UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

DISCIPLINA: COMPUTAÇÃO DE ALTO DESEMPENHO

Paralelizando o treinamento de uma rede neural

Integrantes:
Alexandre Costa Ferro - 202105832
Elisa Ayumi Masasi de Oliveira - 202105837
Iago Alves Brito - 202109766
Rafael Alves Goiás - 202105865

- INTRODUÇÃO

O constante avanço da tecnologia de redes neurais tem revolucionado o campo da aprendizagem de máquina, permitindo a resolução de tarefas complexas em diversas áreas, como reconhecimento de padrões, processamento de linguagem natural e visão computacional. Uma das áreas cruciais no aprimoramento do desempenho das redes neurais é o paralelismo, que se tornou uma abordagem indispensável para lidar com a crescente demanda computacional.

Neste contexto, este relatório apresenta a implementação e análise do paralelismo nos estágios de treinamento de uma rede neural para a resolução do problema que fez com que o avanço do estudo de redes se desenvolvesse, o do 'ou exclusivo'. Reconhecendo que a eficiência computacional é um fator determinante para a viabilidade desses algoritmos, exploramos estratégias de paralelização visando otimizar o tempo de treinamento e maximizar a utilização dos recursos de processamento.

Análise que foi concretizada por meio da implementação de um problema didático, o XOR (OU Exclusivo), e da implementação de uma análise de sentimentos com bag of words, que traz um dataset anotado pelos próprios integrantes da turma 2021.

Ao longo deste relatório, examinamos as diferentes formas de paralelismo aplicadas às etapas de propagação direta (feed-foward) e retropropagação (backpropagation). Nosso foco se direcionou para explorar o potencial do paralelismo multi-core usando a biblioteca OpenMP, bem como a alavancagem dos recursos de multi-processamento da GPU através da plataforma CUDA.

- OBJETIVO

O objetivo principal deste trabalho é explorar e implementar estratégias de paralelização nos estágios de treinamento de uma rede neural. Com base nas etapas de propagação direta e retropropagação, nosso objetivo é alcançar uma aceleração significativa no processo de treinamento, reduzindo o tempo necessário para atingir uma convergência satisfatória do modelo. Especificamente, pretendemos:

Analisar Cargas de Trabalho para Paralelismo: Identificar os pontos críticos que podem ser paralelizados. Avaliar o impacto do uso eficiente de recursos de processamento multi-core através da biblioteca OpenMP e da alavancagem de multiprocessadores em GPUs utilizando a plataforma CUDA.

Implementar Paralelismo Multi-core e GPU: Desenvolver implementações paralelas eficazes para as etapas de treinamento, utilizando OpenMP e CUDA, respectivamente. Buscaremos aproveitar a natureza paralela intrínseca das redes neurais para distribuir as tarefas computacionais entre os núcleos de processamento e os multiprocessadores da GPU.

Avaliar Desempenho e Aceleração: Realizar experimentos comparativos entre as implementações sequenciais, paralela com OpenMP e paralela com CUDA, medindo o tempo de treinamento e o desempenho. Buscaremos quantificar a aceleração obtida em cada abordagem além de comparar múltiplas configurações da nossa rede (épocas, número de camadas e etc)

Compreender Limitações e Potencial: Investigar as limitações das implementações paralelas em relação à escalabilidade e aos ganhos obtidos. Com base nos resultados, buscaremos entender quando e como o paralelismo pode ser mais vantajoso, considerando as características da rede neural e dos recursos de processamento disponíveis.

- DESENVOLVIMENTO

Na condução deste trabalho, adotamos três abordagens distintas que se concentraram em explorar o potencial de paralelismo de dados em nossa rede neural. As estratégias selecionadas abrangeram o uso de multi-core, multiprocessadores em GPU e a aplicação de processamento em lote. Para implementar e testar nossas abordagens, foi utilizada a máquina do discente Rafael Alves Goiás, que proporcionou um ambiente propício para execuções eficientes. Nossa escolha baseou-se na utilização do processador I5 de 10° Geração de 6 núcleos, além da GPU GTX 1650 oferecendo recursos potentes para explorar o paralelismo.

