Entrega 04 - MAC6931

Frederico Meletti Rappa - 15601846

1 Introdução

Conforme mencionado nos relatórios anteriores, 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 iniciais da execução de inferência de Graph Neural Networks (GNNs) e Convolutional Neural Networks (CNNs) em FPGAs. 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, em continuidade ao exposto no relatório anterior.

Além disso, após os experimentos com CNNs, foram retomados os estudos de modelos de GNNs e retomadas tentativas de aceleração em FPGAs. Desde a antrega anterior, foram consultados os pesquisadores responsáveis pelo desenvolvimento de uma aplicação de GNN em FPGA já experimentada anteriormente, de modo que foram feitas novas tentativas de execução, descritas nas seções a seguir.

2 Metodologia

2.1 Convolutional Neural Networks

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

Novamente, o modelo proposto visa classificar imagens do dataset MNIST (*Modified National Institute of Standards and Technology*), que contém 10.000 imagens de dígitos manuscritos de 0 a 9 em resolução de 28x28 pixels e está em *greyscale*. Novamente, o modelo utilizado foi o mesmo do experimento anterior, descrito na Figura 2, relativamente simples a fim de obter resultados experimentais, e o que se objetiva com FPGA é acelerar a classificação de imagens por um modelo pré-treinado.

Novamente, 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 foi adaptado com diretivas específicas de High-Level Synthesis (HLS), permitindo sua simulação e análise de desempenho em FPGAs. No relatório anterior, foram expostos os experimentos com a simulação do modelo: para a síntese em hardware, são utilizados remotamente comandos de geração de hardware na máquina conectada a uma placa Alveo U55C. O processo envolve a geração de arquivos `.xo` do kernel que computa a inferência, que geram arquivos binários que definem a arquitetura da placa. Finalmente, um *host code* C++ processa as imagens e faz a chamada ao kernel executado no hardware.

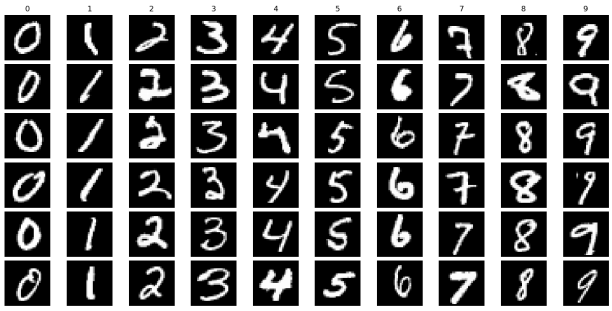

Figure 1: Exemplos de imagens do dataset MNIST. Cada imagem representa um dígito manuscrito entre 0 e 9.

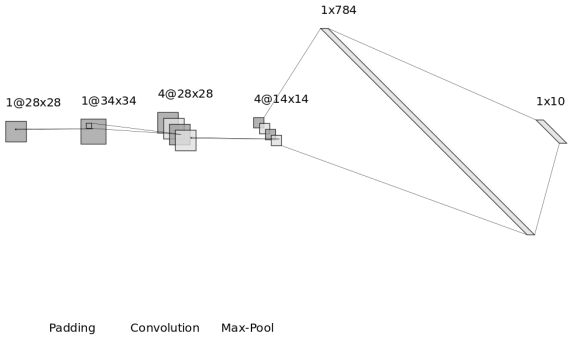

Figure 2: Diagrama da arquitetura do modelo utilizado

2.2 Graph Neural Networks

Conforme mencionado, foram retomados experimentos de aceleração de GNNs em FPGAs, cuja implementação se baseou na implementação descrita no artigo "Accelerating GNN-based SAR Automatic Target Recognition on HBM-enabled FPGA" [1]. O funcionamento da aplicação foi exposto na Entrega 02, de modo que será retomada a fim de contextualizar os experimentos. Os experimentos, como o anterior, foram executados em um cluster de FPGAs Xilinx U55C, a partir da descrição da descrição do hardware e C++ com diretivas de HLS.

O artigo propõe o uso de uma GNN para Reconhecimento Automático de Alvos (ATR) em Radares de Abertura Sintética (SAR) e um acelerador em FPGA que a processe. Diferentemente da abordagem anterior que interpreta imagens puramente como matrizes de valores numéricos, as imagens de radar são transformadas em grafos conectando com arestas pixels vizinhos e removendo os nós correspondentes a pixels desnecessários [1, 2], como pode ser visto na Figura 3. Desse modo, apesar do problema semelhante ao anterior, o artigo define uma abordagem com grafos uma vez que a maioria dos pixels das imagens é irrelevante para a classificação e, portanto podem ser ignorados. A Figura 4 mostra camadas de GNN (GNNL), pooling para redução de dimensionalidade, atenção para estabelecimento de relações entre nós, e *Nulti Layer Perceptron* (MLP), que faz a classificação a partir das características extraídas. Mais uma vez, os Kernels são desenvolvidos em C++ com diretivas de HLS que definem como regiões serão sintetizadas no hardware.

