UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

FELIPE DE AMORIM BORBA

Redes neurais profundas e *ensemble* de classificadores: uma aplicação em imagens médicas

Campinas

2021

Felipe de Amorim Borba

Redes neurais profundas e *ensemble* de classificadores: uma aplicação em imagens médicas

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Matemática Aplicada e Computacional.

Orientador: João Batista Florindo

Este trabalho corresponde à versão final da Dissertação defendida pelo aluno Felipe de Amorim Borba e orientada pelo Prof. Dr. João Batista Florindo.

Campinas

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

B644r

Borba, Felipe de Amorim, 1977-
Redes neurais profundas e *ensemble* de classificadores : uma aplicação em imagens médicas / Felipe de Amorim Borba. – Campinas, SP : [s.n.], 2021.

Orientador: João Batista Florindo.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Redes neurais convolucionais. 2. Análise de imagem. 3. Reconhecimento de texturas. 4. Imagens médicas. I. Florindo, João Batista, 1984-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Deep neural networks and classifier *ensemble* : an application in medical imaging
Palavras-chave em inglês:
Convolutional neural networks
Image analysis
Texture recognition
Medical images
Área de concentração: Matemática Aplicada e Computacional
Titulação: Mestre em Matemática Aplicada e Computacional
Banca examinadora:
João Batista Florindo [Orientador]
Núbia Rosa da Silva
Antônio Maria Pereira de Resende
Data de defesa: 31-05-2021
Programa de Pós-Graduação: Matemática Aplicada e Computacional

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-1566-6601
- Currículo Lattes do autor: http://lattes.cnpq.br/0354334504202805

Dissertação de Mestrado Profissional defendida em 31 de maio de 2021 e aprovada pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). JOÃO BATISTA FLORINDO

Prof(a). Dr(a). NÚBIA ROSA DA SILVA

Prof(a). Dr(a). ANTÔNIO MARIA PEREIRA DE RESENDE

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Agradecimentos

Ao Deus Eterno criador de tudo o que há no universo, ao único digno de receber toda honra, glória, louvor e adoração, ao Deus Pai, Filho e Espírito Santo, a Ele dedico minha eterna gratidão por tudo o que tenho e sou, em especial, por ter me dado a graça de concluir o que, para mim, era impossível. Por tanto, devolvo para Ele toda honra que porventura seja dirigida a mim, pois nunca desejarei ocupar o lugar que é somente dEle.

Aos humanos que estão próximos de mim, dedico meu primeiro agradecimento àquela que pacientemente esperou, noite após noite, dia após dia, seu esposo terminar de estudar para provas ou terminar de concluir um experimento desse trabalho, ou ainda de escrever mais uma página dessa dissertação. Andréa, sem seu apoio, sem seu encorajamento, sem seu amor, eu jamais teria chegado até aqui, muito obrigado. Te amo e te amarei por toda vida.

Outros humanos próximos de mim também merecem grande destaque: meus filhos e meus pais. Meus pais (Tavares e Nilda) por terem forjado em mim o caráter necessário para ter estabilidade social e emocional diante de tão grandes desafios, além do constante apoio e encorajamento que me motivam a cada dia; e os meus filhos (Igor, Larissa e Levi), por tornarem meus dias mais valorosos através de seus atos de carinho, mostrando-me diariamente como Deus foi bondoso comigo ao me proporcionar uma família tão linda e tão especial. Amo muito todos vocês.

Ao meu orientar, Professor Dr. João Batista Florindo, agradeço pela orientação, disponibilidade, e por tão grande paciência. Reconheço que nem sempre fui um orientando fácil e precisei de ajustes tanto na área técnica quanto na emocional para concluir esse trabalho.

Agradeço aos professores que ministraram as disciplinas do curso, com certeza esse foi o maior desafio da minha vida acadêmica, os Senhores e as Senhoras estarão para sempre em minha história, agradeço o compartilhar de saberes e a disponibilidade para não somente ministrar as aulas, mas também em tirar dúvidas seja presencialmente ou em formato digital.

Aos amigos que fiz durante o curso, em especial ao Roger, João, Lívia e Josué agradeço pela amizade e pelo companheirismo.

Agradeço aos meus colegas de trabalho por me ajudarem nas dissiplinas durante o curso, não somente transmitindo conhecimento, mas também me dando apoio moral para que eu nunca desistisse: Dr. Bruno Giordano, Dr. Luis Fernando Gouveia Morais, Dr^{a} Ivana Yoshie Sumida e Ms André Yoshimi Kusumoto.

Agradeço também às minhas chefes: Mara Oliveira e Liliana Penha, por me apoiarem e por serem tão solicitas aos meus pedidos de ausência, quando o curso assim exigiu.

Agradeço também aos meus amigos, em especial: João Iraldo e Aída sua esposa, Marcos Jorge e sua esposa Valquíria, Marcos Manfredine e sua esposa Renata, Adolfo Mota e sua esposa Simone, Antônio e sua esposa Ana Rubélia, por sempre trazerem palavras de apoio e incentivo, e sempre demonstrar amor e carinho à minha família, o que contribuiu emocionalmente para que essa trajetória fosse menos turbulenta. Como dizem as Escrituras Sagradas: “existe amigo mais apegado que um irmão”. (Provérbios 18.24)

Resumo

O uso das redes neurais profundas e do aprendizado de máquina para a classificação de imagens é tema bastante recorrente em trabalhos acadêmicos que usam as técnicas da inteligência artificial. Esta dissertação trata de uma aplicação para esse tema, focando na classificação de imagens de texturas e em como *ensembles* de classificadores constituem ferramentas úteis na melhoria do desempenho das modernas arquiteturas de redes neurais neste tipo de aplicação. O trabalho também apresenta uma aplicação em imagens médicas, em especial, imagens de três tipos de cistos mandibulares bastante agressivos. A rede neural profunda que utilizamos é do tipo convolucional e passa por um processo de transferência de aprendizado, possuindo uma arquitetura já consolidada no meio acadêmico, que é a ResNet-18. Além de seu uso como um classificador convencional, uma rede neural convolucional (abreviada aqui por CNN, de sua sigla em inglês) pode produzir vetores de valores reais baseados nas imagens processadas. Esses vetores, chamados vetores de descritores, são aqui armazenados para serem utilizados como fonte de dados em um *ensemble* de classificadores, que são algoritmos de aprendizado de máquina especialmente aplicados em nosso trabalho para a classificação de imagens de texturas. A fim de parametrizar os experimentos realizados e confrontar com trabalhos acadêmicos já consolidados no meio científico, utilizamos imagens de texturas de bases de *benchmark*. Essas imagens fazem parte de bancos de imagens já amplamente difundidos em outros trabalhos na literatura. Por fim, mostramos os resultados em termos da acurácia da classificação tanto nas imagens médicas quanto nas não médicas, evidenciando as várias diferenças ocorridas durante os experimentos que realizamos, suas possíveis aplicações no mundo real e como os métodos que utilizamos podem contribuir para o avanço do diagnóstico médico por meio de imagens e da classificação de imagens de texturas em um contexto mais geral.

Palavras-chave: Redes Neurais Convolucionais, *Ensembles* de Classificadores, Análise de Imagens, Reconhecimento de Texturas, Imagens Médicas.

Abstract

The use of deep neural networks and of machine learning for image classification is a quite recurring theme in academic works that use artificial intelligence techniques. This dissertation is about an application on this topic, focusing on the classification of texture images and on how classifier ensembles constitute useful tools for the improvement of performance in modern architectures of neural networks for this kind of application. The work also presents an application to medical images, particularly, images from three groups of aggressive jaw cysts. The deep neural network used here is convolutional and undergoes a process of transfer learning. We employ an architecture already consolidated in the literature, which is the ResNet18. In addition to its use as a conventional classifier, a convolutional neural network (CNN) can produce vectors of real values based on the processed images. These vectors, called feature vectors, are used here as the input to classifier ensembles, which are the machine learning algorithms applied in our work for texture classification. In order to parameterize the experiments accomplished and compare them with other state-of-the-art works in the literature, we employed benchmark texture images. These images are part of databases frequently used in other studies of texture recognition in the literature. Finally, we present the results, in terms of classification accuracy, in the medical images as well as in the non-medical ones, evidencing the many differences occurred during the experiments that were carried out, their possible applications in the real world and how the methods that we investigated can contribute for the advance of the computer-aided medical diagnosis and classification of texture images in a general context.

Keywords: Convolutional Neural Networks, Classifier Ensembles, Image Analysis, Texture Recognition, Medical Images.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AB	AdaBoost
AD	Árvores de decisão
CHA	Com hiperparâmetros ajustados
CNN	Convolutional Neural Network
EmpSel	Empilhamento seletivo
EmpTot	Empilhamento total
FA	Floresta aleatória
IA	Inteligência artificial
KNN	K-vizinhos mais próximos
MisSel	Mistura seletiva
MisTot	Mistura total
MLP	Multilayer Perceptron
NB	Naive Bayes
RNA	Rede Neural Artificial
RLin	Regressão linear
RLog	Regressão logística
SHA	Sem hiperparâmetros ajustados
SVC	Support vector classification
Vot	Votação

Sumário

Introdução

Descrição e Motivação

Neste trabalho será estudada a identificação de três tipos de cistos bucais mandibulares por aprendizado de máquina: os queratocistos odontogênicos (OKC) do tipo K e do tipo S[1]; e os cistos radiculares. As texturas visuais analisadas são obtidas a partir de um base de dados de imagens histológicas do revestimento epitelial.

Em (KRAMER, 1974) observamos a seguinte definição para cistos: "uma cavidade patológica com conteúdo fluido, semifluido ou gasoso e que não é criada pelo acúmulo de pus". Restringindo nosso foco para os cistos bucais, a estomatologia[2] destaca três grandes grupos para essa patologia (SHEAR; SPEIGHT, 2007):

- Cistos mandibulares;
- Cistos associados ao antro maxilar; e
- Cistos dos tecidos moles da boca, rosto, pescoço e glândulas salivares.

De acordo com (SHEAR; SPEIGHT, 2007, p.2), os cistos radiculares são os de ocorrência mais comum, abrangendo aproximadamente 52% dos casos. Os queratocistos ondogênicos ocorrem em aproximadamente 12% dos cistos mandibulares.

Segundo Florindo *et al.*, existem algumas diferenças histológicas entre OKCs sindrômicos e esporádicos catalogados em literatura. Porém, o uso desse material histopatológico em arbordagens automatizadas ou semiautomáticas não é trivial, pois o mesmo exige descrições contextuais de células e tecidos que não são diretamente traduzidos em recursos quantitativos que os algoritmos de aprendizado de máquina podem atualmente manipular (FLORINDO et al., 2013).

Classificar texturas visuais dos cistos mandibulares é um constante desafio da estomatologia, devido às suas diferenças nos padrões visuais complexos como brilho, densidade, rugosidade, granulação, dentre outras características das texturas, conforme descreve (MATERKA; STRZELECKI, 1998). E assim, os métodos de análise de imagens que visam resolver esse tipo de problema acabam recorrendo a soluções de uma outra grande área da ciência: a Inteligência Articial (IA), e mais espeficamente ainda aos métodos de aprendizado de máquinas.

[1] Tipo K: OKC esporádico; tipo S: OKC sidrômico.

[2] *Estomatologia*: Especialidade médica que estuda e trata das doenças da boca e do sistema dentário (7GRAUS, 2020).

Os trabalhos pioneiros de Charles Babbage e Alan Turing[3] são uma demonstração de como o homem vem há muitas décadas tentando de forma artificial imitar a inteligênica humana, gerando alguns dos embriões da inteligência artificial, afim de resolver problemas como o que expomos (reconhecer texturas em imagens médicas) de forma automática com menos erros e subjetivismo.

Após décadas de estudos, a IA apresentou um avanço promissor com a criação do neurônio artificial de McCulloch & Pitts (MCCULLOCH; PITTS, 1943), que culminou nas redes de neurônios artificiais contendo várias camadas de neurônios e que por isso apresentam a característica de aprendizado profundo, conforme trata Yann LeCun, *et al.* em (LECUN; BENGIO; HINTON, 2015, p.437).

Utilizar redes neurais profundas para classificar imagens não é algo novo. Um bom exemplo disso é o aclamado trabalho de LeCun *et al.* no fim da década de 1980 (LECUN et al., 1989).

De igual forma, utilizar *ensemble* de classificadores (nas mais diversas aplicações do aprendizado de máquina) não é algo inovador. Como exemplo, destacamos os trabalhos (THEPADE; CHAUDHARI, 2020), (MEDINA-QUERO et al., 2018) e (ABDULHAY et al., 2020) que utilizam *ensemble* de classificadores em aplicações diferentes.

Existem várias arquiteturas de redes neurais com aprendizado profundo. Dentre essas arquiteturas, as redes convolucionais[4] apresentam excelentes resultados na identificação de imagens de diversos tipos, por isso, é motivador tentar utilizar esse tipo de rede neural profunda para classificar texturas visuais.

As redes neurais convolucionais (CNN), realizam transformações matemáticas nas imagens que atravessam sua rede de neurônios e, com isso, produzem vetores de atributos.

Vale ressaltar contudo que a IA não utiliza apenas redes neurais para classificar imagens. Existem também outros algoritmos de aprendizado de máquinas, que aqui, por estarem voltados para a tarefa de classificação, serão chamados genericamente de *classificadores*.

