O Uso de Placa Gráfica Aplicada ao Reconhecimento Facial: Um Estudo de Caso com Redes Neurais Artificiais

Marcos R. M. Saavedra[1], **Flávio R. A. Souza**[2], **Josivaldo de S. Araujo**[3]

[1]Instituto de Ciências Exatas e Naturais – Universidade Federal do Pará (UFPA)
Rua Augusto Corrêa nº 01 Caixa Postal 66075-110 – Belém – PA – Brazil

marcosaavedraa3@gmail.com, souramon@hotmail.com, josivaldo@ufpa.br

Abstract. *Biometric apps that identify individuals from facial images are becoming increasingly popular. This article performs a facial recognition technique with neural networks using a graphics card in order to reduce the network training and testing time, using TensorFlow and Pytorch libraries.*

Resumo. *Aplicativos biométricos que identificam indivíduos a partir de imagens faciais estão se tornando cada vez mais populares. Este artigo executa uma técnica de reconhecimento facial com redes neurais utilizando uma placa gráfica com o objetivo de reduzir o tempo de treinamento e teste da rede, e para isso, foram utilizadas as bibliotecas TensorFlow e Pytorch.*

1. Introdução

Com o avanço da tecnologia é possível acessar informações, equipamentos e até mesmo lugares utilizando a autenticação, seja através do uso de cartões de acesso, ou de senhas digitadas. No entanto, esses sistemas podem ser contornados e o usuário pode ter os seus dados revelados. Por conta disso, o reconhecimento facial tem se tornado uma importante área de pesquisa para a comunidade em Segurança Computacional, pois proporciona a identificação de cada indivíduo como único, através das suas características pessoais.

Este trabalho apresenta um estudo de caso para reconhecimento facial através do uso de placa gráfica com duas bibliotecas paralelas, o TensorFlow e o Pytorch, que são utilizadas em uma Rede Neural Artificial, com o objetivo de obter ganhos de desempenho no processo de treinamento e teste. Os resultados se mostraram satisfatórios com a redução do tempo de processamento quando comparado com a execução sequencial.

2. Trabalhos Relacionados

O trabalho apresentado [Terraza Arciniegas et al. 2022] trouxe uma proposta capaz de analisar a atenção do aluno em aulas online e determinar quando ele está atento a determinada explicação. Para isso foi desenvolvida uma técnica automatizada baseada em algoritmo RNA de visão computacional com a finalidade de verificar e alterar a dinâmica das aulas quando necessário.

No trabalho de [Cristiano A. Künas and Padoin 2020] é utilizada uma rede neural artificial para detecção de sentimentos, pois é uma área recorrente da Visão Computacional. O trabalho tem como objetivo treinar uma base de dados pública e aplicar no reconhecimento de sentimentos expressados em textos e postagens utilizando python junto com os módulos *keras*, *Tensorflow* e *Scikit-Learn*.

3. Reconhecimento Facial

Para realizar o reconhecimento facil foi utilizada uma estrutura de cinco etapas que se inicia com a captura da imagem (aquisição) e finaliza com a conversão da mesma em um vetor binário que é usado como entrada para alimentar a Rede Neural Artificial (RNA). Todo o procedimento pode ser analisado em [Souza and Araujo 2021].

4. Paralelização

Atualmente existem várias bibliotecas, definidas em diversas linguagens, que são capazes de executar funções prontas em paralelo. Neste trabalho foram utilizadas as bibliotecas: TensorFlow, onde os dados são definidos em forma de tensores e processados de acordo com a definição dos dados para execução [Géron 2019], e Pytorch, que é uma biblioteca expressiva para trabalhos envolvendo aprendizado de máquina (*Machine Learning*) [Howard and Gugger 2020]. Ambas são escritas em Python.

Os testes foram executados em um computador com processador Intel Core i7 3770, 8 cores, com 8 GB de RAM, utilizando sistema operacional Linux Fedora 36 (kernel 5.18.13-200 e distribuição i3 Spin). A placa gráfica é uma Nvidia GeForce GTX 1660, com memória global de 6 GB, versão CUDA 11.6 e com 1.280 CUDA Cores.

