Avaliando o desempenho do PyTorch sobre GPUs embarcadas

**Angelo Nery Vieira Crestani, Paulo Silas Severo de Souza,
Wagner dos Santos Marques, Marcos Paulo Konzen, Fábio Diniz Rossi**

[1]Instituto Federal de Educação, Ciência e Tecnologia Farroupilha (IFFar)
Campus Alegrete – RS 377, KM 27 – Alegrete – RS – Brasil

angelo.vcrestani@gmail.com, {paulo.souza,wagner.marques}@email.com

{marcos.konzen,fabio.rossi}@iffarroupilha.edu.br

Resumo. *A aprendizagem de máquina emergiu com a necessidade de atender a uma demanda crescente por processamento de grandes volumes de dados de maneira rápida e precisa. Esta abordagem consiste no reconhecimento de padrões, a fim de treinar modelos sobre dados com características específicas para a realização de predições sobre novos dados com as mesmas características. Várias bibliotecas foram propostas a fim de suprir a necessidade de desempenho. Assim, esse trabalho apresenta uma avaliação da biblioteca de programação PyTorch em GPUs embarcadas. Os resultados indicam que a utilização das GPUs proporciona desempenho de 87,6 vezes melhor quando comparado com execução sobre CPUs.*

1. Introdução

Aprendizagem de máquina (*machine learning*) [Condie et al. 2013] consiste no ramo da computação que visa reconhecimento de padrões através do treino de modelos sobre dados com certas características, e depois deste modelo treinado, infere predições ou classificações sobre novos dados de mesma característica. O crescente volume, complexidade e variedade de dados associado ao aumento no poder computacional com baixo custo, fez com que técnicas de aprendizagem de máquina ganhem um novo impulso no que tange à análise desses dados de forma rápida, precisa e automática. Uma das técnicas de aprendizagem de máquina que está fortemente ligada a análise de grandes volumes de dados é a aprendizagem profunda (*deep learning*) [Deng and Yu 2014]. Essa técnica é baseada em redes neurais convolucionais, ou seja, apresenta redes neurais multicamadas inspiradas em modelos biológicos do cortex visual, onde neurônios corticais individuais respondem a estímulos de regiões restritas representando sub-regiões da visão. Isso pode melhorar o desempenho das aplicações pois permite um melhor balanceamento da quantidade de filtros por estágio e por profundidade. Porém, isso exige grande poder de processamento.

A fim de sanar este problema, bibliotecas de programação foram desenvolvidas com o objetivo de atender a demanda por processamento e análise de grandes volumes de dados via aprendizagem de máquina, e ao mesmo tempo executar tal processamento sobre arquiteturas computacionais atuais que oferecem grandes quantidades de processadores gráficos (*GPUs - Graphic Processor Units*) [Harris 2008]. A biblioteca avaliada por este trabalho é o PyTorch[1], que permite características de alto nível, tais como computação de

[1]http://pytorch.org/

tensores e redes neurais profundas. As métricas avaliadas foram: desempenho, consumo de energia e EDP (*Energy-Delay Product*) [Blem et al. 2013], que avalia o equilíbrio entre desempenho e economia de energia). O artigo está organizado da seguinte maneira: a Seção 2 apresenta o ambiente de testes, *benchmarks* e parâmetros avaliados; a Seção 3 apresenta e discute os resultados obtidos; finalizando na Seção 4 com nossas conclusões e trabalhos futuros.

2. Ambiente de Execução

A Figura 1 apresenta o testbed utilizado neste trabalho - NVidia Jetson TX2 - um *System On Chip* (SoC) que combina os processadores NVidia Denver 2 *dual-core* 2.0-GHz e ARM Cortex-A57 2.0-GHz *quad-core*, além de uma GPU NVIDIA Pascal integrada de 256 núcleos. O SoC provê uma memória DRAM de 8GB. Além disso, o SoC provê leitura de métricas de leituras digitais da potência, tensão e corrente via um sensor de corrente INA226. O sistema operacional presente no SoC, trata-se do Ubuntu 16.04 LTS (kernel 4.4.15-tegra). A medição da corrente de energia em miliamperes foi obtida através do circuito integrado INA 226 (sendo que esses valores foram posteriormente convertidos para Joule), e os resultados são médias de 10 execuções.