Usar os vetores de atributos gerados por uma CNN como fonte de dados de treino e teste para outros classificadores poderia melhorar o desempenho destes mesmos classificadores? Seria possível utilizar os classificadores dessa forma para classificar texturas visuais médicas? E usar vários classificadores em conjunto (*ensemble*) para fazer essas classificações seria uma boa solução para o problema do diagnóstico das imagens médicas dos cistos mandibulares? Tentar responder a todas essas perguntas também motiva nosso

[3] Falaremos com mais detalhes sobre esses pesquisadores no Capítulo 1.

[4] O aprendizado profundo também será denominado neste trabalho como *deep learning*, e terá suas características mais exploradas no Capítulo 3. Naquele Capítulo também traremos uma breve abordagem das redes neurais convolucionais.

trabalho.

Portanto, no intuito de aumentar a possibilidade de um diagnóstico mais preciso e mais rápido, este trabalho porpõe o uso da Inteligência Artificial (IA), em especial, das redes neurais convolucionais e de um conjunto de algoritmos de aprendizado de máquina especialmente adaptados para classificar as texturas visuais dos cistos mandibulares que descrevemos.

Objetivos Gerais

Objetivo Teórico

Este trabalho tem como objetivo teórico mostrar o funcionamento de um *ensemble* de classificadores com descritores de imagens de texturas fornecidos por uma CNN.

Objetivo Aplicado

O objetivo aplicado deste trabalho é investigar o desempenho de uma CNN e de um *ensemble* de classificadores usando vetores de descritores da CNN na classificação das imagens médicas.

Objetivos Específicos

Neste trabalho vamos combinar as técnicas, aproveitando suas características mais interessantes para melhorar a identificação de imagens médicas, mais especificamente as texturas visuais de cistos mandibulares, pois há poucos trabalhos que tratam da classificação automática de queratocistos, conforme destaca (FLORINDO et al., 2013). Os trabalhos (FRYDENLUND; ERAMIAN; DALEY, 2014), (LANDINI, 2006) e (FLORINDO et al., 2013) são alguns dos poucos exemplos que podemos dar do uso de métodos computacionais para a classificação de queratocistos.

Para essa combinação de técnicas, a CNN irá fornecer aos classificadores os seus vetores de descritores[5], alimentando-os com os dados a serem utilizados em seu treinamento para que assim sejam capazes de classificar texturas visuais. As classificações geradas após esse processo de treino serão empilhadas e misturadas[6] com os vetores de descritores produzidos pela CNN. Esse empilhamento e mistura vão compor novos dados para treinar

[5] Os vetores de descritores serão melhor explicados no Capítulo 4, mas em síntese, são vetores que contêm o resultado das transformações matemáticas que a CNN faz em cada imagem processada por ela, antes da classificação final. Eles também são chamados de vetores de atributos, ou de *feature maps*.

[6] O empilhamento e a mistura dos descritores das imagens serão explicados no Capítulo 4.

e testar novamente os classificadores e assim será possível reavaliar seu desempenho em relação à acurácia da classificação de imagens.

Todo nosso trabalho está construído em linguagem Python e o código foi devidamente testado visando dar confiabilidade a seus resulados quando efetivamente utilizado para classificar as imagens médicas.

Para realizar esses testes, utilizamos quatro bases de dados (KTH-TIPS-2b, UMD, FMD, UIUC[7]) que já serviram como *baseline* de comparação em diversos outros trabalhos acadêmicos na área. Com isso, nossos resultados puderam ser comparados com esses trabalhos já publicados. Uma vez testado o código, podemos realizar com o conjunto de texturas visuais de cistos mandibulares (cuja base de dados é aqui chamada de CYST), a execução das técnicas que nos levarão a propor novas formas de automatizar o processo de análise desse tipo de imagem.

Para facilitar a escrita do trabalho, chamamos os métodos que utilizamos[8] pelas siglas que a seguir descrevemos:

- **CNN** (rede neural convolucional): as imagens de textura inicialmente alimentam a CNN que, além de gerar suas próprias classificações, também produz os vetores de atributos que irão fornecer os dados de treino e de teste para os classificadores;
- **SHA** (classificadores sem hiperparâmetros ajustados): os classificadores no modo *default* são treinados e testados utilizando os vetores de atributos;
- **CHA** (classificadores com hiperparâmetros ajustados): os classificadores, após sofrerem ajustes em alguns de seus hiperparâmetros (esses ajustes são feitos de forma automática), são novamente treinados e testados utilizando os vetores de atributos;
- **EmpTot** (empilhamento total): as classificações geradas pelos classificadores com seus parâmetros ajustados são empilhadas, ou seja, farão parte de uma única base de dados a ser utilizado para novamente treinar e testar os classificadores;
- **EmpSel** (empilhamento seletivo): as classificações geradas pelos classificadores com parâmetros ajustados e com melhores desempenho em acurácia[9] serão empilhados e farão parte de uma única base de dados para serem utilizados em um novo treino e teste dos classificadores;

[7] Alguns detalhes desses bancos de imagens são descritos no Capítulo 5, bem como a forma de utilização de cada um deles.

[8] Todos os métodos estão agrupados em um único programa, ou seja, fazem parte do mesmo código em Python. Com isso, o único ajuste que se faz antes de iniciar a execução do programa é com relação à base de dados que será utilizada.

[9] Como serão utilizadas apenas as classificações dos classificadores que obtiveram os melhores desempenhos em acurácia no método EmpTot, justificamos o termo *empilhamento seletivo*.

- **MisTot** (mistura total): as classificações geradas pelos classificadores com seus parâmetros ajustados são misturadas aos vetores de atributos, ou seja, farão parte de uma única base de dados;
- **MisSel** (mistura seletiva): as classificações geradas pelos classificadores com seus parâmetros ajustados e com melhores desempenhos em acurácia são misturadas aos vetores de atributos, ou seja, farão parte de uma única base de dados; e
- **Vot** (votação): os classificadores com seus parâmetros ajustados e utilizando os vetores de atributos passam por um processo de votação para verificar qual classificador obtem o melhor desempenho em acurácia, sendo este então o vencedor da votação.

Rodamos o código com a base CYST utilizando imagens dos cistos dos tipos K, R e S; e também rodamos o código apenas para os tipos K e S, uma vez que são ambos queratocistos odontogênicos, trazendo uma dificuldade a mais para a classificação a ser realizada.

Resultados

Quando testamos o código que produzimos sobre as bases de imagens KTH-TIPS-2b, UMD, FMD e UIUC, obtivemos resultados com altas médias de acurácia. Quando esses resultados são comparados com outros trabalhos[10] que já utilizaram essas bases de dados. A Tabela 1 mostra, de forma resumida, os maiores resultados que obtivemos frente a outros trabalhos com essas bases de dados.

Nessa tabela, **menor** significa a menor média entre os trabalhos analisados, **maior** significa a maior média de acurácia desses trabalhos, e **NR** significa *nossos resultados*, fazendo referência à média que obtivemos ao testar o código nessas bases de imagens.

Tabela 1 – Comparação da média de acurácias (%) em diferentes trabalhos que usaram as bases de dados: KTH-TIPS-2b (HAYMAN et al., 2004), FMD (SHARAN; ROSENHOLTZ; ADELSON, 2009), UIUC (LAZEBNIK; SCHMID; PONCE, 2005), UMD (XU; JI; FERMüLLER, 2009).

Base de dados	Menor	Maior	NR
KTH-TIPS-2b	46.3	77.5	74.5
FMD	22.1	78.8	82.9
UIUC	48.2	99.4	99.3
UMD	85.7	99.9	99.7

[10] A Tabela 15 traz as referências e os resultados médios dos trabalhos a que estamos nos referindo.

Ao analisarmos a Tabela 1 chegamos às nossas primeiras conclusões: o código e, consequentemente, os métodos que utilizamos são confiáveis o suficiente para serem utilizados na base CYST. Além disso, o código apresenta um excelente desempenho com as bases de imagens KTH-TIPS-2b, UMD, FMD e UIUC, e esses resultados podem servir de referência para outros trabalhos.

A CNN foi capaz de classificar a base de dados CYST com acurácia média de 99.4%, sendo esse o método mais bem sucedido nessa utilização da base CYST. Quando a base foi utilizada apenas com as classes K e S[11], a CNN superou os demais métodos que utilizamos, obtendo média de acurácia de 99.3%.

Frente aos resultados obtidos por (FLORINDO et al., 2013), que obtiveram 72% e 68% de acurácia média, respectivamente, com as bases CYST e CYST_KS, concluímos que nossos resultados obtiveram excelente desempenho.

Diante desses resultados, podemos concluir que nosso trabalho atingiu o objetivo de contribuir para o avanço dos estudos no diagnóstico por imagem de texturas dos cistos mandibulares de forma automática, além de ter investigado e validado um conjunto de algoritmos com ampla aplicabilidade a problemas não só da área médica, mas de outras áreas também.

Organização

Visando dar uma breve introdução conceitual sobre IA e suas ramificações, o Capítulo 1 abordará de forma sucita os principais aspectos desse ramo do conhecimento humano, com uma revisão bibliográfia sobre o tema, trazendo suas principais áreas de aplicação e também focando em aprendizado de máquinas e redes neurais.

As redes neurais artificiais terão suas características mais exploradas e aprofundadas no Capítulo 2, onde faremos uma correlação entre o neurônio biológico e o modelo matemático que produz o neurônio artificial utilizado nos programas de computador para gerar as redes neurais artificiais.

No Capítulo 3 abordaremos os principais aspectos do *deep learning*, da CNN e do *ensemble* de classificadores. Nesse capítulo também trataremos da transferência de aprendizado, ou seja, como podemos utilizar o aprendizado de uma rede neural base para treinar rapidamente outra rede neural a fim de minimizar o tempo de aprendizado desta última para o objetivo ao qual ela se destina.

A forma como utilizamos a CNN e o *ensemble* de classificadores será abordada no Capítulo 4, onde trataremos das metodologias propostas para a classificação das texturas visuais com as quais trabalhamos.

[11] Neste trabalho, quando a base CYST é utilizada apenas com as classes K e S, usamos a cigla CYST_KS

No Capítulo 5, traremos uma breve descrição dos bancos de imagens dos cistos mandibulares e dos bancos de imagens gerais (KTH-TIPS-2b, UMD, FMD, UIUC), bem como algumas referências do uso dessas imagens em outros trabalhos acadêmicos. Nesse Capítulo trataremos também da forma como esses bancos de imagens são processados pelos modelos que construímos.

Os resultados que obtivemos estão descritos no Capítulo 6, no qual também fazemos uma breve comparação com os resultados obtidos em trabalhos semelhantes e, assim, nesse capítulo tecemos algumas discussões sobre os métodos que utilizamos e os resultados que obtivemos.

No Capítulo 7 concluímos nosso trabalho, justificando o fato de termos alcançado os objetivos que almejávamos, bem como abordamos algumas propostas para a continuidade dos estudos aqui realizados.

1 Revisão Bibliográfica

O ser humano, ao longo de décadas, tentou responder as seguintes perguntas: as máquinas são capazes de aprender? Seriam os computadores capazes de aprender? Para respondê-las, pesquisadores de diversos ramos do conhecimento humano, desde que o homem começou a dominar os circuitos eletrônicos, utilizam-se desses equipamentos para automatizar processos computacionais, e tentam implantar os recursos da aprendizagem nos computadores. Alcançar esse objetivo foi e continua sendo, uma meta de longo prazo, com desafios que fomentam a inteligência artificial.

Nesse contexto, vale a afirmação: "O estudo e a modelagem computacional dos processos de aprendizado em suas múltiplas manifestações constituem o assunto do aprendizado de máquina" Michalski, Carbonell e Mitchell (1986, p.3).

1.1 Inteligência Artificial

"Inteligência artificial pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente" Luger (2004, p.23). Essa definição não serve para finalizar o assunto, mas será um norte para nossa abordagem doravante, haja vista que a inteligência artificial (como ciência) é jovem o suficiente para se permitir não estar restrita a uma única definição, mas estar aberta a romper e a expandir os paradigmas da computação moderna e, consequentemente, de se redefinir com o passar do tempo. Porém, a definição exposta nos impulsiona a pensar em algoritmos estruturados para resolverem problemas através do conhecimento que o computador adquiriu por meio de alguma técnica específica.

Dotar um computador de alguma forma de inteligência que se assemelhe ao pensamento humano precisa estar fundamentado em alguma estrutura sólida de como tal pensamento é logicamente formado. Devemos a axiomatização formal do raciocínio humano ao legado deixado por Gottlob Frege, Bertrand Russel, Kurt Göbel, Alan Turing, Alfred Tarski e outros. Contudo, suas raízes remotam aos filósofos gregos da antiguidade, tais como Aristóteles.

Um dos pioneiros da inteligência artificial foi Charles Babbage (cientista, matemático, filósofo e engenheiro mecânico nascido em Londres no século XIX). Ele conseguiu conectar os alicerces filosóficos à prática da computação científica ao projetar uma das primeiras máquinas capazes de executar cálculos de forma mecanizada. A máquina de Charles Babbage trouxe à tona a materialização de fundamentos filosóficos pois realizava cálculos ao manipular padrões, e essa aplicação da abstração e sua manipulação a uma

forma foi tema discorrido por Aristóteles.

