Aprimorando o algoritmo SLIDE-GPU para Problemas de Classificação Extrema

Michel Brasil Cordeiro [1]

Wagner M. Nunan Zola [1]

[1]Departamento de Informática, UFPR

24 de abril de 2024

Sumário

Classificação Extrema

- Classificação Extrema é uma categoria de problema de aprendizado de máquina supervisionado em que muitas classes precisam ser consideradas
- Problemas de Classificação Extrema podem envolver redes neurais que processam mais de um milhão de classes e entradas com mais de cem mil dimensões
- Alto custo computacional no processo de treinamento dessas redes

Justificativa

- A Classificação Extrema é uma área de pesquisa ativa e em rápido crescimento, podendo ser aplicada em tarefas como:
  - **Classificação de Documentos e Textos:** Em aplicações como categorização automática de documentos em áreas específicas e identificação de tópicos
  - **Recomendação de Conteúdo:** Em plataformas de streaming de música, vídeo ou livros, no qual o objetivo é recomendar itens para usuários entre um grande conjunto de opções possíveis

Justificativa

- - **Classificação de Genes e Proteínas em Biologia Computacional:** Na área de bioinformática, para classificar genes, proteínas ou sequências genômicas em diferentes categorias funcionais
    - As bases de dados de expressão gênica geralmente possuem um grande número de características e um pequeno número de amostras
- Esses são apenas alguns exemplos. A aplicação de algoritmos de Classificação Extrema continua a crescer à medida que novas tecnologias são desenvolvidas

Sub-Linear Deep Learning Engine (SLIDE)

- Sub-Linear Deep Learning Engine (SLIDE) [Chen et al. 2020] é um algoritmo para o problema de Classificação Extrema
- Esse algoritmo constrói tabelas *hash* sensíveis à localidade (LSH) para selecionar neurônios com alta ativação em tempo sublinear
- Sendo assim, o SLIDE não calcula todos os pesos de todos os neurônios durante o processo de treinamento

Sub-Linear Deep Learning Engine (SLIDE)

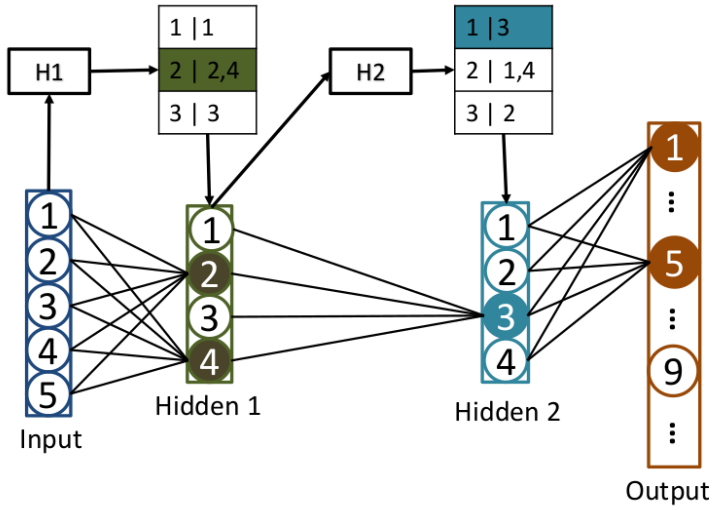

Figure: Fonte: [Chen et al. 2020]

- Primeiramente, é obtido o código hash a partir dos dados de entrada
- Então, a tabela de hash é consultada para obter os neurônios que serão ativados na primeira camada

Sub-Linear Deep Learning Engine (SLIDE)

