UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Detecção de câncer de pele usando Cluster Raspberry PI e a plataforma Pytorch

Dissertação de Mestrado

Elias Rabelo Matos

São Cristóvão – Sergipe

2022

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Elias Rabelo Matos

Detecção de câncer de pele usando Cluster Raspberry PI e a plataforma Pytorch

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Edward David Moreno Ordonez
Coorientador(a): Kalil Araujo Bispo

São Cristóvão – Sergipe

2022

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Matos, Elias Rabelo
M433d Detecção de câncer de pele usando Cluster Raspberry PI e a plataforma Pytorch / Elias Rabelo Matos ; orientador Edward David Moreno Ordonez. - São Cristóvão, 2022.
80 f.; il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2022.

1. Computação de alto desempenho. 2. Inteligência artificial. 3. Pele - Câncer. 4. Raspberry PI (Computador). I. Ordonez, Edward David Moreno do orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: Elias Rabêlo Matos

Em 24 dias do mês de janeiro do ano de dois mil e vinte dois, com início às 10h00min, realizou-se na Sala virtual https://meet.google.com/mty-sewd-eyn. A Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Elias Rabêlo Matos**, que desenvolveu o trabalho intitulado: **"*Detecção de câncer de pele usando Cluster Raspberry PI e a plataforma Pytorch*",** sob a orientação do Prof. Dr. **Edward David Moreno Ordonez**. A Sessão foi presidida pelo Prof. Dr. **Edward David Moreno Ordonez** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Victor Manuel Rodrigues Alves** (Uminho) e em seguida, o Prof. Dr. **Bruno Otávio Piedade Prado** (Procc/UFS) e Prof. Dr. **Kalil Araujo Bispo (**Procc/UFS**)**. Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) APROVADO "*(aprovado/reprovado)"*. Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), Resolução nº 25/2014/CONEPE e da Portaria nº 413 de 27 de maio de 2020 (Banca por videoconferência) que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 24 de janeiro de 2022.

Prof. Dr. Edward David Moreno Ordonez
(PROCC/UFS)
Presidente

Documento assinado digitalmente
gov.br Kalil Araujo Bispo
Data: 27/01/2022 18:20:12-0300
Verifique em https://verificador.iti.br

Prof. Dr. Kalil Araujo Bispo
(PROCC/UFS)
Examinador Interno - Coorientador

Assinado por: **Víctor Manuel Rodrigues Alves**
Num. de Identificação: 05910026
Data: 2022.01.30 17:19:54+00'00'

Prof. Dr .Victor Manuel Rodrigues Alves
(UMinho)
Examinador Externo

Documento assinado digitalmente
gov.br Bruno Otavio Piedade Prado
Data: 28/01/2022 15:44:54-0300
Verifique em https://verificador.iti.br

Prof. Dr.
(PROCC/UFS)
Examinador Interno

Elias Rabêlo Matos
Candidato

UNIVERSIDADE FEDERAL DE SERGIPE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO - PROCC
Sala do Seminário do DCOMP – Departamento de Computação / UFS, Av. Marechal Rondon, S/N - Jardim Rosa Elze - Tel. (79) 3194-6353. CEP: 49100-000 - São Cristóvão - Sergipe - Brasil
E-mail: secretaria.pos@dcomp.ufs.br

Para todos que insistem em fazer ciência nesse país

Agradecimentos

Quero agradecer primeiramente à minha família, à minha mãe, Lindicelma, mulher guerreira e educadora por ter me ensinado a importância dos estudos. Ao meu pai, José Araujo, por ter me ensinado tudo que não estava ligado à colégio ou faculdade. Quero agradecer a minha irmã, Ana Maria (aia), por todas as vezes que brigamos e sorrimos juntos durante toda a vida. Quero agradecer a minha avó Josefa por ser minha mãe do mesmo jeito que minha mãe. Agradeço aquele a quem dedico este trabalho e quem eu queria que estivesse aqui, meu avô José Martins, nunca esqueço das suas histórias, nem dos seus conselhos, nem paro de imaginar seu orgulho em me ver formado e agora mestre mesmo que não entendesse muito bem que curso era o meu.

Após a minha família gostaria de agradecer a todos aqueles que compartilharam seus ensinamentos comigo, desde colégio até agora, mesmo aquelas aulas que eu odiei. Gostaria de agradecer a todos os autores de artigos, livros, ou qualquer outra fonte que me ajudaram a produzir este trabalho. Ao meu coorientador Kalil, gostaria de agradecer e espero ter sido digno de sua orientação. Kalil foi meu professor apenas em Fundamentos da Computação no já longínquo primeiro período da graduação, ainda sim, um dos que mais me ensinaram . Meu orientador Edward, obrigado por ter me ajudado de perto nesses anos que passamos juntos, obrigado por ser uma inspiração na área de Computação de Alto Desempenho para qualquer um e por ser um dos professores mais divertidos que já tive. Obrigado também aos membros dessa banca.

Por ultimo gostaria de agradecer a meus velhos amigos. Obrigado a todos meus amigos de Antas, Cícero e Fátima, Aracaju e BH por terem compartilhado os melhores momentos da minha vida.

Já que o tempo fez-te a graça de visitares o Norte,
leva notícias de mim
Diz àqueles da província que já me viste a perigo: na cidade grande enfim
Conta aos amigos doutores que abandonei a escola pra cantar em cabaré
(Belchior)

Resumo

O câncer de pele provoca mortes e tem um aumento de número de casos todos os anos, ao redor do mundo. Uma das formas mais comuns de detectar o câncer de pele é o uso da regra ABCDE, regra aplicada em um conjunto de características sobre uma lesão da pele do paciente. A partir dessa forma de diagnóstico o câncer de pele pode ser tratado pela Ciência da Computação como um problema de classificação de imagens usando a Inteligência Artificial. Desde 2016 o ISIC *(International Skin Cancer Challenge)* lança competições anuais acerca da detecção de câncer de pele. Neste trabalho foi selecionado o *dataset HAM1000* que contem imagens de pele divididos em sete diferentes tipos de lesões e que faz parte dos arquivos do ISIC. Com esse *dataset* foram definidos os objetivos do trabalho: Treinar uma rede neural convolucional em um *Cluster* de Alto Desempenho de Baixo Custo usando a plataforma *Raspberry Pi*; e aplicar a técnica de transferência de conhecimento de uma rede neural convolucional para uma rede MLP que fosse executada na plataforma *Raspberry PI*. O treinamento no *cluster* não foi possível com a configuração usada nesse trabalho. E com a transferência de conhecimento foi atingido o valor de 80% de acurácia na melhor das configurações utilizadas.

Palavras-chave: Inteligência Artificial, Computação de Alto Desempenho, Câncer de Pele, *Raspberry PI*

Abstract

Skin cancer causes deaths and has an increase in the number of cases each year worldwide. One of the most common ways to detect skin cancer is the use of the ABCDE rule, applied to a set of features on a patient's skin lesion. From this form of diagnosis, Skin Cancer can be treat by Computer Science as an image classification problem using Artificial Intelligence. Since 2016 the ISIC (International Skin Cancer Challenge) has launched annual competitions towards the detection of skin cancer. In this work, the selected dataset is the HAM1000, which contains skin images divided into seven different types of lesions and which is part of the ISIC files. With this dataset, the objectives of the work were defined. Training of a convolutional neural network in a Low-Cost High-Performance Cluster using the Raspberry Pi platform; and apply the knowledge transfer technique from a convolutional neural network to an MLP network running on the Raspberry PI platform. Cluster training was not possible with the configuration used in this job. And with the transfer of knowledge, was achieved the value of 80% accuracy in the best settings used.

Keywords: Artifical intelligence, High-Performance Computing, Skin Cancer, Raspberry PI.

Lista de tabelas

Tabela 1 – Comparativo entre os estudos relacionados 44

Tabela 3 – Divisões de *dataset* 64

Tabela 4 – Divisões de *dataset* 70
Tabela 5 – Acurácia das redes 71
Tabela 6 – Resultados para a MLP treinada com a *Densenet* como professora. 74

Lista de abreviaturas e siglas

HPC - *High Performance Computing*

ISIC - *International Skin Cancer Challenge*

Colab - *Google Collaboratory*

MIPS - *Millions Instructions Per Second*

HDFS - *Hadoop Distributed File System*

ARM - *Advanced RISC Machine*

RNA - Redes Neural Artificial

Sumário

1 Introdução 14
1.1 Contextualização e motivação 14
1.2 Justificativa 15
1.3 Objetivos 16
1.4 Metodologia 17
1.5 Estrutura do documento 17

2 Fundamentação Teórica 19
2.1 Computação de Alto Desempenho 19
2.1.1 Cluster 20
2.1.2 Apache Hadoop 21
2.1.3 Colab 22
2.2 Sistemas embarcados ARM 23
2.2.1 Arquitetura ARM 23
2.3 Inteligência Artificial 24
2.3.1 Redes Perceptron 24
2.3.2 Redes Neurais Artificiais 25
2.3.2.1 Arquiteturas de rede Neurais 25
2.3.2.2 Rede Perceptron de Multicamadas 26
2.3.2.3 Redes Profundas 27
2.3.3 Tipos de Aprendizagem 27
2.3.4 Medidas de desempenho 28
2.4 Pytorch 28
2.5 Câncer de pele 28

3 Computação de Alto Desempenho em Plataformas Embarcadas 30
3.1 Revisão Sistemática 30
3.1.1 Bases Utilizadas 31
3.1.2 Termos de busca e String de busca 31
3.1.3 Critério de seleção e relevância dos artigos encontrados 32
3.2 Discussão acerca dos trabalhos relacionados 32
3.2.1 *The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures* 32
3.2.2 *A Container-based Edge Cloud PaaS Architecture based on Raspberry Pi Clusters* 33
3.2.3 *Modelling Low Power Compute Clusters for Cloud Simulation* 34

3.2.4 Comparison of Load Balancing Methods for Raspberry–Pi Clustered Embedded Web Servers . . . 34
3.2.5 *Study of Raspberry Pi 2 Quad-core Cortex-A7 CPU Cluster as a Mini Supercomputer* . . . 35
3.2.6 *Performance of a Low Cost Hadoop Cluster for Image Analysis in Cloud Robotics Environment* . . . 36
3.2.7 Avaliação de Cluster Raspberry Pi para Execução de Aplicações de Análise de Imagens Microscópicas Médicas . . . 37
3.2.8 *Performance Analysis of a Low Cost Cluster with Parallel Applications and ARM Processors* . . . 39
3.2.9 ZCluster: A Zynq-based Hadoop Cluster . . . 40
3.2.10 Acceleration of RSA Processes based on Hybrid ARM-FPGA Cluster . 41
3.2.11 Portable Parallel Computing with the Raspberry Pi . . . 41
3.2.12 A novel single-arm-worn 24 h heart disease monitor empowered by machine intelligence . . . 42
3.2.13 Performance analysis of single board computer clusters . . . 43
3.3 Interpretação dos Resultados . . . 45

4 ISIC - International Skin Imaging Colaboration . . . 48
4.1 Padrões propostos pelo ISIC . . . 48
4.2 Sumario de estudos do ISIC . . . 49
4.3 Artigos selecionados . . . 53
4.3.1 Combining deep learning and hand-crafted features for skin lesion classification . . . 54
4.3.2 Skin lesion classification from dermoscopic images using deep learning techniques . . . 54
4.3.3 Araguaia Medical Vision Lab at ISIC 2017 Skin Lesion Classification Challenge . . . 55
4.3.4 Fully Convolutional Neural Networks to Detect Clinical Dermoscopic Features . . . 55
4.3.5 Apply lightweight deep learning on internet of things for low-cost and easy-to- access skin cancer detection Apply Lightweight Deep Learning on Internet of Things for Low-Cost and Easy-To-Access Skin Cancer Detection . . . 56
4.3.6 Recognition of skin melanoma through dermoscopic image analysis . . 56
4.3.7 Optimal selection of features using wavelet fractal descriptors and automatic correlation bias reduction for classifying skin lesions . . . 57
4.4 Justificativa sobre textos selecionados . . . 57

5 Preprocessamento e escolha de redes pre-treinadas . . . 59

5.1 O Modelo 59
5.2 Redes Pre-treinadas Selecionadas 60
5.2.1 Densenet 61
5.2.2 Resnet 62
5.2.3 VGG 62
5.2.4 Inception 63
5.3 Construção do modelo 63
5.4 Treinamento do Modelo 66
5.5 Considerações final do capítulo 69

6 Resultados e Discussões 70
6.1 Seleção da melhor rede 70
6.2 *Cluster* Embarcado usando o *Apache Hadoop* 71
6.3 Transferência de conhecimento 72
6.4 Considerações sobre os resultados 74

7 Conclusão 76

Referências 78

1 Introdução

Esse Capítulo apresenta a visão geral do trabalho, trazendo sua motivação e justificativa, objetivos gerais e específicos, bem como a metodologia utilizada e a estrutura do trabalho.

1.1 Contextualização e motivação

No contexto da Segunda Grande Guerra Alan Turing e sua equipe de matemáticos trabalharam para os aliados no contexto da segunda grande guerra ao decifrar os códigos da máquina nazista Enigma, dessa forma ajudando a vencer a guerra. Este evento está ligado intimamente com o que é considerado o nascimento da Ciência da Computação por se tratar de uma das primeiras máquinas programadas para resolução de um problema (SIPSER, 1996). A máquina usada para decifrar as mensagens encriptadas da Enigma foi chamada de *The Bomb* e processava os dados capturados e tentava descobrir as ordens que haviam sido enviadas aos nazistas naquele dia. Mas a grande quantidade de combinações existentes fez com fosse necessário otimizar os cálculos de alguma forma e conseguir resolver o problema antes que a codificação usada pelos nazistas fosse mudada.

A história da quebra do segredo Enigma é um caso muito explicativo de como a Computação resolve problemas do cotidiano humano. Outro problema também já evidenciado é a necessidade de processar dados de forma rápida. Para resolver o problema da necessidade de processamento a computação tem usado o paralelismo e o uso de *clusters*, assim, obtendo um grande poder de processamento desde os anos 60, onde os primeiros *clusters* foram construídos (PFISTER, 1998). Um *cluster* é a união de dois ou mais computadores.

Nas décadas subsequentes, essas técnicas foram aprimoradas e a partir de 2013, com o lançamento da plataforma *Raspberry PI*, foi observada a possibilidade de construção de *clusters* de baixo custo usando essa plataforma, tornando assim a Computação de Alto Desempenho mais acessível. O poder de processamento de um *cluster* formado por *Raspberries PI* já foi observado

em trabalhos como Qureshi et al. (2016), Lima, Moreno e Dias (2016), Ramos, Ralha e Teodoro (2016).

Com o conhecimento sobre o poder computacional do processamento combinado de vários *Raspberries PI*, o presente trabalho então apresentará uma união entre a Computação de Alto Desempenho de baixo custo de um *cluster* formado por *Raspberries PI* e as áreas de Processamento de Imagens e a Inteligência Artificial para criar um *cluster* capaz de detectar câncer de pele com base na analise de imagens médicas.

A motivação da escolha do câncer de pele como objeto de pesquisa reside no fato de esse tipo de câncer ser um dos mais comuns em todo o mundo. No Brasil, segundo dados da associação brasileira de câncer de pele, 180 mil novos casos são detectados todos os anos segundo o Instituto Nacional do Câncer (INCA) (SBD, 2017). Já nos Estados Unidos, o melanoma é o tipo mais comum de câncer de pele e tem 87 mil novos casos por ano, com cerca de 9370 mortes registradas em 2014 (American Cancer Society, 2014).

Para a obtenção de imagens dermatológicas usadas nesse trabalho, será usado o arquivo de imagens do ISIC *The International Skin Imaging Collaboration*, que contém mais de vinte mil imagens de lesões de pele, acompanhadas por sua classificação como benignas ou parte de algum tipo de lesão ou câncer de pele.

1.2 Justificativa

Dermatoscopia é um importante exame feito pelos dermatologistas para detectar lesões na pele e possivelmente câncer de pele. Com a subjetividade intrínseca do olho humano, os diagnósticos podem variar entre um especialista e outro. No entanto, de maneira geral, os dermatologistas usam um conjunto de características conhecido como a regra ABCDE, onde: A é assimetria, B é a borda da lesão, C é a cor, D o diâmetro e E a elevação (ILLIG, 1987).

A análise de imagens e reconhecimento de padrões realizados pelos médicos fez com que a Ciência da Computação fosse atraída de maneira natural para essa área de pesquisa. A partir disso, técnicas de inteligencia artificial e de processamento de imagens podem ser aplicadas para auxiliar os profissionais da área de saúde a realizar uma dermatoscopia mais acurada. De fato, o primeiro trabalho nesse sentido foi feito por BINDER et al. (1994), onde imagens de lesões de pele foram usadas para treinar uma rede neural.

Com o surgimento das redes de aprendizado profundo *(deep learning)* por volta de 2012, a Inteligência Artificial passou a ter um poder maior de resolução de problemas. Dentre os problemas que passaram a ter melhores resoluções estão aqueles que fazem o uso de imagens, dentre eles o ramo de pesquisa sobre imagens dermatológicas.

Desde 2016, o ISIC *(The International Skin Imaging Collaboration*), que é uma colaboração da academia com a indústria focado na construção de aplicações digitais para ajudar a detecção

e consequentemente reduzir a mortalidade de melanoma (Ver Capítulo 4), lança competições anuais acerca do uso de redes profundas para resolver algum problema usando o *dataset* de vinte mil imagens disponíveis para aqueles que desejem participar da competição. As competições envolvem fatores como segmentação das lesões, padrões de diagnóstico, classificação de lesões, etc.

Por ser uma competição de cunho científico, os resultados das competições do ISIC geralmente acabam se tornando publicações com o que há do estado da arte da pesquisa em aplicação de técnicas de inteligencia artificial para a detecção de câncer de pele.

Embora o uso de supercomputadores e maquinas mais potentes mostrem uma grande acurácia nos testes feitos nas competições do ISIC, é necessário que a computação agregue valor à sociedade fazendo com que esses experimentos cheguem de fatos nas mãos dos dermatologistas e democratize o acesso à dermatoscopia, principalmente às populações de baixa renda. Com uma maior detecção em estágios iniciais, é possível que o numero de mortes causadas pelo câncer de pele seja menor.

Sabendo do bom desempenho de máquinas de baixo custo como o *Raspberry PI* para Computação de Alto Desempenho, como demostrado nos artigos supracitados, o presente trabalho visa aliar o desempenho de computadores de baixo custo com a inteligência artificial para fazer a detecção de câncer de pele com poucos recursos financeiros. Para isso, serão utilizadas as redes de aprendizado profundo pré-treinadas como a RESNET *(Residual neural network)* e a VGG (Ver Seção 5.2), e técnicas adicionais que possibilitem a utilização em *Raspberry PI* (Ver Capítulo 5).

1.3 Objetivos

O objetivo desta dissertação é desenvolver e encontrar a forma mais eficaz de detectar possíveis casos de câncer de pele usando computadores de baixo custo e a Inteligência Artificial focada em aprendizado de maquina usando redes neurais convolucionais pretreinadas. Dessa maneira, é esperado que seja possível obter uma acurácia no nível do estado da arte com bases nas taxas obtidas nas competições do ISIC. Assim, espera-se validar a possibilidade de obter resultados que permitam a construção de um futuro protótipo de ferramentas auxiliares à dermatologistas usando poucos recursos computacionais e financeiros.