Outro personagem fundamental para o desenvolvimento da inteligência artificial foi George Boole (matemático e filósofo britânico do século XIX), pois seus trabalhos, que tratam das leis da lógica, são essenciais para a computação moderna. Apesar de a álgebra booleana ser fundamental para o projeto dos circuitos lógicos digitais que conhecemos hoje, Boole realizou suas pesquisas baseado nas operações mentais e formalizou uma linguagem de poder extraordinário, apesar de ter concebido de forma simples as operações "E", "OU" e "NÃO", dando assim o cerne do cálculo lógico (IDOETA; CAPUANO, 1984, p.41).

A relação simbiótica entre o computador digital e os alicerces da inteligência artificial foram consolidadas pela formulação da ciência e matemática ocorridas nos séculos XVIII, XIX e início do século XX, apesar de o computador digital ter sido introduzido nesse contexto apenas no final dos anos 40.

A capacidade dos transistores de funcionarem como "flip-flops" (IDOETA; CAPUANO, 1984, p.231), possibilitou o armazenamento dos "bits" dando assim o poder de processamento requerido pelos programas inteligentes. Com isso, tornou-se possível produzir sistemas de raciocínio formal em processadores lógicos e testar sua capacidade de simular a inteligência humana.

Cabe destacar aqui também Alan Turing (matemático, lógico, criptoanalista e cientista da computação britânico, nascido em 1912), que produziu um dos primeiros artigos sobre inteligência de máquina, em 1950. Ele elaborou um teste que mede o desempenho de uma máquina supostamente inteligente, comparada ao desempenho de um ser humano. O teste de Turing ainda é utilizado para avaliar programas de IA modernos, apesar de sofrer críticas por parte de alguns membros da comunidade científica.

A representação do conhecimento e a busca para a solução de problemas são os principais objetos de estudo dos pesquisadores da inteligência artificial.

A representação do conhecimento se preocupa em conceber uma linguagem que se adeque de forma eficaz à implementação dos algoritmos no computador. A busca para a solução de problemas visa "os estágios sucessivos e alternativos no processo de solução do problema" Luger (2004, p.38).

1.1.1 Áreas de Aplicação da Inteligência Artificial

Trataremos a seguir de algumas das principais áreas de atuação da inteligência artificial. Porém, como dissemos no início dessa seção, IA é uma ciência com muitos ramos em estado embrionário, com taxa de crescimento muito alta. Portanto, o leitor pode se deparar com aplicações de IA que não foram listadas aqui devido à natureza inovadora que a inteligência artificial oferece.

1.1.1.1 Jogos

As primeiras pesquisas de IA basearam-se nos jogos de tabuleiro, tais como xadrez, damas, entre outros. Uma das características interessantes dos jogos que os tornam especiais para IA é possuírem regras bem definidas. Outra é a plataforma de interação humana ser de fácil adaptabilidade aos computadores, como no caso dos tabuleiros, que não exigem pesados processadores de imagem para serem produzidos. Além do que, os jogos, em sua grande maioria, não apresentam problemas éticos e são facilmente testados por humanos não especialistas em computação.

As heurísticas [1] dos jogos reforçam sua importância em IA, pois ao produzirem fortes soluções, com fácil manipulação de teste e execução, geram a oportunidade de explorar o uso dessas heurísticas em outros problemas de forma rápida e eficiente.

Apesar de parecer uma aplicação meramente voltada ao entreterimento humano, jogos com inteligência artificial são utilizados, por exemplo, em simuladores de voo, e podem treinar pilotos em situações onde pilotar uma aeronave ou uma espaçonave poderia gerar elevados custos operacionais e riscos desnecessários à tripulação.

1.1.1.2 Raciocínio Automático e Prova Automática de Teoremas

Sendo utilizados como ferramenta de auxílio para os matemáticos na construção da Matemática formal, os provadores de teorema são amplamente destinados a realizar tarefas básicas (mas não menos laboriosas) no tratamento rigoroso das provas matemáticas, permitindo aos matemáticos gastarem-se em "decompor um problema grande em subproblemas e para conceber heurísticas de busca no espaço de provas possíveis" Luger (2004, p.40).

O raciocínio automático e a prova automática de teoremas é dos ramos mais antigos de aplicação de IA e um dos que possui mais resultados em suas aplicações na Ciência Matemática.

1.1.1.3 Sistemas Especialistas

"O conhecimento especialista é uma combinação de um entendimento teórico do problema com uma coleção de regras heurísticas para resolver problemas, que a experiência demostrou ser efetiva no domínio" Luger (2004, p.40).

Os sistemas especialistas são normalmente concebidos através de um trabalho conjunto em que o especialista do domínio (um médico, geólogo, engenheiro, ou outro) fornece ao especialista em IA o conhecimento necessário para que o sistema resolva problemas similares ao que o especialista do domínio é capaz de resolver.

[1] Heurísticas são "técnicas poderosas para determinar quais alternativas devem ser exploradas no espaço do problema" Luger (2004, p.39).

Os sistemas especialistas possuem algumas limitações, porém são ferramentas de uso amplo em diversas áreas do conhecimento humano e possuem um forte potencial científico (LUGER, 2004, p.41).

1.1.1.4 Compreensão da Linguagem Natural

De aplicativos para celulares a carros inteligentes, a compreensão e a produção da linguagem humana pelos computadores já é realidade na vida da maioria das pessoas com acesso à tecnologia moderna.

Portanto, sua aplicabilidade tornou-se óbvia o suficiente para não nos gastarmos em descrever sua importância ao estudo de IA nesse segmento.

1.1.1.5 Planejamento e Robótica

Robôs capazes de tormar decisões frente a um cenário complexo, dividindo e subdividindo o problema que ele tem para resolver, chegar à solução e executá-la. Esse é o desafio da IA na robótica.

Hoje vemos carros ocuparem uma vaga sozinhos, sendo essa uma demonstração simples de como os robôs podem analisar um problema e resolvê-lo sem intervenção humana.

1.1.1.6 Aprendizado de Máquina

A seguir há uma seção exclusiva sobre o aprendizado de máquina, pois este é certamente uma das seções mais importantes em IA no mundo moderno e é também a principal ferramenta usada neste trabalho.

As aplicações com aprendizado de máquina tentam resolver problemas da forma mais assertiva possível, valendo-se de seus acertos e erros para o ajuste de seus parâmetros na tentativa de minimazar tais erros, o que não é possível em sistemas especialistas que utilizam apenas o conhecimento do especialista do domínio para executar suas tarefas.

Apesar de se mostrar uma excelente aplicação de IA, o aprendizado de máquina perde desempenho devido a limitações de hardware quando seu algoritmo envolve muitos cálculos complexos e, também, quando sua base de dados ocupa muita memória. Porém, vivemos um avanço constante no desenvolvimento de novas tecnologias para que essas limitações sejam superadas, principalmente com a possibilidade do uso de uma quantidade astronômica de dados a que hoje temos acesso (o comumente chamado "*Big Data*") e também do uso de GPU [2] para o cálculo matricial em grande escala.

[2] GPU significa Unidade de Processamento Gráfico, mas em IA é amplamente utilizada para realizar (muitas vezes com um ganho de mais de 1000% em termos de tempo) operações com matrizes que levariam horas ou dias para serem processadas em uma CPU.

Esses são alguns dos motivos do aprendizado de máquina ser uma aplicação de IA com tantas frentes de pesquisa na atualidade.

1.2 Aprendizado de Máquina

Dos grandes desafios da IA, as habilidades humanas que envolvem a linguagem e a cognição possuem destaque por suas intrínsecas dificuldades em serem reproduzidas por um computador.

Dentre outros motivos, essas habilidades envolvem competências adquiridas por meio da experiência. Utilizando-se, muitas vezes, das emoções que reforçam a capacidade de aprender novos saberes e de expressá-los por meio da fala.

Visando romper essas dificuldades, as pesquisas em aprendizado de máquina têm se ocupado, principalmente, em discorrer sobre: **estudos orientados a tarefas**, **simulação cognitiva** e **análise teórica** (MICHALSKI; CARBONELL; MITCHELL, 1986, p.3). Daremos ênfase à simulação cognitiva, pois se ocupa em investigar a computação baseada na experiência humana de aprendizado.

O aprendizado de máquina pode ser classificado em aprendizado **simbólico**, **conexista** e **emergente** (LUGER, 2004, p.331). Dentre outras ferramentas de aprendizado, o simbólico utiliza-se do viés indutivo, "que é aprender uma generalização a partir de um conjunto de exemplos" Luger (2004, p.334).

Aliás, a generalização é uma característica importante do aprendizado de máquina. Controlar a generalização de tal forma que ela esteja sempre convergindo para o estado ótimo de solução do problema é um fator importante a ser considerado e será abordado em seções posteriores.

O aprendizado simbólico também se vale de algoritmos que "não usam qualquer conhecimento prévio do domínio do problema" Luger (2004, p.335). Algoritmos construídos dessa forma são chamados *algoritmos baseados em similaridade*. O aprendizado conexista inspira-se na forma neural ou biológica de aprender.

Nessa abordagem, os símbolos perdem ênfase e o aprendizado ocorre por adaptação e interação entre componetes simples chamados **neurônios**. "A solução de um problema é paralela no sentido que todos os neurônios dentro do conjunto ou camada processam as suas entradas simultaneamente e independentemente" Luger (2004, p.391).

Aqui, os padrões de um domínio são convertidos em vetores numéricos e operações matemáticas ocorrem entre os neurônios e através deles para que o problema seja resolvido.

A escolha da arquitetura da rede de neurônios é impregnada pela intuição do especialista em IA, ou seja, o projetista interpreta, com base em sua experiência, qual

arquitetura se adapta melhor à resolução de um problema específico.

A abordagem conexista oferece bons resultados quando aplicada: à classificação, ao reconhecimento de padrões, à evocação de memória, à predição, à otimização e à filtragem de ruído.

Uma das formas do computador aprender é por meio de exemplos que lhe são oferecidos. Esses exemplos podem estar devidamente classificados, concebendo o chamado *aprendizado supervisionado*. Porém, se o aprendizado ocorre idependente da classificação dos exemplos, mas observando suas propriedades e dividindo os exemplos em conjuntos por similaridades dessas propriedades, temos o chamado *aprendizado não supervisionado*.

1.3 Breve Contextualização Histórica das Redes Neurais

Dentre as formas de aprendizado de máquina existentes, vamos destacar a seguir alguns fatos que marcaram o estudo das Redes Neurais Artificiais[3] (RNA).

Há uma forte analogia entre o neurônio biológico e o artifical, e o trabalho pioneiro e inovador em neurociência escrito por Santiago Ramón y Cajal[4], no início do século XX (CAJAL, 1909), fomentou os estudos iniciais dos neurônios artificiais.

Com base em seus resultados, podemos entender que: nos organismos vivos "os neurônios são as unidades celulares básicas que compreendem redes neurais responsáveis pelo fluxo de informações no sistema nervoso" Davies e Morris (2004, p.75).

As RNAs, como conhecemos hoje, tiveram seu embrião no aclamado trabalho realizado por McCullock & Pitts (MCCULLOCH; PITTS, 1943), no qual eles "descrevem um cálculo lógico das redes neurais que unificava os estudos de neurofisiologia e da lógica matemática" Haykin (2008, p.63).

Em 1949, no livro de Hebb (HEBB, 1949), é apresentada uma formulação explícita de uma metodologia da aprendizagem que ocorre nos organismos vivos durante a modificação sináptica. Na modificação sináptica, a conectividade cerebral está sempre em transformação conforme o ser vivo é exposto a tarefas funcionais diferentes e os agrupamentos neurais surgem durante essas modificações.

Segundo o postulado de aprendizagem de Hebb, "a eficiência de uma sinapse variável entre dois neurônios é aumentada pela ativação repetida de um neurônio causada pelo outro neurônio, através daquela sinapse" Haykin (2008, p.64).

Uma das limitações do trabalho de McCulloch & Pitts (MCCULLOCH; PITTS, 1943) é que não previa um aprendizado automatizado, como conhecemos hoje; em 1949,

[3] O Capítulo 2 tratará dos principais aspectos das Redes Neurais Artificiais.

[4] Santiago Ramón y Cajal foi um médico e histologista espanhol, considerado o "pai da neurociência moderna".

Hebb (HEBB, 1949) adapta esse trabalho, usando aprendizado não supervisionado. Rosemblatt em 1958 (ROSEMBLATT, 1958) desejava, com seu Perceptron, além de outros objetivos, superar essa deficiência usando aprendizado supervisionado.

Widrow e Hoff em 1962 com seu neurônio ADALINE (WIDROW; HOFF, 1962), além de mostrar ser possível ajustar automaticamente seus pesos, mostra que seu neurônio poderia trabalhar com memória associativa e desenvolveu o algoritmo do mínimo quadrado médio (LMS).

Destacamos ainda os trabalhos realizados por Grossberg (GROSSBERG, 1969), em que os neurônios são usados de forma não supervisionada e são capazes de aprender, lembrar e reproduzir padrões. Grossberg também estabeleceu um novo princípio de auto-organização conhecido como teoria da *ressonância adaptativa* (ART, *Adaptative Resonance Therory*), de acordo com esse princípio "se um padrão de entrada e o padrão realimentado aprendido coincidirem, então ocorre um estado dinâmico chamado *ressonância adaptativa*" Haykin (2008, p.66).