OpenMP

A primeira abordagem adotada concentrou-se no paralelismo multi-core, empregando a biblioteca OpenMP. No estágio de propagação direta (feedforward), aplicamos a paralelização às operações de multiplicação de matrizes entre as camadas. A utilização do OpenMP facilitou a soma paralela dos produtos elementares, resultando em uma execução mais ágil. No processo de retropropagação, estabelecemos um fluxo similar de paralelismo para calcular os gradientes de erro e armazenar os deltas de maneira paralela. As atualizações das matrizes de peso também foram executadas de forma paralela, consolidando uma abordagem de treinamento eficaz.

CUDA

Nossa segunda abordagem buscou aproveitar a capacidade de processamento da GPU, empregando a tecnologia CUDA. Inicialmente, aplicamos a paralelização através da multiplicação de matrizes durante a etapa de propagação direta. No entanto, nossa primeira tentativa de criar uma matriz extensa para a multiplicação de matrizes revelou-se menos eficiente do que o esperado. Posteriormente, adotamos um enfoque que se beneficiou da arquitetura de blocos de threads da GPU, realizando cálculos de multiplicação vetorial por bloco. Isso permitiu uma distribuição mais eficaz do processamento.

- RESULTADOS

Nossa dedicação à execução do projeto nos permitiu alcançar todos os planos iniciais estabelecidos. No contexto das nossas conquistas, obtivemos um aumento notável de velocidade, que se correlacionou diretamente com o número de núcleos da máquina, ao empregar a biblioteca OpenMP. Além disso, essa fase do projeto também viu a utilização bem-sucedida de chamadas de kernel tanto do OpenMP quanto do CUDA, explorando assim o potencial do paralelismo de dados que estava à nossa disposição.

Além disso, cronometramos os segundos/iteração específicos da aplicação, para determinar o tempo que nossa rede neural necessitou para percorrer todas as observações de treinamento por thread. Assim como observado a seguir para o problema XOR e para a Análise de Sentimentos utilizando bag of words, respectivamente:

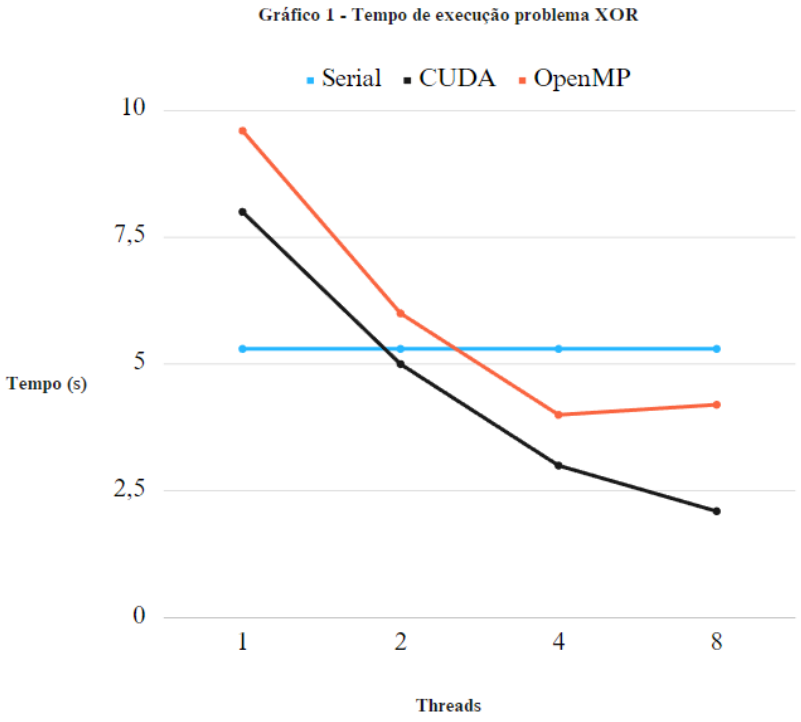
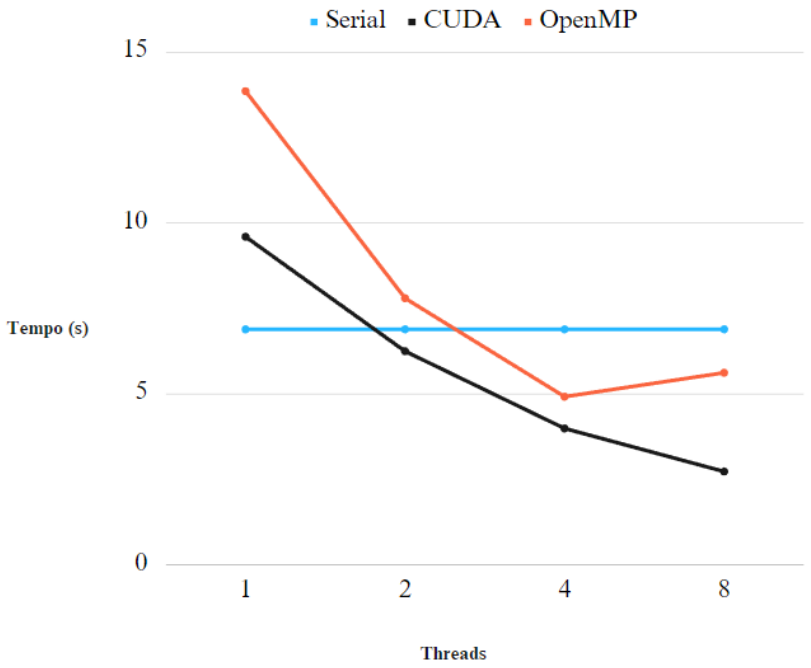