5. Resultados

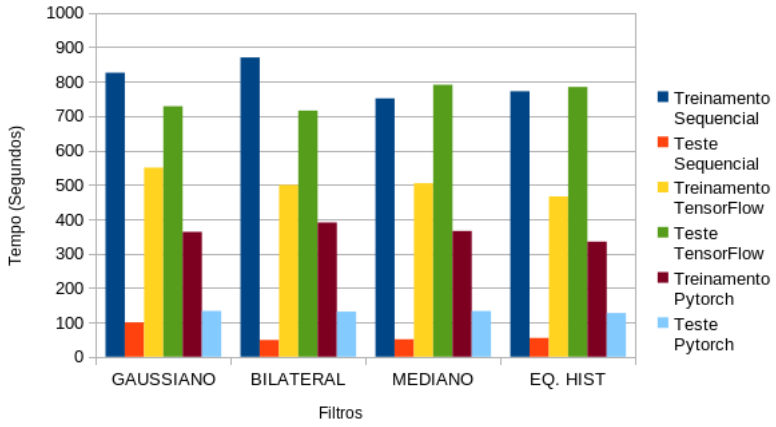
A primeira análise feita, após a execução dos algoritmos propostos no trabalho, foi a média dos tempos na função treinamento e teste para os filtros (Gaussiano, Bilateral, Mediano e Equalização de Histograma) utilizados na etapa de pré-processamento. Os resultados foram obtidos através da média de 10 execuções de cada algoritmo. O objetivo deste teste é verificar o comportamento da Rede Neural Artificial (RNA) em cada um desses filtros, os resultados podem ser observados na **Figura 1**.

Os códigos sequenciais obtiveram os maiores tempos médios de treinamento, em comparação aos outros dois códigos que utilizam as funções paralelas, contudo alcançaram os menores tempos de teste, devido ao custo de processamento da herança da classe de dados com outras classes que manipulam os cálculos nas bibliotecas do TensorFlow e Pytorch. O segundo motivo seria o *overlay* da realocação de memória na placa gráfica, nesse caso, a GPU precisa verificar os espaços livres para fazer a alocação, visto que toda essa informação é lida a partir da memória e não do disco rígido, como é feito no treinamento. Foi verificado, também, um tempo maior no treinamento sequencial utilizando o filtro bilatera. O treinamento mais eficiente foi atingido pelo filtro mediano, seguido dos filtros equalizador, gaussiano e bilateral. A fase de teste do algoritmo sequencial resultou na melhor eficiência do filtro bilateral, seguido do mediano, equalizador e gaussiano

As execuções em TensorFlow possuem uma características diferentes dos códigos sequenciais e Pytorch devido aos melhores resultados serem obtidos na fase treinamento das execuções. A ordem de eficiência na fase de treinamento ficou com o equalizador, depois bilateral, mediano e gaussiano.

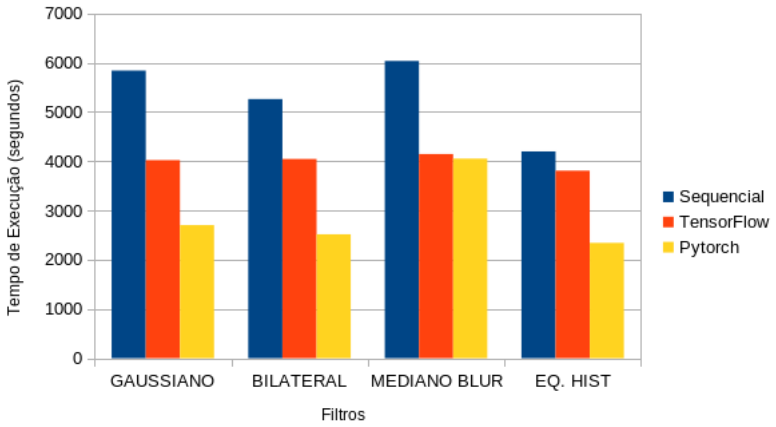
A mesma equiparação de dados para a função teste define uma proximidade entre os filtros bilateral e gaussiano, também entre o equalizador e o mediano e a melhor média de execução foi para o filtro bilateral e o pior resultado ocorreu para o mediano. Na fase

Figure 1. Média dos tempos de treinamento e teste na execução dos algoritmos no alvo 1 por filtro.



de teste, os melhores resultados obtidos foram pelo filtro bilateral, seguido pelos filtros gaussiano, equalizador de histograma e mediano.