Para facilitar a compreensão do objetivo geral e expor de forma mais detalhada cada uma das abordagens, o que permitirá decidir quão viável e quanto de acurácia será possível conseguir usando as técnicas de inteligencia artificial em *clusters* de Alto Desempenho de Baixo Custo e com isso validar a viabilidade dos experimentos propostos. São eles:

- Treinamento diretamente no *Raspberry PI* usando a plataforma *Pytorch* para treinar uma rede RESNET ou GoogleNet;

- Treinamento em um *cluster Raspberry PI* usando o *framework Apache Hadoop* aliado a plataforma *Pytorch* para treinar uma rede RESNET ou GoogleNet.

1.4 Metodologia

O trabalho será dividido em duas partes, sendo elas: Investigação teórica e implementação dos experimentos.

Na parte teórica do trabalho serão apresentados todos os conceitos teóricos da Ciência da Computação necessários para o completo entendimento deste trabalho. Esses conceitos serão abordados em um capítulo específico.

Depois disso, será apresentada uma revisão sistemática da literatura acerca do uso de computadores de baixo custo para Computação de Alto Desempenho, usando as definições de revisão tal como apresentada por Kitchenham (2004). Dessa forma, ficará claro ao leitor as perguntas de pesquisa, os resultados encontrados, o que é possível obter com computadores de baixo custo, junto com o resumo dos artigos selecionados e considerações acerca da Computação de Alto Desempenho a baixo custo.

Para a revisão sobre o uso da Inteligência Artificial, o foi utilizado o portal do ISIC dispõe de uma seção onde estão listados dos os trabalho científicos publicados acerca das imagens e competições disponíveis desde 2016. A partir da sumarização dos resultados e comentários acerca dos trabalhos, será permitido ao leitor a compreensão do estado da arte do uso de Inteligência Artificial para a detecção do câncer de pele.

Com base nos resultados e problemas encontrados na parte teórica poderá ter inicio a parte da implementação dos experimentos. Para tornar clara a fase dos experimentos, ela será dividida em três etapas. Na primeira serão apresentadas as ferramentas e os códigos utilizados, os códigos completos estarão nos anexos do trabalho e partes serão inseridas no corpo do texto para deixar mais clara as explicações. Na segunda fase, todos os resultados serão apresentados, e por fim, serão feitas as conclusões com base nos resultados obtidos.

1.5 Estrutura do documento

Para melhor entendimento de como o trabalho será estruturado, aqui é apresentada a organização do trabalho em capítulos.

- Capítulo 2 - Fundamentação Teórica: Neste capitulo serão apresentados todos os conceitos computacionais necessários para o entendimento do trabalho.
- Capítulo 3 - Computação de Alto Desempenho com Baixo Custo: Neste capitulo é apresentada uma revisão sistemática acerca do uso de computadores de baixo custo na

Computação de Alto Desempenho.

- Capítulo 4 - Literatura do ISIC: O capitulo abordará os resultados obtidos e disponíveis na literatura usando o banco de imagens do ISIC.
- Capítulo 5 - Descrição do modelo: O capítulo abordará a descrição e implementação do modelo usado nos experimentos.
- Capítulo 6 - Resultados e discussões: O capítulo apresentará os resultados dos experimentos e discutirá sobre eles.
- Capítulo - Conclusões: Apresentará as conclusões e trabalhos futuros.

2

Fundamentação Teórica

Este Capítulo apresentará as definições teóricas necessárias para o entendimento do presente trabalho, são elas: Computação de Alto Desempennho, *Cluster*, *Apache Hadoop, Pytorch*, Arquitetura ARM e Inteligência Artificial. Seguem descritas nas seções a seguir.

2.1 Computação de Alto Desempenho

Mesmo com o aumento exponencial de desempenho que os computadores tiveram nas últimas seis décadas, o aumento não foi suficiente para acompanhar a necessidade de processamento. Embora o processamento disponível em computadores pessoais seja suficiente para a realização de todas as tarefas de um usuário comum, não é o suficiente para os problemas enfrentados por diversas áreas do conhecimento. Nesse contexto surge a Computação de Alto Desempenho (PACHECO, 2011).

O uso da Computação Científica não é exclusivo da Ciência da Computação. Biólogos necessitam de supercomputadores para sequenciar DNA, Químicos e Físicos representam seus modelos teóricos, múltiplas engenharias realizam seus cálculos em computadores. A Computação de Alto Desempenho é a área que busca tornar experimentos dessas e de outras áreas possíveis, estudando forma de realizar os cálculos e outra operações realizada sobre os dados mais rápido. Muitas das técnicas de Computação de Alto Desempenho estão relacionadas ao uso de paralelismo e o uso de *clusters* (ver Seção 2.1.1).

Existem diversas técnicas para mensurar o que é uma Computação de Alto Desempenho. A performance de um sistema computacional é um dos pontos mais importante quando precisa-se medir a eficiência, enquanto os usuário comuns estão interessados no tempo de reposta, isso é, o tempo que demora entre o acionamento de um programa e o término de sua execução. A Computação de Alto Desempenho está mais interessada em medir os *throughputs*, que é a média de trabalhos que cada unidade de processamento pode executar em uma unidade de tempo.

As performances dos sistemas computacionais utilizados em computação de Alto Desempenho são muitas vezes medidas em MIPS (*Millions Instructions Per Second*). O cálculo é feito exclusivamente no número de instruções, não importando que algumas instruções demorem mais do que outras. Por esse motivo, a taxa de MIPS não significa necessariamente o tempo de execução de um programa. Além disso, o uso de programas de *benchmark* são usados constantemente para definir a performance. Os *benchmarks* geralmente avaliam a performance de componentes de *hardware*, por exemplo, a performance da operação de ponto flutuante (RAUBER; RNGER, 2013).

2.1.1 Cluster

Um *cluster* é um sistema heterogêneo de dois ou mais computadores capazes de realizar uma tarefa, onde cada computador é conectado por rede (TANENBAUM, 2005). Geralmente essa rede é uma rede local, conectando os diversos computadores que são chamados de nós do *cluster*. Os nós têm um mesmo *hardware* e usam o mesmo Sistema Operacional, mas isto não é uma regra.

Um *cluster* deve ser visto como um sistema homogêneo que realiza uma tarefa específica, além da necessidade de continuar funcionando mesmo sem um dos seus nós. Geralmente os *clusters* são programados e controlados por *softwares*.

Os primeiros *clusters* foram desenvolvidos na década de 60, o que demostra que a busca por mais desempenho é algo que não é recente. A implementação de um *cluster* geralmente é motivada pela necessidade de performance. A utilização de um *cluster* representa um potencial ganho de desempenho, pois os nós trabalhando em conjunto propiciam paralelismo, que é a a execução de mais de uma tarefa ao mesmo tempo pelo computador, sendo estas tarefas iguais ou não (TANENBAUM, 2005).

Clusters são organizados principalmente de duas maneiras distintas: Os *clusters* centralizados, onde os computadores usados estão reunidos em uma mesma sala, geralmente em um *hack*; e os *clusters* descentralizados, onde os computadores estão em salas diferentes ou até mesmo em prédios diferentes. Como a conexão entre os nós é realizada via rede, é importante observar que a distância entre os nós pode representar um fator a ser considerado durante a construção de um *cluster*.

A Figura 1 mostra um modelo de *cluster* conectado em rede. Cada nó com memória e CPU própria e realiza uma tarefa, que é enviada via rede, e os dados são armazenados e processados na memória de cada nó.

No modelo de construção de um *cluster* são utilizados geralmente vários computadores (pelo menos dois) onde um computador é chamado de servidor ou mestre, e os restantes são chamados de escravos. Os computadores de um *cluster* geralmente utilizam um *hardware* comum, ou seja, todos os nós do *cluster* são computadores idênticos. E os nós do *cluster* são conectados

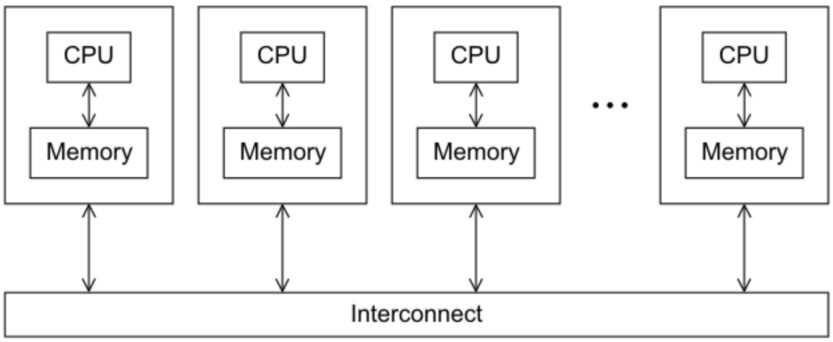

Figura 1 – Memória distribuída em um *cluster*

Fonte – (PACHECO, 2011)

via rede e *software*.

2.1.2 Apache Hadoop

O *Apache Hadoop* é um *framework* escrito principalmente em Java, que permite o desenvolvimento de processamento distribuído de uma grande quantidade de dados em *clusters*. Para isso são usados modelos de programação simples. O *framework* é projetado para ser escalável, desde servidores únicos a centenas de máquinas em um *cluster*, oferecendo processamento e armazenamento locais (Apache Software Foundation, 2017).

Embora a maior parte do *framework* seja escrito em Java (também há partes escritas em C e em Shell Script), o código do *MapReduce* do java é comum. O MapReduce é um modelo de programação distribuída usada para gerar e processar conjuntos muito grandes de dados de forma paralela usando algoritmos distribuídos em um *cluster*.

A chave de utilização do MapReduce está em duas funções. A função Map, que que filtra e ordena os dados a serem processados, enquanto a função Reduce de fato realiza as operações sobre os dados (DEAN; GHEMAWAT, 2008). O MapReduce funciona no Hadoop usando o *JobTracker* e o *TaskTracker*. O *JobTracker* envia dados para os nós que estão ociosos e sabe em quais nós os dados estão armazenados, enquanto o *TaskTracker* gera uma máquina virtual Java separada em cada nó para impedir que os nós falhem e constantemente enviam seu *status* ao *JobTracker*.

Dessa maneira é possível através do *Hadoop Streaming*, escrever códigos em qualquer linguagem de programação que implementem o *Map* e o *Reduce*. O núcleo do *Hadoop* é formado principalmente por quatro módulos, todos os módulos do *Hadoop* são desenvolvidos com a fundamental suposição que o *hardware* pode falhar, dessa forma, o *Hadoop* deve via *software* lidar automaticamente com essas falhas (BAPPALIGE, 2014).

Os quatro módulos são:

- *Hadoop Common*: Contém as bibliotecas e as ferramentas utilizadas em outros módulos;
- *Hadoop Distributed File System*: É um sistema de arquivos que distribui os arquivos através dos nós do *cluster Hadoop*, provendo uma largura de banda muito alta;
- *Hadoop YARN*: É a plataforma de gerenciamento de arquivos responsável por gerenciar os recursos dos computadores no *cluster* e usa-los para o agendamento das aplicações de usuário;
- *Hadoop MapReduce*: É a implementação do *Hadoop* para um modelo de programação em larga escala desenvolvido pela Google.

Os dois principais componentes do *Hadoop* são o *Hadoop Distributed File System* (HDFS) e o *Hadoop MapReduce*. ambos são baseados em projetos *open source* disponibilizados pela Google.

O HDFS é escalável, distribuído e portável. Cada nó em um *cluster* tipicamente tem um *namenode* e um agrupamento de *datanodes* forma o HDFS. Cada *datanode* envia blocos de dados através de um protocolo específico para o HDFS. O sistema de arquivos usa a camada de TCP/IP para se comunicar, enquanto os clientes se comunicam por chamadas de procedimento remoto. O sistema armazena arquivos grandes distribuídos através de múltiplas máquinas e a confiabilidade é dada pela replicação de arquivos através dos *hosts*. Também é parte do HDFS um *namenode* secundário, embora o nome secundário confunda fazendo acreditar que o secundário entra em ação caso o primário falhe, esta não é a sua função. O *nomenode* secundário funciona entrando em contado regulamente com o primário e armazenando algumas informações (BAPPALIGE, 2014).

2.1.3 Colab

O Colab é como é chamado o *Google Colaboratory*. Uma ferramenta de uso gratuito disponibilizada pelo a Google como parte do incentivo ao uso e aprendizagem tanto de ciência de dados quanto de aprendizagem de maquina. O único requerimento para uso da ferramenta é uma conta Google.

O Colab permite que o usuário crie um *notebook* para a linguagem Python, nesse notebook os códigos podem ser divididos em trechos chamados de células, onde o resultado das células podem ser armazenado, fazendo com que ele não precise ser executado cada vez que seja necessário. Isso faz com que seja fácil armazenar partes do aprendizado de máquina, como o preprocessamento.

Além de um ambiente de desenvolvimento python com todas as ferramentas da linguagem, incluindo a possibilidade de instalar novas bibliotecas, o Colab também permite que o usuário também execute seus códigos usando CPU, GPU ou TPU (processador logico usado para acelerar tensores).

2.2 Sistemas embarcados ARM

Tradicionalmente os sistemas embarcados são aqueles computadores construídos com um proposito único, geralmente encapsulado em uma única placa contendo toda tecnologia necessária para realizar sua tarefa proposta. Exemplos tradicionais de sistemas embarcados são os controles de televisão, os primeiros celulares e a televisão em si.

A grande diferença dos sistemas embarcados em relação a os sistemas de propósito geral é a quantidade de ações que o sistema pode realizar. Por esse motivo as arquiteturas de sistema embarcados seguem alguns critérios diferentes daqueles levados em consideração para a construção de um sistema de proposito geral:

- Baixo consumo de energia;
- Dimensão física reduzida;
- Baixo custo;
- Capacidade Computacional compatível com a aplicação.

No entanto, com a criação das arquiteturas ARM, os sistemas embarcados deixaram de ser usados apenas para tarefa especificas, pois as arquiteturas ARM seguem os critérios previamente expostos como requisitos para um sistema embarcado. No entanto, permite programação de aplicações além daquelas definidas em fábrica.

Esses são os chamados *System-on-a-chip (SoC)* e podem ser largamente usados para computação de aplicações programadas e têm uma capacidade de desempenho considerada relativamente alta. Desde 2012, com a popularização do *Raspberry PI*, muitos SoCs são usados como uma opção de baixo custo (custando algumas centenas de dólares) para a Computação de Alto Desempenho (ver Capitulo 2.1).

2.2.1 Arquitetura ARM

As arquiteturas ARM (do inglês: *Advanced RISC Machine*) são um conjunto de arquiteturas atualmente desenvolvidas pela empresa britânica ARM Holdings. No entanto, as primeiras arquiteturas foram desenvolvidas pela também britânica Acorn, em 1983, onde a responsabilidade pelo desenvolvimento do conjunto de instruções ficou com Sophie Wilson (LIMA; MORENO; DIAS, 2016).

Ainda na década de 1980 houve um interesse por parte da Apple em utilizar processadores ARM, no entanto, algumas mudanças no projeto foram necessárias. Assim, em novembro de 1990, em uma parceria da Acorn, Apple e VLSI surge a empresa ARM Holdings.

A produção de chips nunca foi o objetivo da ARM Holdings, por esse motivo a empresa costuma licenciar sua propriedade intelectual para várias outras empresas. Essa característica leva

as arquiteturas ARM ao topo das arquiteturas quanto ao número de chips produzidos, contando com bilhões de chips.[1]

Segundo o manual de referência (SEAL, 2000), as arquiteturas ARM suportam um amplo espectro de pontos de performance. A simplicidade dos processadores levam a uma pequena implementação. Os processadores ARM são baseados em um conjunto de instruções RISC e, por isso, o conjunto de instruções é simplificado. Por conta do conjunto de instruções RISC que permitem essa implementação simples, essa arquitetura é desejada por fabricantes de processadores para dispositivos portáteis. Os registradores internos são usados para processamento de dados, onde a CPU lê os dados da memória principal e armazena nos registradores internos (LIMA; MORENO; DIAS, 2016).

As Principais características da arquitetura ARM são: Conjunto de instruções RISC; instruções de tamanho fixo de 32 bits ou 64 bit; 16 registradores de propósito geral de 32 bits cada; tamanho do núcleo reduzido; baixo consumo de energia.

2.3 Inteligência Artificial

A Inteligencia Artificial é a capacidade de uma maquina computacional simular o comportamento inteligente na solução de problemas. Para isso, na Ciência da Computação são usadas diversas técnicas de programação que auxiliam na solução algorítmica do problema e também técnicas que simulam o modo de pensar e agir do ser humano ou da natureza para solucionar problemas inerentemente difíceis.

A inteligência artificial é um campo de estudo da ciência da computação e engenharia que data do período logo após a segunda guerra mundial. E o termo Inteligência Artificial em si é usado desde 1956. A inteligência artificial tem vários subconjuntos de estudos que vão de como jogar xadrez à como dirigir um carro. Historicamente, quatro principais abordagens são usadas, sendo elas: Agir humanamente, pensar como humano, pensar racionalmente e agir racionalmente (RUSSELL; NORVIG, 2009).

2.3.1 Redes Perceptron

Mcculloch e Pitts (1943) apresentaram o primeiro modelo de Inteligência Artificial, onde o modelo proposto extraia apenas o essencial para representar um modelo biológico do cérebro, abstraindo assim todos os aspectos desnecessários. O modelo proposto tem a seguinte configuração:

- **Entradas:** São os terminais de entrada que simulam os detritos do cérebro humano (MARSLAND, 2009).

[1] Disponível em: https://community.arm.com/iot/b/blog/posts/arm-from-zero-to-billions-in-25-short-years

- **Pesos:** Representam as sinapses do cérebro, adquirem valores positivos os negativos, podendo então serem chamados de excitatórios ou inibidores (MARSLAND, 2009);
- **Combinador linear:** É o somador de valores da entrada e corresponde a função da membrana que recebe descargas elétricas do cérebro (MARSLAND, 2009);
- **Função de ativação:** É uma função matemática que toma pra si a responsabilidade de decidir a ação do neurônio (MARSLAND, 2009);
- **Saída:** Corresponde ao resultado final e único do neurônio, representada de forma binaria (MARSLAND, 2009).

Mas foi somente em 1958 que Rosenblatt (1958) apresentou o Perceptron. Nele, o conceito de RNA foi introduzido. O Perceptron é basicamente um neurônio que é capaz de ajustar os pesos de entrada. Mas levou quatro anos para que o teorema de convergência do Perceptron fosse provado assim atestando que está correto para problemas linearmente separáveis (ROSENBLATT, 1962).

Depois dos questionamentos de Minsky e Papert (1969), a pesquisa sobre o Perceptron foi praticamente esquecida na década de 70 e primeira metade da década de 80. Foi só com o algoritmo de *backpropagation* (RUMELHART; MCCLELLAND; GROUP, 1986) que o Perceptron tornou-se um dos mais importantes algoritmos de treinamento.

2.3.2 Redes Neurais Artificiais

Uma rede neural artificial é uma abstração de uma rede biológica. As redes neurais não têm o objetivo de replicar totalmente uma rede biológica, mas sim, servir como um modelo de aprendizagem e resolução de problemas mais complexos. Uma rede neural é uma estrutura complexa de ligação entre vários elementos simples que simulam a função de um neurônio biológico. Nas redes neurais, o conhecimento é adquirido na rede por meio da percepção do ambiente onde o conhecimento adquirido é armazenado (RUSSELL; NORVIG, 2009). Haykin (2009) diz que "uma rede neural é um processador paralelo composto por unidades mais simples com uma propensão natural para o armazenamento de conhecimento e torná-lo disponível ao uso."