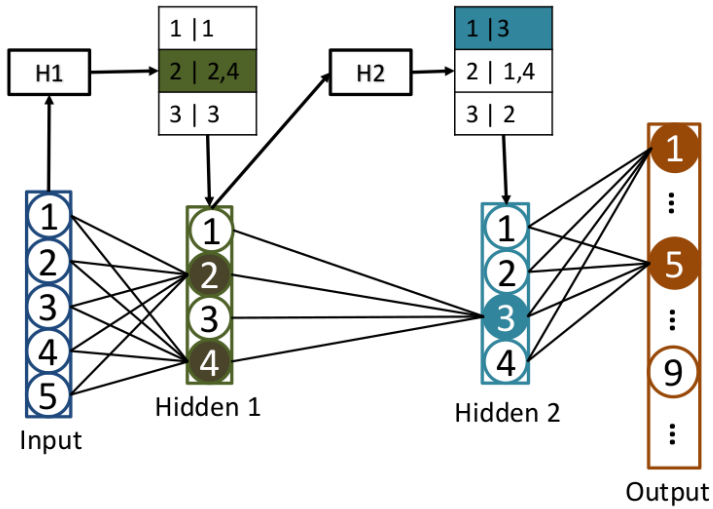

Figure: Fonte: [Chen et al. 2020]

- Utilizando a saída da camada anterior, um novo código hash é obtido, e uma nova consulta na tabela é feita, obtendo os neurônios que deverão ser ativados
- O mesmo é feito para as camadas subsequentes, obtendo uma ativação esparsa de neurônios

Sub-Linear Deep Learning Engine (SLIDE)

- Durante o backpropagation, a saída da rede é comparado com o rótulo da entrada e o erro é retropropagado para calcular o gradiente e atualizar os pesos
- Os gradientes parciais são propagados apenas para os neurônios ativados nas camadas anteriores

Sub-Linear Deep Learning Engine (SLIDE)

- A implementação do SLIDE utiliza somente paralelismo em CPU
- Durante os experimentos, o SLIDE foi comparado com a implementação otimizada para GPU do TensorFlow

Sub-Linear Deep Learning Engine (SLIDE)

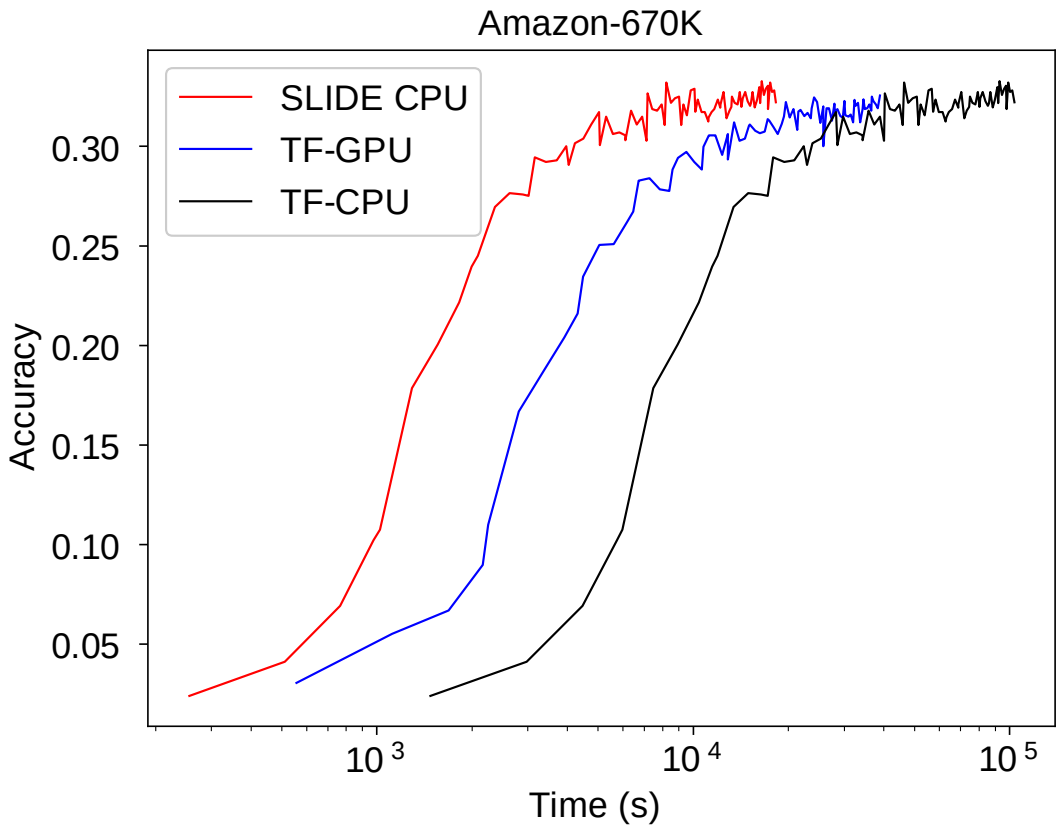

Figure: Fonte: [Chen et al. 2020]