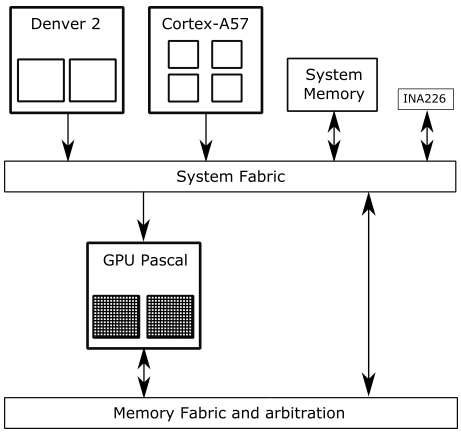


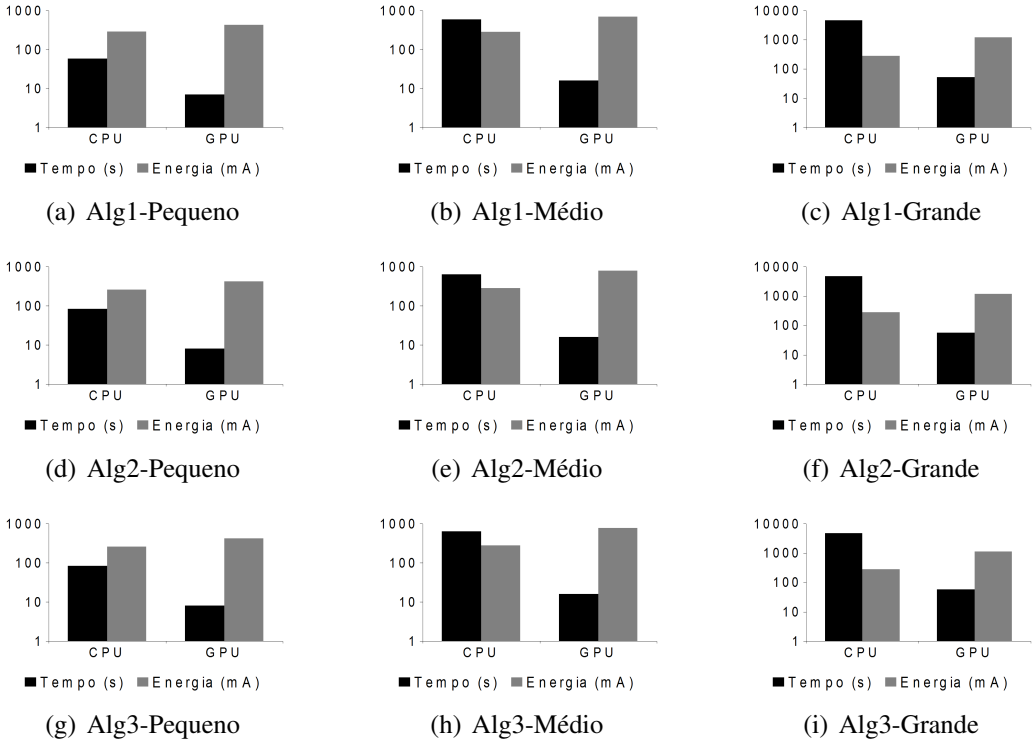
Figura 1. Arquitetura da NVidia Jetson TX2

As aplicações consideradas nesta avaliação foram: *Alg1 - PyTorch Tensors*: é uma implementeção do algoritmo *PyTorch* que segundo [Paszke et al. 2017] caracteriza-se pela utilização de tensores sendo esses generalizações de uma matriz que pode ser indexada em n-dimensões. A utilização deste algotitmo provém ao PyTorch possibilidades como a utilização de GPUs para a aceleração de cálculos numéricos atravéz da definição de um novo tipo de dados. *Alg2 - PyTorch: Variables and autograd*: fornece classes e funções que implementam a diferenciação automática de funções com valores escalares arbitrários para todas as operações em tensores. A diferenciação automática pode ser usada para automatizar a computação de caminhos em redes neurais com o uso de variáveis, podendo definir redes complexas que podem ser executadas em GPUs [Paszke et al. 2017]. *Alg3 - PyTorch* com funções adicionais: permite definir novas funções com subclasses do *torch.autograd*, modificando as funções para realizar operações específicas como, por exemplo, implementar uma função de ativação chamada *Rectified Linear Units* (ReLU), de modo que a rede neural processe os dados com base nessa função de ativação.

A Tabela 1 apresenta os três tamanhos de entrada utilizados nos testes, além de seus respectivos parâmetros: tamanho do lote (N), dimensão de *input* (D_in), dimensão da camada oculta (H) e dimensão de *output* (D_out).

Tabela 1. Entradas de dados utilizadas nos 3 algoritmos.

