Entrega 03 - MAC6931

Frederico Meletti Rappa - 15601846

1 Introdução

A aceleração de redes neurais em FPGAs é uma tarefa desafiadora devido a crescente complexidade dos modelos, além dos desafios inerentes de síntese de hardware especializado. Nos relatórios anteriores, foram expostas tentativas da execução de inferência de Graph Neural Networks no hardware. Essa tarefa mostrou-se desafiadora tanto a partir de reprodução de hardwares descritos em artigos como de ferramentas de High Level Synthesis (HLS), em especial devido à irregularidade e complexidade das estruturas dos modelos e à relativamente escassa literatura relacionada ao tema. Desse modo, a abordagem dos estudos em andamento foi direcionada para redes neurais convolucionais, uma vez que a literatura no tema é mais extensa e os modelos apresentam estrutura menos complexa se comparada a GNNs. Desse modo, o relatório apresenta um experimento de síntese de hardware para inferência de uma rede neural convolucional simples, e comparação com sua execução em CPU e GPU.

2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) são uma classe de redes neurais aplicadas especialmente no processamento de imagens, como reconhecimento de imagens, detecção de objetos e segmentação. Sua principal característica é a capacidade dos modelos de extrair padrões por meio de "camadas convolucionais", aplicando operações de convolução sobre as imagens e reduzindo a necessidade de engenharia manual de *features* com características que as descrevem. Essas redes exploram a estrutura local das imagens para capturar informações relevantes de forma eficiente, tornando-se fundamentais para diversas aplicações de inteligência artificial, em especial em visão computacional. Assim, as CNNs se beneficiam da organização espacial fixa das imagens, que permite a aplicação sequencial de filtros por meio de convoluções para a extração e processamento de características.

Conforme mencionado, CNNs são amplamente utilizadas em tarefas de visão computacional. Para tal, comumente utilizam-se ferramentas como Pytorch e Keras, baseadas em Python, para a definição dos modelos, uma vez que têm usabilidade e flexibilidade simples, além de permitirem a execução do treinamento e inferência por meio de código compilado em CPUs e GPUs de forma nativa.

3 Metodologia

Como citado anteriormente, este relatório tem como objetivo descrever os experimentos iniciais realizados com redes neurais convolucionais (CNNs) em ambientes de execução com CPU e GPU, bem como os primeiros projetos de síntese de um modelo simples para implementação em FPGA.

O modelo proposto visa classificar imagens. Para isso, foi utilizado o dataset MNIST (*Modified National Institute of Standards and Technology*), amplamente utilizado na área de aprendizado de máquina. Esse conjunto de dados contém imagens de dígitos manuscritos de 0 a 9, e é frequentemente empregado como benchmark para validação de modelos de classificação de imagens, principalmente em experimentos iniciais e

comparativos, uma vez que não requer grande complexidade na arquitetura do modelo para ser processado. O dataset contém 60.000 imagens para treinamento e 10.000 para teste, e cada imagem possui resolução de 28x28 pixels e está em *greyscale*, facilitando o pré-processamento e o treinamento de modelos simples. Apesar de existirem arquiteturas de modelos e conjuntos de dados mais complexos, o MNIST se mostra uma maneira de validar abordagens iniciais e estabelecer uma base para experimentos mais avançados. Além disso, seu formato simples é adequado para exploração em arquiteturas reconfiguráveis como FPGAs, tornando possível analisar limitações, consumo de recursos e desempenho sem as complexidades adicionais de tarefas como classificação de imagens coloridas ou inferência em grafos.