2.3.2.1 Arquiteturas de rede Neurais

As arquiteturas da RNAs (Redes Neurais Artificiais) são variadas e estão na grande maioria dos casos intimamente ligadas ao problema que será tratado e qual algoritmo que será utilizado. No entanto, quatro parâmetros são fundamentais para a arquitetura de uma RNA, sendo eles:

- **Numero de camadas de rede:** Podendo ser uma ou varias;

- **Quantidade de nós em cada camada de rede:** Não há uma regra para definição desse numero, muitas vezes são usados números com bons resultados em outros experimentos ou definidos empiricamente;
- **Tipo de conexão entre os nós:** Cada nó pode estar parcialmente ou totalmente conectado;
- **Topologia da rede:** Existem duas topologias, sendo elas: *feedfoward* e *feedback*;
  - **feedforward:** A saída de um nó não pode ser usada como entrada para nós da mesma camada ou camadas anteriores;
  - **feedback:** Onde a saída de um nó pode ser usada como entrada em um nó da mesma camada ou camadas anteriores.

2.3.2.2 Rede Perceptron de Multicamadas

As redes perceptron multicamadas é semelhante as redes perceptron, mas enquanto o perceptron só é eficaz para problemas linearmente separáveis (ROSENBLATT, 1962), os perceptrons multicamadas podem resolver os problemas não linearmente separáveis usando camadas extras na rede (RUSSELL; NORVIG, 2009).

Essas redes são caracterizadas pela presença de ao menos uma camada intermediária entre as camadas de entrada e de saída. Essa rede intermediaria é chamada de camada oculta, pode ser uma ou várias, geralmente a quantidade de camadas ocultas é definida experimentalmente (MARSLAND, 2009). Uma característica importante é a versatilidade, e por isso, as redes perceptron multicamada são usadas nos mais variados problemas.

Dependendo do problema e dos tipos de dados de entrada é possível que as redes multicamadas convirjam para um mínimo local ou que seja necessário e demorado para encontrar uma solução, para isso faz-se necessário o aprendizado da rede (RUSSELL; NORVIG, 2009).

Nessas redes, a medida que o processamento da entrada à saída, o resultado do processamento de cada nó está intimamente ligado aos resultados de cada nó anteriormente ligado a ele, ou seja, é usada uma topologia do tipo *feedforward*, onde não há retroalimentação.

A quantidade de camadas intermediárias é um problema a ser enfrentado durante a definição da rede. Uma rede muito pequena pode não chegar a resultados esperados, enquanto uma rede muito longa pode fazer com que a medida de erro seja menos precisa a cada propagação à camada anterior.

A forma mais comum de treinamento de um perceptron multicamadas é o *backpropagation*, definido por Rumelhart, McClelland e Group (1986) e que pode ser explicado em cinco passos:

1. **Inicialização:** Pesos são iniciados de maneira aleatória ou seguindo algum padrão
2. **Cálculo de erro**: O erro é calculado segundo as ativações de cada neurônio;

3. **Passo reverso**: O Erro é propagado de forma retroativa na rede;
4. **Teste de parada**: Se o critério de parada for atingindo pare o treinamento;
5. **Reinicie**: Volte ao passo 2 até o teste de parada ser satisfeito.

2.3.2.3 Redes Profundas

As definições de redes neurais variam, mas de forma predominante entre as definições existentes está a necessidades de camadas intermediárias entre a entrada e saída. Essas camadas possuem como entrada a saída da camada anterior e suas saídas ativam as camadas sucessoras. Não há uma definição de a partir de quantas camadas uma rede pode ser considerada profunda. Mas essas redes podem ter dezenas, centenas ou até milhares de camadas. O aprendizado nessas redes pode ser supervisionado ou não supervisionado.

Geralmente cada camada durante o processo de aprendizagem fica responsável por identificar uma característica do problema que está sendo tratado pela rede em questão, esse processo é conhecido como extração de características. Muitas das vezes são observadas características redundantes no processo, essas características são retiradas, fazendo as redes ficarem menores sem perderem sua capacidade.

2.3.3 Tipos de Aprendizagem

A capacidade de aprender é o que relaciona a inteligência artificial com a inteligência humana. Esses é um dos destaques mais relevantes das redes neurais artificiais. O processo de treinamento ocorre em etapas e é um processo interativo, onde ao final os pesos e parâmetros são armazenados e essa rede é testada em um conjunto chamado conjunto de testes (RUSSELL; NORVIG, 2009). O processo de aprendizado pode ser dividido em dois: Supervisionado e o Não-supervisionado.

- **Supervisionado:** O aprendizado é chamado de supervisionado quando a entrada e a saída desejadas são informadas por um supervisor. Conhecendo a saídas desejadas, a rede ajusta seus pesos até atingir um critério de parada, geralmente determinado por um número definido de interações ou quando a rede consegue acertar as saídas para os dados fornecidos pelo supervisor. A grande desvantagem do aprendizado supervisionado é que a rede não consegue adaptar-se a cenários não descritos pelo supervisor durante o processo de aprendizagem;
- **Não-supervisionado:** Nesse modo de aprendizado não há nada que dê conhecimento a priori a rede e apenas dados de entrada são fornecidos. Nesse tipo de aprendizagem a rede identifica padrões nos dados de entrada e se organiza a partir desses padrões. Esse tipo de abordagem só é possível quando há redundância nos dados de entrada, tornando possível para rede identificar padrões e características.

2.3.4 Medidas de desempenho

Na inteligência artificial existem várias medidas de desempenho. No escopo deste trabalho três delas são essenciais:

- **Acurácia:** É a taxa geral de acertos. Isto é, a divisão de acertos pelo conjunto totais de dados.
- **Precisão:** Relação entre as previsões positivas acertadas e todas previsões positivas.
- **Sensibilidade:** É relação de casos positivos corretamente previstos.

2.4 Pytorch

O Pytorch é um *framework open souce* de aprendizado de maquina que é indicado para uso em pesquisa até o uso em produção. O Pytorch funciona sobre o Python, sendo assim, possível ser executado em todos os sistemas operacionais e arquiteturas com interpretadores Python, incluindo as arquiteturas ARM (ver 2.2.1). Podendo ser instalado via PIP ou Anaconda.

Além da interface Python, pensada para facilitar o desenvolvimento e aprendizado do *framework*, o Pytorch também dispõe de uma interface construída em C++

O Pytorch foi escolhido para esse trabalho por contar com uma versão estável para arquitetura ARM, enquanto ,no momento do inicio da pesquisa, a versão para ARM do TensorFlow ainda não estava funcional.

2.5 Câncer de pele

O câncer de pele é mais comum entre os cânceres no Brasil. Dados do INCA (Instituto Nacional de Câncer) mostram que 180 mil casos novos são detectados a cada ano. Existem vários tipos de câncer de pele, entre os quais o mais comum é o melanoma. O melanoma tem a forma mais mortal de câncer de pele (American Cancer Society, 2014).

O melanoma é um perigo mundial. Nos Estados Unidos, por exemplo, em 2017, 87 mil casos foram detectados com 9730 mortes. O câncer de pele é provocado por células de crescimento anormal da pele. Essas células formam camadas e a maneira como é afetada determina o tipo de câncer (SBD, 2017).

O melanoma é mais difícil e mortal, mas curável desde que detectado nos estágios iniciais. Para detectar câncer de pele em estágio inicial e tornar a doença altamente curável, os dermatologistas geralmente devem ser consultados para exames médicos.

A dermatoscopia é um exame importante feito por dermatologistas, no entanto, devido à subjetividade do diagnóstico humano, pode diferir entre os dermatologistas. Mas, em geral,

os dermatologistas examinam um conjunto de características em uma lesão de pele. Essas características são chamadas de regra ABCDE, onde: A é assimetria, B é Borda, C é Cor, D é diâmetro e E é Elevação / Evolução (ILLIG, 1987).

Desde 2016, o ISIC (*International Skin Cancer Challenge*) lança uma competição anual para a detecção de melanoma na análise de imagens. O arquivo ISIC possui mais de vinte mil imagens com lesões cutâneas classificadas como benignas ou com um tipo de câncer de pele, a maior parte delas é o melanoma.

3

Computação de Alto Desempenho em Plataformas Embarcadas

Com intuito de estabelecer as bases para os objetivos descritos no Capítulo 1, foi realizada uma revisão acerca dos trabalhos que fizessem o uso de técnicas já conhecidas de *clustering* utilizadas com uma abordagem diferente, ou seja, voltadas para computadores ARM, sobretudo o *Raspberry PI*. O objetivo da revisão feita foi conhecer quais trabalhos que já foram publicados e quais experimentos já foram feitos. Para a obtenção de tais informações foram utilizadas as principais bases de trabalhos científicos da área de computação.

De posse dos resultados encontrados no presente Capítulo foi possível definir quais experimentos seriam feitos no presente trabalho e de qual forma representaria algum avanço para a área de Computação de Alto Desempenho. Parte dessa revisão foi publicada em 2018 como parte do Trabalho de Conclusão de curso do autor, esse trabalho é a continuação desta pesquisa iniciada na graduação e complementada no mestrado.

O processo de pesquisa, a discussão sobre os trabalhos encontrados e, por fim, os objetivos delineados a partir dos resultados encontrados serão descritos nas Seções a seguir.

3.1 Revisão Sistemática

Seguindo o procedimento indicado em Kitchenham (2004), esse Capítulo faz uma pesquisa em bases cientificas em busca de trabalhos a partir de *strings* de busca, depois seleciona e qualifica os artigos de acordo com a relevância. Após um refinamento entre todos os artigos encontrados, apenas os selecionados farão parte da discussão do restante do Capítulo.

As pesquisas na bases foram realizadas inicialmente no período que vai de dezembro de 2017 a janeiro de 2018 e atualizado entre fevereiro e março de 2020, durante o desenvolvimento dessa dissertação. Para eliminar a restrição de download das bases foi utilizado o portal Capes.

3.1.1 Bases Utilizadas

Para a realização da pesquisa foram utilizadas as seguintes bases:

- ACM Digital Library;
- IEEE Xplorer;
- Science Direct;
- Spring Link.

As seguintes bases não foram utilizadas por não disporem de um mecanismo avançado de busca.

- Google Scholar;
- BDBComp.

3.1.2 Termos de busca e String de busca

Para tornar possível a busca de artigos foram formulados seis termos de busca, são eles:

1. Cluster
2. Clustering
3. Raspberry PI
4. High Perfomance Computing
5. Parallel Processing
6. Performance Evaluation

A partir da combinação deste termos foi cunhada uma *string* de busca genérica, que foi utilizada igualmente em todas as bases.

- ((cluster OR clustering) AND Raspberry PI OR Raspberry PI AND(High Perfomance Computing OR Perfomance Evaluation))

com a atualização do trabalho foram adicionados os seguintes termos de busca:

- ARM
- Machine Learning

e cunhada duas novas *strings* de busca, sendo elas:

- ((cluster OR clustering)) AND (Raspberry PI OR ARM)
- ((cluster OR clustering)) AND machine learning AND (Raspberry PI OR ARM))

3.1.3 Critério de seleção e relevância dos artigos encontrados

Para delimitação dos artigos que seriam selecionados, foi realizada a leitura do resumo e conclusão de todos artigos encontrados nas bases de busca. A partir daí foram selecionados artigos que apresentassem de forma teórica e/ou prática a utilização de *cluster* para o desenvolvimento de Computação de Alto Desempenho e posteriormente foram considerados artigos que usassem qualquer computador ARM e *Machine Learning*.

Quanto aos critérios de exclusão, foram excluídos artigos encontrados em mais de uma base. Também não foram utilizados aqueles que não foi possível o *download* do texto integral ou não estivessem no escopo da pesquisa, foi considerado fora do escopo qualquer artigo que abordasse uso de computadores ARM sem relação com Computação de Alto Desempenho ou sem relação com aprendizagem de máquinas.

Por fim, a avaliação da relevância dos trabalhos selecionados foi dada a partir da leitura completa dos mesmos.

Acabado o processo de busca, inicia-se o processo de discussão acerca dos resultados encontrados. Buscou-se nesse Capítulo detectar quais experimentos parecidos já foram realizados e por fim detectar algo ainda inexplorado.

3.2 Discussão acerca dos trabalhos relacionados

O lançamento do primeiro modelo Raspberry PI foi em 2012[1], e o primeiro trabalho encontrado nas bases pesquisadas relacionado ao uso como um sistema distribuído data de 2013. Em Tso et al. (2013) foi formulado a *"The Glasgow Raspberry PI Cloud"*, onde seus autores propuseram um modelo de nuvem escalável baseado no Raspberry PI. As seguintes subseções descreverão os objetivos, metodologia e conclusão de cada um dos artigos selecionados. Findada a descrição dos artigos, encontra-se a Tabela 1 fazendo uma comparação entre os artigos.

3.2.1 *The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures*

Tso et al. (2013) argumentam que a computação em nuvem é um novo paradigma de processamento baseado na infraestrutura do *pay-as-you-go*. Eles argumentam que existem um

[1] Data disponível no site oficial: www.raspberrypi.org

número de problemas que precisam ser discutidos acerca do modelo tradicional de nuvens em *Data Centers*, um deles é o custo. Também salientam que mesmo as pesquisas em simulação de nuvem tenha grande importância para o desenvolvimento de algumas áreas da computação em nuvem, a simulação é falha para muitas coisas, sobretudo, na simulação do tráfego. A partir disso eles propõem uma nova alternativa na pesquisa de nuvem, o que eles chamam de *CloudPi*. Um modelo de nuvem escalável composta por 56 *Raspberry PI*.

A arquitetura do sistema do *Cloud PI* foi feita usando os 56 nós conectados por uma arvore topológica de múltiplas raízes, essa organização foi escolhida com intuito de refletir a organização dos *Data Centers*. Cada parte do sistema roda um *Raspbian* em um *SD-Card* de 16GB e acima da camada do Sistema Operacional, são usados *Containers* para que seja possível a virtualização. Para conseguir virtualizar no *Raspberry* modelos tradicionais de virtualização como o *VMware* não são possíveis por conta da pouca quantidade de RAM disponível, por isso na construção da nuvem foi utilizados *containers* que usam apenas 30MB de RAM.

Os autores argumentam que com a *CloudPi* eles podem testar qualquer aspecto de uma nuvem convencional. Também afirmam que conhecendo a maneira que a nuvem é virtualizada, eles podem testar novos algoritmos e avaliar empiricamente qualquer outro aspecto das outras camadas. Por exemplo, é possível medir o congestionamento da rede na nuvem.

O custo estimado por Tso et al. (2013) para a construção da *CloudPi* foi 1960 dólares estadunidenses enquanto uma nuvem de 56 computadores x86 custaria 112.000 dólares estadunidenses. Por fim, os autores concluem que o principal valor da construção da *CloudPi* é tornar a experiência na gestão de *Data Centers* mais real do que os simuladores.

3.2.2 *A Container-based Edge Cloud PaaS Architecture based on Raspberry Pi Clusters*

Em Pahl et al. (2016), também é abordada a forma de construção de *cluster* de *Raspberry PI* como opção para a construção de uma nuvem. No entanto, a diferença está no fato que, segundo os autores, as nuvens não estão mais centralizadas em *Data Centers* e sim distribuídas. Essas nuvens são chamadas de *edge clouds*. Esta abordagem está relacionada a necessidade de conectar *Data Centers* com aparelhos pequenos conectados a internet. No entanto existe uma barreira que é a virtualização, que para ser usada custa uma grande quantidade de memória RAM, que muitas das vezes, esses aparelhos não dispõem. Por isso é preciso criar *containers* menos custosos quanto ao consumo de memória RAM. O *Raspberry Pi* é uma alternativa para testar esses *containers*.

Os autores definem a arquitetura para essas *edge clouds*, tais como a necessidade dos *containers*, e a forma de gerenciá-los. Após a construção de uma nuvem composta por 300 nós, é analisado o custo benefício, baseados em fatores como custo, consumo de energia e robustez.

3.2.3 *Modelling Low Power Compute Clusters for Cloud Simulation*

Kecskemeti, Hajji e Tso (2017) discutem no seu trabalho a possibilidade de construir um *Data Center* baseado em computadores ARMs, pensando, além do baixo custo de comprar essas placas em quantidade, no baixo consumo de energia. Inspirados pela *CloudPi* definida em (TSO et al., 2013), comparam o desempenho de uma *Raspberry Pi* com uma nuvem formada por computadores x86.

O artigo trata as limitações causadas na implementação de nuvens em computadores ARM por conta da maioria das ferramentas terem sido projetadas para arquitetura x86, por isso os autores precisaram modificar os simuladores existentes para que pudessem ser utilizados em computadores ARM. Foram feitos testes em cenários reais e simulados.

O artigo defende que os *Data Centers* gastam muita energia e que o uso de computadores com baixo consumo como o *Raspberry Pi* está sendo ignorado no desenvolvimento dos simuladores. E iniciativas como a de criação de simuladores que funcionem em arquiteturas ARM possam mudar essa perspectiva.

3.2.4 Comparison of Load Balancing Methods for Raspberry–Pi Clustered Embedded Web Servers

Maduranga e Ragel (2016) abordadam a criação de servidores embarcados. Para isso, os autores criaram um *cluster* com três *Raspberry Pi*. Um deles funciona como mestre e os outros dois como escravos. O objetivo do trabalho é verificar o desempenho e escalabilidade do *cluster*. Usando o *Apache Web Server*, é medido o desempenho de quatro algoritmos de balanceamento de rede em um servidor web embarcado.

Os quatro algoritmos são:

- *By_Request*: O servo com maior *lbstatus* é selecionado, então o valor do *lbstatus* é decrementado do *lbfactor*; Consequentemente a soma de todos *lbstatus* e o *request* são distribuído;
- *By_Traffic*: Funciona de maneira bastante a o *By_Request*, no entanto o balanceamento não é feito pelo tráfego na rede;
- *By_Busyness*: O número de *requests* em cada servo é armazenado. Quando chega um novo *request*, o servo com menor número fica com o *request*;
- *By_Heartbeat*: Cada *request* é entregue ao servo mais ocioso.

O cálculo para comparação foi feito com base nas seguintes métricas:

- Transações: O total de transações é dado pelo número de usuário multiplicado pela quantidade de *requests* permitida por usuário;

- Tempo Gasto: O tempo decorrido entre o início e o termino do experimento;
- Dados Transferidos: A soma de todos o dados de usuário como resposta a todos os *requests* feitos;
- Taxa de Transações: O número de transações por segundo;
- Concorrência: A média de transições simultâneas de um usuário pela média de todos os usuários;
- Transações Bem Sucedidas: A quantidade de transações que são consideradas bem sucedidas, ou seja, aquelas que o server retorna um número menor de 400.

Os testes foram realizados com 10, 100 e 1000 usuários. No aumento de 10 para 100 só foram apresentadas mudanças significativas nas taxas de transição e concorrência. Mas na mudança de 100 para 1000 todos parâmetros apresentam mudanças significativas. O algoritmo *by request* foi o que se saiu melhor nos testes.

3.2.5 *Study of Raspberry Pi 2 Quad-core Cortex-A7 CPU Cluster as a Mini Supercomputer*

O trabalho de Mappuji et al. (2016) assemelha-se com o objetivo deste trabalho, porque os autores tocam no ponto do alto custo e grande consumo de energia na Computação de Alto Desempenho usando os supercomputadores convencionais. Baseado em estudo anterior (MOORE, 2014) já sabia que um *cluster* de quatro nós de um *Raspberry PI single core* e com 0.7 GHz de *clock* atinge 846 MFLOPS usando o *benchmark* HPL. Dessa maneira, o objetivo do estudo é identificar o desempenho de *cluster* de 4 nós (*quad- core*) quando o *clock* é aumentado para 0.9 Ghz.

Os estudos foram realizados com base em duas comparações: consumo de energia versus quantidade de nós e performance computacional versus quantidade de nós. Na análise do consumo de energia foi identificado que todos os nós consumem 1.19W quando está ocioso e 1.49W quando está realizando processamento. Na análise de desempenho ficou constatado que o desempenho com um nó é de 150 MFLOPS e após adição dos outros três nós o resultado final é de 850 MFLOPS nos computadores *quad-cores*.

