Arquitetura de Hardware Dedicada para CNNs: Uma Análise comparativa entre FP32 e INT8

Vanessa Aldrighi, Denis Maass, Ruhan Conceição, Marcelo Porto, Luciano Agostini

[1]Video Technology Research Group – ViTech
Universidade Federal de Pelotas – UFPEL
Pelotas, Rio Grande do Sul, Brazil

{vanessa.a, dlmaass, radconceicao, porto, agostini}@inf.ufpel.edu.br

Resumo. *Para viabilizar a implementação de Redes Neurais Convolucionais (CNNs) de alto custo computacional em sistemas restritos, a quantização de modelos, como a conversão de representações de ponto flutuante (float32) para inteiros de 8 bits (int8), é uma técnica essencial. Este trabalho quantifica os ganhos da técnica através da comparação de uma camada convolucional e uma totalmente conectada, descritas em VHDL e sintetizadas para um ASIC de 40 nm. A análise mensura a redução em área e potência da implementação em int8 frente à de float32, validando sua eficácia para hardware de baixo consumo.*

Abstract. *To enable the implementation of computationally expensive Convolutional Neural Networks (CNNs) on resource-constrained systems, model quantization, such as converting 32-bit floating-point (float32) representations to 8-bit integers (int8), is an essential technique. This work quantifies the gains of this technique by comparing a convolutional layer and a fully connected layer, described in VHDL and synthesized for a 40 nm ASIC. The analysis measures the reduction in area and power of the int8 implementation against the float32 version, validating its effectiveness for low-power hardware.*

1. INTRODUÇÃO

Redes Neurais Convolucionais (CNNs) são fundamentais na área de inteligência artificial [Ren et al. 2017, Schroff et al. 2015, Krizhevsky et al. 2012]. Contudo, seu elevado custo computacional, derivado do grande volume de operações de multiplicação-acumulação (MAC), representa um desafio significativo para sua implementação em sistemas com recursos restritos.

Tradicionalmente, CNNs utilizam aritmética de ponto flutuante de 32 bits (float32). Embora precisa, sua implementação em hardware impõe um alto custo em área e energia [Horowitz 2014] e pode gerar inconsistências entre plataformas [Conceição et al. 2025]. Para mitigar esses problemas, a quantização de modelos se tornou uma técnica essencial, consistindo na conversão dos dados da rede para inteiros de 8 bits (int8). Essa abordagem permite uma redução drástica no custo computacional com um impacto frequentemente mínimo na acurácia do modelo [Wu et al. 2016].

A eficácia da quantização em hardware é evidenciada por aceleradores de ponta [Chen et al. 2017]. Inspirado por esses avanços, este trabalho apresenta a análise comparativa de uma camada convolucional e uma totalmente conectada, descritas em VHDL e sintetizadas para um ASIC de 40 nm, com o objetivo de isolar e quantificar os ganhos em área e potência obtidos com a quantização.

2. METODOLOGIA

2.1. Arquitetura de Redes Neurais Convolucionais

As CNNs processam dados visuais em camadas sequenciais. A principal, a camada convolucional, aplica filtros à imagem de entrada para gerar mapas de características que detectam padrões. Nesta operação (Equação 1), a saída $y_{k_{i,j}}$ é calculada a partir da soma ponderada entre os canais de entrada x_c e os pesos do filtro $w_{k,c}$, e seguida pela ativação, que pode ser uma Unidade Linear Retificada (*Rectified Linear Unit* – ReLU). Frequentemente, camadas de pooling são intercaladas entre as camadas convolucionais com o objetivo de reduzir a dimensão desses mapas, diminuindo o custo computacional [LeCun et al. 1998].

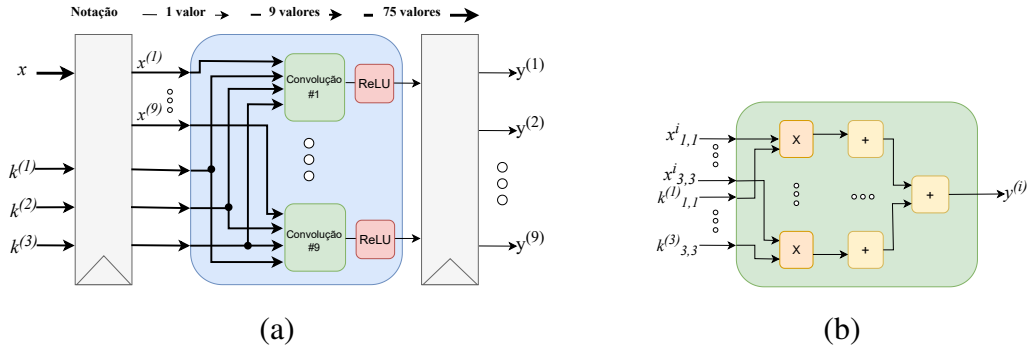
$$y_{k_{i,j}} = \max\left(0, \sum_{c=0}^{C-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} w_{m,n}^{k,c} \cdot x_{i+m,j+n}^{c}\right) \tag{1}$$

Ao final da rede, a camada totalmente conectada utiliza as características extraídas para a classificação ou regressão. A saída é obtida multiplicando o vetor de entrada x pela matriz de pesos W, seguida pela aplicação de uma função de ativação (Equação 2). A escolha da função de ativação é crucial: enquanto camadas totalmente conectadas intermediárias comumente utilizam ReLU, a camada de saída emprega funções específicas da tarefa, como Softmax para classificação ou uma ativação linear para regressão. Para o escopo deste trabalho, os blocos de hardware implementados utilizam a função ReLU.

$$y = \max(0, W \cdot x) \tag{2}$$

2.2. Projeto de Hardware e Metodologia de Avaliação

Para a análise comparativa, foram projetados em VHDL dois módulos de hardware MAC-dominantes. A camada convolucional (Figura 1) foi projetada para aplicar três filtros 3x3 sobre uma janela de entrada 5x5. Sua estrutura interna consiste em 27 multiplicadores em paralelo e uma árvore de somadores para gerar um único pixel de saída. A camada totalmente conectada (Figura 2), por sua vez, é composta por cinco neurônios operando em paralelo, onde cada um possui nove multiplicadores e uma árvore de somadores para processar seu vetor de entrada.