Gráfico 2 - Tempo de execução análise de sentimentos



Como pode ser observado, tanto no problema do XOR, quanto na análise de sentimentos, a implementação em CUDA teve um desempenho superior ao OpenMP e à implementação sequencial, apresentando os respectivos speedups:

Tabela 1 - Speedup problema do XOR

SPEEDUP			
THREADS	**SERIAL**	**CUDA**	**OPENMP**
1	1	0,625	0,521
2	1	1,06	0,883
4	1	1,766	1,325
8	1	2,52	1,26

Tabela 2 - Speedup problema análise de sentimentos

SPEEDUP			
THREADS	**SERIAL**	**CUDA**	**OPENMP**
1	1	0,563	0,332
2	1	0,987	0,795
4	1	1,34	1,045
8	1	2,297	1,791

A principal hipótese levantada pelo grupo para explicar essa diferença de desempenho consiste no nível de atuação de cada uma das ferramentas. Apesar do OpenMP ser uma biblioteca de fácil implementação, ela não permite uma personalização tão grande das funções a serem paralelizada, possibilitando apenas o uso de diretivas. Já utilizando CUDA, foi possível ter esse grau de liberdade e construir funções específicas que utilizassem do máximo poder computacional possível através da paralelização, mas, em contrapartida, o tempo para que essas funções fossem implementadas foi bastante custoso.

Ao analisar apenas um único thread, também notamos que o sequencial se saiu melhor que ambas as soluções. Isso se deve ao fato de que, ao utilizar apenas uma thread, o código é executado sequencialmente, e o CUDA e OpenMP utilizam de tempo de processamento para alocar espaço na memória enquanto solução sequencial não.

- CONCLUSÃO

Nossos resultados refletem nosso empenho em compreender, implementar e otimizar as estratégias de paralelização em nossa rede neural. De maneira que, nossas expectativas foram atingidas, e assim como o esperado, a paralelização foi benéfica para ambas as aplicações abordadas. Pela natureza do problema, obtivemos resultados mais positivos na aplicação em CUDA, com uma leve diferença de desempenho em detrimento do OpenMP, e uma diferença maior da implementação serial.

Continuamos a explorar novas direções para aprimorar ainda mais o desempenho e a eficiência, mantendo nosso compromisso com a busca por soluções que possam revolucionar a eficácia do aprendizado de máquina.