Baseado na normalização da curva de performance com o número de nós, concluem que o número ideal são 4 nós, pois há um ganho significativo entre as curvas de 2 e 4 nós, aliado com o fato que 2 nós são muito pouco para um *cluster* de Alto Desempenho. Também fica claro que quantidade de *cores* em cada *Raspberry* não representa ganhos significativos, pois o *cluster single-core* obteve 836 MFLOPS no *benchmark* enquanto o *quad-core* atingiu 850.

3.2.6 *Performance of a Low Cost Hadoop Cluster for Image Analysis in Cloud Robotics Environment*

Inspirados pelo rápido desenvolvimento das *cloud robotics*, cujo o mercado popularizou sob uso dos *drones*, permitindo aplicações em diferente modalidades. Este artigo aborda seu uso na captação de imagens aéreas.

Qureshi et al. (2016) procuram estudar a performance de um *cluster* baseado em *Raspberry PI* aliado a essas *clouds*. A razão para essa pesquisa é que os *drones*, controlados por controle remoto, usam de forma principal a sua câmera para permitir sua navegabilidade. As imagens capturadas pela câmera precisam ser processadas para permitir o uso de *drones* em aplicações reais, tais como: identificação de objetos suspeitos.

A análise dessas imagens requer um grande processamento, pois cada operação em uma imagem requem um calculo na matriz desta imagem (PEDRINI; SCHWARTZ, 2008), o processamento no *drone* é limitado e este processamento seria lento. Por isso uma das estratégias empregadas para contornar esse problema é a transferência dos dados capturados para nuvens, que realizam o processamento e transferem o resultado de volta para o *drone*. Com isso, o objetivo do trabalho é analisar a performance de um *cluster* construído com o *framework Hadoop* (ver Seção 2.1.2) para análise de imagens. No entanto o *Hadoop* apresenta algumas limitações no uso de imagens, como:

- Não existe uma interface para ler ou escrever imagens;
- A maioria das aplicações para processamento de imagem ou video não são compatíveis com o *Hadoop*;
- A performance dos algoritmos de análise de imagem deixa muito a desejar por conta da biblioteca HIPI.

O *cluster* foi construído com vinte *Raspberry Pi 2*, que foram conectados através de uma estrutura de rede baseada em topologia de estrela, conectado em 24 portas de *gigabit* por segundo. A versão instalada do *Hadoop* foi a 2.6, com configurações que permitem um maior uso de memória, garantindo o melhor desempenho. Mesmo com o *Hadoop* provendo inúmeras interfaces para leitura e escrita de texto, o mesmo não é verdade para imagens. Portanto, foram necessárias alterações na biblioteca HIPI que permitissem a realização dos testes.

Os testes foram realizados com base em três métricas para a avaliação de desempenho: performance computacional, performance da entrada e saída de texto e performance para entrada e saída de imagens. Os experimentos foram realizados com base em três configurações diferentes do *cluster*, e o tempo de execução foi comparado em cada uma das configurações. Em todas as configurações os *Raspberry PI* eram escravos enquanto um computador *Intel I7* era o mestre.

As três configurações foram:

- Configuração 1: Cada nó configurado com o *clock* padrão de 700 MHz, enquanto o *Hadoop* estava configurado para o HDFS ter o tamanho padrão de 128MB e o *MapReduce* e *YARN* com alocação máxima de memória padrão de 426MB.
- Configuração 2: Cada nó configurado com o *clock* de 1000 MHz, enquanto o *Hadoop* estava configurado para o HDFS ter o tamanho padrão de 128MB e o *MapReduce* e *YARN* com alocação máxima de memória padrão de 852MB.
- Configuração 3: Para fins de *benchmark* foi configurado um *cluster* composto por cinco computadores *Intel I7* de 3 GB de RAM, rodando nós em máquinas virtuais e conectados ao mestre de forma semelhante a dos *Raspberry PI*.

Como *benchmark* os autores usaram três critérios, que foram:

- Critério A: Execução de um programa de calculo de PI usando um algoritmo quase Monte Carlo para encontrar **m** dígitos binários da constante PI.
- Critério B: Execução do *WordCount* para verificar a ocorrência de palavras em arquivos de 3MB, 30MB e 300MB.
- Critério C: Analisar o tempo de execução para imagens JPEG de alta resolução usando a biblioteca HIPI.

Com base no critério A, a configuração 2 foi melhor que a 1, mas foi inferior a configuração 3. No critério B a configuração 2 é melhor em 50% para arquivos de 3MB e 27% e 19% para os arquivos de 30MB e 300MB respectivamente. A análise do critério C mostrou que a configuração 3 atinge um tempo considerado aceitável e que para configurações 1 e 2 existe uma diferença considerável de desempenho quando a biblioteca HIPI é modificada em relação a original. Com base nos critérios B e C foi constatado que o *cluster Raspberry PI* perde em desempenho em relação ao uso de máquinas virtuais, quando o conjunto de dados é grande.

3.2.7 Avaliação de Cluster Raspberry Pi para Execução de Aplicações de Análise de Imagens Microscópicas Médicas

Á medicina é uma das muitas Áreas do Conhecimento que necessitam da Computação de Alto Desempenho para realizar suas tarefas. Uma dessas aplicações é a análise de imagens microscópicas para o reconhecimento de câncer cerebral. Dessa maneira, o trabalho de Ramos, Ralha e Teodoro (2016) trata da construção de um *cluster* formado por 16 *Raspberry PI* como uma alternativa para um ganho de desempenho com um baixo custo de produção.

Além do *cluster Raspberry PI* foram usados outros dois computadores convencionais *core2duo* e I7 respectivamente. Para fins de análise comparativa entre os três foram levados em

consideração fatores como o custo, consumo de energia e desempenho. Os 16 nós do *cluster* foram configurados com o Sistema Operacional *Raspbian* e conectados por um *switch* 24 portas de 100MB. A comunicação e gerenciamento dos processos foi realizada usando MPI.

Para a análise das imagens microscópicas é necessária a realização de algumas operações, que são caracterizadas quanto a quantidade de acesso de dado e a intensidade computacional. As operações de acesso aos dados podem ser caracterizadas por dois tipos, as regulares e as irregulares. São ditas como regulares quando fazem acessos contínuos na imagem e irregulares quando fazem acesso a pontos espalhados pela imagem. Essa fase de analise das imagens é chamada de segmentação e foi implementada em C++ usando MPI e a execução é feita utilizando o modelo Mestre/Escravo, onde o mestre armazena a informação sobre as imagens que precisam ser processadas e assinala as mesmas para execução nos escravos sob demanda. Cada escravo então lê a imagem a ser processada e invoca o estágio de segmentação que é executado sequencialmente.

Os experimentos foram realizados com 512 imagens de 1K x 1K pixeis totalizando um tamanho de 6GB. O primeiro fator a ser analisado foi a velocidade de acesso aos dados, que foi uma preocupação inicial dos autores pois o *Raspberry Pi* usa cartões *MicroSD* que são mais lentos que os HDs SATA, além disso o processador trabalha em uma velocidade de *clock* menor. Ficou constatado que o tempo de leitura do *Raspberry PI* é consideravelmente inferior que o de computadores com HD SATA e que isso pode ser um gargalo na execução de problemas com operações de entrada e saída mais intensivas.

Quanto a análise de desempenho por tipo de operação, o *Raspberry PI* foi inferior as plataformas *Core2Duo* e I7, nas operações regulares pois tem seu tempo de acesso a memória mais lento e uma memória cache menor, ficando assim em desvantagem. No entanto, nas operações irregulares, como nenhuma arquitetura otimiza o acesso aleatório e a memória cache não exerce influência, o *Raspberry PI* levou vantagem.

Com relação à escalabilidade ficou comprovado que a execução paralela é 47x mais rápida que a execução sequencial, levando a uma eficiência de quase 75%, quando os quatro núcleos do *Raspberry PI* são utilizados, a eficiência é de 80%. É verificado que a perda de eficiência não está relacionada ao uso de múltiplas máquinas e sim de múltiplos núcleos, essa perda de eficiência é causada pela quantidade de acesso a memória. Como os núcleos usam a mesma memória acabam tornando um gargalo para aplicação. Por esse motivo o uso de todos os núcleos no I7 faz com que ele perca em desempenho para o *Core2Duo*, enquanto o *cluster Raspbery Pi* vence ambos.

Quando comparado o tempo de execução total, o *cluster Raspberry PI* é 10.6 vezes mais rápido que o computador *Core2Duo* e 2 vezes mais rápido que o I7. Quando ao consumo de energia, o *cluster* é duas vezes mais eficiente que ambos os computadores.

3.2.8 *Performance Analysis of a Low Cost Cluster with Parallel Applications and ARM Processors*

Lima, Moreno e Dias (2016) iniciam seu trabalho falando sobre como o poder de processamento aumentou muito, no entanto, a quantidade de dados processados também aumentou. No aumento da capacidade de processamento, a lei de Moore deixou de ser aplicável, devido as limitações físicas enfrentadas na construção de transistores cada vez menores. Por conta disso deixou de ser objetivo principal dos *designers* de processadores o aumento de desempenho de um único processador e sim a exploração de soluções paralelas.

Dessa forma, o trabalho de Lima, Moreno e Dias (2016) busca explorar o paralelismo na Computação de Alto Desempenho. Para isso, foi construído um *cluster* formado por quatro *Raspberry PI* onde a comunicação entre os nós é feita por uma rede local LAN (*Local Area Network*) e o aproveitamento do paralelismo é dado pelo uso de bibliotecas MPI. Durante o trabalho foi realizado experimentos de multiplicação de matrizes e de cálculo de produto escalar. Com a ajuda do *benchmark* HPL foi possível obter de desempenho como o tempo de execução e *Gflops*.

Quanto ao ambiente de teste, os testes foram realizados em um *cluster* formado por quatro *Raspberry PI 2 Model B*, todos usando como Sistema Operacional o *Raspbian*. Todos os nós foram conectados em um *switch* de rede local e toda a comunicação entre os nós se deu via protocolo SSH (*Secure Shell*). Para calcular o desempenho médio os autores usaram a função *MPI_Wtime()* disponível na API MPI para calcular o tempo médio de execução dos algoritmos depois de 100 execuções.

Com base em simulações com diferentes números de nós e ordem das matrizes, ficou constatado que o *cluster* tem um desempenho superior na execução das multiplicações de matrizes, chegando a 66% de melhora quando comparado o desempenho do *cluster* de quatro nós, em relação algoritmo sequencial em um nó. Foi constatado que quando as matrizes de ordem 100 não há ganho de desempenho quando o número de nós é incrementado. Isso é dado pelo fato que para quantidades pequenas de dados o tempo gasto com troca de informações entre os nós é maior que o tempo de processamento interno. Também é visto que o *speed-up* [2] aumenta junto com o número de nós para matrizes da mesma ordem.

Nas simulações de produto escalar de dois vetores com tamanho 100, 250 e 500, pode-ser concluir que a paralelização representou um ganho de 52%. Ao analisar o tempo de execução dos vetores de ordem cem pode-se concluir que não houve mudanças significativas de desempenho. No entanto, ao realizar comparações com vetores maiores é possível perceber um aumento no desempenho do *cluster*.

Na execução do *benckmark* HPL foi realizado a resolução de um sistema linear denso

[2] *speed-up* em arquitetura de computadores é uma medida referente a performance de dois sistemas para o mesmo problema.

de ordem 4000 para medir o tempo necessário para a resolução. Além disso, é possível obter usando o HPL o número de operações de ponto flutuante por segundo (*Gflops*). Também nesse cenário de testes, ficou comprovado que há um ganho de desempenho em relação a execução sequencial para a execução paralela com quatro nós. Neste caso, foi observado que a execução paralela foi 64% mais rápida. Adicionalmente foi possível concluir que a arquitetura ARM tem um bom desempenho na execução de operações matemáticas massivas. Com relação ao número de operações realizadas, ouve um aumento de aproximadamente 65%. Enquanto o número de operações com um nó é um pouco menor de 300 *Gflops*, para quatro nós o número foi superior a 1000 *Gflops*

As conclusões dos autores foram que: O aumento no número de nós implica na redução considerável do tempo de execução; Um aumento na quantidade de dados reflete no ganho de desempenho do *cluster*, dado que para quantidade pequenas de dados o tempo gasto na comunicação é maior que o tempo gasto com o processamento; E finalmente, com base nas aplicações executadas juntamente com o teste de *benchmark* realizado conclui-se que o *cluster Raspberry PI* tem bom desempenho na execução de algoritmos paralelos.

3.2.9 ZCluster: A Zynq-based Hadoop Cluster

O trabalho de (LIN; ZED, 2013) se diferencia dos abordados anteriormente por trazer o uso do FPGA integrado disponível na placa Xilinx Zynq SoC aliado com *framework Hadoop*.

Na introdução, os autores argumentam que computadores baseados em ARM tem sido usados no *design* de *data centers* e *clusters* pela indústria e pela academia, mas os computadores ARM são melhores em aplicações sem um processamento intensivo, basicamente fazendo operações de entrada e saída.

O *cluster* foi construído usando uma CPU de alta performance com i5 de 3.3Ghz e 4GB de RAM DDR3 como o mestre e os nós escravos são as placas Xilinx XC7Z020 Zynq SoC cada uma com ARM Cortex-A9 MPCore e o FPGA Artix-7 FPGA com 85 mil células lógicas conectados pelo um *switch* Netgear de 100Mbps.

Os autores clamam serem o primeiro trabalho a usar aceleradores de hardware para aumentar o processamento de computadores ARM. Para isso, o FPGA interage com o processador pela interface padrão disponibilizada pela Xillybus. A comunicação ARM para FPGA é feita por uma interface genérica de 32 bits, e do FPGA para o ARM é realizada através de uma porta AXI de alta performance.

Os resultados dos experimentos foram obtidos com arquivos transferidos para o HDFS (*Hadoop Distributed File System*) pelo nó mestre e o tempo de cópia não foi levado em consideração no resultado final. Os resultados foram divididos em resultados da aceleração de *hardware* e experimentos do Hadoop. Nos experimentos, a adição da aceleração de *hardware* obteve *speed-up* de 2.4. Na comparação do tempo de execução do Hadoop com a adição de nós é

relevante apenas após a adição do terceiro nó.

3.2.10 Acceleration of RSA Processes based on Hybrid ARM-FPGA Cluster

Assim como o trabalho de Lin e Zed (2013), o trabalho de Bai et al. (2017) usa a combinação de computadores baseados em ARM com o uso de um FPGA integrado para a construção de um *cluster*. No entanto neste *cluster* foi utilizado o protocolo MPI para a resolução de algoritmos RSA.

O tempo de processamento para algoritmos RSA é maior em CPUs tradicionais com a arquitetura x86, pois esses processadores não dispõem de algumas operações específicas para a multiplicação modular, com o uso auxiliar do FPGA esse trabalhou buscou unir a versatilidade de processadores convencionais com o alto paralelismo disponibilizado por um *hardware* dedicado no FPGA. Neste trabalho foram utilizados 48 nós de placas Xilinx Zynq SoC.

As placas usadas no *cluster* foram as Xilinx Zynq-7020 SoC e foram conectadas através de um *switch gigabit* de 48 portas e um algoritmo de escalonamento foi utilizado para aumentar a taxa de transferência entre os nós. Para a comunicação interna ARM para FPGA ou FPGA para ARM foi utilizado a interface AXl4 e a comunicação entre os nós foi feita pelo protocolo MPI.

Em comparação com um i7 de oito *cores* os resultados obtidos no *cluster* foram de um *speed-up* entre 6 e 9 vezes, em algoritmos de 512, 1024 e 2048 bits e foi bastante similar a um *cluster* com oito processadores Tilera GX36 de 36 *cores*. O *cluster* também foi duas vezes e meia mais eficiente no consumo de energia.

3.2.11 Portable Parallel Computing with the Raspberry Pi

O trabalho de Matthews et al. (2018) traz uma abordagem acerca da falta de paralelismo no currículo dos cursos de ciência da computação. Para isso, eles apontam como o fator decisivo o custo muito alto de hardwares que proporcionem a possibilidade de ensino de computação paralela, como os FPGAs. Mas com o desenvolvimento de GPUs e principalmente a popularização dos *system-on-a-chip* o custo de manter aulas sobre paralelismo no currículo diminuiu bastante. Partindo dessa abordagem, o trabalho apresenta um método detalhado do uso de Raspberry PI no ensino de paralelismo com memória compartilhada.

Para justificar a escolha do Raspberry PI, os autores apresentaram quatro principais motivos. Sendo eles: Padronização: o Raspberry faz com que seja possível todos os estudantes terem acesso a uma mesma imagem no *SDCard*; Fácil manutenção: é só inserir o *SDCard* e começar a programar; Custo: por custar apenas 35 dólares é possível garantir que todos estudantes tenham um; E imediatismo: os alunos podem ver de fato o hardware que estão usando.

Na decisão onde o Raspberry PI foi eleito como plataforma de ensino, foram considerados:

Dispositivos móveis, que foi descartado por não ter uma gama de IDEs e outros softwares de programação disponíveis; Notebooks pessoais, descartados por não ser possível padronizar e serem caros; Máquinas virtuais, que apresentam baixo desempenho; E servidores remotos, descartados por requererem contas, por ser complicado garantir a não interferência entre alunos acessando o mesmo servidor e exigir muita abstração do aluno, visto que o hardware não é palpável. Outro ponto em favor do Raspberry foi o fato de ser *multicore*, que também torna possível o uso de paralelismo com memória partilhada.

Para o ensino de paralelismo foram realizados vários *workshops* e cada aluno respondia cinco questões antes do *workshop* e as mesmas questões ao sair. As questões eram:

1. O quão confidente você está que pode decompor um problema em *multithreads* e decompor em um *loop* paralelo?
2. O quão confidente você está para descrever as vantagens e desvantagens do uso de programação paralela com memória compartilhada com alguém familiarizado com programação?
3. O quão confidente você está para definir o que é o *speed-up* e explicar para alguém familiarizado com programação?
4. O quão confidente você está para definir o que é uma condição de corrida e como evitar isso ao escrever programas paralelos de memória compartilhada?
5. Até que ponto um computador barato que permite programação paralela (como o Raspberry PI) pode te incentivar a aprender mais sobre programação paralela no futuro?

As questões deveriam receber notas de 1 a 5, onde 1 era pouco confiante, 3 com confiança média e 5 muito confiante. Após os experimentos todas as médias tiveram melhoras significativas, com isso o Raspberry Pi apresenta-se como opção viável ao ensino de computação paralela.

3.2.12 A novel single-arm-worn 24 h heart disease monitor empowered by machine intelligence

Doenças de coração são a principal causa de morte no mundo segundo a Organização Mundial de Saúde. Evitar, predizer e diagnosticar essas doenças são um grande desafio para a sociedade. O trabalho de Zhang, Zhou e Zeng (2018) estuda o complexo QRS que é parte central de um eletrocardiograma. Pois é com esses dados que é possível relacionar algumas doenças relacionadas ao coração. Geralmente, eletrocardiogramas são feitos utilizando eletrodos no peito do paciente e isso acaba sendo desconfortável para o usuário, então os autores propõem e constroem uma faixa que é colocada no braço do paciente e com isso é possível capturar os batimentos. O autores alegam serem os primeiros a usarem um medidor cardíaco vestível no braço e usam uma SVM para identificação dos batimentos. Eles também conseguiram agrupar os

batimentos em três *clusters* possibilitando assim o uso de métodos de inteligência artificial por agrupamentos.