A arquitetura de hardware proposta implementa módulos para processamento dos kernels de Feature Aggregation - combinação de características de nós vizinhos - e Feature Update - atualização da matriz de *features* de um nó. Além disso, implementa kernels de MLP. A implementação no hardware original Xilinx Alveo U280 obteve 5,2x e 1,57x menor latência, 36,2x e 4,9x maior eficiência energética e 10x e 3,3x maior throughput em comparação com implementações em CPU e GPU, respectivamente.

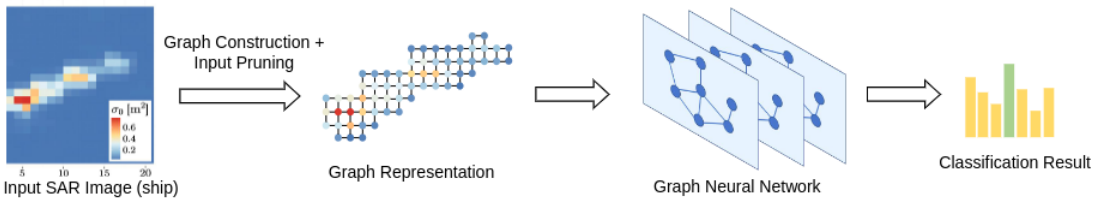

Figure 3: Representação de imagens de radas como grafos, reproduzida de [2]

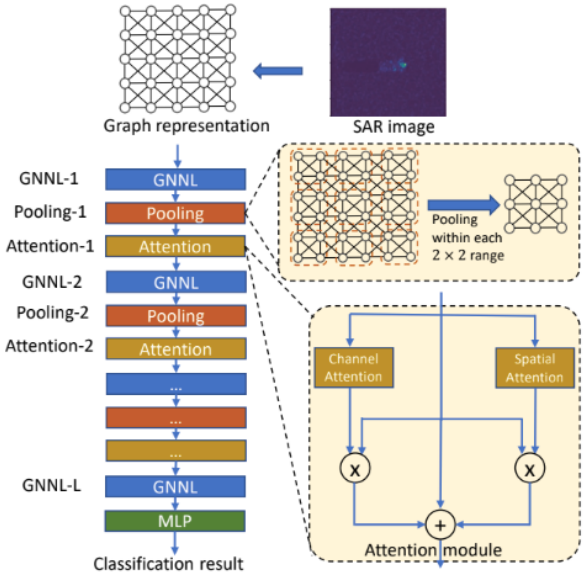

Figure 4: Diagrama de arquitetura do modelo de GNN utilizado, reproduzido de [1]

Como mencionado em relatórios anteriores, tentativas iniciais foram feitas a partir da implementação do acelerador cedida pelos pesquisadores responsáveis pelo seu desenvolvimento. Entre os diversos desafios encontrados podem ser citados a ausência de documentação detalhada e de partes do código-fonte, além de incompatibilidades de versão entre o as ferramentas de HLS e as bibliotecas utilizadas no projeto original. As configurações específicas da placa utilizada no experimento original, U280 foram adaptadas para commpatíveis com a mais recente U55C disponível para execução via SSH, entretanto apesar do esforço de adaptação do código às versões mais recentes da ferramenta e da infraestrutura disponível, não foi possível obter uma versão funcional e completa do acelerador, com erros de difícil identificação e foram problemas na compilação do código C++.

3 Resultados e desafios

3.1 Convolutional Neural Networks

Conforme exposto no relatório anterior, o modelo foi executado em duas CPUs a partir das implementações em Python e C, e em GPU a partir da implementação em Python. O comportamento no hardware foi simulado apresentou maiores tempos de inferência entre os experimentos, possivelmente por sobrecargas da siulação em CPU e oportunidades de otimização do hardware.

Embora a síntese do modelo convolucional tenha sido concluída com sucesso e os testes por meio da simulação do kernel tenham apresentado comportamento próximo ao esperado, não foi possível realizar a inferência do modelo diretamente no hardware real. Foi criado um *host code* simples em C++ usando a função de inferência, que seria processada na FPGA. O processo de síntese foi capaz de gerar um arquivo que representa o kernel (`.xo`), mas inicialmente, na etapa final de implementação que envolve a geração do binário (`.xclbin`) e a execução via *host code* no dispositivo Alveo U55C ocorreram falhas que impediram a execução completa. Nesse caso, entre os erros listados estavam erro na interface AXI de comunicação da placa. Entretanto, após fazer a correção recomendada, os mesmos erros se mantinham. A correção incluiu a revisão das diretivas HLS no código C++ para garantir compatibilidade com a plataforma da FPGA Alveo U55C.

Apesar do sucesso parcial nessa etapa, entretanto, após a compilação do *host code*, o kernel não era corretamente identificado durante a execução. Mesmo com o binário gerado corretamente e a placa identificada, ao carregar o `.xclbin` no dispositivo por meio das ferramentas Xilinx, o programa retornava mensagens de erro indicando que o kernel especificado no código não estava presente ou não podia ser localizado no binário carregado.