As **redes neurais convolucionais** foram introduzidas nos estudos de redes neurais pelos trabalhos realizados por Wiesel e Hubel: (WIESEL; HUBEL, 1959) e (WIDROW; HOFF, 1962), sendo que suas conclusões foram baseadas na análise do córtex visual do gato.

O *feedforward* em redes com múltiplos percetprons deu origem aos primeiros modelos de aprendizado profundo e os primeiros sistemas que apresentaram essas características foram as redes treinadas pelo Método de Treinamento de Dados em Grupo (GMDH) feitos por Ivakhnenko e Lapa (IVAKHNENKO; LAPA, 1965), Ivakhnenko et al. (IVAKHNENKO; LAPA; MCDONOUGH, 1967), e Ivakhnenko (IVAKHNENKO, 1971a) e (IVAKHNENKO, 1971b).

Em 1979, surgiu o Neocognitron de Fukushima (FUKUSHIMA, 1979), que é uma rede utilizada para reconhecer padrões de forma não-supervisionada do tipo "vencedor leva tudo", usando como base a similaridade geométrica (Gestalt) de suas formas, sem afetar suas posições. O Neocognitron é baseado nas conclusões de Wiesel & Hubel (WIESEL; HUBEL, 1959) e (WIDROW; HOFF, 1962), sendo a primeira rede neural convolucional (CNN), com algumas das características que ainda hoje encontramos nas CNNs modernas, como matrizes com mais de uma dimensão e camadas de sub-amostragem, por exemplo.

O desenvolvimento da retropropagação por descida de gradiente tem suas bases na minimização de erros apresentada por Hadamard (HADAMARD, 1908), e sua aplicação às redes neurais tem sido discutida desde 1960, em trabalhos[5] que usam a regra da cadeia para conseguir uma descida mais acentuada do gradiente na atualização dos pesos. Um dos primeiros trabalhos a utilizar o método do gradiente estocástico para classificação

[5] (BRYSON, 1961) e (KELLEY, 1960), por exemplo.

adaptativa de padrões pode ser visto em (AMARI, 1967).

Uma abordagem eficiente da retropropagação de correção do erro, usando o gradiente descendente como conhecemos hoje, foi descrita em 1970 por Linnainmaa (LINNAINMAA, 1970).

Em 1969, o livro de Minsky & Papert (MINSKY; PAPERT, 1969) mostrou como os perceptrons lineares de uma camada possuíam a limitação de classificar apenas conjuntos de dados linearmente separáveis. Com isso, por mais de quinze anos, poucos trabalhos promoveram avanços significativos nas redes neurais. Este período é hoje conhecido como o "(grande) inverno da IA".

Esse quadro foi alterado quando o *backpropagation* (RUMELHART; HINTON; WILLIAMS, 1986) foi aplicado em redes com poucas camadas, pois o teorema de Kolmogorov [6] (KOLMOGOROV, 1965) motivava, naquela época, o estudo de redes de muitos neurônios com poucas camadas.

Nesse contexto, devemos destacar o trabalho de Ackley, Hinton e Sejnowski (ACKLEY; HINTON; SEJNOWSKI, 1985), onde foi apresentada a "máquina estocástica conhecida como máquina de Boltzmann, que foi a primeira realização bem-sucedida de uma rede neural de múltiplas camadas" Haykin (2008, p.67).

Em 1989, Hornik *et al.* (HORNIK; STINCHCOMBE; WHITE, 1989) provaram que as redes MLP[7] são aproximadores universais de funções, ou seja, é possível representar qualquer função com uma rede neural adequada. Com isso, desde que um problema possa ser matematicamente modelado, teoricamente, é possível resolvê-lo por redes neurais. Nesse mesmo ano, o *backpropagation* foi aplicado às redes convolucionais nos trabalhos de Yan LeCun e seus colaboradores em (LECUN, 1989) e (LECUN et al., 1989).

Em (LECUN et al., 1989) há o relato do desempenho de uma rede inspirada no Neocognitron de Fukushima (FUKUSHIMA, 1979), aplicando o algoritmo de retropopagação[8] para identificar dígitos manuscritos obtidos dos correios dos EUA, com base nos manuscritos da agência de Buffalo. O objetivo era identificar nas correspondências o código postal, usando uma câmera conectada às placas de vídeo de um computador do tipo PC.

Nessa rede de múltiplas camadas [9], são utilizados os *detectores de atributos (features)* para que os atributos locais de cada imagem fossem, pelo método da convolução

[6] O Teorema de Kolmogorov trata da representação de funções de múltiplas variáveis por um cubo n-dimensional ($n \geqslant 2$). Ele possíbilita representar qualquer função contínua, estabelecendo-se uma certa precisão, por uma rede neural com uma camada oculta, usando-se um número suficiente de neurônios.

[7] MLP: *multilayer perceptron*.

[8] Que no trabalho é chamado de *retropropagação restrita*.

[9] Em (LECUN et al., 1989) foram utilizadas exatamente três camadas ocultas, o que para a época era um grande desafio e considerado uma inovação em RNA.

(FUKUSHIMA, 1979), formando o que no trabalho foi chamado "*feature maps*" (mapas de atributos); essa RNA também utilizou o compartilhamento de pesos com a finalidade de reduzir o número de parâmetros livres e para expressar informações sobre a geometria e a topologia da imagem a ser trabalhada. Destacamos também o uso do gradiente estocástico e a implantação de uma versão especial do algoritmo de Newton, usando aproximação diagonal positiva da matriz Hessiana.

As conclusões expressas em (LECUN et al., 1989) mostram que a rede convergiu rapidamente, trazendo um inovador procedimento para reconhecimento de imagens, usando um pre-processamento mínimo. "Este trabalho também introduziu o conjunto de dados MNIST de digitos manuscritos, que ao longo do tempo se tornou talvez a referência mais famosa do *Machine Learning*" Schmidhuber (2015, p.93).

Apesar de trabalhos como os de LeCun *et al.* (LECUN et al., 1989) mostrarem que o *backpropagation* é eficiente quando aplicado em redes neurais profundas, há um inconveniente nessa técnica, conforme relata Hanin: "Um obstáculo fundamental de redes neurais profundas usando otimização baseada em gradiente é o problema da explosão e desaparecimento do gradiente" Hanin (2018, p.1).

Em 1991, a tese (HOCHREITER, 1991) identificou formalmente o problema e vários trabalhos abordando o assunto, desde então, tem sido feitos para mitigá-lo, mas ainda não há uma solução única ou definitiva. Para redes recorrentes com *backpropagation*, o LSTM [10] (HOCHREITER; SCHMIDHUBER, 1997) é uma proposta de solução apresentada por Hochreiter & Schmidhuber em 1997. Em (HANIN, 2018) vemos mais uma proposta de solução para esse problema para redes neurais profundas totalmente conectadas e ativadas com a função ReLU.

Em 1995, Vladmir N. Vapnik, no livro (VAPNIK, 1995), afirma que o problema de aprendizagem de máquina é tão geral que praticamente todos os estudos nessa área possuem análogo na Estatística, inclusive, "alguns resultados gerais muito importantes foram encontrados primeiramente nessa teoria e depois reformulados nos termos estatísticos" Vapnik (1995, p.viii).

Na época em que esse livro foi escrito, o aprendizado de máquina valia-se de poucas amostras. Por exemplo, no trablaho de LeCun *et al* (LECUN et al., 1989) a RNA foi treinada com pouco mais de nove mil exemplos. Hoje podemos usar milhões de imagens para treinar uma RNA com relativa facilidade.

Diante da realidade que viviam, Vapnik em (VAPNIK, 1995) mostra os avanços da teoria estatística, além dos novos métodos de inferência, obtidos por meio do aprendizado de máquina usando pequenas amostras.

Ainda com base na teoria de pequenos tamanhos de amostras, Vapnik mostra

[10] LSTM: *Long short-term memory*

o funcionamento da *máquina de vetores de suporte* (SVM), que se tornaria uma aplicação prática e comercialmente viável para o reconhecimento de padrões em anos futuros, como, por exemplo, quando em 2009 uma rede convolucional tridimensional (com *pool* máximo), combinada com SVM (JI et al., 2013), ganhou três categoriass no concurso TRECVID 2009, quando usada para identificar ações humanas em vídeos de vigilância.

Os anos de 2006 e 2007 tiveram relevantes destaques para o aprendizado de máquina, principalmente porque pesquisadores de RNA dos últimos anos valeram-se do uso da GPU, um equipamento amplamente utilizado em videogames, que é um mercado bastante competitivo e por isso a concorrência faz baratear os preços, facilitando o uso de seus equipamentos no meio acadêmico.

A primeira implementação de redes convolucionais utilizando GPU ocorreu em 2006 (CHELLAPILLA; PURI; SIMARD, 2006), em que sub-rotinas básicas de álgebra linear foram aceleradas com o uso da GPU e nos testes realizados a taxa de erro não foi superior a 1,2%, com uma velocidade 4 vezes maior que usando CPU, um excelente resultado para a época.

Já em 2007 ocorreu a primeira aplicação do *max-pooling* [11] em redes convolucionais: em (RANZATO et al., 2007) foi proposto um modelo de aprendizado não supervisionado para o pré-treinamento das camadas profundas da rede, usando extratores de atributos invariantes a pequenas distorções nas imagens de objetos que se desejava reconhecer.

Com o empilhamento de vários níveis da rede, a arquitetura resultante ficou idêntica ao Neocognitron de Fukushima (FUKUSHIMA, 1979). Nessa rede, a camada final foi treinada no modo supervisionado executando a classificação das imagens. Redes convolucionais criadas com base nessas características, iriam adquirir o reconhecimento de padrões visuais sobre-humano em 2011 (CIRESAN et al., 2011).

No desafio de reconhecimento visual em grande escala do ImageNet [12] (ILSVRC [13]) de 2012, Krizhevsky *et al.* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) propuseram uma CNN que posteriormente viria a se chamar AlexNet, que foi treinada para classificar 1,2 milhões de imagens, distribuídas em 1000 classes diferentes, alcançando um erro entre as 5 principais predições de 15,4% no Imagenet, sendo que o melhor resultado de classificação obtido até aquele momento era um erro de 26,2%.

O ImageNet se tornou também uma base de dados comum para *transfer learning*

[11] Quando redes convolucionais são combinadas com *max-pooling*, elas se tornam uma rede convolucional chamada de HMAX.

[12] ImageNet é uma grande base de dados visual com mais de 10 milhões de imagens catalogadas em mais de 20.000 categorias.

[13] O ILSVRC ocorreu de 2010 a 2017 para que equipes de pesquisa em deep learning avaliassem seus algoritmos usando o ImageNet. A partir de 2018 o Kaggle também tem realizado desafios similares ao ILSVRC.

(transferência de aprendizado), conforme é apresentado em (PAN; YANG, 2010). O *transfer learning*[14] já atrai a atenção dos pesquisadores em IA desde 1995 com diferentes nomes: "*Learning to learn, life-long learning, knowledge transfer, inductive transfer, multitask learning, knowledge consolidation, context-sensitive learning, knowledge-based inductive bias, metalearning, incremental/cumulative learning*" Pan e Yang (2010, p.1543).

Em 2014, Zeiler & Fergus (ZEILER; FERGUS, 2014) propuseram uma melhoria na AlexNet através da chamada DeconvNet: "Uma DeconvNet pode ser pensada como um modelo de rede convolucional que usa os mesmos componentes (filtragem, *pool*), mas que em vez de mapear pixels para os *features*, faz o oposto" Zeiler e Fergus (2014, p.820).

Em (RUSSAKOVSKY et al., 2015), é feita uma análise das lições aprendidas com os desafios porpostos no ILSVRC de 2010 a 2015, como o ImageNet influenciou os avanços no reconhecimento de objetos por *machine learning*, e faz um resumo dos algoritmos mais bem-sucedidos nesses desafios.

Destacamos duas lições relatadas por Russakovsky *et al.*: "Todas as tarefas de inteligência humana precisam ser excepcionalmente bem projetadas" Russakovsky et al. (2015, p.244), e "a expansão de dados sempre revela desafios inesperados" Russakovsky et al. (2015, p.244).

Apesar de não considerar o conjunto de dados do ImageNet suficientemente desafiador, com erros de notação e tendo o ILSVRC regras muito restritivas, (RUSSAKOVSKY et al., 2015) enaltecem o valor que o ImageNet promoveu para o desenvolvimento de novos conjuntos de dados e algoritmos de reconhecimento de objetos.

Nos últimos 10 anos, o *Deep Learning* tem consumido grande parte das pesquisas envolvendo aprendizado de máquina, e não poderíamos deixar de destacar o resumo teórico (LECUN; BENGIO; HINTON, 2015) dos ganhadores do prêmio Turing [15] de 2018: LeCun, Bengio e Hinton; nesse trabalho encontramos os principais aspectos do *Deep Learning*, suas principais aplicações e as perspectivas de avanços cientifícos esperados para essa área de IA para os próximos anos.

[14] A transferência de aprendizado será abordada em seção específica.
[15] O prêmio Turing é considerado o Nobel da Computação.

2 Rede Neural Artificial

Na Seção 1.3, vimos que o neurônio natural (Figura 1) é uma fonte inspiradora para o trabalho com redes neurais artificiais. Faremos, neste capítulo, várias analogias entre o neurônio natural e o artificial, expandindo essa idéia para as redes de neurônios artificiais.