Para realizar os estudos foi usado um *hardware* customizado onde um *system-on-chip* foi colocado em uma faixa contendo eletrodos, possibilitando a captura dos dados. Usando um *dataset* previamente reportado (ZHANG; ZHOU; ZENG, 2017), foi possível classificar os dados coletados e realizar comparações com estudos prévios. Usando as técnicas de agrupamento das batidas do coração descritas pelos autores foi possível identificar um método de agrupamento K para cada um dos três grupos que foram definidos. Em comparação com os métodos de detecção tradicionais usando eletrodos no peito, a nova solução demostrou ser mais robusta. No entanto, os próprios autores exibem os problemas e defendem que a principal contribuição é mostrar ser possível criar um monitor de ECG vestível.

3.2.13 Performance analysis of single board computer clusters

Desde de 2012 com o lançamento *Raspberry PI*, o poder de um cluster formado por *system-on-a-ship* aumentou quatro vezes. Para validar essa comparação, o artigo de Basford et al. (2020) apresenta a construção de *cluster* usando o que eles chamam de *PiStack*. Para comparar a eficiência foram montados três *clusters* de 16 nós cada, o primeiro modelo usado foi o *Raspberry PI 3 Model B*, o segundo *Raspberry Pi Model B+* e o terceiro usava o *Odroid C2*. Embora apresente limitações na construção de *clusters* usando outras placas, a *PiStack*, segundo os autores, mostrou-se uma técnica válida para construção de *clusters*.

Para colher dados para o resultado o *benchmark* HPL foi usado 1500 vezes, e dada a conhecida falta de performance da biblioteca ATLAS foi usada no lugar a biblioteca OpenBLAS. A otimização dos cálculos da biblioteca ATLAS foi utilizada para testar a mudança de performance com a otimização. Para habilita o uso do HPL entre os nós foi usado a biblioteca de MPI MPICH. Nos três *clusters* usados é importante frisar que além de diferentes velocidades de CPU as placas *Odroid C2* têm mais memória RAM que ambos os modelos do *Raspberry PI* e tem uma memória interna eMMC, que faz com que custe 12 dólares a mais.

Nos testes de placas individuais, embora o *Odroid C2* tenha desempenho parecido com o *Raspberry PI 3 Model B+*, é usado apenas a metade da porcentagem da memoria RAM disponível. Nos testes com apenas duas placas o *Raspberry PI 3 Model B+* e o *Odroid C2* tiveram resultados muitos próximos e superiores aos resultados do *Raspberry Pi 3 Model B*, no entanto ao escalar o *cluster* aumentando o número de nós o *Odroid C2* se torna significativamente mais rápido. Também foi constatado que com a adição de novos nós os ganhos se tornam cada vez menores representando assim uma máxima performance teórica. Na comparação de uso de energia, o *Odroid C2* se saiu melhor entre os três e na relação GFLOPs por dólar gasto, o melhor desempenho foi do *Raspberry PI 3 Model B+*.

Na comparação com trabalhos anteriores foi confirmado a melhora exponencial de

desempenho em *clusters* formados por *Raspberry PI*, onde um *Raspberri PI Model B+* tem mais GFLOPs que um *cluster* de 64 *Raspberry PI 1*. Na comparação com *cluster* tradicionais, ficou constatado que na lista de 500 melhores computadores do ano de 1993, o *cluster Odroid C2* ficaria em primeiro e o *cluster Raspberry PI 3 Model B+* em terceiro.

Tabela 1 – Comparativo entre os estudos relacionados

Título	Ano	Numero de nós	Modelo do Raspberry PI	Tipo de Cluster	Benchmark
The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures	2013	56	*Raspberry PI 1 Model B*	Nuvem	Não Usou
A Container-based Edge Cloud PaaS Architecture based on RaspberryPi Clusters	2016	300	*Raspberry PI 2 Model B*	Nuvem	Não usou
Modelling Low Power Compute Clusters for Cloud Simulation	2017	12	Raspberry PI2 Model B	Nuvem	Não Usou
Comparison of Load Balancing Methods for Raspberry Pi Clustered Embedded Web Servers	2016	3	Raspberry PI2 Model B	Apache Web Server	Não Usou
Study of Raspberry Pi 2 Quad-core Cortex-A7 CPU Cluster as a Mini Supercomputer	2016	4	Raspberry PI 2 Model B	Alto Desempenho	*High Performance Linpack*
Performance of a Low Cost Hadoop Cluster for Image Analysis in Cloud Robotics Environment	2016	20	Raspberry PI 2 Model B	Alto Desempenho Usando Apache Hadoop	Definiu um benchmark próprio
Avaliação de Cluster Raspberry Pi para Execução de Aplicações de Análise de Imagens Microscópicas Médicas	2016	16	Raspberry PI 2 Model B	Alto Desempenho usando MPI	Definiu um benchmark próprio

Performance Analysis of a Low Cost Cluster with Parallel Applications and ARM Processors	2016	4	Raspberry PI2 Model B	Alto Desempenho Usando MPI	*High Performance Linpack*
ZCluster: A Zynq-based Hadoop Cluster	2013	16	Zilinx Zynq SoC	Alto Desempenho Usando Hadoop	*Não usou*
Acceleration of RSA Processes based on Hybrid ARM-FPGA Cluster	2017	48	Zilinx Zynq SoC	Alto Desempenho Usando MPI	*Comparação direta*
Portable Parallel Computing with the Raspberry Pi	2018	1	Raspberry Pi	Alto Desempenho Usando Multicore	*Não usou*
A novel single-arm-worn 24 h heart disease monitor empowered bymachine intelligence	2018	1	Soc Proprio	Inteligência Artificial com SVM	*Comparação*
Performance analysis of single board computer clusters	2020	16	Raspberry PI 3 Model B Raspberry PI 3 Model B+ Odroid C2	Computação de Alto Desempenho	*High Performance Linpack*

3.3 Interpretação dos Resultados

Findada a análise dos artigos encontrados feita na Seção 3.2 podem ser feitas considerações acerca do uso de computadores *Raspberry PI* na construção de *clusters* e o seu uso na computação de Alto Desempenho. Foram encontradas três desses *clusters*, na construção de nuvens, a análise de um *Apache Web Server* e o uso na Computação de Alto Desempenho, sobretudo, fazendo uso das bibliotecas MPI e do *framework Apache Hadoop*, além do uso de ARM para aprendizado de máquina, ainda que não na forma de *cluster*.

A partir dos trabalhos de (TSO et al., 2013; PAHL et al., 2016; KECSKEMETI; HAJJI; TSO, 2017) que foram focados no uso de *Raspberry PI* para a construção de nuvens. Como a construção de uma nuvem não é um dos objetivos deste trabalho, os três artigos citados não

apresentam contribuições significativas para a realização de Computação de Alto Desempenho, visto que o objetivo deste trabalho é analisar o desempenho para grandes quantidade de dados e principalmente a realização de cálculos. Como pontos positivos retirados da leitura deste trabalhos é possível indicar que além do baixo custo, *clusters Raspberry PI* são escaláveis e consumem uma pequena quantidade de energia, além da impossibilidade de usar aplicações de arquitetura x86 diretamente nos computadores ARM.

Mesmo o trabalho de (MADURANGA; RAGEL, 2016) sendo focado na construção de um *Web Server Apache*, já foi possível identificar um problema na construção de um *cluster* de Alto Desempenho. O problema é que quando a quantidade de usuários no servidor era pequena e não havia mudanças significativas no desempenho. Isso é dado pela demora de comunicação entre os nós ser maior que o tempo de processamento interno. Este problema foi confirmado com a leitura dos outros artigos selecionados.

Os trabalhos (MAPPUJI et al., 2016; QURESHI et al., 2016; RAMOS; RALHA; TEODORO, 2016; LIMA; MORENO; DIAS, 2016) foram dentre os encontrados os arquivos mais significativos para esse trabalho. Especialmente Mappuji et al. (2016) falam sobre o custo como um fator importante para o estudo de Computação de Alto Desempenho em computadores ARM, enquanto Lima, Moreno e Dias (2016) trazem o problema que motiva esse trabalho como uma das suas motivações, isso é, mesmo com a capacidade de processamento aumentando o aumento de dados para serem processados também aumentou muito.

O trabalho de Mappuji et al. (2016) ajudou a identificar que o aumento de *cores* em um único *Raspberry PI* não apresenta contribuições significativas no desempenho. Desta maneira conclui-se que a melhor forma de explorar o paralelismo está no número de nós e não na quantidade de *cores* no processador. Também foi identificado que o uso de *overclocking* no processador ARM representa ganhos de desempenho.

Os trabalhos de Qureshi et al. (2016) e Ramos, Ralha e Teodoro (2016) foram semelhantes ao fazer análise de imagens, porém usando métodos diferentes para aplicações diferentes. Enquanto o primeiro usou o *framework Apache Hadoop* o segundo uso as bibliotecas MPI. O primeiro trabalho confirmou o que já havia sido encontrado em (MAPPUJI et al., 2016) que o *overclocking* representa um ganho de desempenho e que o desempenho de um *cluster Raspberry* é inferior ao uso de máquinas virtuais. Já o segundo identificou que o *cluster* construído é melhor do que o desempenho sequencial em computadores *Core2Duo* e I7. Ambos trabalhos confirmaram que a comunicação e o tempo de acesso a memória são gargalos.

Os trabalhos de Lin e Zed (2013) e (BAI et al., 2017) foram pioneiros ao apresentar o uso dos *clusters* de computadores ARM com FPGAs como uma boa alternativa para o ganho de desempenho e os resultados confirmaram essa viabilidade.

já o trabalho de Zhang, Zhou e Zeng (2018) é o único dentre o selecionados a abordar o uso de aprendizado de maquinas usando ARM, no entanto ainda é um estudo inicial e não treina

uma rede de alta complexidade.

A partir das conclusões de (LIMA; MORENO; DIAS, 2016) conclui-se que um *cluster* formado por Raspbery PI apresenta um bom desempenho nas operações matemáticas de multiplicação de matrizes e do produto vetorial. Também confirma que o ganho de desempenho está relacionado a quantidade de nós, aliado disso o desempenho é melhor para grandes quantidades de dados.

A partir da revisão sistemática concluem-se os seguintes pontos positivos na construção de um *cluster Raspberry PI*:

- Escalabilidade;
- Baixo Custo;
- Bom desempenho em operações matemáticas;
- O *overclocking* do processador representa ganhos significativos;
- Baixo consumo de energia.

E os seguintes pontos negativos:

- Operações em disco são lentas;
- A comunicação entre os nós é um gargalo;
- Falta de aplicações que estão disponíveis apenas para x86.

Com os pontos positivos e negativos encontrados nesta Revisão Sistemática é possível concluir que o sistema embarcado *Raspberry PI* apresenta-se como uma opção viável para a realização de uma Computação de Alto Desempenho. Os trabalhos mais relevantes encontrado na revisão foram os de Qureshi et al. (2016), Ramos, Ralha e Teodoro (2016) e Lima, Moreno e Dias (2016). Com a semelhança de tópicos ao comparar à análise de imagem, mesmo em aplicações diferentes apresentadas em Qureshi et al. (2016) e (RAMOS; RALHA; TEODORO, 2016), tem-se uma análise de performance entre o *framework Apache Hadoop* e o protocolo MPI. No entanto o único trabalho que usa aprendizado de maquina é o de Zhang, Zhou e Zeng (2018) ainda não pode-se afirmar quanto a viabilidade e escalabilidade de um *cluster* de computadores ARM para o aprendizado de máquina. De forma que o presente trabalho munido das informações quando ao desempenho dos computadores ARM, buscará alinhar o desempenho já conhecido com a aprendizagem de máquinas para construir um *cluster* que seja capaz de detectar lesões na pele usando *Raspberry PI*.

4

ISIC - International Skin Imaging Colaboration

O *International Skin Imaging Collaboration (ISIC)* é uma colaboração da academia com a indústria focado na construção de aplicações digitais para ajudar a detecção e consequentemente reduzir a mortalidade de melanoma. Que embora seja o tipo mais mortal de câncer de pele, é altamente curável se descoberto nos estágios iniciais. O ISIC é uma proposta para padronizar tecnologia, técnicas e terminologias usadas no análise de lesões da pele com o objetivo de detectar melanoma ou outros tipos de câncer de pele. Além disso o ISIC mantém um acervo publico de imagens e incentiva o desenvolvimento de pesquisas realizando uma competição anual.

Embora as lesões de pele sejam visíveis a olho nu, elas são muito difíceis de distinguir causando um grande número de diagnósticos errados. E mesmo quando são feitas biopsias, muitas delas são desnecessárias e poderiam ser feita apenas com a analise da lesão. Dermatologistas têm usado as técnicas de fotografia e dermartoscopia para analises de lesões e permitir diagnósticos mais cedo de melanomas. Com isso sistemas de aquisição, armazenamento e pesquisa de imagens médicas têm sido desenvolvidos. Essas imagens ajudam a constituir o *dataset* do ISIC e assim permitir a exploração do problema como o problema de Inteligência Artificial.

O objetivo primordial do ISIC é ajudar a diminuir o número de mortes causadas por melanoma e também o número de biopsias desnecessárias com o aumento do diagnostico inicial dos melanomas.

4.1 Padrões propostos pelo ISIC

Para que as imagens sejam utilizadas é necessário que três características sejam cumpridas.

- Qualidade: é necessário uma qualidade miníma entre as imagens usadas;
- Privacidade: os dados do paciente precisam ser desassociados das imagens;

- Interoperabilidade: os metadados de imagens devem ser efetivamente compartilhado por servidores e por provedores.

Baseado nessas características, o ISIC propõe três padrões:

- Padrões de tecnologia: Esses padrões devem assegurar a adequação e compatibilidade entre as imagens com relação a resolução, processamento, metadata, compressão e encriptação;
- Padrões de técnicas: Definir padrões de abordagem a pacientes e a lesões, identificação e documentação dos atributos de uma lesão. Bem como a forma de obtenção das imagens assegurando qualidade, privacidade e interoperabilidade;
- Padrões de terminologia: Esses padrões devem facilitar , o ensino, aprendizagem, busca nas bases e a comparações de imagens. Devem ser divididos em dois tipos:
  - Metadata: Dados relacionados ou embarcados em cada imagem devem ser padronizados para prover interoperabilidade;
  - Privacidade: Padrões sobre a privacidade deve assegurar a manutenção da privacidade e o consentimento para uso das imagens e seus metadados associados.

4.2 Sumario de estudos do ISIC

O site do ISIC dispõe de uma lista de trabalhos gerados a partir do seus desafios e *datasets*, a Tabela 2 representa um sumário dos trabalhos publicados que fazem o uso de inteligência artificial. São descritos rapidamente os resultados e objetivos de cada um deles.

Estudos publicados listados pelo site do ISIC

Titulo	**Rede Usada**	**Principais Resultados**	**Dataset**	**Breve resumo**
Multi-scale classification based lesion segmentation for dermoscopic images	Multiscale Classification	Coenficiênte de Dice 0.91 e Acuracia de 94%	International Skin Imaging Collaboration e Pedro Hispano Hospital	Segmenta cada pixel da como lesão ou pele ao redor e consegue detectar bordas com precisão.
Combining deep learning and handcrafted features for skin lesion classification	Combinação de RSurf com Padrões binários locais	ISIC Dataset	82.6% de acurácia , 0.533 de sensividade e 0.898 de especificidade	O artigo apresenta um método eficiente para detecção de melanomas.

A general algorithm for automatic lesion segmentation in dermoscopy images	Classificador Bayesiano	Coeficiente de Dice 0.89 e de Jaccard 0.79.	900 imagens do ISIC	O artigo é pioneiro ao realizar uma sequencia de processamentos na imagens antes de usa-la no classificador.
An intelligent system for the diagnosis of skin cancer on digital images taken with dermoscopy	Algoritmo C-means	88% de acurácia	ISIC Dataset	O artigo é o primeiro a usar a tecnica ABCD usada por dermatologistas na Inteligência Artificial.
Recognition of skin melanoma through dermoscopic image analysis	SVM	Coeficiente de Jacard 0.7 e acuracia de 63.3%	ISIC Dataset	É o primeiro artigo usar SVM e a baixa acurácia demostra como esse problema é melhor solucionado por redes profundas.
Skin lesion classification from dermoscopic images using deep learning techniques	VGGnet e SVM	77% de acuracia	ISIC Dataset	É um artigo pioneiro ao usar transferência de conhecimento entre as redes.
Deep learning for skin lesion segmentation	VGG	Supera todos os participantes em termos de segmentação	Subconjunto do Dataset do ISIC	O artigo foca na importância na segmentação de imagens e eliminação de viés.
Araguaia Medical Vision Lab at ISIC 2017 Skin Lesion Classification Challenge	GoogleNet e AlexNet	95% de acurácia para lesões de melanoma e 84% para keratosis	Subconjunto do dataset ISIC	Apresenta estudos focados em lesões especificas.

A hybrid dermoscopy images segmentation approach based on neutrosophic clustering and histogram estimation	HBCE	96.3% de Acuracia	Subconjunto do ISIC.	O trabalho foca na aglomeraçãao de pixels com o objetivo de detectar lesòes baseados em intensidade e características morfológicas.
Optimal selection of features using wavelet fractal descriptors and automatic correlation bias reduction for classifying skin lesions	SVM-RFE	97.63% de sensibilidade, 100% de especificidade e 98.28% de acurácia	4094 imagens de lesões	O trabalho busca encontrar as caraactericas mais importantes para a classificação, tornando assim a rede mais simples.
A Novel Skin Lesion Detection Approach Using Neutrosophic Clustering and Adaptive Region Growing in Dermoscopy Images	C means	95.3% de acurácia	Dataset ISIC	As imagens foram divididas em verdadeiras, indeterminadas e falsos verdadeiros, então um filtro foi aplicado e as imagens aglomeradas pelo algoritmo de classificação.
Automated skin lesion segmentation of dermoscopic images using GrabCut and k-means algorithms	K-means	Coeficiente de dice 0.82 para o ISIC e 9.91 para o PH^2.	ISIC E $PH^2 datasets$.	O artigo explora a detecção das lesões usando duas fases. Na primeira é feita o preprocessamento e depois usando o Grabcut na segmentação e só então o k-means é usado.

Fully Convolutional Neural Networks to Detect Clinical Dermoscopic Features	Rede Neural convolucional	Primeiro lugar no desafio de 2017 do ISIC	ISIC Dataset	Os autores argumentam que o problema de detectar melanoma pode ser abordado como um problema de segmentação.
Skin Lesion Analysis towards Melanoma Detection Using Deep Learning Network	FCRN	75.3% para tarefa 1, 84.8% para tarefa 2 e 91.2% para tarefa 3	Dataset ISIC.	A primeira tarefa focou em segmentação, a segunda em extração de características e a terceira em classificação da lesão.
Deep convolutional neural networks as a decision support tool in medical problems – malignant melanoma case study	Rede Neural convolucional	84% de acurácia	Dez mil imagens de lesões	O autores propoem uma maneira de lidar com o pequeno dataset.
Apply lightweight deep learning on internet of things for low-cost and easy-to-access skin cancer detection	Redes Hibridas	80%	ISIC Dataset	o tabalho é focado no uso de redes pretreinadas que podem ser embarcadas em dispositivos menores e de baixo custo como o Raspberry PI.
Adversarial learning with multi-scale loss for skin lesion segmentation	Redes adversarias	Primeiro lugar no desafio ISIC de 2017	Dataset ISIC	É primeiro artigo a abordar o problema usando redes adversárias.
Skin Lesion Analysis By Multi-Target Deep Neural Networks	DCNN , GoogleNet e U-Net	DCNN é superior as outras duas	Dataset ISIC	É o o primeiro trabalho a atacar os problemas de segmentação e classificação simultaneamente usando uma rede convolucional.

