتر (FP16) استفاده می‌شود. این کار حجم مدل را کاهش داده و سرعت محاسبات را روی GPU هایی که از FP16 پشتیبانی می‌کنند، به طور قابل توجهی افزایش می‌دهد، البته با احتمال کاهش بسیار جزئی دقت.

- **اپراتور NMS سفارشی با پشتیبانی از :Dynamic Batch Size** یکی از مراحل نهایی در تشخیص اشیاء، **Non-Maximum Suppression (NMS)**است که کادرهای مرزی (Bounding Box) اضافی و همپوشان را حذف می‌کند. نویسندگان متوجه شدند که پلاگین NMS استاندارد TensorRT از **اندازه دسته پویا (Dynamic Batch Size)** پشتیبانی نمی‌کند. این قابلیت برای استفاده مؤثر از ویژگی Dynamic Batching در Triton ضروری است. بنابراین، آنها یک **نسخه سفارشی از پلاگین NMS**توسعه دادند که با اندازه‌های دسته متغیر کار می‌کند.

- **دسته‌بندی پویا (Dynamic Batching) در:Triton**

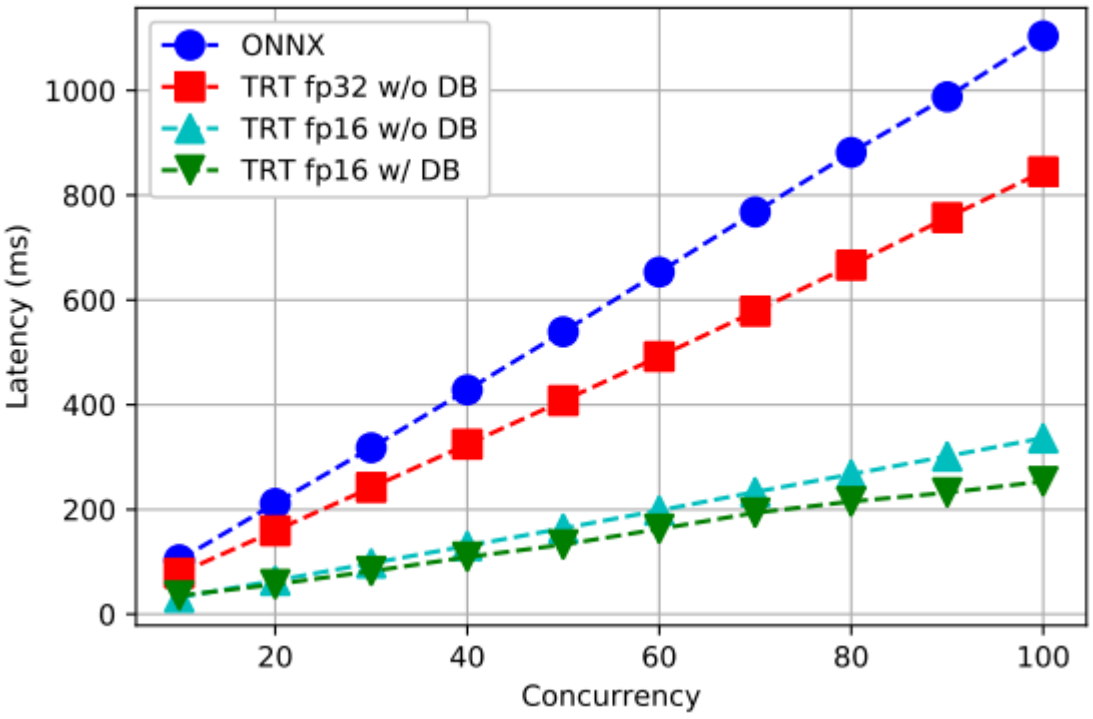

Fig. 6. Latencies of the ONNX model, TensoRT fp32 model without dynamic batching, TensorRT fp16 model without dynamic batching and TensorRT fp16 model with dynamic batching

- این ویژگی به Triton اجازه می‌دهد تا درخواست‌های ورودی را برای مدت کوتاهی جمع‌آوری کرده و آنها را به صورت یک دسته *(Batch)* بزرگتر به GPU ارسال کند. پردازش دسته‌ای به جای پردازش تکی، از قابلیت پردازش موازی GPU بسیار بهتر استفاده می‌کند و توان عملیاتی کلی سیستم (تعداد درخواست پردازش شده در ثانیه) را به شدت افزایش می‌دهد، به خصوص زمانی که تعداد زیادی کاربر همزمان درخواست ارسال می‌کنند.(High Concurrency)

۵. **نتایج آزمایش‌ها :**آزمایش‌ها روی GPU های NVIDIA 2080 Ti انجام شد. نتایج به وضوح نشان داد که:

- مدل) TensorRT حتی در حالت (FP32 سریع‌تر از مدل ONNX پایه است.
- استفاده از کوانتیزاسیون **FP16**سرعت و توان عملیاتی را به طور چشمگیری افزایش می‌دهد.
- فعال کردن) **Dynamic Batching**با استفاده از NMS سفارشی (باعث بهبود قابل توجه دیگری در توان عملیاتی و کاهش تأخیر (Latency) می‌شود، به ویژه در شرایط بار کاری بالا Concurrency) بالا.(

جمع‌بندی و اهمیت:

این مقاله یک راهکار عملی و مؤثر برای یکی از چالش‌های رایج در هوش مصنوعی یعنی استقرار بهینه مدل‌های سنگین مانند YOLOv5ارائه می‌دهد. با ترکیب هوشمندانه ابزارهایی مانند Triton Inference Server وTensorRT ، و انجام بهینه‌سازی‌های خاص مانند توسعه پلاگین NMS سفارشی و استفاده از کوانتیزاسیون FP16 وDynamic Batching ، می‌توان عملکرد این مدل‌ها را در کاربردهای واقعی به طور قابل توجهی بهبود بخشید. این کار به ویژه برای سیستم‌هایی که نیاز به پردازش آنی ویدئو یا تعداد زیادی تصویر دارند (مانند سیستم‌های نظارت تصویری هوشمند) بسیار حیاتی است. آموخته اصلی این است که صرفاً داشتن یک مدل دقیق کافی نیست، بلکه نحوه استقرار و بهینه‌سازی آن برای سخت‌افزار هدف، نقشی کلیدی در موفقیت عملیاتی آن ایفا می‌کند.