- Mesmo utilizando apenas paralelismo em CPU, SLIDE apresentou tempo de convergência 2,7 vezes menor do que o TensorFlow

Sub-Linear Deep Learning Engine (SLIDE)

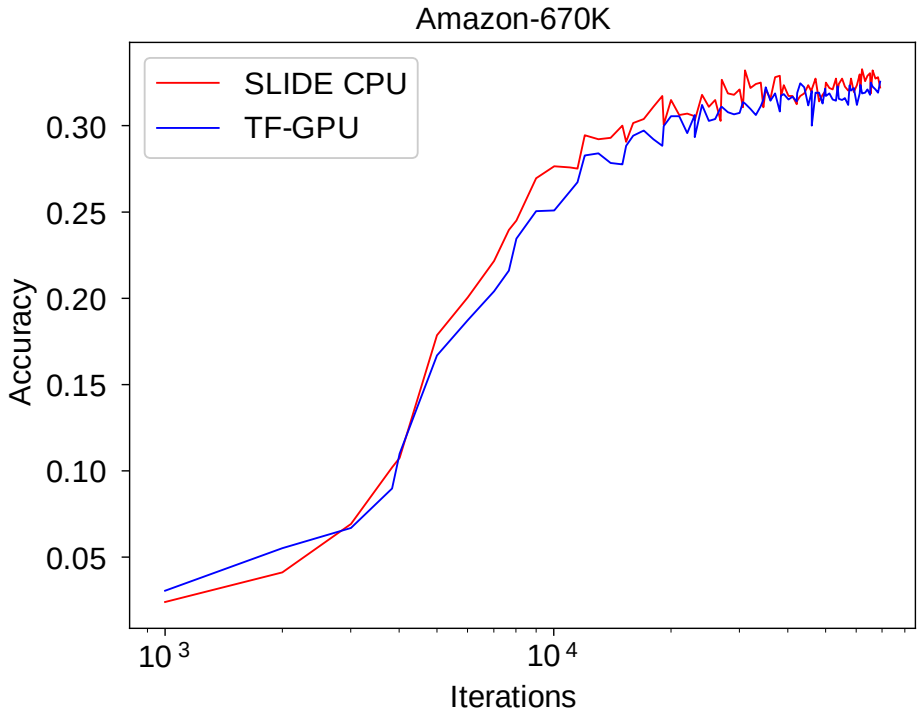

Figure: Fonte: [Chen et al. 2020]

- Quando comparado em termos de iterações, o comportamento de convergência é semelhante, confirmando que a aceleração se deve às estratégias algorítmicas da implementação

Sub-Linear Deep Learning Engine (SLIDE)