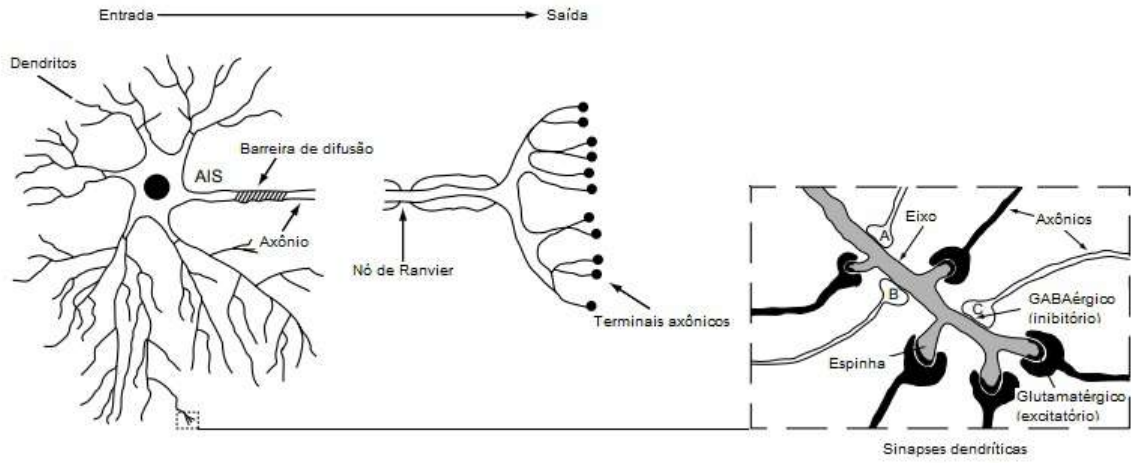


Figura 1 – Neurônio Natural.

"Os neurônios não são cabos elétricos passivos, mas unidades integradoras, coletando informações de várias fontes" Davies e Morris (2004, p.75).

A recepção e a transmissão das informações ocorre nas sinapses. "Uma sinapse converte um sinal elétrico pré-sináptico em um sinal químico e então volta em um sinal elétrico pós-sináptico" Haykin (2008, p.33).

Os axiônios funcionam como a linha de trasmissão do neurônio, similar ao fio de um equipamento elétrico. Já os dentritos realizam a conexão proximal, similar ao conector que encontra-se na extremidade do fio, possibilitando a corrente elétrica fluir da tomada para o equipamento através do fio.

Apesar de ser muitas vezes mais lento que o semicondutor de silício, a rede de neurônios naturais possui uma capacidade de transmissão e armazenamento de dados que supera os mais modernos computadores atuais.

O neurônio artificial possui semelhanças de operação com o natural, porém a rede neural cerebral possui uma diversidade de funções e habilidades tão complexas e eficientes que as RNAs estão longe de atingir.

2.1 Um modelo de neurônio artificial

Na Figura 2 vemos a representação esquemática de um neurônio artificial, baseado no neurônio formulado por McCuloch e Pitts em (MCCULLOCH; PITTS, 1943). Nessa representação, podemos observar três elementos básicos que correlacionam o neurônio artificial com o natural.

1º) **Sinapses**: Cada dado de entrada x_p da sinapse p conectada ao neurônio k recebe um peso ou uma *força* quando multiplicada pelo peso sináptico w_{kp}.

2°) **Somador**: nele ocorre a combinação linear das entradas após terem recebido seus respectivos pesos sinápticos, sendo similiar ao núcleo do neurônio biológico.

3°) **Função de Ativação** ou **Função Restritiva**: Essa função interpreta os resultados da combinação linear dos sinais de entrada e ativa ou não a saída, de acordo com os critérios para sua ativação, sendo similar ao axiônio do neurônio natural.

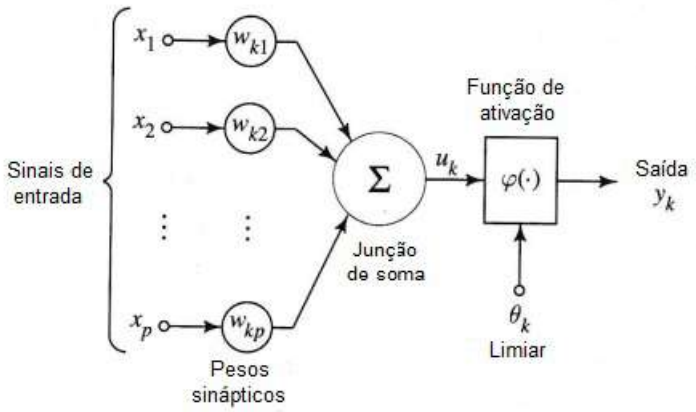


Figura 2 – Modelo não linear de neurônio artificial.

Matematicamente, podemos descrever um neurônio artificial k através de duas equações básicas 2.1 e 2.2:

$$u_k = \sum_{j=1}^{p} w_{kj}\, x_j \tag{2.1}$$

e

$$y_k = \varphi(\mu_k - \theta_k). \tag{2.2}$$

O parâmetro θ_k (Equação 2.2) é chamado *bias* e mostraremos sua importância quando analisarmos o Perceptron na Seção 2.3.

2.1.1 Função de Ativação

A função de ativação, representada na Figura 2 por $\varphi(\cdot)$, define a saída do neurônio com base no valor assumido pelo parâmetro u_k. Em (HAYKIN, 2008, p.38) são mostradas três funções de ativação básicas: *função de limiar*, *função linear por partes* e *função sigmóide*.

Na rede neural deste trabalho, foi utilizada a função **ReLU** [1], pois a rede é do tipo convolucional [2]. A não linearidade da função ReLU é uma característica que a torna desejável em redes convolucionais, sendo a função de ativação mais comumente utilizada para construir esse tipo de arquitetura de RNA, como afirma Yann LeCun *et al.* em (LECUN; BENGIO; HINTON, 2015, p.437). Abaixo mostramos a equação característica dessa função. Com base na Equação 2.2 e fazendo $x = \mu - \theta$ na Equação 2.3, temos:

$$\begin{gathered} f_{ReLU} : \Re \to \Re \\ y_k = f(x) = max(0, x) = \begin{cases} x \ se \ x > 0 \\ 0 \ se \ x \leqslant 0 \end{cases} \end{gathered} \tag{2.3}$$

Nas redes neurais, tanto o valor da função de ativação quanto de sua derivada são importantes e a função ReLU também apresenta essa vantagem: o cálculo do valor da função e de sua derivada são tarefas computacionalmente fáceis, ou seja, a derivada será 1 se x for positivo e 0 se x for negativo.

Com isso, mesmo que a rede precise realizar muitas multiplicações com a derivada da ReLU, esse cálculo tende a ser mais rápido, se comparado a quando se usa a sigmóide como função de ativação (MAAS; HANNUN; NG, 2013, p.2).

2.2 Aprendizagem por correção de erro

As redes neurais artificiais possuem parâmetros que são matematicamente ajustados durante o processo de aprendizagem. Nas redes com aprendizagem por correção de erro, existe um neurônio de saída k (Figura 3), que é alimentado por diversos outros neurônios "ocultos" [3].

O neurônio de saída é ajustado por um *sinal de erro* $e_k(n)$ (Equação 2.4), onde n é o passo de tempo discreto relacionado ao ajuste do peso sináptico do neurônio k, y_k a saída real do neurônio e d_k a saída desejada.

$$e_k(n) = y_k(n) - d_k(n), \tag{2.4}$$

[1] ReLU vem de *Rectified Linear Unit* em inglês e refere-se ao fato dessa função realizar a ativação de forma unitária e linearmente retificada.

[2] Trataremos das Redes Neurais Convolucionais na Seção 3.4.

[3] Neurônios ocultos são os que estão entre a camada de entrada da rede e a saída, que nesse caso é um único neurônio.

Os ajustes corretivos visam minimizar o erro entre a saída desejada do neurônio e sua saída real. Para atingir esse objetivo, uma **função de custo** ξ (Equação 2.5) deve ser minimizada [4].

$$\xi(n) = \frac{1}{2}e^2{}_k(n). \tag{2.5}$$

Supondo que $w_{kj}(n)$ represente o valor de peso sináptico do neurônio de saída k, excitado por um elemento $x_j(n)$ do vetor de sinal $X(n)$ que aciona o neurônio k, podemos escrever, com base na regra delta, a Equação 2.6:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n). \tag{2.6}$$

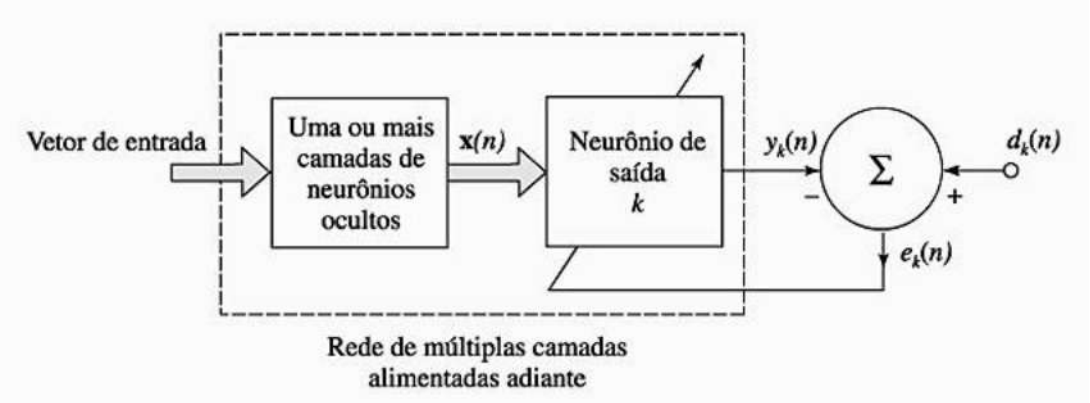


Figura 3 – Aprendizagem por correção de erro.

Na Equação 2.6, η é "uma constante positiva que determina a taxa de aprendizado quando avançamos em um passo no processo de aprendizado" Haykin (2008, p.78); η deve ser ajustado para que o processo de correção de erro atinja a estabilidade do processo de aprendizado iterativo.

2.3 O Perceptron

Em (ROSEMBLATT, 1958), é apresentado o Perceptron como uma forma de "ilustrar algumas das propriedades fundamentais dos sistemas inteligentes em geral, sem se envolver demais nas condições especiais ... de organismos biológicos específicos" Rosemblatt (1958, p.387).

O Perceptron foi formulado com base "no modelo atual em termos da teoria da probabilidade, e não da lógica simbólica" Rosemblatt (1958, p.387).

[4] Minimizar ξ é comumente chamada de *regra delta* ou *regra de Widrow-Hoff* em homenagem aos seus criadores (WIDROW; HOFF, 1960).

Com o Perceptron de Rosemblatt (que doravante chamaremos apenas perceptron), "é possível prever curvas de aprendizado a partir de variáveis neurológicas e, da mesma forma, prever variáveis neurológicas a partir de curvas de aprendizado" Rosemblatt (1958, p.407).

Um modelo de grafo de fluxo do perceptron é visto na Figura 4, onde $w_1, w_2, ..., w_m$ representam os pesos sinápticos; $x_1, x_2, ..., x_m$ representam as entradas de dados; o *bias* aplicado externamente é representado na figura por θ; v (Equação 2.7) é chamado limitador abrupto e definido por:

$$v = \sum_{i=1}^{m} w_i x_i - \theta. \tag{2.7}$$

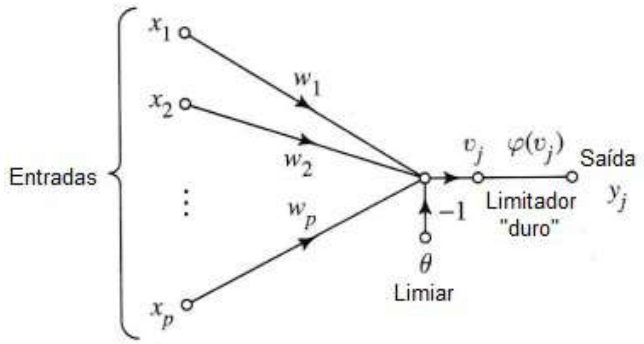


Figura 4 – Grafo de fluxo do sinal do perceptron.

Vamos começar nossa análise admitindo o perceptron como sendo capaz de classificar, o mais corretamente possível, o conjunto de dados $X = x_1, x_2, ..., x_m$ em uma de duas classes C_1 e C_2. Porém, expandir essa abordagem para mais de duas classes é admissível, como é mostrado em (HAYKIN, 2008, p.400).

A regra de decisão para determinar se um elemento x_k de X pertence à classe C_1 ou à classe C_2 é: se a respectiva saída y_k for $+1$, x_k pertence a C_1, se y_k for -1, x_k pertence a C_2.