A fim de investigar e contornar o problema, entre as abordagens experimentas, tentou-se a geração no binário novamente pela linha de comando por comandos do Vivado, da CLI Vitis e, finalmente, pela interface gráfica das ferramentas. Além disso, foram feitas tentativas de geração do projeto a partir de um novo ambiente Vitis totalmente limpo, garantindo que não houvesse resíduos de builds anteriores que pudessem causar causando inconsistências. Outras tentativas incluiram modificação dos scripts de compilação e execução do código e kernel, o que também não gerou resultados positivos.

Esse problema é ilustrativo de um desafio recorrente no uso de FPGAs com ferramentas de alto nível como o HLS exemplificados nos relatórios anteriores: a complexidade da integração entre o kernel e o código host, além da complexidade do hardware dificultam o desenvolvimento e análise de erros. Além disso, a complexidade dos modelos de redes neurais traz um desafio adicional à depuração dos erros e geração de hardware de maneira simples.

Finalmente, em função dos novos alinhamentos sobre a execução de GNNs, os esforços foram direcionados a esses experimentos e não foi possível fazer investigações mais detalhadas.

3.2 Graph Neural Networks

Apesar dos esforços na retomada dos experimentos com Graph Neural Networks em FPGAs após comunicações com os pesquisadores responsáveis por [1], novamente foram encontrados desafios técnicos para que o modelos fosse executado em hardware com sucesso. Buscou-se utilizar o código-fonte fornecido pelos autores do artigo após as correções indicadas, que inclui a descrição de kernels em C++ com diretivas de HLS, scripts de compilação e estrutura de diretórios para integração com ferramentas Xilinx. No entanto, novamente, apesar das recomendações dos pesquisadores originais, a ausência de documentação detalhada e de componentes do código como arquivos de configuração comprometeu novamente a reprodutibilidade da solução.

Em primeiro lugar, o código atualizado continha outros erros de compatibilidade entre bibliotecas e

sintáticos. Apesar de corrigidos, estes erros apresentaram uma dificuldade adicional em um processo que, em teoria, seria simples. Na sequência, após seguir os comandos de compilação e geração de kernels atualizados, a execução do *host code* apresentou erros na identificação dos nomes, em que não eram encontrados kernels com prefixo `copy_1_`, que não foram mencionados em outros trechos da aplicação. Apesar de ser um erro de correção simples, removendo os prefixos incorretos, os erros levantaram dúvidas se a geração dos kernels com prefixos era o comportamento correto e, portanto havia erros nos experimentos feitos que os gerou com identificação incorreta; ou se novamente havia documentação e código-fonte incorreto. Durante os experimentos, erros internos de bibliotecas das plataformas Xilinx também aconteceram, em que foi necessário sua substituição ou reimplementação por funcionalidades padrão C++ com comportamento equivalente.

Além disso, após a compilação do *host code*, sua execução passou a falhar sem mensagens de erro aparente. Este erro não pôde ser resolvido, de modo que serão necessários alinhamentos futuros com os pesquisadores para a identificação de pedências. Mais uma vez, o projeto original foi desenvolvido para Alveo U280, enquanto a infraestrutura utilizada no presente experimento é baseada na U55C, mais recente, o que potencialmente é a fonte de parte das incompatibilidades. As diferenças como organização da memória e conectividade, por exemplo, exigiram modificações manuais nos arquivos de configuração, nas diretivas HLS e nos parâmetros de *linking* do projeto. Entretanto, os erros de compilação e execução persistiram.

Essas falhas evidenciam uma limitação significativa da abordagem baseada em HLS para aplicações com GNNs: a difícil portabilidade e reusabilidade de aceleradores, conforme mencionado em relatórios anteriores. Embora a aplicação original tenha obtido resultados expressivos em desempenho e eficiência energética, seu forte acoplamento tornam sua adaptação para outros domínios extremamente custosa. Além disso, novamente, a alta complexidade estrutural de GNNs impõe desafios adicionais à modelagem e apuração em HLS.

4 Conclusão e próximos passos

Os experimentos descritos neste relatório evidenciam novamente os desafios significativos associados à aceleração de redes neurais em FPGAs, especialmente no contexto do uso de ferramentas de High-Level Synthesis (HLS). Ao comparar estruturas de CNNs e GNNs, nota-se que apesar de modelos convolucionais apresentarem uma estrutura mais regular e uma literatura mais madura se comparados a GNNs, desafios similares impossibilitaram a obtenção de resultados práticos em aceleração das redes.

Como próximos passos, espera-se aprofundar a investigação das causas específicas das falhas de integração entre kernel e host no caso das CNNs e GNNs a fim de isolar e diagnosticar os erros e obter resultados para comparação com os tempos de inferência em CPU e GPUs descritos anteriormente. Em paralelo, objetiva-se esclarecer pendências que explicam os desafios na execução de GNNs, para que se possa também avançar nos estudos desses modelos.

References

[1] Bingyi Zhang, Rajgopal Kannan, Viktor Prasanna, and Carl Busart. Accelerating gnn-based sar automatic target recognition on hbm-enabled fpga. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7, 2023.

[2] Bingyi Zhang, Sasindu Wijeratne, Rajgopal Kannan, Viktor Prasanna, and Carl Busart. Graph neural network for accurate and low-complexity sar atr. *arXiv preprint arXiv:2305.07119*, 2023.