Construction of Saliency Map and Hybrid Set of Features for Efficient Segmentation and Classification of Skin Lesion	SVM, Obtém sucesso em remover características desnecessárias da rede.	PH2	ISIBI2016 e ISIC	O artigo aborda o uso de SVM para otimizar caracteristicas.
Fusing fine-tuned deep features for skin lesion classification	Rede Neural Convolucional	87% para melonoma e 95.5% para ceratose seborreica	ISIC Dataset	A rede usada nesse artigo consiste em um conjunto de CNNs pre treinadas e cada uma tem foco em uma característica da lesão.
Domain-specific classification-pretrained fully convolutional network encoders for skin lesion segmentation	ResNet34 e VGG16	Resnet-34 é mais eficaz que a VGG16	ISIC Dataset	O artigo conclue que a complexidade nas bases da segmentação tem um grande impacto nas métricas e devem ser levadas em consideração.
Attention Residual Learning for Skin Lesion Classification	ARL-CNN	Atinge o estado da arte no desafio do ISIC 2017	ISIC dataset	A rede ARL-CNN foi proposta neste trabalho e apresenta bom desempenho focando em partes discriminativas da lesão.

4.3 Artigos selecionados

Dentre os arquivos sumarizados na tabela 2, 7 deles foram selecionado por abordar aprendizado profundo ou SVM que são os tópicos de interesse desse trabalho, portando essa seção traz um resumo mais detalhado de cada um deles.

4.3.1 Combining deep learning and hand-crafted features for skin lesion classification

O trabalho de Majtner, Yildirim-Yayilgan e Hardeberg (2017) aborda o problema de detecção do melanoma. Mesmo sendo mortal o melanoma é altamente curável caso seja descoberto ainda em estágios iniciais. No entanto é fácil a confusão entre células malignas e as células benignas que são chamadas de moles. O trabalho apresenta uma alternativa para a detecção usando uma combinação de características reunidas manualmente com características aprendidas pela rede convolucional.

O primeiro método usado é o *RSurf*, baseado na ideia de dividir a imagem em sequencias paralelas de intensidades locais, conhecidos como *slopes*. O conjunto de *slopes* é usado para extrair características das imagens. Ao final cada função é caracterizada em um histograma, e a junção dos histogramas forma os vetores de características. Nos testes realizados no trabalho o melhor vetor tinha o tamanho de 2000 elementos.

Local Binary Patterns (LBP) é um descritor baseado em modelos de textura e onde cada imagem é descrita através de um espectro de textura. Cada pixel da imagem é comparado com oito *pixels* vizinhos e classificado como 0 ou 1. Depois é gerado um histograma que por sua vez dá origem ao vetor de características.

Após a extração dos vetores de características do *Rsurf* e do LBP, duas redes SVM são executadas, a primeira delas usa as características do *Rsurf* e do LBP, e a segunda um SVM derivado de uma rede convolucional e usa características profundas aprendidas pela rede. Ao final os dois resultados são comparados e chega-se ao resultado final.

Os resultado final foi de 82,6% de acurácia , 0,533 de sensividade e 0,898 de especificidade. Fazendo assim a quinta colocação no desafio do ISIC de 2016.

4.3.2 Skin lesion classification from dermoscopic images using deep learning techniques

O diagnostico de melanoma tem uma acurácia clínica entre 65 e 80% e o uso de imagens dermatoscópicas pode representar um ganho de 49%. As diferenças entre lesões malignas e as benignas podem ser sutis e difícil de distinguir até por especialistas. No entanto essas imagens podem ser analisadas com a ajuda da inteligência artificial e o trabalho de Romero Lopez et al. (2017) busca encontrar a melhor configuração de redes CNN para esse problema.

Três redes neuronais foram treinadas usando o dataset de ISIC, na primeira delas a rede foi treinada sem nenhum conhecimento anterior e serviu de parâmetro para o desempenho das outras duas redes. Na segunda a CNN usou a técnica de transferência de conhecimento ao usar a rede pre-treinada VGG16 que foi treinada no dataset ImageNet, e por fim, na última rede a transferência de conhecimento também foi usada porém associada com a técnica de *fine-tuning*,

usada nesse trabalho para congelar as camadas inferiores.

Os resultados para cada uma das rede foram explorados em cinco métricas, sendo elas: perda, sensibilidade, precisão, especificidade e acurácia. Os resultados obtidos para a primeira rede foram considerados aceitáveis e com uma acurácia acima da chance e não apresentou sinais de *overfit* ou *underfit*. Os resultados da terceira rede foram superiores no conjunto de testes e os resultados altos da segunda rede no conjunto de treinamento mostra um *overfit*. Os resultados finais foram mais promissores nos critérios de sensibilidade e de precisão, atingindo 78.66 e 79.74%, respectivamente, no critério de acurácia atingiu 91,5%.

4.3.3 Araguaia Medical Vision Lab at ISIC 2017 Skin Lesion Classification Challenge

Sousa e Moraes (2017) apresentam uma abordagem que considera não apenas as lesões de melanoma, mas também a ceratose seborréica, seus diferentes algoritmos foram treinados com os modelos do GoogleNet e AlexNet. As imagens foram redimensionadas para 256x256 na fase de pré-processamento e foram usadas 72 épocas na GoogleNet 256, 50 épocas na GoogleNet 224 e 30 para AlexNet. O resultado da média aritmética de três redes foi de 84,7%

4.3.4 Fully Convolutional Neural Networks to Detect Clinical Dermoscopic Features

O trabalho de Kawahara e Hamarneh (2019) propõe uma abordagem para reformular o superpixel na abordagem na tarefa de classificação transformando em um problema de segmentação e usa o *finetune* em uma CNN pretreinada para detectar características clínicas estudadas. Essa nova abordagem é baseada no argumento que os superpixels apenas classifica cada *pixel* individualmente ignorando os contexto em volta, como a posição do *pixel* em relação ao restante da lesão.

A transformação em segmentos é realizada na fase do preprocessamento e depois o treinamento é feito usando uma rede VGG16 pre-treinada o ImageNet e é mapeado as respostas e características ao longo da rede.

Os resultados dessa abordagem levaram a rede proposta ao primeiro lugar na segunda parte do desafio do ISIC de 2017 com a maior média no operador de curva das lesões tanto no conjunto de validação público quanto no privado. Também atingiu a maior pontuação no AUROC que analisa as redes de pigmentos, as redes negativas e as sequências. Também ouve bons resultados quando a rede foi testada na primeira parte do desafio, focada em detecção de características clínicas. Mostrando que ambos os problemas podem ser abordados da mesma maneira.

4.3.5 Apply lightweight deep learning on internet of things for low-cost and easy-to- access skin cancer detection Apply Lightweight Deep Learning on Internet of Things for Low-Cost and Easy-To-Access Skin Cancer Detection

O trabalho de (SAHU; YU; QIN, 2018) usa um Raspberry Pi para avaliar a análise de lesões cutâneas com o udo de um computador de mão, como o próprio Raspberry, smartphones ou SoCs (*system-on-a-chip*). Eles propuseram e implementaram uma abordagem híbrida baseada no uso de modelos de aprendizado profundo e bases de domínio específicas do conhecimento em recursos que os dermatologistas usam para detectar câncer de pele.

Obtendo os recursos de aprendizado profundo do modelo pré-treinado do Google MobileNet. Eles também treinaram uma rede no Raspberry PI e detectaram que o Movidius Neural Compute Stick é cinco vezes mais rápido que o Raspberry PI.

A tarefa final deles foi transferir o conhecimento para uma rede SVM e rodar em um Raspberry. Eles usaram o conjunto de dados de desafio do ISIC 2017 com 2700 imagens e obtiveram um resultado entre os 10 principais concorrentes.

4.3.6 Recognition of skin melanoma through dermoscopic image analysis

O Trabalho de Gomez e Herrera (2017) aborda o contexto de que as análises dematoscópicas são uma analise de padrões sobre as imagens, sobretudo usando as características da regra ABCDE. No entanto como o ISIC dispõe apenas de uma imagem por lesão a evolução foi retirada dos parâmetros estudados. A razão apresentada pelos autores para o interesse é facilitar o acesso a diagnóstico principalmente em áreas onde a saúde não é acessível a todos como a Colômbia.

Para a segmentação das imagens foram realizados preprocessamentos, dentre eles distinguir as cores das imagens e depois as imagens foram tranformadas em diferentes canais para descobrir em qual canal que a lesão era melhor detectada. Também foram removidas bordas pretas das imagens. Para evitar o *overfitting* imagens com lesões malignas e benignas foram normalizadas para terem a mesma quantidade.

Após o preprocessamento o dataset foi treinado em um SVM e testado com diferente *kernels*, foram extraídas as características do ABCD durante a criação da rede.

Foram treinadas redes com diferentes argumentos e configurações, onde o melhor desempenho no conjunto de testes foi de 63,3% de acurácia. Sendo que a maioria dos registados são favoráveis. No entanto nesse trabalho é possível observar que há uma grande diferença de acurácia atingida nas redes profundas para as redes puramente SVM.

4.3.7 Optimal selection of features using wavelet fractal descriptors and automatic correlation bias reduction for classifying skin lesions

O trabalho de Chatterjee, Dey e Munshi (2018) usa SVM-RFE, uma técnica de eliminação de características da rede, mantendo apenas as mais relevantes para a classificação de melanomas.

O trabalho pode ser sumarizado em três grandes contribuições: O uso de um descritor fractal para a caracterização de padrões na área da lesão, e então extrair a distribuição de textura através da lesão. Para detecção da irregularidade da borda, foi usado o cálculo de séries de distancia dos pixéis da borda para o pixel central. E a classificação dos resultados foi feita usando a seleção automática de características possibilitada pelo SVM-RFE.

A grande novidade desse trabalho é a abordagem da borda da lesão como uma geometria fractal, isso é objetos com não usual e alto nível de complexidade em formas irregulares.

Para a criação da rede o algoritmo RFE é usado com o SVM, onde o SVM é primeiro treinado com todas as características detectadas e cada uma dessas características é ranqueada de acordo com seu valor estimado. A cada interação a característica de menor valor é retirada. Para um conjunto de muitas faturas o tempo necessário é grande e pode inserir o viés da correlação. Para reduzir o viés características correlatas são agrupadas e excluidas em grupo.

A performance da classificação foi calculada nos parâmetros padrões de sensibilidade, especificidade e acurácia. Chegando a 98.53%, 100% e 98.28% respectivamente. Embora outros fatores, como a abordagem fractal das formas geométrica, provavelmente, tenham exercido influência. É possível supor que o uso de SVM-RFE traz resultados melhores que o SVM puro, principalmente se comparado com os resultados de Gomez e Herrera (2017).

4.4 Justificativa sobre textos selecionados

Todos os sete trabalho selecionados usam SVM ou redes CNN profundas de diversas maneiras. Dessa forma a partir da leitura detalhada de cada um deles é possível extrair detalhes cruciais para a realização dos experimentos dessa pesquisa.

O trabalho de Gomez e Herrera (2017) traz grandes contribuições ao mostrar os ganhos de desempenho provocados pela normalização das imagens benignas e malignas. Também é importante extrair a contribuição ao mostrar que a evolução da regra ABCDE pode ser descartada durante o processo de aprendizagem, pois o database do ISIC não dispõe de mais de uma imagem para a mesma lesão.

O trabalho de Chatterjee, Dey e Munshi (2018) complementa o trabalho de Gomez e Herrera (2017) ao mostrar que é possível obter resultados mais promissores usando variações do SVM, no entanto sendo ainda mais leve em questões de processamento que as redes CNN. Esta mesma técnica pode ser encontrada no trabalho de Majtner, Yildirim-Yayilgan e Hardeberg

(2017).

No entanto entre os trabalhos selecionados o de maior contribuição para esse trabalho é o de Sahu, Yu e Qin (2018) que ao usar a técnica de transferência de conhecimento para o *Raspberry* demonstra que os objetivos desse trabalho podem ser promissores.

Após a leitura do estado da arte do aprendizado de maquina usando o database do ISIC e seus subconjuntos será possível definir as implementações que são usadas na parte prática deste trabalho.

5

Preprocessamento e escolha de redes pre-treinadas

Esse capitulo tem por objetivo esclarecer as decisões tomadas para a escolha do modelo usado nesse trabalho. Também será descrita a forma de obtenção da média de tamanho das imagens, o balanceamento do *dataset*. Por fim também será descrita cada uma das redes pre-treinadas usadas no presente trabalho.

5.1 O Modelo

Para conseguir uma analise dos dados, o processo de construção do modelo foi dividido em partes: Analise dos dados, processamento de dados,escolha de redes pre-treinadas utilizadas , a construção do modelo em si e por fim, a otimização do modelo.

A primeira tarefa realizada na analise dos dados foi a construção de um dicionario relacionando as imagens do *dataset* com sua classificação obtida com a ajuda do csv anexo ao *dataset* do ISIC usado. Com isso cada imagem está relacionada a uma das 7 classes disponíveis no *dataset*

Listing 5.1 – Código para obter as imagens do Google Drive e criar os dicionarios

```
  data_dir = '../content/drive/My Drive/input'
  all_image_path = glob(os.path.join(data_dir, '*', '*.jpg'))
  imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x for x in all_
lesion_types_dict = {
  'NV': 'Melanocytic nevi',
  'DER': 'dermatofibroma',
  'BKL': 'Benign keratosis-like lesions ',
  'BCC': 'Basal cell carcinoma',
  'AK': 'Actinic keratoses',
  'VS': 'Vascular lesions',
```

```
    'DF': 'Dermatofibroma'
}
```

No segundo passo do preprocessamento nos calculamos a média e desvio padrão para o *dataset* inteiro. Levando em consideração que o cálculo para um *dataset* deste tamanho o tempo desse preprocessamento é considerável. Essa fase do preprocessamento foi feita em uma tarefa separada e o resultado foi usado como uma constante.

Listing 5.2 – Média e desvio padrão para o dataset

```
    norm_mean = [0.7630373, 0.5456457, 0.57003844]
    norm_std = [0.14092845, 0.15261292, 0.16997094]
}
```

Após o cálculo da média e desvio padrão todas as imagens do *dataset* foram lidas e de acordo com seus identificadores (ids) foram checadas se há duplicidade no em imagens. Na divisão do *HAM1000* há imagens duplicadas de uma mesma lesão e isso deve ser levado em consideração na aplicação do treinamento. Nessa fase do preprocessamento foi idenficado que há 5514 imagens não duplicadas e 4501 duplicadas. Ao final dessa fase foi possível obter o número de imagens para cada um dos tipos de lesões. Na figura 2 se observa a quantidade de lesões em cada classe nesse dataset, é possível constatar que há um desbalanceamento entre cada tipo de lesão no *dataset*. Esses valores representam o numero de classes malignas no *dataset*, o restante das imagens são de lesões benignas. Na figura **??** é possível visualizar cada tipo de lesão.

```
Melanocytic nevi                 883
Benign keratosis-like lesions     88
dermatofibroma                    46
Basal cell carcinoma              35
Actinic keratoses                 30
Vascular lesions                  13
Dermatofibroma                     8
Name: cell_type, dtype: int64
```

Figura 2 – Imagens por classe antes do balanceamento

Para contornar o desbalanceamento usamos uma função de balanceamento e obtemos a seguinte proporção:

5.2 Redes Pre-treinadas Selecionadas

Para performar essa tarefa de aprendizado foram selecionadas quatro redes pretreinadas e com uso do *framework pytorch* mudanças minímas no código são necessárias para a utilização de cada uma das redes. Esta seção dá um panorama geral sobre as particularidades de cada uma das redes pretreinadas que são utilizadas.

```
Melanocytic nevi                  5822
Dermatofibroma                    5350
dermatofibroma                    5335
Vascular lesions                  5160
Benign keratosis-like lesions     5055
Basal cell carcinoma              4790
Actinic keratoses                 4455
Name: cell_type, dtype: int64
```

Figura 3 – Imagens por classe depois do balanceamento

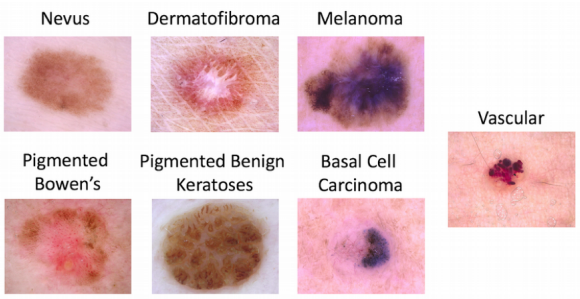

Figura 4 – Tipos de lesões de pela

5.2.1 Densenet

A *densenet* foi uma rede desenvolvida em conjunto entre a Universidade de Cornwell , a Universidade de Tsinghua e o *Facebook AI Research(FAIR)* (HUANG; LIU; WEINBERGER, 2016). Que com uma conexão mais densa e menos parâmetros atinge acurácias maiores quando comparadas as *Resnet*.

Em uma rede convolucional normal as imagens de entrada passam por múltiplas convoluções para obter as *features* da rede. Em redes como a *ResNet* o mapeamento de identidade é passado de uma camada para outra por meio de um gradiente de propagação. Na *DenseNet* cada camada obtém entradas adicionais enviadas por todas as camadas anteriores e cada camada consequentemente insere seu mapeamento de *features* nas camadas seguintes.

Com todas as camadas recebendo *features* de todas as camadas anteriores a rede pode ser mais fina e compacta, assim um número menor de canais pode ser usado no treinamento da rede.

A arquitetura da rede *DenseNet* é composta em cada camada por uma ativação *Batch Norm (BN)* seguida por um ReLU e então uma convolução de 3x3.

Algumas vantagens das redes *Densenet* são: Um forte fluxo de gradiente, eficiência de parâmetros computacionais, é possível obter *features* mais diversificadas e manter as *features* mais complexas.

5.2.2 Resnet

A rede *ResNet* é a vencedora do ILSVRC 2015 classificação de imagens, detecção e localização, além de ter também vencido o MS COCO 2015 em detecção e segmentação(HE et al., 2015). As redes *ResNet* podem ser muito profundas, chegando até 152 camadas aprendendo funções de representação residual no lugar de representação direta do sinal. A rede *ResNet* introduz o conceito de *skip connection* para contornar os problemas de *Vanishing/Exploding Gradient*.

As redes convolucionais comuns usualmente tem camadas convolucionais completamente conectadas sem nenhum *skip* ou *shotcurt*, assim sendo chamadas de redes planas. Quando essas redes se tornam profundas o problema de *vanishing/ exploding gradient* ocorre. Isto é: durante a fase de *backpropagation* quando uma derivada parcial tem um erro no peso atual esse efeito é propagado n vezes, onde n é o número de camadas. Quando a rede é profunda um erro pequeno multiplicado por n irá se tornar próximo de zero *(vanish)*, enquanto um erro de um número grande fará com que esse erro cresça *(exploded)*.

Para contornar isso a *ResNet* introduz o *skip/shortcut connection* resolvendo o problema adicionando o input de uma camada x diretamente na saída de uma camada a frente. Dessa maneira mesmo se o erro *vanishing/ exploding gradient* ocorre ele é mitigado pelo x vindo diretamente de algumas camadas anteriores.

5.2.3 VGG

A rede VGG é uma evolução da *AlexNet*, rede vencedora do ILSVRC 2012. Ela é focada em entradas menores e passos mais largos na primeira camada convolucional e tem um foco maior na profundidade de rede que as redes convolucionais anteriores (SIMONYAN; ZISSERMAN, 2015).

