Acelerando Convoluções em Dispositivos Reprogramáveis

Welbert Hime Lino Castro[1], **Fábio Luís Livi Ramos**[1], **Bruno Silveira Neves**[1]

[1]Universidade Federal do Pampa (Unipampa)
Av. Maria Anunciação Gomes Godoy, 1650 - 96460-000 - Bagé - RS - Brazil

{welbertcastro.aluno,fabioramos,brunoneves}@unipampa.edu.br

Resumo. *As Redes Neurais Convolucionais (RNC) executam tarefas como classificação e detecção de objetos, exigindo alto poder de processamento. Este trabalho descreve duas arquiteturas para dispositivos FPGA para aceleração da inferência das RNC, otimizando os acessos aos pixels nas convoluções. A versão com paralelismo apresenta ganho em latência proporcional ao número de unidades de operação utilizadas com baixo custo em área do dispositivo.*

1. Introdução

Hoje, as inteligências artificiais têm aplicações como automação de rotinas, interpretação de textos e imagens, diagnósticos médicos e suporte para pesquisas científicas, resolvendo rapidamente problemas difíceis para humanos, especialmente quando o prazo para geração da solução é exíguo, porém fáceis para computadores [Goodfellow et al. 2016]. A maioria das Redes Neurais Convolucionais (RNC) se caracteriza pela grande quantidade de camadas, parâmetros e alto tempo de execução para desenvolver com eficácia suas atividades [Paszke et al. 2016], como classificação e segmentação semântica. No caso de processamento de vídeos, cada *frame* é tratado como uma entrada da RNC. Nessas condições, torna-se inviável a inferência de imagens de alta resolução em RNC em tempo real sem a implementação de técnicas de aceleração [Sanchez et al. 2018]. Este trabalho visa, portanto, apresentar uma arquitetura voltada para dispositivos FPGA com o objetivo de acelerar o processo de inferência executado por uma RNC. Esta arquitetura tem foco na execução das operações de convolução.

2. Organização da Arquitetura

Entre os principais componentes da arquitetura se encontram os seguintes componentes: conjuntos de **Memórias de Imagens** (para imagem original e a imagem gerada), **Banco de Registradores**, **Módulo de Convolução** e contadores, apresentados na Figura 1 (a). Dentre os contadores, dois deles são denominados como **Contador x** e **Contador i**, identificam dentro das linhas, qual coluna deve ser, respectivamente, lido ou escrito, enquanto os outros dois, **Contador y** e **Contador j**, controlam a linha a ser, respectivamente, lida ou escrita nas memórias. O componente denominado **Banco de Registradores**, recebe os pixels lidos da imagem original selecionados para a janela de convolução. Este componente também organiza os pixels de forma que o **Módulo de Convolução** receba as posições corretas em relação ao filtro de convolução a ser aplicado.

Operações de convolução compartilham dados entre diferentes campos receptivos, assim, a utilização das unidades de memória para armazenamento das entradas e saídas das operação é inviável. A utilização de conjuntos de memórias para representação das imagens permite a paralelização dos acessos a diferentes janelas de convolução na

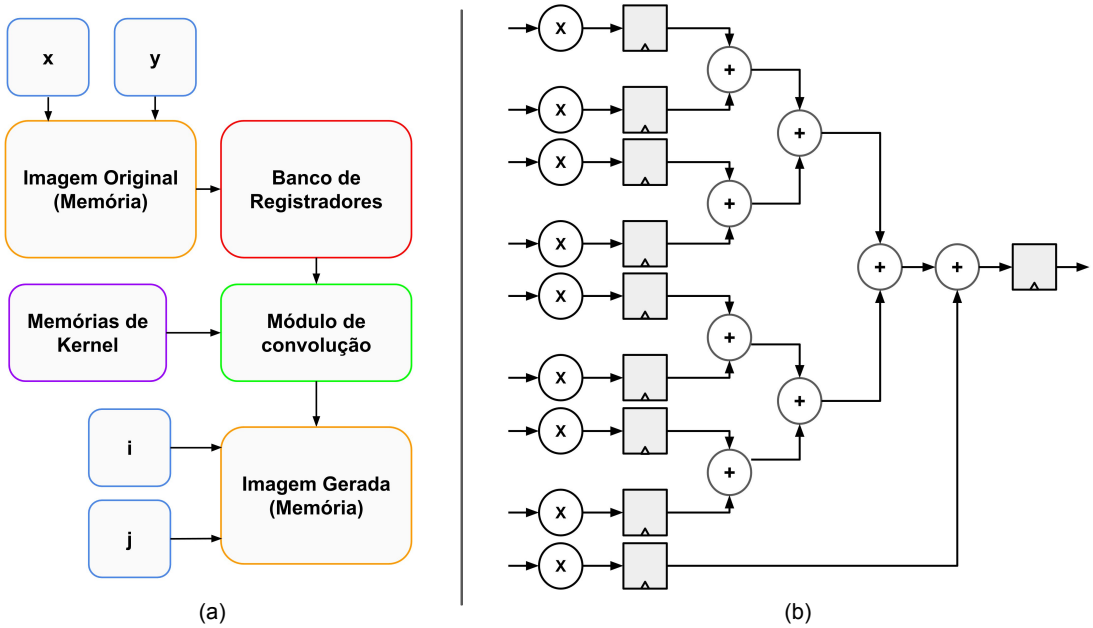

Figura 1. a) Principais blocos da arquitetura. b) Organização dos registradores, multiplicadores e somadores no Módulo de Convolução.

