Arquitetura Adaptável para Execução de Redes Neurais Artificiais em Dispositivos FPGA

Welbert H. L. Castro[1], Gabriel L. Silva[1], Igor O. Fonseca[1], Milton R. Heinen[1], Bruno S. Neves[1]

[1] Universidade Federal do Pampa (UNIPAMPA) – Bagé – RS – Brasil

{welberthime, gabriel18.lopes, igorfonsecaigor}@gmail.com

{milton.heinen, brunoneves}@unipampa.edu.br

RESUMO

Dentro do campo de Inteligências Artificiais, as Redes Neurais Artificiais (RNA) recebem destaque pela capacidade de aprender através de processos de treinamento e sua pluralidade de aplicações, que vão desde a classificação de padrões até o cálculo de funções. A implementação de algoritmos em hardware permite a paralelização de etapas e, então, a aceleração de processamento. Este trabalho propõe uma arquitetura de hardware de propósito geral para a execução de RNA em dispositivos FPGA. Implementada através da linguagem VHDL, a arquitetura proposta processa uma camada em média a cada 3 ciclos de clock. Simulada no dispositivo EP3C25F324C6, foi atingida a frequência de clock de 106.53 MHz e necessários 65.5 Kb de memória.

1 INTRODUÇÃO

Uma Rede Neural Artificial (RNA) é um sistema projetado para emular o comportamento que o cérebro humano exerce para executar determinadas tarefas. Com ela, é possível realizar o processamento distribuído e paralelo através de unidades de processamento simples, denominadas neurônios artificiais, com propósito de armazenar conhecimento experiencial e disponibilizá-lo para utilização [Haykin, 2009]. Para isso, RNA são sistemas capazes de aprender através de conjuntos de exemplos [Nielsen, 2015]. As RNA são adequadas para problemas nos quais os dados de treinamento podem corresponder a dados de sensores complexos e ruidosos, como entradas de câmeras e microfones, além de serem aplicáveis a problemas para os quais são usadas representações simbólicas, assim como tarefas de aprendizagem com árvores de decisão o [Mitchell 1997].

Existem algumas alternativas para implementação de RNA, sendo as duas principais: (i) execução em software, com computadores convencionais, e (ii) solução em hardware específico, capaz de decrementar o tempo de execução

[Rojas, 1996]. Na prática, implementações em software têm sua velocidade limitada em processadores sequenciais. Este problema é resolvido pelo alto paralelismo que pode ser alcançado em circuitos VLSI, enquanto a implementação em dispositivos FPGA dispõe do paralelismo e das adaptações possíveis em software [Omondi et al. 2006].

Este trabalho propõe uma arquitetura que permita a execução em dispositivos FPGA de RNA já treinadas, com pesos e funções de ativação previamente definidas. Para isso, é proposto um método de codificação das camadas e os respectivos pesos de seus neurônios para alocação em memória com a implementação de funções de ativação, seu cálculo e alimentação nas camadas seguintes. A arquitetura paraleliza as operações dos neurônios e acessos a memória em uma mesma camada, formando um pipeline a partir da reutilização da mesma estrutura em todas as camadas. Assim, o atraso de propagação das entradas até as saídas se torna linearmente dependente do número de camadas até o preenchimento do pipeline, após, o hardware passará a produzir novas saídas a cada ciclo.

2 METODOLOGIA

Algumas informações importantes são necessárias para a parametrização de uma rede neural. Sua execução necessita do conhecimento de: (i) as entradas, (ii) os pesos utilizados em cada neurônio, (iii) as funções de ativação, (iv) o número de neurônios em cada camada, (v) o número de camadas e (vi) o número de saídas. Nas redes perceptron todos os neurônios de uma camada alimentam as entradas da camada seguinte com suas saídas, assim, todos os neurônios recebem os dados gerados pela camada anterior. Esta característica permite que a implementação em hardware reutilize a mesma estrutura para gerar os dados de cada camada, intercalando entre as transições das camadas, acessos às memórias buscando os parâmetros para o próximo ciclo de execução.

Embora ao adicionar mais neurônios e camadas as RNA possam executar funções mais poderosas [Goodfellow, 2016], o número de camadas e neurônios se torna um fator crucial para a área ocupada e a latência total na produção de respostas da arquitetura proposta. Além disso, o número máximo de camadas, neurônios e a precisão dos dados calculados podem ser ajustados incrementando ou decrementando a largura de palavra utilizada na arquitetura [Rojas 1996], de

modo que a solução se torne escalável e capaz de suportar redes densas, respeitando as limitações do hardware utilizado.

Unidades de memória

Implementações de redes neurais em dispositivos FPGA para processamento de imagens em tempo real utilizam memórias individuais para cada neurônio afim de possibilitar a paralelização das operações entre eles, uma vez que o uso de uma única uma unidade de memória compartilhada entre todos os neurônios de uma camada inviabilizaria a leitura em paralelo de todas os pesos de neurônio [Yang, 2003]. A estratégia de utilização de memórias individuais dedicadas a configuração de cada neurônio se torna custosa em área, porém permite a configuração em paralelo dos pesos para os neurônios, viabilizando um processamento mais efetivo do pipeline entre as camadas.