A entrada da VGG usa um canal RGB de 224x224 *pixels* , cada camada convolucional tem um filtro muito curto de 3x3 e também um filtro de 1x1 que funciona como uma transformação linear. A VGG tem três camadas completamente conectadas onde as 2 duas primeiras tem 4096 canais e a última tem 1000 canais. Todas as suas camadas ocultas usam um ReLU e não usa normalização de resposta local.

As principais inovações em relação a *AlexNet* estão nas entradas menores de 3x3 em comparação as entradas de 11x11 em relação ao uso de 3 unidades de ReLU ao invés de apenas uma, a função de decisão consegue ser mais discriminativa.Com isso a VGG necessita de menos parâmetros para funcionar. O uso de filtro 1x1 nas camadas fazem que a função de decisão seja mais não linear. Com o tamanho de convolução menos VGG tem o número maior de camadas.

5.2.4 Inception

A rede *inception* é um marco no desenvolvimento de redes convolucionais, pois antes dela as redes mais populares apenas ficavam mais profundas na esperança de obter mais performance. A *inception* por outro lado usou vários artifícios para ganho de performance, tanto na velocidade quanto na acurácia. Ela foi desenvolvida em maneira sequencial tendo assim quatro versões famosas (Szegedy et al., 2015).

Na primeira versão os autores detectaram que escolher o tamanho ideal do *kernel* de convolução é uma tarefa bastante complicada, pois enquanto um *kernel* maior é melhor para identificar informações distribuídas globalmente, um *kernel* menor é usado para informações locais. Outro problema é que redes muito profundas podem causar *overfitting* e apenas tornar a rede maior pode ser computacionalmente muito custoso.

Para isso os autores decidiram usar múltiplos filtros em uma mesma camada, assim são realizadas três operações convolucionais em cada camada com três diferentes filtros, 1x1, 3x3 e 5x5. Apos os filtros uma operação de *max pooling* é realizada e o resultado concatenado é enviado para a próxima camada. O filtro de 1x1 é usado para diminuir o número de canais tornando assim as convoluções menos custosas.

As versões 2 e 3 foram prublicadas em um mesmo trabalho e propõem um número de atualizações que melhoram a perfomance da inception (SZEGEDY et al., 2016). Na segunda versão o filtro 5x5 foi substituído por duas convoluções de 3x3, pois ainda assim é menos custoso do que realizar uma convolução de 5x5. Na versão 3 foram introduzidas e otimizadas as *RMSProp*, convoluções de 7x7 factorizadas, o uso de *BatchNorm* nos auxiliadores de classificação e a tecnica de *Label Smoothing* usada para previnir *over fitting*.

Para a versão 4 (SZEGEDY et al., 2017) os autores identificaram que alguns módulos da rede estavam muito mais complexos que o necessário, para mitigar esse problema e ganhar performance, o conjunto de operações performadas antes de cada bloco de *inception* foi alterado.

5.3 Construção do modelo

Inicialmente o *dataset* balanceado é dividido em três partes: Conjunto de treinamento, conjunto de validação e conjunto de treinamento. Durante esse trabalho foram as divisões de dataset descritas na Tabela 3 (onde aparecem as divisões I, II e III, e suas características).

Com a divisão do *dataset* realizada é possível então inicializar o modelo (conforme indicado no código 5.3)

Listing 5.3 – Código para inicialização do modelo

```
def initialize_model(model_name, num_classes,
            feature_extract, use_pretrained=True):
```

Tabela 3 – Divisões de *dataset*

	Porcentagem da divisão		Quantidade de imagens		
Divisão do Dataset	Treinamento	Teste	Treinamento	Validação	Teste
I	75	25	6000	1500	2500
II	80	20	6400	1600	2000
III	90	10	7200	1800	1000

```
model_ft = None
input_size = 0

if model_name == "resnet":
    model_ft = models.resnet50(pretrained=use_pretrained)
    set_parameter_requires_grad(model_ft, feature_extract)
    num_ftrs = model_ft.fc.in_features
    model_ft.fc = nn.Linear(num_ftrs, num_classes)
    input_size = 224
elif model_name == "vgg":
    model_ft = models.vgg11_bn(pretrained=use_pretrained)
    set_parameter_requires_grad(model_ft, feature_extract)
    num_ftrs = model_ft.classifier[6].in_features
    model_ft.classifier[6] = nn.Linear(num_ftrs,num_classes)
    input_size = 224
elif model_name == "densenet":
    model_ft = models.densenet121(pretrained=use_pretrained)
    set_parameter_requires_grad(model_ft, feature_extract)
    num_ftrs = model_ft.classifier.in_features
    model_ft.classifier = nn.Linear(num_ftrs, num_classes)
    input_size = 224
elif model_name == "inception":
    model_ft = models.inception_v3(pretrained=use_pretrained)
    set_parameter_requires_grad(model_ft, feature_extract)
    num_ftrs = model_ft.AuxLogits.fc.in_features
    model_ft.AuxLogits.fc = nn.Linear(num_ftrs, num_classes)
    num_ftrs = model_ft.fc.in_features
    model_ft.fc = nn.Linear(num_ftrs,num_classes)
    input_size = 299

else:
```

```
        print("Invalid model name, exiting...")
        exit()
    return model_ft, input_size
```

Com o Código 5.3 é possível criar o modelo para o treinamento apenas passando o nome da rede desejada como parâmetro, o número de classes e se é desejável ou não a extração de caracteristicas. O ultimo parâmentro é usado sempre como verdadeiro, assim sempre é usada uma rede pretreinada que facilita a convergência da rede.

Em seguida a criação do modelo, foi criada a classe HAM10000, que é uma classe que será utilizada pelo *dataloader* do *Pytorch*, o Código 5.4 mostra a implementação da classe:

Listing 5.4 – Código da clase HAM10000

```
class HAM10000(Dataset):
def __init__(self, df, transform=None):
    self.df = df
    self.transform = transform

def __len__(self):
    return len(self.df)

def __getitem__(self, index):
    # Load data and get label
    X = Image.open(self.df['path'][index])
    y = torch.tensor(int(self.df['cell_type_idx'][index]))

    if self.transform:
        X = self.transform(X)

    return X, y
```

Com a função de inicializar modelos pronta e a classe HAM1000, é então possível inicializar o modelo para qualquer uma das redes pre-treinadas selecionadas:

Listing 5.5 – Código da clase HAM10000

```
model_name = 'densenet'
num_classes = 7
feature_extract = False
model_ft, input_size = initialize_model(model_name, num_classes,
        feature_extract, use_pretrained=True)
```

```
device = torch.device('cuda:0')
model = model_ft.to(device)
training_set = HAM10000(df_train, transform=train_transform)
train_loader = DataLoader(training_set, batch_size=32,
        shuffle=True, num_workers=4)
validation_set = HAM10000(df_val, transform=train_transform
val_loader = DataLoader(validation_set, batch_size=32,
    shuffle=False, num_workers=4)
optimizer = optim.Adam(model.parameters(), lr=1e-3)
criterion = nn.CrossEntropyLoss().to(device)
```

Na linha 1 do Código 5.5 o modelo de rede pretreinada é escolhido, no caso a DenseNet, então na linha 2 é dito para o modelo quantas classes existem no *dataset*, na linha 3 é definido se as características serão extraídas em camadas intermediarias ou apenas na ultima, Por fim na linha 5 o modelo é iniciado. Nas linhas 6 e 7 é informado ao *pytorch* que esse modelo irá ser executado em GPU. Das linhas 8 a 12 são executadas transformações nos conjuntos de treinamento e validação, as transformações são realizadas usando funções disponibilizadas pelo pytorch. Nas linhas 13 e 14 é definido o otimizador Adam para usar entropia cruzada como a função de perda do modelo.

Com o modelo iniciado e configurado desta maneira é possível então realizar o treinamento.

5.4 Treinamento do Modelo

Depois da fase de contrução do modelo é possível então iniciar a fase de treinamento deste modelo. Para isso foi preciso de uma classe que serve para armazenar a perda e acurácia do treinamento representada pelo Código 5.6.

Listing 5.6 – Código da clase AverageMeter

```
class AverageMeter(object):
def __init__(self):
    self.reset()

def reset(self):
    self.val = 0
    self.avg = 0
    self.sum = 0
    self.count = 0

def update(self, val, n=1):
```

```
        self.val = val
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count
```

Por fim então foi criada a função de treinamento: Primeiro para obter uma sequencia de perda e acurácia durante cada época do treinamento foram criados dois *arrays* que armazenam as taxas obtidas a cada 100 iterações durante o treinamento, com esses dados é possível ver o comportamento da rede durante todo o treinamento . Esses *arrays* são criados na linha 1 do Código 5.7. Logo em seguida é definida a função de treinamento. Que recebe por parâmetro o conjunto de treinamento, o modelo, o *criterion*, e a época que está sendo treinada na ocasião. A cada iteração a perda e a acurácia são atualizadas e ao final elas são retornadas pela função de treinamento.

Listing 5.7 – Código da função de treinamento

```
total_loss_train, total_acc_train = [],[]
def train(train_loader, model, criterion, optimizer, epoch):
model.train()
train_loss = AverageMeter()
train_acc = AverageMeter()
curr_iter = (epoch - 1) * len(train_loader)
for i, data in enumerate(train_loader):
    images, labels = data
    N = images.size(0)

    images = Variable(images).to(device)
    labels = Variable(labels).to(device)

    optimizer.zero_grad()
    outputs = model(images)

    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()
    prediction = outputs.max(1, keepdim=True)[1]
    train_acc.update(prediction.eq
    (labels.view_as(prediction)).sum().item()/N)
    train_loss.update(loss.item())
    curr_iter += 1
    if (i + 1) % 100 == 0:
```

```
            total_loss_train.append(train_loss.avg)
            total_acc_train.append(train_acc.avg)
    return train_loss.avg, train_acc.avg
```

Em seguida na função de validação descrita pelo Código 5.8 o modelo que foi treinado é testado contra o conjunto de validação. Na função de validação também são calculadas as perdas e acurácia para o conjunto de validação.

Listing 5.8 – Código da função de validação

```
def validate(val_loader, model, criterion, optimizer, epoch):
model.eval()
val_loss = AverageMeter()
val_acc = AverageMeter()
with torch.no_grad():
    for i, data in enumerate(val_loader):
        images, labels = data
        N = images.size(0)
        images = Variable(images).to(device)
        labels = Variable(labels).to(device)

        outputs = model(images)
        prediction = outputs.max(1, keepdim=True)[1]

        val_acc.update(prediction.eq(
        labels.view_as(prediction)).sum().item()/N)

        val_loss.update(criterion(outputs, labels).item())
```

A fase do final do treinamento consiste em criar um *loop* em que repetidas vezes o modelo é treinado e colocado contra ao conjunto de validação. A cada época, a perda e acurácia tanto para o conjunto de treinamento quanto validação são salvas e com base nisso o modelo pode ser avaliado (Ver Capitulo 6). O Código 5.9 representa o uso de épocas durante a fase de treinamento.

Listing 5.9 – Código de execução de épocas

```
epoch_num = 10
best_val_acc = 0
total_loss_val, total_acc_val = [],[]
for epoch in range(1, epoch_num+1):
```

```
loss_train, acc_train = train(train_loader, model,
   criterion, optimizer, epoch)
loss_val, acc_val = validate(val_loader, model,
   criterion, optimizer, epoch)
total_loss_val.append(loss_val)
total_acc_val.append(acc_val)
```

5.5 Considerações final do capítulo

Neste capitulo foram abordados os passos realizados para o preprocessamento do *dataset* do HAM1000. Também foram apresentadas de forma resumida, as quatro redes pre-treinadas usadas neste trabalho e suas particularidades. As quatro redes são: *Densenet*, *Resnet*, *VGG* e *Inception*. Por fim foi mostrado passo-a-passo a criação do modelo e e como realizar o treinamento.

Os trechos de códigos exibidos nesse capítulo tem por objetivo ajudar a compreensão dos resultados que são exibidos e analisados no Capítulo 6.

6

Resultados e Discussões

Neste Capítulo aborda-se os resultados obtidos com o uso do modelo descrito no Capítulo 5. Com base nos resultados serão apresentadas discussões sobre cada uma das descobertas e como cada uma delas impactou o processo de desenvolvimento do modelo para esse trabalho.

Na primeira fase foi investigada a acurácia e o tempo de convergência do modelo para cada uma das redes pretreinadas que foram abordadas no Capítulo 5. Na Tabela 5 mostramos a acurácia para cada uma das redes usando as divisões de *dataset* definidas na Tabela 4, cada divisão representa o número de imagens que foram selecionadas para os conjuntos de treinamento,validação e teste. O tempo médio utilizado no treinamento para cada uma das redes foi de em média 154,5 minutos para atingir 89% de precisão em dez épocas. Para calcular o tempo médio usamos as bibliotecas de tempo do *Python* para obter uma média de tempo utilizado desde preprocessamento, até a finalização da décima época, os tempos foram salvos 12 vezes e o tempo maior e menor foram descartados, fazendo a média dos dez restantes.

Tabela 4 – Divisões de *dataset*

	Divisão do *dataset*		Rede		
Dataset de treinamento	Treinamento	Teste	Treinamento	Validação	Teste
I	75	25	6000	1500	2500
II	80	20	6400	1600	2000
III	90	10	7200	1800	1000

6.1 Seleção da melhor rede

Os resultados apresentados na Tabela 5 foram coletados com a ajuda do Google Collaboratory (COLAB) usando uma GPU. O principal intuito destes destes testes foram identificar em qual das redes pre-treinadas o modelo que foi proposto no Capítulo 5 apresentavam maior

Tabela 5 – Acurácia das redes

	Acurácia				
	Divisão do Dataset				
Rede Pretreinada	I	II	III	Média da Rede	Média Geral
DenseNet	83	82	89	84,6	82,3
Resnet	82	81	83	82	
VGG	76	80	83	79,6	
Inception	79	82	82	81	

acurácia. Na Tabela 5 os valores para cada rede representam a média de 12 execuções onde os valores maiores e menores foram descartados para a retirada de possíveis *outlines*.

Dentre as redes pretreinadas a rede que saiu melhor em todas as divisões do *dataset* foi a rede *DenseNet*. Com uma acurácia de 89% na terceira divisão do *dataset* fazendo com que fosse o melhor resultados que foi obtido pelo modelo.

É possível identificar com à análise de toda a tabela, que para todas as redes a divisão de número 3 do *dataset* é sempre a melhor. Enquanto as divisões I e II acabam sendo pior. No entanto é possível identificar que a média entre as três divisões de *dataset* é bem próxima para as quatro redes.

A análise desse resultado indica que pode existir um *overfitting* quando é usado um conjunto de treinamento. Para solucionar tal dúvida é necessário investigar o número de entrada de cada classe que ficou no conjunto de teste e se houve uma aprendizagem maior em alguma das classes. No entanto tal problema não foi explorado neste trabalho, visto que o principal objetivo dele é trabalhar com as redes na plataforma *Raspberry*, como seria uma investigação mais voltada para a parte da inteligência artificial, foi excluída deste trabalho.

No entanto como o objetivo deste trabalho é o uso da Plataforma *Raspberry PI*, tal dúvida foi pouco explorada por esse trabalho. Sendo assim escolhemos nosso modelo com acurácia de 89% para implantar no *Raspberry PI*. Para a implantação direta no *Raspberry* esse trabalho aborda duas soluções, o treinamento direto no *Raspberry PI* usando um *cluster Hadoop* e o modelo de transferência de conhecimento conhecido como método professor e estudante.

6.2 *Cluster* Embarcado usando o *Apache Hadoop*

Selecionamos a divisão III do *dataset* que contém 90 % das imagens para o conjunto de treinamento e 10 % do conjunto de teste para realizar a fase de treinamento em nosso cluster Raspberry PI Hadoop.

Sabe-se que a acurácia permanece a mesma, visto que a divisão do *dataset* e o código usado no pytorch é o mesmo. Com base nisso o *cluster* foi montado com quatro *Raspberry PI Model 2 B*

Com o melhor modelo identificado conforme explicado da Seção 6.1 e Tabela 5 foi possível treinar a mesma rede em nosso *cluster Raspberry PI Hadoop* e esperáva-se atingir a mesma acurácia com um tempo de treinamento mais lento. Mas isso não revela ser verdade, o *cluster Raspberry PI* construído provou não ser capaz de treinar a rede *DenseNet*.

O *cluster* usado para essas tentativas é formado por quatro *Raspberry PI Model 2 B* conectados uns aos outros em uma rede *gigabit*. Onde o nó principal atua como mestre e trabalhador no *Hadoop*, enquanto os outros nós atuam apenas como trabalhadores *cluster* e com apenas 4 Raspberry PI conta apenas com 4GB de memoria RAM.

É importante lembrar que apesar de construído em Java o *Apache Hadoop* também consegue executar códigos em Java. Dessa forma foi usado exatamente o mesmo código usado no COLAB.

E por causa dessa limitação de processamento, o foco passou a se tentar entender por que o *cluster* não pode executar a fase de treinamento da rede *Densenet*. Após executar novamente a fase de treinamento da rede no *Google Colab* para medir a quantidade de Memória RAM necessária para realizar este processamento de treinamento, foi detectado que é necessário entre 3,5 GB e 5 GB de RAM durante o treinamento para chegar ao modelo final.

Como o *cluster* tem 4 GB de RAM e em cada nó o *Raspbian* OS usa cerca de 80 MB de RAM no melhor cenário e para construir o *cluster Hadoop* em cada nó usando uma máquina virtual java usa cerca de 254 MB, perde-se 334 MB de RAM disponível em cada nó. Portanto, 1,33 GB de RAM sendo usado apenas para construir o *cluster*. Portanto, foi detectado que é inviável usar um *cluster* de aprendizado de máquina com apenas quatro nós de *Raspberry PI*, para a fase de treinamento da rede desenvolvida nesse trabalho.

6.3 Transferência de conhecimento

Como um de nossos objetivos é usar uma plataforma embarcada de baixo custo e, portanto, com menor capacidade computacional, precisamos ter algoritmos mais leves do ponto de vista computacional. Por esse motivo, precisamos de algoritmos leves. Para executar um algoritmo leve com boa precisão no *Raspberry PI*, precisamos traduzir o conhecimento obtido pela rede *Densenet* para um modelo MLP usando o método *teacher-student* (professor e aluno) que é baseado no propósito de destilar conhecimento por (HINTON; VINYALS; DEAN, 2015) que é maneira simples de melhorar o desempenho de quase qualquer algoritmo de aprendizado de máquina.

Este método usa o conhecimento prévio que faz previsões usando um conjunto completo de modelos que é complicado e pode ser muito caro computacionalmente. Em seguida, comprima o conhecimento em um conjunto em um único modelo que é muito mais fácil de implantar. Este processo torna um pequeno modelo computacionalmente mais barato que a rede anterior.

A técnica chamada *teacher-student* usa uma rede *DenseNet* totalmente treinada como professor e o conhecimento é passado para um algoritmo MLP usando o conhecimento e faz com que o algoritmo MLP atinja mais precisão em menos épocas. A técnica funciona da seguinte forma: a rede de professores altamente complexa é primeiro treinada separadamente usando o conjunto de dados completo. Esta etapa requer alto desempenho computacional. Em seguida, uma correspondência entre a rede de professores e alunos deve ser criada. Na terceira etapa, os dados são passados pela rede do professor para obter todos os resultados intermediários. Agora as saídas da rede do professor são usadas para fazer a relação de correspondência com a rede do aluno e retro-propagar o erro na rede.