imagem. Foi optada pela implementação das **Memórias de Imagens** paralelizando os acessos por linhas. Assim, para imagens de 32x32 pixels, por exemplo, são utilizadas 32 unidades de memória. Desta maneira, os acessos entre colunas diferentes ficam a cargo do **Contador x**, sendo cada pixel armazenado em uma palavra de memória diferente.

3. Unidades Operativas

As entradas do **Módulo de Convolução**, controladas externamente e armazenadas no **Banco de Registradores**, recebem os pixels da imagem em que será aplicada a convolução. O número de entradas no módulo é equivalente ao número de valores do *kernel* de convolução. Para deslizar na imagem é utilizada a técnica descrita por [Solovyev et al. 2018]. Entretanto, a utilização de um único **Módulo de Convolução** pode tornar lenta a aplicação de um filtro, especialmente durante a execução de RNC, onde são aplicadas diversas operações de convolução em paralelo. A Figura 2 ilustra a utilização de três janelas de acesso simultâneo. Cada janela de acesso se desloca primeiramente no eixo horizontal da imagem, posteriormente, ao alcançar a última coluna da imagem, há o deslocamento entre linhas. A aplicação das janelas de acesso simultâneo resulta na redução do tempo proporcional ao número de janelas utilizadas, com o custo da replicação dos blocos **Banco de Registradores** e **Módulo de Convolução**. A primeira janela de cada linha necessita do acesso de todas as colunas nas memórias, enquanto o restante das janelas se alimenta do deslocamento a esquerda dos dados já consultados.

Visando paralelizar a leitura dos *kernel* de convolução são utilizadas unidades de memória para cada um dos valores, denominadas **Memórias de Kernel**. Por exemplo, com um filtro 3x3, são utilizadas 9 unidades de memória. A quantidade de endereços das **Memórias de Kernel** é proporcional as camadas convolucionais aplicadas. Os valores de saída das **Memórias de Kernel** são direcionadas aos **Módulos de Convolução**. Estes módulos efetuam o somatório de multiplicações entre os valores dos filtros e os pixels da

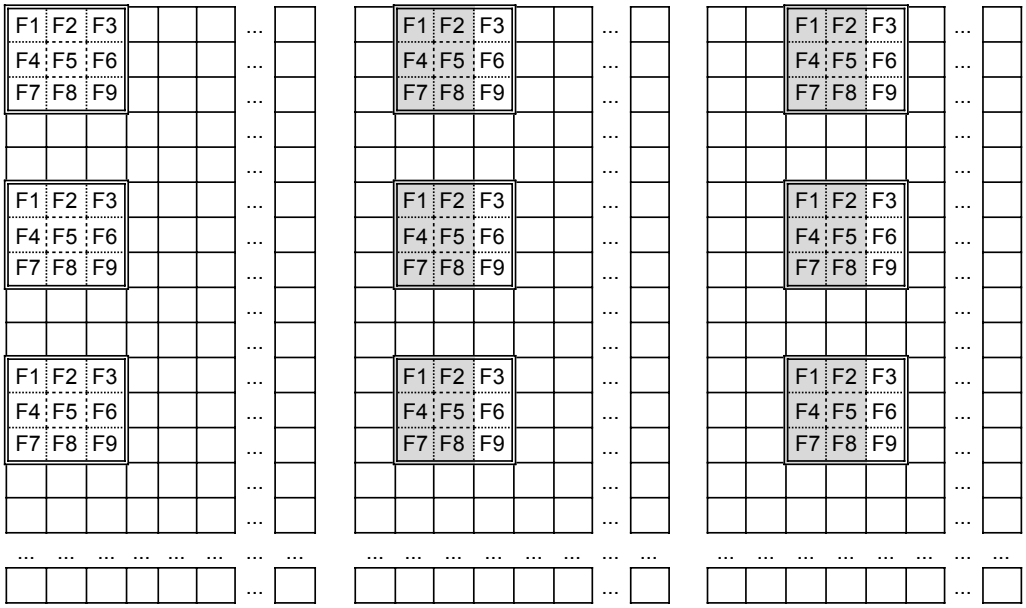

Figura 2. Aplicação simultânea do filtro de convolução em pontos diferentes.

imagem. Entre as multiplicações e somas são utilizados registradores para manutenção de estado entre os estágios do *pipeline*, apresentados na Figura 1 (b).