Figure 2. Média dos tempos de treinamento e teste na execução dos algoritmos em todos os alvos por filtro.



O Pytorch resultou nas melhores médias de tempo para o treinamento, contudo ele é inferior em relação a fase de teste do código sequencial para todos filtros utilizados. O filtro equalizador de histograma possui o menor tempo de execução para o treinamento realizado.

A segunda comparação foi a partir da média total dos tempos obtidos a partir da execução dos algoritmos por filtro (gaussiano, bilateral, mediano e equalizador), foram feitas 10 execuções dos algoritmos em cada filtro e foi gerada a média dessas execuções, **Figura 2**. O Pytorch tem a melhor média de tempo, depois o TensorFlow e o sequencial. Esse fato ocorre devido a forma de organizar os dados em tensores multidimensionais, esses tensores são otimizados pelo Pytorch. Por fim, o paralelismo executado pelo Pytorch é distribuído automaticamente, tornando o processo mais rápido. O TensorFlow possui o paralelismo atribuído manualmente, tornando o processo um pouco mais demorado.

No comparativo de tempo total de execução dos testes, o pytorch com media de 2487,33 segundos tem a melhor média de tempo, depois o tensorFlow com 3842,25 segundos e o sequencial com 8258,18 segundos.

6. Conclusão

O objetivo deste trabalho foi apresentar o melhor desempenho registrado entre os testes realizados, utilizando rede neural, para um problema de visão computacional com algoritmos sequenciais e em paralelo utilizando as bibliotecas do Python (TensorFlow e Pytorch). Na comparação dos filtros utilizados na rede neural para o caso utilizado, os dados apontam que o bilateral possui o pior desempenho na fase de treinamento, exceto no tensorFlow onde obteve o segundo melhor tempo.

Na fase de testes, o filtro gaussiano possui um comportamento parecido com o treinamento para a fase de teste, dando ao mediano o pior resultado ao tensorFlow. Para os melhores resultados, o equalizador de histograma foi mais eficiente nos algoritmos com bibliotecas paralelas, e o mediano teve um desempenho melhor no algoritmo sequencial. Já na fase de teste, o bilateral possui o melhor desempenho em todos casos.

Por fim, os testes mostram que o Pytorch possui uma taxa de aprendizado melhor que o TensorFlow na execução total dos testes, tornando o resultado mais eficiente dentre os casos propostos.

References

Cristiano A. Künas, L. P. H. and Padoin, E. L. (2020). Análise de desempenho de redes neurais artificiais em plataformas cpu e gpu aplicadas no reconhecimento de sentimentos em textos. *XXI Simpósio em Sistemas Computacionais de Alto Desempenho*.

Géron, A. (2019). *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn TensorFlow*, volume 1ª Edição. Editora Alta Books.

Howard, J. and Gugger, S. (2020). *Deep Learning for Coders with fastai and PyTorch*, volume 1ª Edição. O'Reilly Media, Inc.

Souza, F. R. A. and Araujo, J. S. (2021). Analysis and evaluation of digital image enhancement techniques applied to facial recognition with neural networks. *International Conference On Information Systems And Technology Management(CONTECSI)*, pages 375–378.

Terraza Arciniegas, D. F., Amaya, M., Piedrahita Carvajal, A., Rodriguezx002D;Marin, P. A., Duquex002D;Mux00F1;oz, L., and Martinezx002D;Vargas, J. D. (2022). Students' attention monitoring system in learning environments based on artificial intelligence. *IEEE Latin America Transactions*, 20(1):126–132.