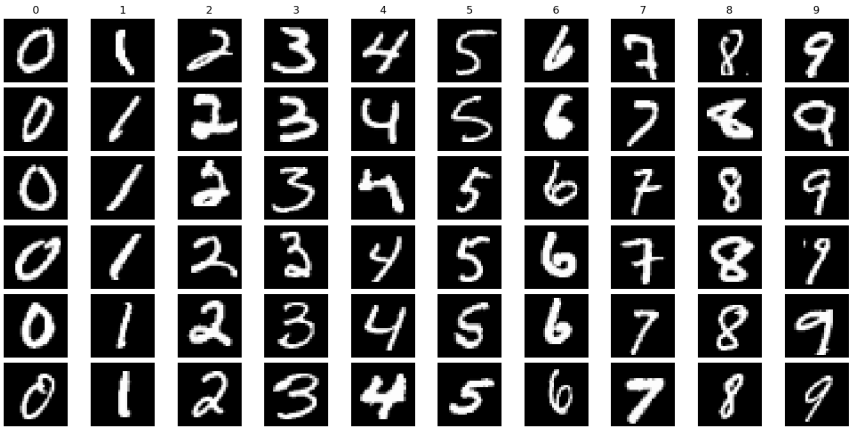

Figure 1: Exemplos de imagens do dataset MNIST. Cada imagem representa um dígito manuscrito entre 0 e 9.

Por se tratar de um problema relativamente simples, como mencionado, a tarefa de classificação do MNIST pode ser resolvida com um modelo de relativamente baixa complexidade. No contexto de CNNs, isso se traduz em uma arquitetura com poucas camadas, reduzido número de parâmetros e operações computacionais menos intensivas, o que favorece sua posterior implementação em hardware como FPGAs. Com base em estudos anteriores de CNNs, propõe-se a arquitetura de CNN exposta na Figura 2. O modelo consiste em apenas uma camada convolucional, responsáveis pela extração inicial de padrões espaciais nas imagens por meio de operações de convolução; seguida por uma camada de *pooling* para redução de dimensionalidade. Finalmente, uma camada densa *fully connected* faz a classificação a partir das features extraídas. Por ser um modelo relativamente simples, tem um número comparativamente baixo de parâmetros treináveis, 8050.

O modelo foi desenvolvido utilizando Pytorch, conforme mencionado anteriormente, e treinado com 60.000 imagens do dataset. O treinamento gera matrizes de pesos e bias para cada camada, que foram exportadas e convertidas em matrizes em arquivos de cabeçalho C. Finalmente, o modelo é reproduzido usando C, a fim de comparar com a performance de inferência obtida usando o modelo em Python e posteriormente na implementação em FPGA. A fim de representar o dataset de teste para os testes em C, as imagens e classificações foram reproduzidas em arquivos textos, parcialmente baseado no que foi feito em um projeto desenvolvido na Universidade de Parma. Embora existam métodos mais eficientes de codificação e armazenamento de imagens, optou-se por manter a abordagem simples para simplificar a depuração e do código gerado.

Por fim, o modelo em C foi adaptado com diretivas específicas de High-Level Synthesis (HLS), permitindo sua simulação e análise de desempenho em FPGAs. Essa adaptação visa avaliar a latência de inferência, fornecendo uma base para a otimização futura do modelo para aplicações em hardware.

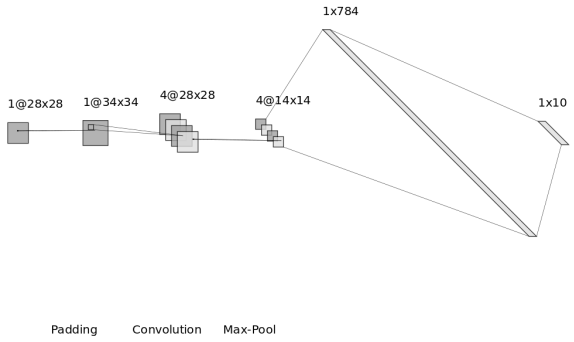

Figure 2: Diagrama da arquitetura do modelo utilizado

4 Resultados

O modelo foi executado em CPU a partir da implementação em Python e C em duas máquinas, com processadores Intel i5-8265U com 1,60 GHz e i7-8700 3,20 GHz. Além disso, foram executados em uma GPU GeForce GTX 1070 e simulados em FPGA Xilinx Alveo U55C. Os resultados comparados podem ser vistos na Tabela 1.