O valor recebido na entrada do neurônio é especificado por um multiplexador com sinal seletor recebido da unidade de controle, descrita na subseção *Unidade de controle*. Cada neurônio, então, recebe uma memória dedicada a memorizar sua função de ativação naquela camada, descrita como *Memória de funções*, logo, são necessárias também *n* memórias com este objetivo tendo ao menos *c* endereços.

Unidade de processamento dos neurônios

A unidade denominada *Neurônio* é responsável pelo cálculo do somatório das entradas já multiplicadas por seus pesos e por passar esse resultado à unidade responsável pelo cálculo da função de ativação através da *Unidade de Ativação*. A quantidade de unidades deste tipo é igual ao maior número de neurônios em uma camada na rede implementada, anteriormente referenciado como *n*. Fora da unidade, sua saída é registrada para utilização nas camadas posteriores.

Uma das grandes dificuldades encontradas nas implementações de redes neurais em hardware são as funções de ativação por exigirem lógicas complexas e alto tempo de processamento. Para reduzir a latência da execução da função de ativação, uma estratégia é utilizar LUT(*look up table*) para aproximação dos valores.

A utilização de LUT permite que os cálculos das funções de ativação sejam efetuados com consultas a uma memória com constantes, resultados da função calculadas previamente. Contudo, a utilização de constantes requer a aproximação

de alguns dos valores, já que o domínio das funções pode ter grandes variações nas casas decimais. Mesmo assim, [Canas, 2006] afirma que a utilização de LUT para aproximação de funções de ativação se mantém eficiente. Destarte, as funções de ativação lineares Degrau e ReLU são implementadas diretamente em hardware.

Unidade de controle

A unidade de controle da arquitetura proposta, opera com seis estados. No estado *reset*, todos os contadores dentro da arquitetura são zerados para possibilitar que o controle dos endereços de memórias seja adequadamente preparado para a execução. O fluxo de execução da arquitetura, baseia-se nas operações com unidades de memória e registradores. No estado *load\inputs* são carregadas paralelamente as entradas para execução da camada de entrada através da alteração do sinal seletor do multiplexador que antecede os neurônios. Na sequência, o sinal seletor é alterado novamente para receber as saídas dos próprios neurônios (na execução das camadas ocultas e de saída).

Após isso, inicia-se a sequência de execução e realimentação das camadas ocultas e camada de saída nos estados. Nesta sequência, são atualizados os contadores de todas as unidades de memória, registrados os somatórios das multiplicações entre entradas e pesos e calculadas as funções de ativação segundo a leitura da *Memória de Funções* de cada *Neurônio*. O ciclo de execução se encerra quando os contadores ativam um sinal indicando que todos os endereços da *Memória de Funções* foram percorridos, assim a última camada executada foi a camada de saída. Portanto, os dados gerados pela camada são direcionados aos pinos de saída e se encerra a execução da arquitetura.

3 RESULTADOS E DISCUSSÃO

Para simular a arquitetura foi implementada uma RNA de 4 neurônios por camada na linguagem *VHDL* e sintetizada com o dispositivo *Altera Cyclone III EPC3C25F324C6*. Para tal, foi gerada a frequência de clock 106.53 MHz, utilizados 294 elementos lógicos e 65.5 Kb de memória, apenas 11% do disponibilizado pelo dispositivo. É esperada a geração de saídas da primeira camada a cada 3 ciclos e então, nos ciclos seguintes os dados gerados são sobrescritos para geração das camadas ocultas, finalizando com a camada de saída.

4 CONCLUSÃO

Neste trabalho foram apresentadas decisões para implementação de uma arquitetura capaz de executar RNA em dispositivos FPGA. Os dados obtidos mostram que a arquitetura proposta pode ser utilizada para RNA mais poderosas mesmo em dispositivos mais simples.

Para trabalhos futuros é interessante a implementação de um módulo capaz de realizar a inicialização de pesos e a adaptação deles através de algoritmos de treinamento como *Back-Propagation*. Desta forma, não só a execução da rede é acelerada, mas também seu aprendizado, exigindo a leitura de amostras e cálculos para o incremento das taxas de acerto das RNA. Outra linha de trabalho futuro a ser desenvolvida é a realização das adequações necessárias para criar suporte para uma RNC (Rede Neural Convolucional), buscando realizar customizações para uso em aplicações de visão computacional.

REFERÊNCIAS

Canas, A., Ortigosa, E. M., Ros, E., and Ortigosa, P. M. (2006).FPGA Implementationof a Fully and Partially Connected MLP, pages 271–296. Springer US, Boston, MA.

Fan Yang and Paindavoine, M. (2003). Implementation of an RBF neural network on em-bedded systems: real-time face tracking and identity verification. IEEE Transactionson Neural Networks, 14(5):1162–1175.

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

Haykin, S. S. (2009).Neural networks and learning machines. Pearson Education, UpperSaddle River, NJ, third edition.

Nielsen, M. A. (2015).Neural networks and deep learning, volume 25. Determinationpress San Francisco, CA, USA.

Nwankpa, C. E., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation functi-ons: Comparison of trends in practice and research for deep learning.arXiv preprintarXiv:1811.03378 [cs.LG].

Omondi, A. R., Rajapakse, J. C., and Bajger, M. (2006).FPGA Neurocomputers, pages1–36. Springer US, Boston, MA.

Rojas, R. (1996). Neural networks: a systematic introduction. Springer Science & Busi-ness Media.