(a) (b)

Figura 1. (a) Arquitetura da camada convolucional e (b) bloco da convolução.

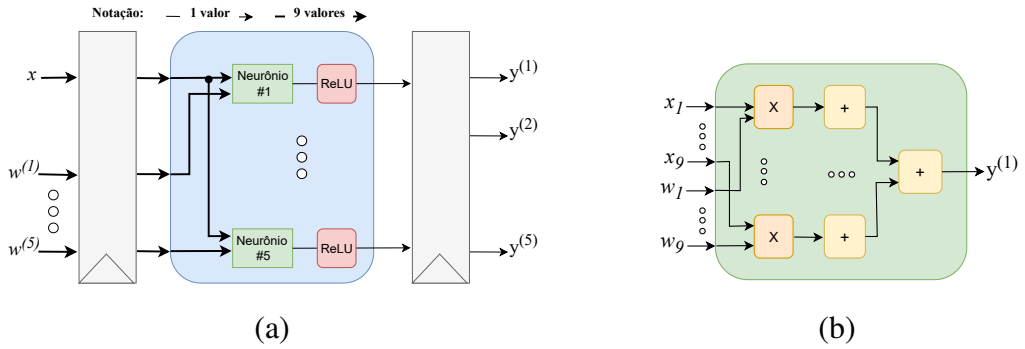


Figura 2. (a) Arquitetura da camada totalmente conectada e (b) bloco do neurônio.

Na descrição em VHDL, a versão float32 teve todas as unidades aritméticas implementadas utilizando um pacote VHDL de ponto flutuante [Bishop 2010]. Para a versão int8, as multiplicações e somas foram descritas com os operadores padrão da linguagem, que são sintetizados em circuitos aritméticos de inteiros. Adotou-se um esquema de quantização simétrica para a versão int8, onde os valores são mapeados para a faixa de [-127, 127].

3. RESULTADOS E DISCUSSÃO

A síntese em ASIC (Tabela 1) confirma que a quantização para int8 promove uma redução drástica de recursos, superando 95% em área e 97% em potência. Essa economia provém da menor complexidade dos circuitos de inteiros frente às custosas unidades de ponto flutuante, validando a abordagem para o desenvolvimento de hardware energeticamente eficiente para dispositivos de borda.

Tabela 1. Resultados de Síntese Comparativos com frequência de 100 MHz. Fonte: Autoria própria.

Módulo	**Ponto Flutuante (32-bit)**		**Inteiro (8-bit)**		**Redução (%)**	
	Potência (mW)	**Área (x10³ Gates)**	**Potência (mW)**	**Área (x10³ Gates)**	**Potência**	**Área**
Camada Convolucional	293,94	1254,30	20,44	100,67	93,04	91,97
Camada Totalmente Conectada	695,23	1197,94	4,23	20,11	99,39	98,31
Total	**989,17**	**2449,24**	**24,67**	**120,78**	**97,50**	**95,06**

As arquiteturas aqui analisadas representam o núcleo computacional para janelas de entrada de tamanho fixo. Sua aplicação em sistemas práticos, que processam imagens maiores, demandaria uma lógica de sistema adicional, como buffers de linha e controle de fluxo, para serializar a operação e reutilizar estes núcleos. Portanto, embora a escalabilidade envolva desafios de integração, a presente análise de custo do núcleo aritmético é fundamental, pois estabelece a base de hardware para o projeto de qualquer acelerador completo e mais complexo. Trabalhos como os de Jacob et al. [Jacob et al. 2018] de-

monstram que a diferença de acurácia entre modelos float32 e suas versões quantizadas é mínima, justificando o foco na otimização de hardware.

4. CONCLUSÕES

Este trabalho demonstrou que a quantização para inteiros de 8 bits em hardware ASIC dedicado para camadas de CNNs resulta em ganhos de eficiência superiores a 95% em área e 97% em potência, frente à implementação tradicional em ponto flutuante. Como trabalhos futuros, destacamos a integração dos módulos em uma rede funcional para avaliar não apenas métricas de hardware, mas também o impacto da quantização na acurácia e a otimização da arquitetura com técnicas de paralelismo.

Referências

Bishop, D. (2010). float_pkg_c.vhdl. [Online]. Available: `https://github.com/FPHDL/fphdl`.

Chen, Y.-H., Krishna, T., Emer, J. S., and Sze, V. (2017). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138.

Conceição, R., Porto, M., Peng, W.-H., and Agostini, L. (2025). Cross-platform neural video coding: A case study. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, Londres, Reino Unido.

Horowitz, M. (2014). Computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 10–14, San Francisco, CA, USA.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823.

Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4820–4828, Las Vegas, NV, USA.