Plataforma	**Backend**	**Tempo Total (s)**	**Tempo Médio ± DP (ms)**	**Acurácia (%)**	**Consumo Energético Médio (W)**	**Recursos**
CPU - i5-8265U	PyTorch	1.5523	0.1659 ± 0.0812	98.07	–	–
	C	1.8155	0.1837 ± 0.0207	98.00	–	–
CPU - i7-8700	PyTorch	21.0468	1.9561 ± 0.5085	98.07	–	–
	C	1.5804	0.1587 ± 0.0151	98.00	–	–
GPU - GTX 1070	PyTorch	1.7359	0.1701 ± 0.3756	97.53	30.18	–
FPGA - Alveo U55C	HLS (simulado)	37.7860	3.7045 ± 0.1923	97.64	– (simulado)	LUTs: 82% DSPs: 29% BRAMs: 41%

Table 1: Comparação de desempenho da inferência da CNN

Observa-se que o modelo apresenta desempenho semelhante em termos de tempo médio de inferência entre a implementação em C e a versão em Python utilizando PyTorch no processador i5-8265U. No entanto, a implementação em C se mostrou ligeiramente mais lenta nesse caso, com um tempo médio de 0.1837 ms comparado a 0.1659 ms da versão em Python. Nesse caso, é possível que a abordagem com Pytorch utilize internamente ferramentas que permitam o melhor uso do processador, como maior paralelismo, por exemplo, o que não foi feito explicitamente na versão em C. Entretanto, dado a pequena dimensão do modelo e a diferença de centésimos de milésimos de segundo, seriam necessários testes em modelos mais robustos para verificar se o comportamento se mantém. No processador i7-8700, por sua vez, a versão em C teve desempenho superior, sendo significativamente mais rápida (0.1587 ms) em relação ao tempo médio da versão em Python (1.9561 ms), mesmo após múltiplos testes, de modo que o tempo de execução foi cerca de 10 vezes superior aos demais casos. Este resultado mostra-se surpreendente considerando que o código executado entre os processadores foi idêntico, e dada a performance obtida no mesmo hardware na versão em C. Além disso, considerando o maior número de cores e maior frequência, seria esperado possivelmente maior performance em relação ao processador i5. Tais diferenças podem ser possivlemnte explicadas por

limitações na versão do Pytorch utilizadas, ou possíveis limitações da máquina utilizada.

Na GPU GTX 1070, o modelo alcançou uma das menores latências, com tempo médio de inferência de 0.1701 ms, o que mostra o potencial das GPUs para acelerar tarefas de inferência mesmo em modelos simples. No entanto, esse desempenho vem acompanhado de maior consumo energético, registrado em média em 30.18 W, o que pode ser um fator limitante em aplicações embarcadas ou de baixo consumo. Além disso, o desvio padrão indica que houve grande variabilidade no valor se comparado aos demais experimentos. O gráfico da Figura 3 mostra o consumo de energia da CPU medido durante a sequência de testes. O consumo foi medido por funcionalidades nativas da CLI da GPU em intervalos de 50 ms, e o gráfico mostra o consumo imediatamente antes e depois da inferência. Nota-se que há um aumento repentino do consumo no início dos testes com GPU, como esperado, que se reduz e retorna a média ao final.