Tamanho	N	D_in	H	D_out
Pequeno	64	15000	100	150
Médio	192	20000	300	200
Grande	384	40000	600	400



(a) Alg1-Pequeno (b) Alg1-Médio (c) Alg1-Grande

(d) Alg2-Pequeno (e) Alg2-Médio (f) Alg2-Grande

(g) Alg3-Pequeno (h) Alg3-Médio (i) Alg3-Grande

Figura 2. Diferentes tamanhos de entrada com 500 interações.

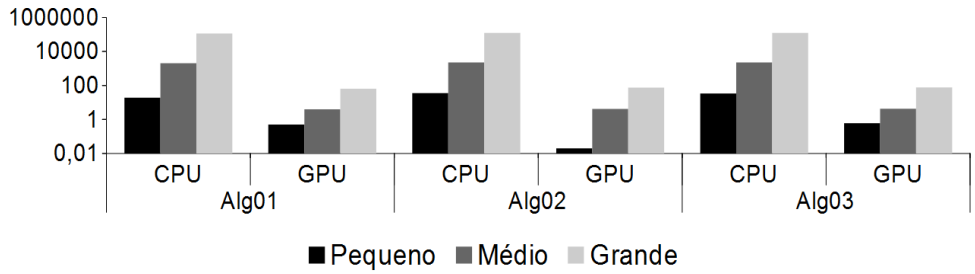


Figura 3. Cálculo do EDP.

3. Resultados

Os resultados indicam que a utilização da GPU gera um aumento significativo no desempenho das aplicações. Todas as implementações, mesmo com diferentes volumes

de dados, apresentaram aumento de desempenho, tendo o primeiro algoritmo com grande carga de trabalho (Figura 2(c)) obtido o resultado mais significativo, com melhora de 87,6 vezes no seu tempo de execução. Os resultados do consumo de energia e do desempenho das demais aplicações podem ser observados junto à Figura 2. O aumento de desempenho obtido através da utilização da GPU ocorre pelo fato de que o PyTorch integra bibliotecas da NVidia que foram desenvolvidas a fim de maximizar o desempenho deste framework com diferentes cargas de trabalho. É possível observar também que as aplicações apresentaram melhora de performance mediante o aumento da carga de trabalho. Isto ocorre devido ao grande número de cores disponíveis pelo embarcado adotado.

Nesse sentido, considerando o acréscimo no desempenho das aplicações e o aumento no consumo de energia, o cálculo do EDP (dado pelo produto do consumo de energia em Joules e o tempo de execução) foi aplicado com o intuito de verificar o equilíbrio entre o consumo de energia e desempenho. Para todos os algoritmos testados, existe melhoria expressiva no EDP em relação à execução apenas sobre CPUs (Figura 3). Os resultados coletados indicam que a utilização da GPU é uma alternativa viável para execução deste framework. Apesar da sua utilização ocasionar o acréscimo do consumo de energia do embarcado, a sua adoção gera aumento relevante no desempenho das aplicações. Assim, mesmo com o aumento no consumo de energia, o dispositivo executa suas tarefas de maneira mais eficiente, o que ocasiona os resultados positivos em EDP.

4. Conclusão e Trabalhos Futuros

A aprendizagem de máquina emergiu com a crescente necessidade de análise de dados de maneira rápida e precisa. Logo, diversas bibliotecas de programação estão sendo desenvolvidas para atender tal demanda. Nesse sentido, este trabalho realizou uma análise de desempenho da biblioteca PyTorch sobre GPUs embarcadas. Os resultados indicam ganho de até 87,6 vezes na utilização da GPU em relação à CPU. Assim, GPUs apresentam-se como uma alternativa viável para a execução da referida biblioteca. Como trabalhos futuros, almeja-se a avaliação das bibliotecas TensorFlow e Theano.

Referências

Blem, E., Menon, J., and Sankaralingam, K. (2013). Power struggles: Revisiting the RISC vs. CISC debate on contemporary arm and x86 architectures. In *Proceedings of the IEEE 19th International Symposium on High Performance Computer Architecture (HPCA2013)*, pages 1–12.

Condie, T., Mineiro, P., Polyzotis, N., and Weimer, M. (2013). Machine learning for big data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 939–942, New York, NY, USA. ACM.

Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. *Found. Trends Signal Process.*, 7(3–4):197–387.

Harris, M. (2008). Many-core gpu computing with nvidia cuda. In *Proceedings of the 22Nd Annual International Conference on Supercomputing*, ICS '08, pages 1–1, New York, NY, USA. ACM.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.