Na Figura 5 vemos um mapa de regiões de decisão que ilustra o comportamento do classificador de padrões, como o perceptron que estamos abordando neste momento. Nesta figura, existem duas regiões separadas por um hiperplano definido na Equação 2.8, e a fronteira de decisão está representada por uma reta. Para que um ponto (x_1, x_2) pertença à classe C_1 ele precisa estar à direita da fronteira de decisão e pertencerá à classe C_2 se estiver à esquerda.

$$\sum_{i=1}^{m} w_i x_i - \theta = 0, \tag{2.8}$$

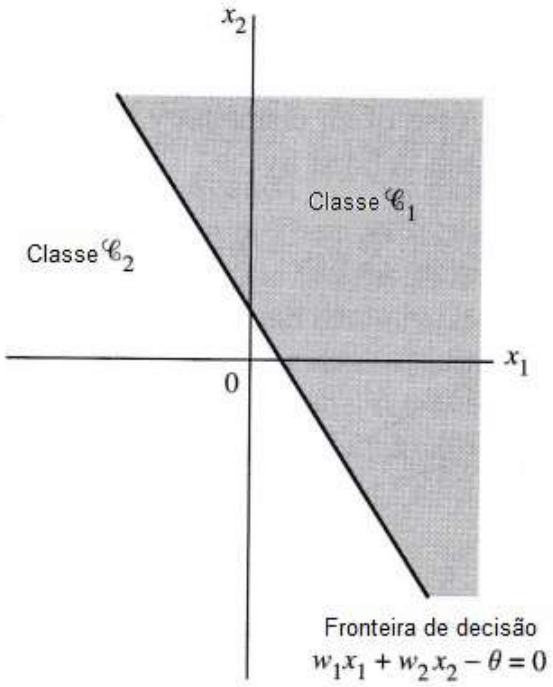


Figura 5 – Ilustração do mapa da região de decisões.

"Para o perceptron funcionar adequadamente, as duas classes C_1 e C_2 devem ser linearmente separáveis" Haykin (2008, p.164). Supondo então que os dados de entrada no perceptron são provenientes de classes linearmente separáveis, seja A_1 o subconjunto de dados de treinamento que pertencem a C_1 e A_2 o subconjunto de dados de treinamento da classe C_2. Usando o conjunto $A = A_1 \cup A_2$ para treinar o perceptron, o algoritmo de treinamento terá como objetivo ajustar o vetor $W = \{w_1, w_2, ..., w_m\}$ de tal forma que C_1 e C_2 sejam linearmente separáveis. Com isso, haverá um vetor $\hat{W} = \{\hat{w}_1, \hat{w}_2, ..., \hat{w}_m\}$ de pesos ajustados, tal que seu comportamento obedece as desigualdades que constam na Equação 2.9. A saber:

$$\begin{cases} \hat{W}X^T - \theta > 0, \forall x_k \in C_1 \\ \hat{W}X^T - \theta \leqslant 0, \forall x_k \in C_2 \end{cases}, \tag{2.9}$$

com $k \in \{1, 2, ..., m\}$. A escolha $x_k \in C_2$ se $\hat{W}X^T - \theta = 0$ foi arbitrária.

2.3.1 A convergência do algoritmo de treinamento do perceptron

O neurônio de McCulloch & Pitts possuía pesos e *bias* ajustávies. Porém, esses parâmetros não eram aprendidos de forma automática. Em (ROSEMBLATT, 1958), é mostrado como superar essa deficiência do modelo apresentado por McCulloch e Pitts usando a mesma arquitetura de neurônio, porém com a possibilidade de treiná-lo através de um algoritmo apropriado e, com isso, fazer o neurônio aprender os melhores valores para os pesos e o *bias*.

Para melhor desenvolvermos o algoritmo de correção de erros do perceptron, vamos redefinir o vetor com os dados de entrada, fixando $x_0 = -1$ e definindo $n \in \{1, 2, ..., r\}$ temos a Equação 2.10:

$$X(n) = [-1, x_1(n), x_2(n), ..., x_m(n)]^T. \tag{2.10}$$

.

Correspondentemente, vamos redefinir o vetor de pesos que é visto na Equação 2.11:

$$W(n) = [\theta(n), w_1(n), w_2(n), ..., w_m(n)]^T. \tag{2.11}$$

Feitas essas considerações, devemos ajustar a Equação 2.7, esse ajuste é visto na Equação 2.12:

$$v = W^T(n)X(n). \tag{2.12}$$

Os vetores $X(1), X(2), ..., X(r)$ formam $(m + 1)$-uplas ordenadas em $\mathbb{R}^{m+1}$. Chamaremos $\hat{X} = \{X(1), X(2), ..., X(r)\}$ de conjunto de treinamento do perceptron. Sendo $s, t \in \mathbb{N}$ e $s + t = r$, devemos supor que s elementos de $\hat{X}$ pertençam C_1 e t elementos de $\hat{X}$ pertençam a C_2, para o caso de classificarmos $X(n)$ em duas classes linearmente separáveis por um hiperplano.

Em (HAYKIN, 1994, p.110), prova-se que o treinamento do perceptron converge para um certo vetor de pesos ajustados $\hat{W}$, tal que após k iterações do algoritmo de correção de erros, ocorre alguma das desigualdades da Equação 2.13.

$$\begin{cases} \hat{W}^T X > 0, \forall X \in C_1 \\ \hat{W}^T X \leqslant 0, \forall X \in C_2 \end{cases} \tag{2.13}$$

Antes de apresentarmos o algoritmo, é necessário proceder com algumas definições. A função $F(\cdot)$, que aparecerá no passo 5 do algoritmo, é definida na Equação 2.14:

$$F(v) = \begin{cases} +1, \ se \ v > 0 \\ -1, \ se \ v \leqslant 0. \end{cases} \tag{2.14}$$

Outra definição é a resposta desejada quantizada, $d(n)$, que é definida pela Equação 2.15:

$$d(n) = \begin{cases} +1, \ \text{se } X(n) \in C_1 \\ -1, \ \text{se } X(n) \in C_2. \end{cases} \tag{2.15}$$

Logo, para cada $X(n) \in \hat{X}$, existe um $d(n)$ correspondente.

O perceptron acerta a classificação de $X(n)$ quando o $y(n)$ (saída do perceptron) correspondente à entrada $X(n)$ for igual a $d(n)$.

No algoritmo isso será verificado pela diferença $d(n) - y(n)$; ou seja, se $d(n) - y(n) = 0$, significa que o perceptron acertou a classificação e não é necessário ajustar $W(n)$; caso $d(n) - y(n) \neq 0$, então $W(n)$ será ajustado com base na diferença $d(n) - y(n)$, que será multiplicada por um fator η chamado de *taxa de aprendizado*; esse parâmetro é uma constante positiva entre 0 e 1, ou seja, $0 < \eta \leqslant 1$.

Normalmente, quando uma arquitetura de RNA está sendo preparada para um novo conjunto X de dados de teste, o programador tende a definir $\eta = 0.5$, a fim de avaliar o desempenho da rede para esse valor de η, podendo reajustá-lo, se necessário.

Outro parâmetro a ser definido é o número de *épocas* [5] de treinamento. Já sabemos que o algoritmo do percpetron irá convergir para um $\hat{W}$ tal que as desigualdades em 2.13 irão ocorrer.

Nosso problema agora é descobrir: quando a convergência ocorrerá? A resposta a essa pergunta depende de vários fatores, como o tipo de dado a ser analisado e a quantidade de elementos de $\hat{x}$.

No algoritmo que iremos descrever a seguir, **p** representa a constante que indica o número de épocas de treinamento, que deve ser o mais próximo possível da convergência. Na verdade se **p** for maior do que esse ponto de convergência não há qualquer problema matemático pois, uma vez que o algoritmo atingiu $\hat{W}$, a rede não faz mais alterações em $W(n)$. O problema no caso é mais em termos de desperdício de recurso computacional, pois a máquina fica em funcionamento por um tempo acima do necessário [6].

2.3.2 O algoritmo de treinamento por correção de erros do perceptron

- **Passo 1**: Defina $W(0) = 0$;
- **Passo 2**: Repita os passos de 3 a 6 **p** vezes;
- **Passo 3**: Para n de 1 até r, repita os passos de 4 até 6;
- **Passo 4**: Leia $X(n)$ e $d(n)$;
- **Passo 5**: Armazene $y(n) = F[W^T(n)X(n)]$; e
- **Passo 6**: Armazene $w(n+1) = W(n) + \eta[d(n) - y(n)]X(n)$.

[5] Cada época de treinamento ocorre quando todos os dados do conjunto de treino passam pela RNA e suas saídas são computadas.

[6] A quantidade de épocas de treinamento é definida pelo programador da RNA.

Uma observação é que, no **Passo 1**, os pesos foram inicializados com 0, formando assim o vetor nulo. Mas existe a possibilidade de iniciar com valores randômicos. Na prática, é necessário implementar o algoritmo e verificar, para o problema específico a ser resolvido, qual é a melhor solução para iniciarlizar os pesos.

2.4 MLP e *Backpropagation*

Na Seção 2.3, descrevemos resumidamente um neurônio artificial (o perceptron), que aprende por correção de erro. Neste momento, analisaremos a associação de vários perceptrons formando uma RNA, que terá características encontradas, inclusive, no aprendizado profundo e nas redes neurais convolucionais[7]. Essa RNA será dividida em camadas[8].

A primeira camada é representada pelas conexões de entrada dos dados que alimentarão a rede, ou seja, os dados de entrada alimentam simultaneamente um conjunto de n neurônios preparados especificamente para receber os dados de entrada.

A última camada recebe os resultados processados pela rede e gera uma saída, que no nosso caso corresponde a uma classificação do dado de entrada. Por isso, chamaremos a última camada de **classificador**.

Entre a primeira e a última camada da rede está a **camada oculta** (podemos ter mais de uma na verdade), onde um segundo grupo de neurônios recebe os dados do primeiro grupo que está na primeira camada da rede. Esse segundo grupo realiza seus cálculos e transmite seus resultados para um terceiro grupo de neurônios (caso haja), que realiza seus cálculos e transmite para um quarto grupo de neurônios (caso haja); esse processo continua até que o k-ésimo grupo de neurônios entrega seus resultados ao classificador da rede. E é a esse entrelaçamento de neurônios, recebendo e transmitindo dados, é que dá-se o nome de **camada oculta** da rede.

Há diversas arquiteturas de RNA com múltiplas camadas. Não há um número pré-estabelecido de quantos neurônios farão parte de cada camada. Uma ilustração do que descrevemos é vista na Figura 6.

[7] O Capítulo 3 tratará do aprendizado profundo e das redes convolucionais.
[8] De onde se justifica chamar MLP, do inglês *multilayer perceptron*.

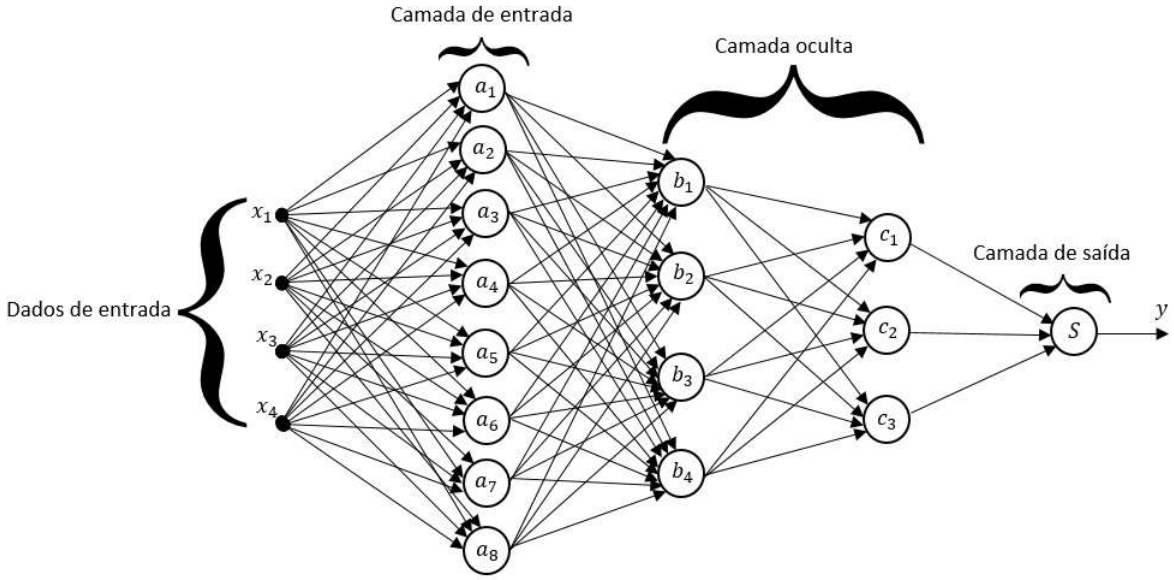


Figura 6 – Múltiplas Camadas de Perceptrons - MLP.

A MLP aceita conjuntos de dados não linearmente separáveis (caso não aceito pelo Perceptron de Rosemblatt), desde que treinada com o *algoritmo de retropropagação de erros* (***backpropagation***). Esse algoritmo consiste de dois passos a serem aplicados nas diversas camadas da rede: um passo para frente e outro para trás, ou seja, a propagação e a retropropagação.

No passo para frente, os dados que entram na rede alimentam progressivamente os neurônios da primeira à última camada, sofrendo a ação dos pesos sinápticos que ficam fixos até que o classificador interprete os resultados da rede.

No passo para trás, a saída do classificador é analisada e uma correção do erro de classificação nos pesos sinápticos e *bias* é realizada, da última para a primeira camada da rede, daí o motivo de se chamar retropropagação.