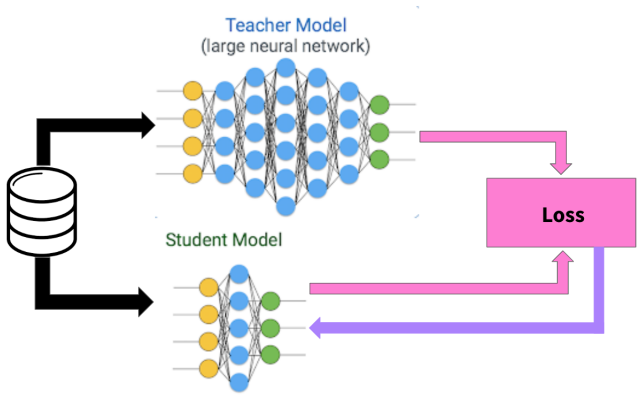

Figura 5 – Fluxograma da técnica de *teacher-student*

A Figura 5 apresenta o fluxograma da técnica usada, essa técnica consiste em quatro passos: 1 - Treinar a rede mais complexa. 2 - Estabelecer correspondência, isso é estabelecer a conexão imediata entre as saídas da rede professora e da rede aluna. 3 - propagar todo o dado através da rede mais complexas e obter todas as saídas intermediarias. 4 - fazer a *backpropagation* de toda rede para a rede estudante, para isso todos os inputs da rede professora e sua correspondência criada no passo 2 são propagadas na rede estudante e ela consegue aprender com o comportamento da rede professora. O Código 6.1 representa o comportamento da função professor.

Listing 6.1 – Código da Funcão *Teacher*

```
def teach (teacher, student, train_loader, database, num_epochs=3):
    learning_rate = 1.e-4
    # loss function and optimizer
    criterion = torch.nn.functional.mse_loss
    optimizer = torch.optim.Adam(student.parameters(), lr=learning_rate)
    for epoch in range(num_epochs):
        total_loss = 0
        for (images, labels) in train_loader:
```

Tabela 6 – Resultados para a MLP treinada com a *Densenet* como professora.

	Acurácia			
Rede	I	II	III	Média
DenseNet	83	82	89	84,6
MPL	74	73	80	75,6

```
            # Forward + Backward + Optimize
            if torch.cuda.is_available():
                outputs = student(images.cuda())
                targets = teacher(images.cuda())
            else:
                outputs = student(images)
                targets = teacher(images)
            # loss = criterion(outputs, labels)
            loss = criterion(outputs,targets)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            total_loss += loss.item() * labels.size(0)

    return student
```

Os modelos *Densenet* de cada uma das divisões do *dataset*(Ver Tabela 4) foram salvo e usados como professores para a MLP. Com o melhor modelo salvo, o processo de transferência para os modelos MLP foi realizado e executamos cada rede MLP no *Raspberry*. A Acurácia para cada rede MLP foi cálculada usando o mesmo conjunto para treinamento e alcançamos uma acurácia entre 74% e 80%. A Tabela 6 resume a acurácia de cada experimento em redes Densenet e MLP, novamente a média foi calculada após 12 execuções excluindo maiores e menores valores.

Como esperado, ao transferir o conhecimento, perde-se alguma acurácia, porém a MLP conseguiu atigir uma acurácia de 80% na divisão III do *dataset*.

6.4 Considerações sobre os resultados

Os objetivos iniciais deste trabalho definidos no Capítulo 1, que eram: Treinar uma rede na plataforma *Raspberry* usando apenas o *Pytorch* e/ou fazer um treinamento usando um *cluster* embarcado usando o *Apache Hadoop*. A Seção 6.2 mostra resultados que vão de encontro a esse objetivo pois não foi possível realizar o treinamento com um *cluster* de apenas quatro nós. No entanto esse resultado não é desanimador para a Computação de Alto Desempenho de Baixo

Custo, pois ainda há chances desses objetivos serem alcançados usando uma escalação horizontal, adicionando mais nós ou, então, com a escolha de outra plataforma que forneça maior poder computacional. Por exemplo: *Banana PI*, *Odroid*, entre outros.

No entanto para a segunda etapa dos experimentos onde a técnica de *teacher-student* foi utilizada, é possível considerar que a perda de apenas 9% de acurácia para a divisão III do dataset, aquela onde a *DenseNet* obteve o seu melhor resultado, como um resultado animador para a inteligência artificial usando plataformas de baixo custo como o *Raspberry*. O resultado de 80% pode ser considerado um bom resultado, mas antes de ser usado para aplicações médicas recomenda-se mais pesquisas na área e a verificação com a comunidade médica para descobrir qual taxa de erro minima para que o software seja aprovado pelos conselhos médicos.

7

Conclusão

Dado que o câncer de pele é um dos mais perigosos e mais comuns em todo o mundo, especialmente no Brasil com 180 mil novos casos por ano segundo a sociedade brasileira de câncer de pele. A ciência da computação busca através da inteligência artificial colaborar com a medicina elaborando estudos de diagnóstico baseados em imagens de lesões de pele. De maneira geral a forma mais comum de identificação do câncer de pele é para união de características avaliadas pela regra do ABCDE. Esse método de diagnóstico é bem similar a extração de características de imagens usadas pela inteligência artificial, por conta disso todos os anos o *International Skin Imaging Collaboration (ISIC)* lança competições onde o principal objetivo é detectar câncer ou diferentes formas lesões de pele. A partir do estudo da literatura do ISIC identificou-se que não há uma tentativa de identificar câncer de pele usando a Computação de Alto Desempenho de Baixo Custo.

Este trabalho investigou a viabilidade de construção de um *Cluster* de baixo custo de Alto Desempenho para a aplicação de técnicas de inteligência artificial com o objetivo de tornar a detecção de câncer de pele menos custosa a partir da aliança entre a Computação de Baixo custo e a Inteligência artificial, treinando redes pretreinadas em na plataforma *Raspberry PI*. Após definidos os objetivos foram feitas: Uma revisão sistemática sobre Computação de Alto Desempenho de Baixo Custo e a revisão literária sobre a literatura do ISIC.

Dentre os trabalhos selecionados na revisão os que trouxeram maior interesse para a investigação realizada foram os trabalhos de Qureshi et al. (2016), Ramos, Ralha e Teodoro (2016) e Lima, Moreno e Dias (2016).Os trabalhos de Qureshi et al. (2016) e Ramos, Ralha e Teodoro (2016), utilizam um *cluster* formado por *Raspberry PI* para realizar processamento em larga escala de imagens usando, respectivamente, o *framework Apache Hadoop* e a ferramenta *OpenMPI*. Enquanto o trabalho de Lima, Moreno e Dias (2016) realiza cálculos matemáticos usando, também, *OpenMPI*. Dentre os cálculos realizados estão a multiplicação de matrizes.

Na Revisão Sistemática realizada não foi encontrado nenhum trabalho que usasse o

framework Apache Hadoop para realizar o treinamento de redes profundas usando inteligência artificial ou mesmo quaisquer outra forma de uso de inteligência artificial usando *Clusters* com a plataforma *Apache Hadoop*. Desta forma identificou-se que o presente trabalho se dispunha a fazer trazia consigo um certo pioneirismo.

A partir da revisão da literatura montada pelo ISIC foi possível entender o estado da arte sobre o estudo de câncer de pele com o uso da inteligência artificial, foram encontrados diversos artigos e *datasets*, dentre os quais os trabalhos de (MAJTNER; YILDIRIM-YAYILGAN; HARDEBERG, 2017), (Romero Lopez et al., 2017), (SOUSA; MORAES, 2017), (KAWAHARA; HAMARNEH, 2019),(SAHU; YU; QIN, 2018),(GOMEZ; HERRERA, 2017) e (CHATTERJEE; DEY; MUNSHI, 2018) foram selecionados como os mais importantes e de maior ajuda na construção deste trabalho. O trabalho de (SAHU; YU; QIN, 2018) é considerado primordial pois nele há uso da plataforma *Raspberry*, no entanto não há tentativa de treinamento usando essa plataforma, apenas a aplicação da técnica de transferência de conhecimento. Após a leitura do estado da arte do estudo de câncer de pele com inteligência artificial foi possível definir os experimentos deste trabalho.

Nesta dissertação, os experimentos estiveram divididos em dois grandes grupos, no primeiro deles foi realizado a tentativa de treinar e executar redes neurais profundas usando a plataforma *Rasberry PI* em um Cluster Embarcado usando o *Apache Hadoop*. No segundo experimento foi realizada uma transferência de conhecimento usando a técnica de *teacher-student*.

A tentativa de executar o treinamento usando o cluster Apache Hadoop mostrou-se infrutífera por limitações técnicas apresentadas em um cluster com apenas quatro nós. Em contrapartida, a acurácia de 80% atingida com a técnica teacher-student é considerada promissora, pois essa taxa está no estado da arte para uma rede MLP.

O resultado negativo em treinar uma rede profunda usando o *cluster* de baixo custo não representa o fim da busca de uma inteligência artificial de alta acurácia em baixo custo pois é possível contornar tal limitação com outras plataformas ou com a adição de mais nós ao *cluster*.

Portanto, a partir deste trabalho é possível propor trabalhos futuros divididos em duas frentes: Na primeira delas, dizendo a respeito da inteligência artificial, pode-se realizar um trabalho usando oito, dezesseis ou trinta e dois nós de *Raspberry* para construir um cluster de maior desempenho computacional, bem como executar testes em outras plataformas como o Banana PI ou o Odroid. Para a parte de inteligência artificial é possível propor estudos com diferentes redes pretreinadas, com diferentes pesos e configurações de *datasets*, também é possível realizar trabalhos isolando a acurácia para cada uma das lesões existentes no *dataset HAM10000*.

Referências

American Cancer Society. American Cancer Society: Cancer Facts & Figures 2014. *Cancer Facts and Figures*, 2014. ISSN 1544-2217. Citado 2 vezes nas páginas 15 e 28.

Apache Software Foundation. *Apache Hadoop*. 2017. Disponível em: <https://hadoop.apache.org>. Citado na página 21.

BAI, X. et al. Acceleration of RSA processes based on hybrid ARM-FPGA cluster. In: *Proceedings - IEEE Symposium on Computers and Communications*. [S.l.: s.n.], 2017. p. 682–688. ISBN 9781538616291. ISSN 15301346. Citado 2 vezes nas páginas 41 e 46.

BAPPALIGE, S. P. *An introduction to Apache Hadoop for big data*. 2014. Disponível em: <https://opensource.com/life/14/8/intro-apache-hadoop-big-data>. Citado 2 vezes nas páginas 21 e 22.

BASFORD, P. J. et al. Performance analysis of single board computer clusters. *Future Generation Computer Systems*, 2020. ISSN 0167739X. Citado na página 43.

BINDER, M. et al. Application of an artificial neural network in epiluminescence microscopy pattern analysis of pigmented skin lesions: a pilot study. *British Journal of Dermatology*, 1994. ISSN 13652133. Citado na página 15.

CHATTERJEE, S.; DEY, D.; MUNSHI, S. Optimal selection of features using wavelet fractal descriptors and automatic correlation bias reduction for classifying skin lesions. *Biomedical Signal Processing and Control*, 2018. ISSN 17468108. Citado 2 vezes nas páginas 57 e 77.

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, ACM, New York, NY, USA, v. 51, n. 1, p. 107–113, jan. 2008. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1327452.1327492>. Citado na página 21.

GOMEZ, C.; HERRERA, D. Recognition of skin melanoma through dermoscopic image analysis. In: . [S.l.: s.n.], 2017. ISBN 9781510616332. ISSN 1996756X. Citado 3 vezes nas páginas 56, 57 e 77.

HAYKIN, S. S. *Neural networks and learning machines*. Third. Upper Saddle River, NJ: Pearson Education, 2009. Citado na página 25.

HE, K. et al. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>. Citado na página 62.

HINTON, G.; VINYALS, O.; DEAN, J. Distilling the Knowledge in a Neural Network. *arXiv e-prints*, p. 1–9, 2015. Disponível em: <http://arxiv.org/abs/1503.02531>. Citado na página 72.

HUANG, G.; LIU, Z.; WEINBERGER, K. Q. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. Disponível em: <http://arxiv.org/abs/1608.06993>. Citado na página 61.

ILLIG, L. *Epidemiologic aspects of malignant melanoma. (Review)*. 1987. Citado 2 vezes nas páginas 15 e 29.

KAWAHARA, J.; HAMARNEH, G. Fully Convolutional Neural Networks to Detect Clinical Dermoscopic Features. *IEEE Journal of Biomedical and Health Informatics*, 2019. ISSN 21682194. Citado 2 vezes nas páginas 55 e 77.

KECSKEMETI, G.; HAJJI, W.; TSO, F. P. Modelling low power compute clusters for cloud simulation. In: *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. [S.l.: s.n.], 2017. p. 39–45. Citado 2 vezes nas páginas 34 e 45.

KITCHENHAM, B. Procedures for performing systematic reviews. v. 33, 08 2004. Citado 2 vezes nas páginas 17 e 30.

LIMA, F. A.; MORENO, E.; DIAS, W. R. A. Performance analysis of a low cost cluster with parallel applications and arm processors. *IEEE Latin America Transactions*, v. 14, n. 11, p. 4591–4596, Nov 2016. ISSN 1548-0992. Citado 7 vezes nas páginas 15, 23, 24, 39, 46, 47 e 76.

LIN, Z.; ZED, Z. ZCluster : A Zynq-based Hadoop Cluster. p. 450–453, 2013. Citado 3 vezes nas páginas 40, 41 e 46.

MADURANGA, M. W. P.; RAGEL, R. G. Comparison of load balancing methods for raspberry-pi clustered embedded web servers. In: *2016 International Computer Science and Engineering Conference (ICSEC)*. [S.l.: s.n.], 2016. p. 1–4. Citado 2 vezes nas páginas 34 e 46.

MAJTNER, T.; YILDIRIM-YAYILGAN, S.; HARDEBERG, J. Y. Combining deep learning and hand-crafted features for skin lesion classification. In: *2016 6th International Conference on Image Processing Theory, Tools and Applications, IPTA 2016*. [S.l.: s.n.], 2017. ISBN 9781467389105. Citado 3 vezes nas páginas 54, 58 e 77.

MAPPUJI, A. et al. Study of raspberry pi 2 quad-core cortex-a7 cpu cluster as a mini supercomputer. In: *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*. [S.l.: s.n.], 2016. p. 1–4. Citado 2 vezes nas páginas 35 e 46.

MARSLAND, S. *Machine Learning: An Algorithmic Perspective*. 1st. ed. [S.l.]: Chapman Hall/CRC, 2009. ISBN 1420067184. Citado 3 vezes nas páginas 24, 25 e 26.

MATTHEWS, S. J. et al. Portable parallel computing with the raspberry Pi. In: *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. [S.l.: s.n.], 2018. ISBN 9781450351034. Citado na página 41.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 127–147, 1943. Citado na página 24.

MINSKY, M.; PAPERT, S. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969. Citado na página 25.

MOORE, J. *Performance Benchmarking a Raspberry PI Cluster*. 2014. Disponível em: <https://opensky.ucar.edu/islandora/object/siparcs3A132/datastream/PDF/download/citation.pdf>. Citado na página 35.

PACHECO, P. *An Introduction to Parallel Programming*. 1st. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 9780123742605. Citado 2 vezes nas páginas 19 e 21.

PAHL, C. et al. A container-based edge cloud paas architecture based on raspberry pi clusters. In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. [S.l.: s.n.], 2016. p. 117–124. Citado 2 vezes nas páginas 33 e 45.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de imagens digitais: princípios, algoritmos e aplicações*. THOMSON PIONEIRA, 2008. ISBN 9788522105953. Disponível em: <https://books.google.com.br/books?id=13KAPgAACAAJ>. Citado na página 36.

PFISTER, G. F. *In Search of Clusters (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998. ISBN 0-13-899709-8. Citado na página 14.

QURESHI, B. et al. Performance of a low cost hadoop cluster for image analysis in cloud robotics environment. *Procedia Computer Science*, v. 82, n. Supplement C, p. 90 – 98, 2016. ISSN 1877-0509. 4th Symposium on Data Mining Applications, SDMA2016, 30 March 2016, Riyadh, Saudi Arabia. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050916300278>. Citado 5 vezes nas páginas 15, 36, 46, 47 e 76.

RAMOS, R.; RALHA, C. G.; TEODORO, G. Avaliação de cluster raspberry pi para execução de aplicações de análise de imagens microscópicas médicas. In: *43º Semish - Seminário Integrado de Software e Hardware*. [S.l.: s.n.], 2016. p. 1795–1806. Citado 5 vezes nas páginas 15, 37, 46, 47 e 76.

RAUBER, T.; RNGER, G. *Parallel Programming: For Multicore and Cluster Systems*. 2nd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2013. ISBN 3642378005, 9783642378003. Citado na página 20.

Romero Lopez, A. et al. Skin lesion classification from dermoscopic images using deep learning techniques. In: *Proceedings of the 13th IASTED International Conference on Biomedical Engineering, BioMed 2017*. [S.l.: s.n.], 2017. ISBN 9780889869905. Citado 2 vezes nas páginas 54 e 77.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, p. 65–386, 1958. Citado na página 25.

ROSENBLATT, F. *Perceptrons and the Theory of Brain Mechanisms*. [S.l.]: Spartan books, 1962. Citado 2 vezes nas páginas 25 e 26.

RUMELHART, D. E.; MCCLELLAND, J. L.; GROUP, C. P. R. (Ed.). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986. ISBN 026268053X. Citado 2 vezes nas páginas 25 e 26.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3rd. ed. USA: Prentice Hall Press, 2009. ISBN 0136042597. Citado 4 vezes nas páginas 24, 25, 26 e 27.

SAHU, P.; YU, D.; QIN, H. Apply lightweight deep learning on internet of things for low-cost and easy-to- access skin cancer detection Apply Lightweight Deep Learning on Internet of Things for Low-Cost and Easy-To-Access Skin Cancer Detection. n. March, 2018. Citado 3 vezes nas páginas 56, 58 e 77.

SBD. *Câncer da pele - Sociedade Brasileira de Dermatologia*. 2017. Disponível em: <https://www.sbd.org.br/dermatologia/pele/doencas-e-problemas/cancer-da-pele/64/>. Citado 2 vezes nas páginas 15 e 28.

SEAL, D. *ARM Architecture Reference Manual*. 2nd. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN 0201737191. Citado na página 24.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: . [S.l.: s.n.], 2015. Citado na página 62.

SIPSER, M. *Introduction to the Theory of Computation*. 1st. ed. [S.l.]: International Thomson Publishing, 1996. ISBN 053494728X. Citado na página 14.

SOUSA, R.; MORAES, L. Araguaia medical vision lab at isic 2017 skin lesion classification challenge. 03 2017. Citado 2 vezes nas páginas 55 e 77.

SZEGEDY, C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: . [S.l.: s.n.], 2017. Citado na página 63.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: . [S.l.: s.n.], 2016. v. 2016-December. ISSN 10636919. Citado na página 63.

Szegedy, C. et al. Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [s.n.], 2015. p. 1–9. ISSN 1063-6919. Disponível em: <https://ieeexplore.ieee.org/document/7298594>. Citado na página 63.

TANENBAUM, A. S. *Structured Computer Organization (5th Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2005. ISBN 0131485210. Citado na página 20.

TSO, F. P. et al. The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures. In: *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*. [S.l.: s.n.], 2013. p. 108–112. ISSN 1545-0678. Citado 4 vezes nas páginas 32, 33, 34 e 45.

ZHANG, Q.; ZHOU, D.; ZENG, X. Highly wearable cuff-less blood pressure and heart rate monitoring with single-arm electrocardiogram and photoplethysmogram signals. *BioMedical Engineering Online*, 2017. ISSN 1475925X. Citado na página 43.

ZHANG, Q.; ZHOU, D.; ZENG, X. A novel single-arm-worn 24h heart disease monitor empowered by machine intelligence. *Biomedical Signal Processing and Control*, 2018. ISSN 17468108. Citado 3 vezes nas páginas 42, 46 e 47.