## گزارش مقایسه عملکرد CPU و GPU

هدف این پروژه مقایسه عملکرد و نشان دادن تفاوت‌های محاسباتی بین CPU و GPU است. ما دو مدل فانکشن محاسباتی برای محاسبه در CPU و GPU تعریف کرده‌ایم: GPUMatrixOperations.py برای محاسبه ماتریس در GPU و CPUMatrixOperations.py برای محاسبه ماتریس در CPU.

در ادامه، در دو ماژول مجزا به نام‌های cpu\_operations.py و gpu\_operations.py الگوریتم زیر را پیاده‌سازی کردیم:

1. `calculate\_average\_execution\_time`: در این تابع، میانگین زمان اجرای یک عملیات ماتریسی با یک سایز مشخص را برای تعداد اجراهای مشخص محاسبه می‌کنیم. برای هر اجرا، تابع `matrix\_multiplication` را از ماژول `CPUMatrixOperations` یا `GPUMatrixOperations` صدا می‌زنیم و زمان اجرا را بدست می‌آوریم. سپس میانگین زمان اجراها را محاسبه و برمی‌گردانیم.

2. `calculate\_and\_save\_averaged\_results`: در این تابع، میانگین زمان اجرا برای اندازه‌های مختلف ماتریس را محاسبه و نتایج را در فایل JSON ذخیره می‌کنیم. ابتدا برای هر اندازه ماتریس، میانگین زمان اجرا را با استفاده از تابع `calculate\_average\_execution\_time` محاسبه می‌کنیم. سپس نتایج را به فرمت JSON در یک دیکشنری ذخیره می‌کنیم و در فایل `averaged\_execution\_cpu\_times.json` یا `averaged\_execution\_gpu\_times.json` ذخیره می‌کنیم.

برای اجرای ماژول‌های جداگانه `cpu\_operations.py` و `gpu\_operations.py` در محیط CPU و

GPU (مانند Google Colab)، نتایج را بدست می‌آوریم و نمودارهای لاین چارت را رسم می‌کنیم. همچنین، ماژول `plot\_average\_from\_json.py` نوشته شده است که میانگین دو فایل JSON `averaged\_execution\_gpu\_times.json` و `averaged\_execution\_cpu\_times.json` را محاسبه کرده و نمودار لاین چارت را رسم می‌کند تا بتوانیم به صورت دقیق‌تری مقایسه کنیم.

با مقایسه نتایج، مشاهده می‌شود که عملکرد ماتریس در GPU سریعتر است. این اتفاق به دلیل وجود تعداد تردهای بیشتر برای محاسبات در GPU رخ می‌دهد. به عبارت دیگر، GPU امکان انجام محاسبات موازی را به ما می‌دهد و با شکستن محاسبات به قطعات کوچکتر و اجرای همزمان در تردهای مختلف، جواب

را بسرعت بدست می‌آوریم. در مقابل، CPU برای محاسبات ماتریسی که نیاز به پردازش خاصی ندارند و محاسباتی کوچک هستند، به دلیل تعداد هسته‌های موجود، سرعت بهتری ارائه می‌دهد.

همچنین در ماژول‌هایی که در بالا نام برده شد، از ماژول numpy برای ایجاد تسک‌ها استفاده شده است. این ماژول قادر است ماتریسی با مقادیر تصادفی ایجاد کند و تابع dot آن را به شکل محاسبات موازی بر روی ضرب ماتریسی انجام دهد.

علاوه بر این، در پوشه image\_processing دو ماژول cpu\_image\_processing.py و gpu\_image\_processing.py نوشته شده است. هدف از این تست، بررسی عملکرد پردازش تصویر در CPU و GPU است. نتایج بررسی در فایل JSON و در عکس نمودار line chart ذخیره شده است (به نام‌های img-proc-cpu.json و img-proc-cpu.png).

همچنین، برای بررسی تأثیر موازی‌سازی در بهبود عملکرد محاسبات، یک ماژول دیگر به نام matrix\_multiplier.py در پوشه performance\_report تعریف شده است. این ماژول ورودی سایز ماتریس را دریافت کرده و دو تابع را پیاده‌سازی کرده است: یکی به صورت موازی و دیگری به صورت سریال. این توابع یک ماتریس با اعداد تصادفی ایجاد شده را با سایز داده شده محاسبه کرده و خروجی محاسبه (یک عدد) را برمی‌گردانند. سپس در ماژول اصلی پروژه به نام performance\_report\_run.py، زمان اجرای این دو تابع را محاسبه و در یک فایل JSON ذخیره می‌کنیم و با نمایش نمودار نوع بار (bar chart) تفاوت را مشاهده می‌کنیم. این بررسی نشان می‌دهد که موازی‌سازی عملیات محاسباتی تأثیر زیادی در سرعت انجام عملیات دارد.

بنابراین، نتیجه‌گیری می‌تواند این باشد که برای انجام محاسبات یا تسک‌هایی که پر از محاسبات کوچک هستند، مثل محاسبه ماتریس، از موازی‌سازی استفاده شود. همچنین، بر اساس نتایج به‌دست آمده، مشخص شد که GPU با تعداد تردهای بیشتری که برای انجام عملیات موازی در اختیار ما قرار می‌دهد، عملکرد بهتری نسبت به CPU دارد.