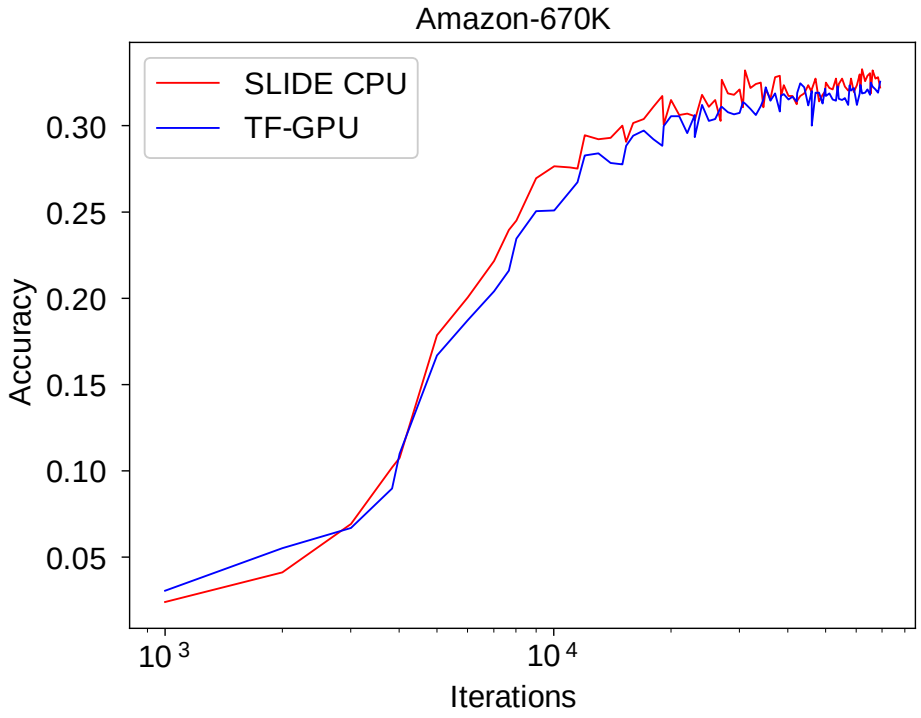

Figure: Fonte: [Chen et al. 2020]

- Isso também significa que a estratégia de selecionar apenas alguns neurônios para ativar não impacta negativamente a convergência do algoritmo

SLIDE-GPU

- Aceleração ainda mais expressiva foi alcançada pelo SLIDE-GPU [Meyer and Nunan Zola 2023]
- Esse algoritmo faz uso da GPU para acelerar a etapa de ativação na rede neural
- Além disso, o SLIDE-GPU substitui as tabelas *hash* por algoritmo de busca aproximada por vizinhos mais próximos, ou *Aproximate Nearest Neighbor* (ANN) para selecionar os neurônios que devem ser ativados
- O algoritmo de ANN utilizado pertence à biblioteca FAISS [Johnson et al. 2019], amplamente utilizada para realizar busca por similaridade em GPU

SLIDE-GPU

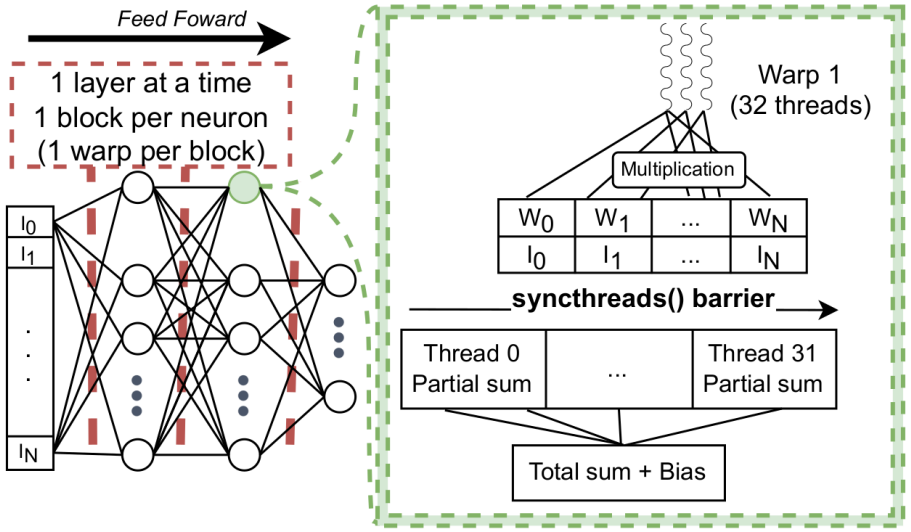

Figure: Fonte: [Meyer and Nunan Zola 2023]

- Para computar a ativação, cada neurônio selecionado pelo ANN é atribuído a uma warp
- Cada um dos pesos relacionados aos neurônios é atribuído a uma thread da warp
- Dessa forma, usando comunicação de warp para fazer a soma completa, é possível evitar o acesso à memória