Existem basicamente três modos de se aplicar o aprendizado por *backpropagation*:

- *Modo on-line, sequencial, padrão ou estocástico.* Nesse modo, a cada novo dado que passa pela rede é feita a retropropagação e os pesos sinápticos e o *bias* são ajustados.
- *Modo por lote.* A cada passagem de um dado novo na rede o erro de classificação é armazenado e ao final de uma época de treinamento, esses erros computados farão parte do cálculo do erro médio que será utilizado no retro-ajuste dos pesos e *bias*.
- *Modo por minilotes.* Esse modo é um meio termo entre o modo *on-line* o modo por lote, pois o cojunto de dados de treinamento é divido em subconjuntos. A cada passagem de um dado do subconjunto, o erro é computado e, ao final da passagem

do último dado desse subconjunto, os erros computados farão parte do cálculo do erro médio e a retropropagação ocorre; daí um novo subconjunto de dados de treino entra na rede e o processo se repete até que todos os dados passem pela rede e se encerre uma época de treinamento.

O modo de aprendizado por minilotes é o mais comumente utilizado, inclusive em redes neurais profundas, pois geralmente utiliza-se GPU para o processo de aprendizado. A entrada de minilotes ocorre em paralelo e os cálculos nos neurônios são processados todos ao mesmo tempo na GPU, daí pode-se rapidamente chegar ao erro médio.

Há um elemento chave no algoritmo de retropropagação, que permite à MLP classificar dados não linearmente separáveis, o uso do **gradiente descendente**. Descrevemos na Equação 2.5 a função de custo da aprendizagem por correção de erro, e o *backpropagation* é uma aprendizagem por correção de erro, então vamos manter aquela notação e abordagem.

Afirmamos na Seção 2.2 que a função de custo ξ precisa ser minimizada; aqui, ξ será minimizada pelo método do **gradiente descendente**.

Retomaremos o último passo do algoritmo de aprendizado do perceptron da Seção 2.3, só que dessa vez o ajuste dos pesos ocorrerá ao término de cada minilote[9] e agora, em vez de simplesmente armazenar $w(n+1) = W(n) + \eta[d(n) - y(n)]X(n)$, haverá uma correção dos parâmetros do perceptron na direção do gradiente negativo. Definimos $w(n+1)$ e Δw, respectivamente, nas Equações 2.16 e 2.17 como:

$$w(n+1) = w(n) + \Delta w(n) \tag{2.16}$$

sendo

$$\Delta w = -\eta \frac{\partial \xi}{\partial w}. \tag{2.17}$$

Para cada amostra i do minilote, o gradiente $\dfrac{\partial \xi}{\partial w}$ pode ser desenvolvido usando a regra da cadeia, conforme consta na Equação 2.18.

$$\frac{\partial \xi_i}{\partial w} = \frac{\partial z_i}{\partial w} \cdot \frac{\partial y_i}{\partial z_i} \cdot \frac{\partial \xi_i}{\partial y_i} \tag{2.18}$$

Como ξ_i é um erro quadrático, ou seja, $\xi_i = \dfrac{1}{2}(y_i - d_i)^2$ (tomando como base a Equção 2.5), então o termo $\dfrac{\partial \xi_i}{\partial y_i}$ pode ser escrito como consta na Equação 2.19.

$$\frac{\partial \xi_i}{\partial y_i} = \frac{\partial[\frac{1}{2}(y_i - d_i)^2]}{\partial y_i} = 2 \cdot \frac{1}{2}(y_i - d_i) = (y_i - d_i) \tag{2.19}$$

[9] Esse é o método mais utilizado.

O termo z_i da Equação 2.18 é a função de ativação, que nesse caso é y_i. Como $z_i = y_i$, o termo $\frac{\partial y_i}{\partial z_i}$ pode ser escrito como consta na Equação 2.20.

$$\frac{\partial y_i}{\partial z_i} = 1 \tag{2.20}$$

Mas $y_i = w_i \cdot x_i$[10], e como $z_i = y_i$, então o termo $\frac{\partial z_i}{\partial w}$ da Equação 2.20 pode ser escrito como visto na Equação 2.21.

$$\frac{\partial z_i}{\partial w} = \frac{\partial (x_i \cdot w)}{\partial w} = x_i \tag{2.21}$$

Assim, a função de atualização completa Δw para a amostra i será dada pela Equação 2.22.

$$\Delta w = -\eta \frac{\partial \xi_i}{\partial w} = -\eta (y_i - d_i) x_i \tag{2.22}$$

Como a operação de ajuste é feita após a passagem de todos os x_i do minilote, e como a média dos erros possui uma resposta mais adequada aos nossos objetivos, assumindo que N é o número de amostras no minilote, podemos reescrever Δw como é visto na Equação 2.23.

$$\Delta w = -\frac{\eta}{N} \sum_{i=1}^{N} (y_i - d_i) x_i \tag{2.23}$$

No algoritmo de *backpropagation*, η é classificado como um hiperparâmetro, ajustando a velocidade da descida do gradiente. Escolher uma taxa de aprendizado correta nem sempre é uma tarefa fácil, pois taxas de aprendizado muito pequenas fazem com que o tempo para a convergência da rede demore muito, e taxas muito altas podem fazer o modelo divergir.

[10] Como na Equação 2.12, só que lá y_i é chamado de v.

3 *Deep Learning*

No Capítulo 2, vimos como o *backpropagation* e o gradiente descendente podem otimizar o ajuste dos pesos da rede de múltiplos perceptrons. Esse modelo de rede neural é a base para a construção de redes neurais com muitas[1] camadas ocultas, construídas com um aspecto bastante parecido com o que é visto na Figura 6. Redes Neurais com muitas camadas ocultas são a base do que se chama em IA de ***deep learning***[2].

Há várias aplicações e arquiteturas de redes neurais profundas, conforme relata (LECUN et al., 1989). Porém, vamos focar no reconhecimento de imagens por *deep learning*. A seguir, trataremos então dos aspectos fundamentais das chamadas Redes Neurais Convolucionais (sigla CNN em inglês), que é o modelo tipicamente utilizado em análise de imagens. Feito isso, mostraremos as principais características dessas redes e como elas aperfeiçoam o trabalho de reconhecimento de imagens.

3.1 Gradiente Descendente Estocástico

Neste trabalho foram utilizadas cinco bases de dados contendo texturas visuais[3]. As bases de dados foram divididas em dados para treino e validação [4]. Os dados de treino foram divididos minilotes de imagens escolhidas de forma aleatória. Essa aleatoriedade dá ao processo de otimização dos pesos a característica estocática que difere do processo que narramos no desenvolvimento da Equação 2.23.

A essa otimização dos pesos por gradiente descentente utilizando-se minilotes de dados escolhidos de forma randômica, dá-se o nome de **Método do Gradiente Descendente Estocástico** (SGD).

Em (BOTTOU; BOUSQUET, 2007) está provado que o SGD, quando utilizado para otimização dos pesos de uma rede neural que é treinada com poucas amostras[5], apresenta melhor desempenho de generalização quando comparado ao método do gradiente descendente padrão.

---

[1] Muitas aqui podem ser dezenas ou centenas de camadas ocultas.

[2] Vamos nos referir ao aprendizado profundo por ***deep learning***, pois é o termo mais usado na literatura.

[3] O Capítulo 5 tratará das texturas visuais que utilizamos.

[4] O aprendizado supervisionado precisa ser validado. Por isso, as amostras de dados são divididas em dados de treino e validação.

[5] Nós utilizamos bancos de dados com no máximo 5000 imagens, que é considerado pouco comparado por exemplo à base pública ImageNet, que possui milhões de imagens.

3.2 Entropia Cruzada

Vimos que o SGD, por *backpropagation*, ajusta os pesos e viéses da RNA a fim de minimizar a função de custo ξ. Em (SHEPHERD, 1997) consta que a função de custo do tipo **entropia cruzada** aumenta a velocidade de convergência da rede neural para a solução ótima de classificação, se comparada à função de custo descrita na Equação 2.5; essa foi a função de custo usada nesse trabalho e sua expressão matemática consta na Equação 3.1, em que para cada amostra i do minilote, com N amostras, temos uma saída real y_i da rede e uma saída d_i desejável.

$$\xi = -\sum_{i=1}^{N} [d_i \ln y_i + (1 - d_i) \ln(1 - y_i)] \tag{3.1}$$

3.3 Classificador SoftMax

Na última camada da CNN que utilizamos para a classificação das texturas visuais, usa-se uma função de ativação chamada **softmax**. Essa função produzirá um conjunto de valores de probabilidades da textura pertencer a uma das classes em questão.

Por exemplo, se forem gerados dez valores de probabilidade de p_0 a p_9, teremos em p_0 a probabilidade para a classe 0, em p_1 a probabilidade para a classe 1, e assim sucessivamente até p_9 que é a probabilidade para a classe 9.

Se p_0 for maior que todas as outras probabilidades, então a imagem é classificada como do tipo 0, se p_1 for maior que todas as outras probabilidades a imagem é classificada como do tipo 1, e assim por diante. A Equação 3.2 é a expressão matemática que representa a função **softmax**, onde $i = 1, ..., k$ e k é o número de classes; e y_i é o valor numérico fornecido pela i-ésima função de ativação ReLU[6] na penúltima camada da rede neural.

$$\sigma(y_i) = \frac{e^{y_i}}{\sum_{i=1}^{k} e^{y_k}}, \tag{3.2}$$

3.4 CNN

Quando aplicadas à classificação de imagens, as **Redes Neurais Convolucionais** (CNN) se diferenciam das demais redes neurais profundas, principalmente por operarem diretamente na imagem enquanto que os outros tipos de redes extraem as características da imagem, por exemplo, utilizando filtros de Fourier (SCHMIDHUBER, 2015).

As CNNs são projetadas para receberem em sua entrada dados dos *pixels* das imagens, disponíveis em várias matrizes com dimensão variável. Na Figura 7, vemos um

[6] A função ReLU foi abordada na Seção 2.1.1.

diagrama ilustrativo contendo algumas das características da CNN que analisaremos a seguir, como os mapas de recursos (*features*) e as camadas de *pooling*.

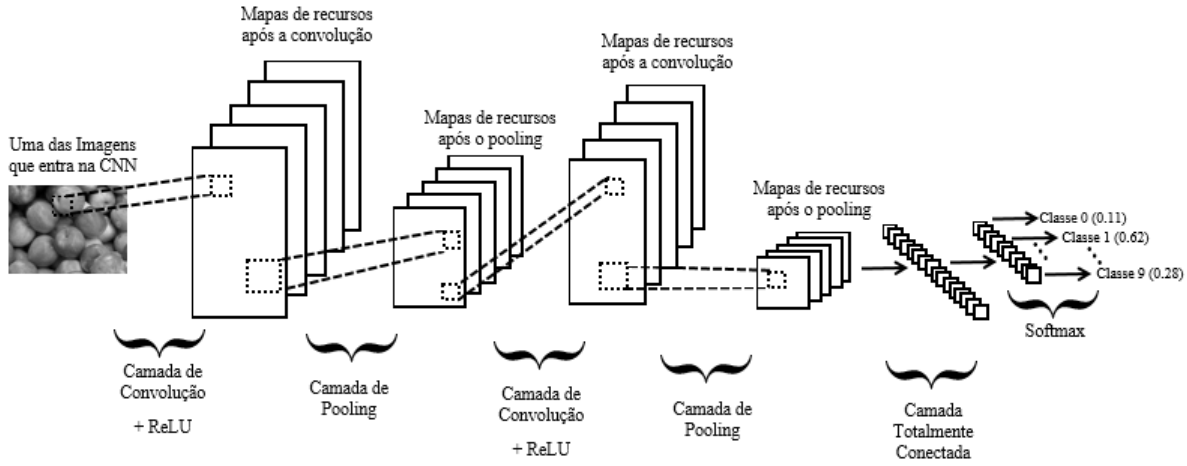


Figura 7 – CNN.

A convolução opera com diversos neurônios, e cada um deles é responsável por aplicar um **filtro** em uma região específica da imagem. Podemos imaginar cada neurônio como sendo conectado a um conjunto de valores gerados na camada anterior e a cada uma dessas conexões se atribui um peso. A fim de que um conjunto de neurônios aplique um mesmo filtro em diferentes posições da imagem, é realizado um compartilhamento de pesos durante o processo de treinamento da rede e dentro de cada camada de convolução.

Em se tratando do processo de classificação de imagens, esse filtro também é chamado de *kernel*, que é uma matriz a ser multiplicada pelos fragmentos dos valores da matriz de dados da imagem. A fim de fornecer uma análise mais precisa da imagem a ser analisada pela CNN, aplica-se o que é comumente chamado de *padding* na imagem, ou seja, adicionam-se pixels com valores (tipicamente nulos) à borda da imagem para que todos os pixels da imagem em questão possam ser processados um mesmo número de vezes. Sem esse preenchimento, os pixels das bordas da imagem passariam pelo *kernel* um menor número de vezes que os pixels afastados das bordas.

Resumindo a convolução: uma parte dos pixels da imagem entra em um neurônio, esses pixels são multiplicados pelo *kernel*, o resultado dessa multiplicação passa pela função de ativação ReLU e seu resultado é armazenado em matrizes comumente chamadas de **mapas de recursos (feature maps)**, conforme vemos na Figura 7.

Outra característica importante no processo de convolução é o *stride*, que define quantos pixels serão pulados em cada janela da imagem a ser convolucionada com o *kernel* da rede.

Uma vez gerados os *mapas de recursos* pela camada de convolução, entra em funcionamento a camada de *pooling*, que tem o objetivo de reduzir a dimensionalidade

desses mapas, gerando mapas menores, diminuindo assim a quantidade de dados a ser analisada pela rede e, consequentemente, diminuindo o tempo de aprendizado.