4. Fluxo de execução

A arquitetura foi projetada para operação em *pipeline* de quatro estágios: leitura dos pixels e deslocamento da janela de visualização, produtos dos valores dos pixels com os valores do filtro de convolução, soma dos valores gerados nas multiplicações e finalmente, a escrita do valor gerado após aplicação do filtro. A cada incremento do **Contador y** a unidade de controle da arquitetura entra em *stall* e interrompe a produção de novas saídas e escritas nas memórias para limpeza e atualização dos dados das janelas de visualização. Em seguida, o ciclo de processamento se inicia com a produção de novas saídas a cada 4 ciclos. A Figura 3 ilustra o fim do processamento de uma linha (em tons de azul) e início do processamento da linha seguinte (em tons de cinza). O fim da execução da arquitetura é identificado quando todos os pixels da imagem de entrada forem lidos e processados. Neste momento os **Contadores x e y** se encontram baixos e com o *bit* de *overflow* ativado.

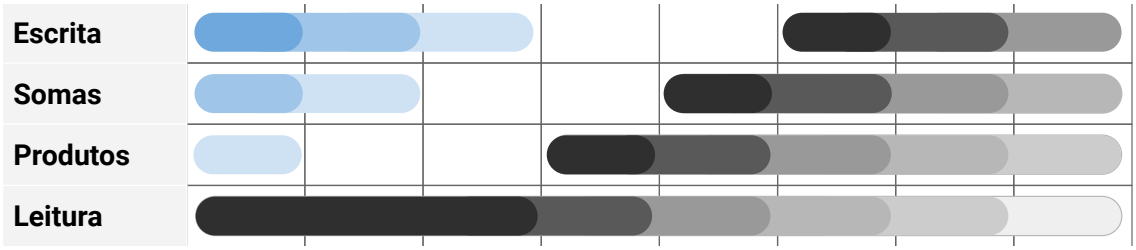

Figura 3. Estágios de processamento após incremento do Contador y.

5. Resultados Experimentais

Para obtenção dos dados de desempenho da arquitetura, foram sintetizadas duas diferentes versões no dispositivo *Xilinx Virtex-6*. A primeira versão opera com apenas uma janela

Módulos em paralelo (#)	Frequência (MHz)	Elementos lógicos (#)	Convoluções completas (#/s)	Latência (μs)
1	198.551	4,546	3,476	287.686
4	198.551	5,589	14,262	70.115

Tabela 1. Desempenho da arquitetura operando sequencial e paralelamente convoluções de filtro 3x3 em imagens de dimensão 240x240 pixels.

de visualização da imagem, enquanto a segunda versão opera com quatro janelas de visualização simultânea, portanto, com quatro **Módulos de Convolução** operando em paralelo. As arquiteturas foram sintetizadas para operação com representação de ponto-fixo de 16 *bits*, sendo 1 *bit* para o sinal de magnitude e 4 *bits* para representação decimal. Os dados de frequência, número de operações por segundo, quantidade de elementos lógicos e latência para geração da imagem final das diferentes versões são apresentados na Tabela 1. A versão da arquitetura com quatro janelas de visualização apresenta os melhores resultados em latência e operações por segundo, ocupando mais 9% dos elementos lógicos disponibilizados no dispositivo FPGA, para multiplexadores e registradores para as novas janelas de visualização, registradores para as somas e multiplicações dos estágios do *pipeline*, além de somadores e multiplicadores de ponto-fixo. Mesmo com ocupação de maior espaço do dispositivo, ainda restam 6,051 elementos lógicos livres para instalação de novas janelas de visualização e **Módulos de Convolução**. Analisando uma solução sequencial em um processador *Intel Core i5-8600* foi obtido tempo de execução de 3568 µs, 52 vezes o tempo da versão operando com quatro módulos.

6. Considerações Finais

Neste trabalho foi apresentada uma arquitetura de *hardware* para aceleração de convoluções. As convoluções são as operações de maior frequência nas RNC, influenciando diretamente no seu tempo de inferência. Foram desenvolvidas em VHDL duas versões da arquitetura para obtenção de dados de desempenho e área ocupada. As duas versões foram sintetizadas para operação com palavras de 16 *bits* representados em ponto-fixo e com sua unidade de controle operando em *pipeline*. Para os próximos passos desta pesquisa, é almejado a adaptação desta arquitetura para operação de camadas totalmente conectadas e a inferência de RNC integralmente ou em co-projeto com FPGA.

Referências

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press, Series: Adaptive computation and machine learnng series.

Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. (2016). ENet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147.

Sanchez, J., Soltani, N., Kulkarni, P., Chamarthi, R. V., and Tabkhi, H. (2018). A reconfigurable streaming processor for real-time low-power execution of convolutional neural networks at the edge. In *International Conference on Edge Computing*, pages 49–64. Springer.

Solovyev, R. A., Kalinin, A. A., Kustov, A. G., Telpukhov, D. V., and Ruhlov, V. S. (2018). FPGA implementation of convolutional neural networks with fixed-point calculations. *CoRR*, abs/1808.09945.