SLIDE-GPU

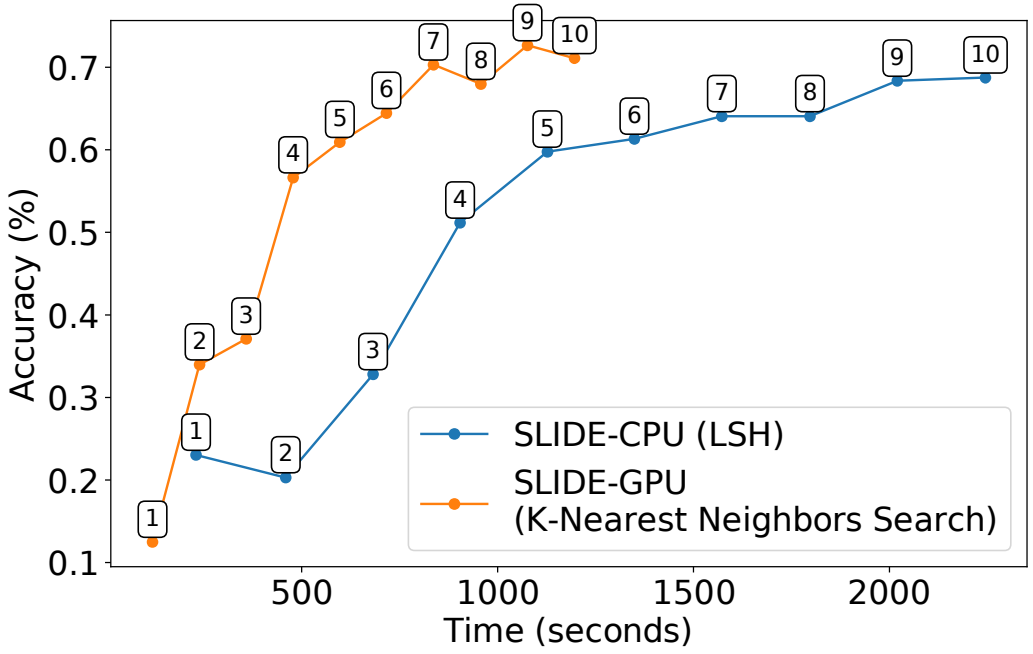

Figure: Fonte: [Meyer and Nunan Zola 2023]

- Embora a fase de backpropagation ainda seja executada em CPU, o SLIDE-GPU foi capaz de alcançar uma aceleração de 268% no processo de treinamento

SLIDE-GPU

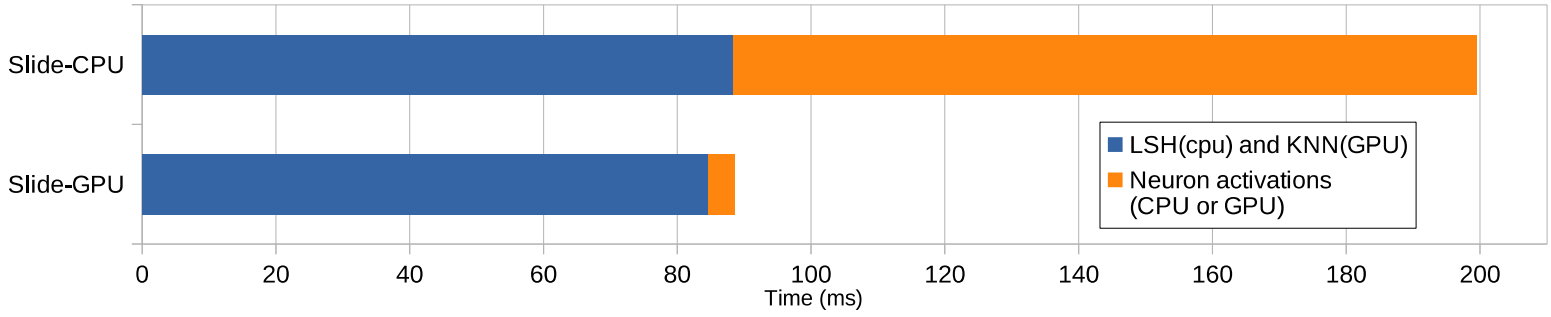

Figure: Fonte: [Meyer and Nunan Zola 2023]