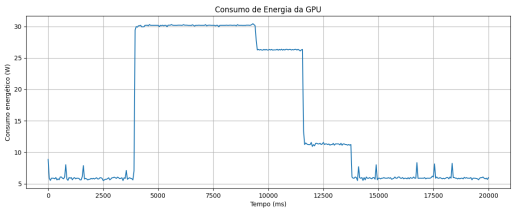

Figure 3: Consumo energético da GPU

Por outro lado, a simulação do modelo no FPGA Xilinx Alveo U55C apresentou o maior tempo médio de inferência entre todas as plataformas testadas, com 3.7045 ms. Nesse caso, a diferencia poderia ser explicada que o valor se refere à simulação via High-Level Synthesis (HLS), que diferentemente da execução real em hardware, poderia introduzir sobrecargas de simulação e não refletir fielmente o desempenho do sistema final implementado. Além disso, é possível dada a estrutura do modelo e dos dados, o hardware simulado realmente não foi capaz de obter performance comparável aos demais, de modo que poderia ser refinado para melhores resultados. A análise de uso de recursos mostra que a implementação ocupa 82% das Look-Up Tables (LUTs) disponíveis, 29% dos DSPs e 41% dos BRAMs da FPGA, o que indica que otimizações adicionais podem melhor explorar os recursos da FPGA. O hardware projetado pode ser visto na Figura 4, gerado a partir da ferramente de síntese Vivado, em que se nota o paralelismo obtido no design.

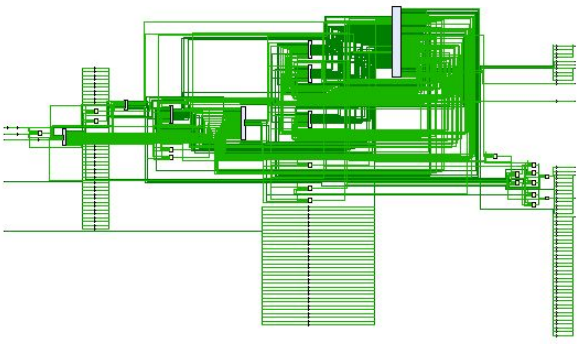

Figure 4: Visualização do esquemático do hardware gerado para o modelo

Nota-se, além disso, que a precisão das previsões se manteve semelhante em todas as plataformas, com pequenas variações. As diferenças podem ser explicadas, potencialmente, por truncamentos na conversão dos pesos entre os modelos treinados em Python e as matrizes definidas em C. Além disso, limitações da HLS podem explicar a diferença no valor na simulação. Além disso, notou-se tempo de inferência médio baixo entre todos os modelos, o que faz sentido considerando a pequena dimensão das imagens e baixo número

de parâmetros do modelo. Desse modo, é possível que modelos mais robustos permitissem resultados mais distintos entre as abordagens, e a melhor comparação de suas características. No que se refere a FPGAs, em especial, modelos maiores requereriam maior detalhamento na síntese do hardware, o que poderia permitir novas otimizações e, potencialmente, melhores ganhos de desempenho.

5 Conclusão e próximos passos

Este relatório apresentou os primeiros experimentos de inferência de uma rede neural convolucional simples em diferentes plataformas de execução: CPUs, GPU e FPGA simulada. Os resultados mostram que, apesar das diferenças de arquitetura e abordagem, todas as plataformas foram capazes de executar a inferência com alta acurácia e tempos médios reduzidos, sendo que a versão em C teve desempenho especialmente bom no processador i7-8700, e a GPU se destacou pela baixa latência, embora com maior consumo energético. Já a simulação em FPGA apresentou o maior tempo médio de inferência, o que pode ser atribuído às limitações do ambiente de simulação via HLS.

Conforme mencionado, o uso de FPGAs para inferência de redes neurais, apesar de desafiador, apresenta uma alternativa promissora, em especial para aplicações que demandam eficiência energética e reconfigurabilidade. No entanto, os experimentos indicam que a complexidade do modelo, bem como a qualidade da síntese e das diretivas aplicadas, são determinantes para o desempenho final. Como próximos passos, pretende-se reformular o modelo e comunicação com FPGA para obter resultados mais distintos entre experimentos. Além disso, as diretivas de HLS devem ser modificadas para melhor refletir a complexidade das operações de inferência. Finalmente, pretende-se fazer a síntese do hardware gerado na placa, a fim de obter dados mais verossímeis para comparação.