- Na etapa de ativação dos neurônios, o SLIDE-GPU alcançou um tempo de execução 28,09 vezes menor em comparação com a implementação em CPU
- Isso sugere que uma aceleração adicional pode ser obtida por meio da aplicação do paralelismo massivo em todo o processo

Proposta do Trabalho

- Este trabalho propõe a investigação de estratégias para acelerar o SLIDE-GPU
- Inicialmente duas estratégias são consideradas:
  - Substituir o algoritmo de seleção de neurônios por outro mais eficiente
  - Implementar todas as etapas do algoritmo em GPU, incluindo *backpropagation*

KNN-PPW

- Aceleração no SLIDE-GPU pode ser obtida ao substituir o algoritmo de ANN da biblioteca FAISS pelo KNN-PPW [Cordeiro and Zola 2023]
- O algoritmo KNN-PPW utiliza estratégias centradas em *warps* para acelerar a busca por vizinhos e demonstrou potencial para superar algoritmos da biblioteca FAISS quando a pesquisa é realizada em pequenos lotes

KNN-PPW

- Levando em consideração que:
  - O SLIDE-GPU utiliza algoritmo de busca da biblioteca FAISS para selecionar neurônios
  - A seleção de neurônios é feita em conjuntos esparsos
- Este trabalho propõe investigar se a utilização do KNN-PPW poderia proporcionar ganhos de desempenho

KNN-PPW

- Além disso, o algoritmo de ANN faz uso de estruturas de dados que particionam o espaço de busca
- Essas estruturas precisam ser atualizadas quando o espaço de busca é modificado, processo que pode ser computacionalmente custos
- Esse custo poderia ser eliminado com o uso do KNN-PPW, pois esse algoritmo não mantém estruturas de dados que necessitam de atualizações

Backpropagation em GPU

- Apesar da esparsidade de computações no SLIDE apresentar desafios para o processamento, considera-se que o uso de técnicas centradas em *warps* permitirá a implementação da etapa de *backpropagation* em GPU de maneira efetiva
- Armazenando na memória da GPU os neurônios que foram ativados em cada camada, o algoritmo proposto pretende utilizar as *warps* para atualizar cada neurônio que foi ativado
- Dessa forma, cada um dos pesos relacionados aos neurônios ativados pode ser atribuído a uma *thread* do *warp*

Backpropagation em GPU

- Essa estratégia facilita o acesso coalescido à memória e evita a divergência de *threads*
- Além disso, utilizando primitivas de comunicação entre *threads* do mesmo *warp*, é possível reduzir o acesso à memória global da GPU (assim como no trabalho de [Meyer and Nunan Zola 2023])
- Por último, a implementação totalmente acelerada em GPU reduzirá a necessidade de transferências de dados entre CPU e GPU, potencializando uma maior aceleração no novo algoritmo

Conclusões

- Problemas de Classificação Extrema pode envolver milhões de classes e dados com centenas de milhares de dimensões
- Este trabalho propôs a investigação de duas estratégias para acelerar o algoritmo de Classificação Extrema SLIDE-GPU:
  - Substituir o algoritmo de ANN da biblioteca FAISS por outro algoritmo que se mostrou ser mais eficiente em buscas esparsas
  - Implementar a etapa de *backpropagation* em GPU de forma eficiente utilizando estratégias centradas em *warps*

Referências I


Chen, B., Medini, T., Farwell, J., Tai, C., Shrivastava, A., et al. (2020).
SLIDE: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems.
Proceedings of Machine Learning and Systems, 2:291–306.


Cordeiro, M. B. and Zola, W. M. N. (2023).
KNN paralelo em GPU para grandes volumes de dados com agregação de consultas.
In *Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho, WSCAD'23*, pages 253–264. SBC.


Johnson, J., Douze, M., and Jégou, H. (2019).
Billion-scale similarity search with GPUs.
IEEE Transactions on Big Data, 7(3):535–547.

Referências II

Meyer, B. H. and Nunan Zola, W. M. (2023).
Towards a GPU accelerated selective sparsity multilayer perceptron algorithm using K-nearest neighbors search.
In *Workshop Proceedings of the 51st International Conference on Parallel Processing*, ICPP W '22.

Muito Obrigado