Na camada de *pooling* de nosso trabalho, o agrupamento do cojunto de dados ocorre em janelas com matrizes 3×3, com um *stride* igual a 2, e uma função de máximo, também chamada ***max pooling***, que verifica qual é o maior valor desse conjunto de dados que representará aquela região analisada, salvando esse valor em um mapa de recursos que servirá de entrada para a próxima camada de convolução.

Após várias camadas de convolução e *pooling*, um conjunto de neurônios totalmente conectados [7] formará a última camada da rede e suas saídas serão transmitidas à função ***softmax*** para que seja produzida a probabilidade da imagem ser classificada em um dos tipos possíveis para aquele conjunto de amostras.

Cabe salientar que a CNN é uma rede de perceptrons e os neurônios aprendem os pesos e *biases* de forma muito similar à MLP que analisamos na Seção 2.4, utilizando o *backpropagation* para otimizar os pesos.

Existem porém diferenças entre uma MLP e uma CNN, como vimos aqui, principalmente no que tange à forma como os neurônios recebem os dados a serem processados e como seus resultados trafegam na rede.

Em nosso trabalho, a CNN foi baseada na arquitetura ResNet-18. A convolução usa matrizes $7 \times 7 \times 3$ como *kernel*[8], *stride* igual a 2 e *padding* igual a 3, ou seja, são acrescentadas três camadas de pixels nulos em volta das imagens a serem processadas. Foram utilizadas cinco camadas de neurônios, com 64 neurônios na primeira camada, 128 na segunda, 256 na terceira e 512 na quarta. A quinta camada é totalmente conectada à quarta e também possui 512 neurônios.

3.5 *Transfer Learning*

Um dos principais problemas que muitos pesquisadores de RNA sempre enfrentaram foi o de não possuir um grande número [9] de exemplos rotulados (pensando em aprendizagem supervisionada), capaz de fazer a rede aprender com uma grande acurácia para, por exemplo, classificar imagens de um determinado tipo específico.

[7] Nas camadas de convolução, os neurônios são conectados em paralelo às saídas dos neurônios das camadas anteriores, não sendo portanto uma conexão de todos com todos, por isso, não são *totalmente conectados*. Porém, na última camada, todas as saídas passam por todos os neurônios, então, podemos dizer que são *totalmente conectados*.

[8] As três dimensões do *kernel* devem-se ao fato de as imagens aqui analisadas serem representadas no padrão de cores RGB, sendo então que os canais de cores são empilhados em uma estrutura tridimensional conhecida como *tensor* no jargão da área.

[9] A ImageNet, que contém milhões de imagens com 1000 categorias, é uma grande base de dados. Ainda assim, está obviamente muito longe de contemplar todos os tipos de imagens possíveis em nosso universo.

Visando mitigar esse problema, em muitas aplicações utiliza-se o procedimento chamado de *transfer learning* (transferência de aprendizado), principalmente quando se tem uma quantidade de exemplos pequena [10] para se treinar a rede.

Em resumo, pensando em classificação de imagens, uma CNN aprende em suas primeiras camadas a reconhecer bordas e contornos, entre outras características básicas presentes em qualquer imagem. Já as camadas mais próximas da saída aprendem a reconhecer detalhes dos objetos em questão.

Então, transferir o aprendizado (no caso os pesos a serem utilizados pelos neurônios) reduz muito o trabalho de treinamento da rede, pois os maiores ajustes serão feitos apenas nas últimas camadas.

Existem várias redes pré-treinadas que podem ser utilizadas para *transfer learning*. Em nosso trabalho, utilizamos a ResNet (HE et al., 2016a), por isso, faremos uma breve abordagem de algumas de suas características neste momento.

ResNet é um acrônimo para Rede Neural Residual, que possui uma característica excencial, os saltos em pares de grupos de camadas da RNA. De forma simplificada, os saltos da ResNet visam evitar que a rede, por ser profunda, seja afetada pelo evanescimento do gradiente (*gradient vanishing*) no empilhamento de mapeamentos de identidade[11]. Com essa estratégia, é possível treinar centenas ou até milhares de camadas, atingindo um excelente desempenho.

Pensando na degradação do gradiente, nas redes mais rasas esse problema praticamente inexiste, fazendo com que essas redes tenham um desempenho melhor que as redes mais profundas. Então, se houver um salto estratégico em algumas camadas das redes profundas, a precisão dessas redes tende a melhorar. Este salto ocorre pulando o treinamento de algumas camadas nas chamadas *conexões de salto*.

[10] Esse é o caso do nosso trabalho, conforme já relatamos na Seção 3.4

[11] Os mapeamentos de identidade são camadas da rede que não acrescentam em nada no aprendizado da rede, conforme relatam os criadores da ResNet.

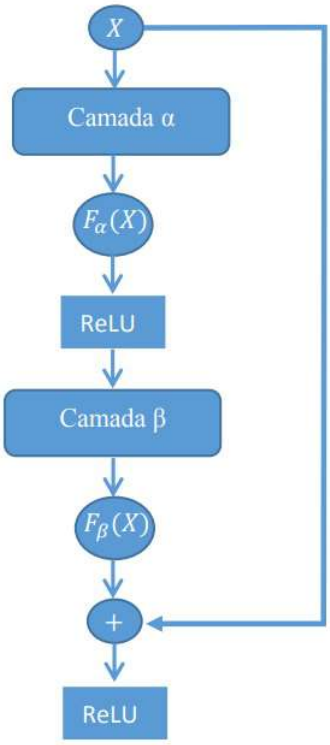


Figura 8 – Bloco Residual da ResNet.

Resumindo o processo de aprendizado residual (Figura 8): consideremos um bloco α da CNN cuja entrada é X e o resultado do processamento de X é $F_\alpha(X)$ que alimenta um bloco β, o resultado do processamento de $F_\alpha(X)$ é $F_\beta(X)$.

Antes de $F_\beta(X)$ ser passado para o próximo bloco ocorre o processo residual, pois $F_\beta(X)$ é somado à X. Se os blocos α e β tiverem desvanecido por conta da degradação do gradiente, o resíduo continuará o caminho pela rede produzindo resultados em camadas superiores.

Para o pré-treinamento da CNN desse trabalho foi utilizada a ResNet-18, que possui 18 camadas de convolução e o processo de aprendizado residual, conforme abordamos aqui, além das especificidades detalhadas em (HE et al., 2016a). Escolhemos a ResNet-18 por apresentar bom desempenho de transferência de aprendizado em comparação a outras redes utilizadas para esse fim, conforme consta em (HE et al., 2016b).

É importante ressaltar que utilizamos o pré-treinamento da ResNet-18 através da biblioteca de aprendizado de máquina *PyTorch*, que exige a normalização das imagens de entrada, com minilotes de imagens RGB de três canais na forma (3 x H x W). Nós utilizamos o padrão recomendado pelo *PyTorch* [12] para a normalização, ou seja, usando um vetor de média mean = [0.485, 0.456, 0.406] e de desvio std = [0.229, 0.224, 0.225].

[12] Os detalhes da ResNet-18 no PyTorch encontra-se em <https://pytorch.org/hub/pytorch_vision_resnet/>, último acesso em 02 de setembro de 2020.

4 Metodologia Proposta

Propomos neste trabalho duas metodologias para classificar as texturas visuais[1], ou seja, o uso direto de descritores extraídos por uma rede convolucional e a aplicação destes descritores a um *ensemble* de classificadores. Pretende-se aplicar e confrontar as duas metodologias, observando suas vantagens e desvantagens.

4.1 Método 1: CNN

O primeiro método utiliza uma CNN[2], baseada na arquitetura ResNet-18[3], utilizando a abordagem conexista com aprendizado supervisionado [4] e transferência de aprendizado a partir da base de dados ImageNet.

A CNN passou por 50 épocas de treinamento e validação, valor este determinado empiricamente. A título de ilustração, a Figura 9 a seguir mostra graficamente o aprendizado da CNN em uma das bases de dados aqui usadas (KTH-TIPS-2b) ao longo das 50 épocas[5].

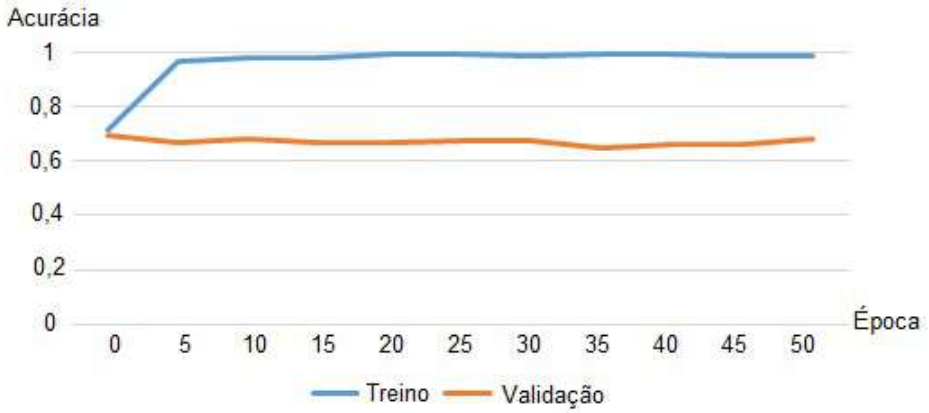


Figura 9 – Desempenho da CNN na Base KTH-TIPS-2b.

No Capítulo 5 está descrito como cada base de dados foi separada em dados para treino e validação. Feita essa separação, a CNN realizou então suas classificações das texturas visuais após passar pelas 50 épocas de treinamento. Da época de treinamento que obteve a melhor média de acurácias na classificação das texturas, duas atividades foram realizadas.

[1] O Capítulo 5 tratará das texturas visuais que utilizamos.
[2] As características da CNN que utilizamos estão descritas nos Tópicos 3.1, 3.2, 3.3 e 3.4.
[3] A transferência de aprendizado e a ResNet-18 foram abordadas na Seção 3.5.
[4] Descrevemos a abordagem conexista e o aprendizado supervisionado na Seção 1.2.
[5] Podemos notar nesse tipo de gráfico como a CNN consegue estabilizar seu aprendizado em 50 épocas de treinamento.

Na primeira, a média de acurácias e seu respectivo desvio padrão foi armazenado para um confronto de desempenho com o segundo método que iremos descrever a seguir.

Já na segunda, os vetores de descritores[6] que foram gerados antes das classificações em si (saída da rede) foram armazenados para serem entregues ao *ensemble* de classificadores.

4.2 Método 2: *Ensemble* de Classificadores

O segundo método utiliza um *ensemble* contendo nove classificadores: AdaBoost (AB), árvores de decisão (AD), floresta aleatória (FA), K-vizinhos mais próximos (KNN), *multilayer perceptron* (MLP), Naive Bayes (NB), *Support Vector Classification* (SVC), regressão linear (RLin) e regressão logística (RLog).

Os classificadores são algoritmos de aprendizado de máquina já consolidados em outros trabalhos na classificação de imagens. Escolhemos os nove classificadores que listamos anteriormente por já possuirem um histórico de bons resultados em classificação de imagens, como pode ser visto nos seguintes trabalhos: (ARAUJO, 2010), (LUZ; SANTOS; ANTUNES, 2009), (MENEZES et al., 2017), (BISNETO, 2011), (ALBOY, 2019), (BITTENCOURT; CLARKE, 2001), (DAMASCENO, 2017).

Doravante, quando utilizarmos o termo "classificador", estaremos nos referenciando a um algoritmo de aprendizado de máquina que foi implementado na linguagem *Python* [7] e que contém um classificador em sua estrutura. Chamar esses algoritmos apenas de classificadores tem o objetivo de facilitar o entendimento, a referência escrita e a compatibilidade com outros trabalhos que tratam de *emsemble* de classificadores.

Este método está dividido nos seguintes submétodos: classificadores sem hiperparâmetros ajustados, classificadores com hiperparâmetros ajustados, votação, mistura e empilhamento. A mistura e o empilhamento estão ainda subdivididos em: mistura total, mistura seletiva, empilhamento total e empilhamento seletivo. A mistura total e o empilhamento total seguem os procedimentos já consolidados em trabalhos similares. A mistura seletiva e o empilhamento seletivo são propostas novas apresentadas neste trabalho. Nas próximas seções, vamos nos referir às subdivisões do método *ensemble* de classificadores como métodos individuais, para facilitar a leitura do texto.

Como os classificadores são algoritmos de aprendizado de máquina, obviamente,

[6] Foram armazenados em estruturas de dados distintas e independentes os vetores de descritores tanto da etapa de treino quanto da etapa de validação da CNN.

[7] Utilizamos a biblioteca *scikit-learn* para essa implementação, portanto todas as definições dos classificadores que faremos a seguir, estão baseadas nessa biblioteca. "O scikit-learn é uma biblioteca de aprendizado de máquina de código aberto que oferece suporte ao aprendizado supervisionado e não supervisionado", definição que consta no site <https://scikit-learn.org/stable/getting_started.html>, última visita em 02 de setembro de 2020.

precisam aprender e, para tanto, precisam ser treinados e seu treinamento precisa ser validado. Para que o treinamento e a validação funcionem, é necessário um pré-processamento das texturas visuais antes de passarem pelos classificadores. Para essa tarefa o *ensemble* contou com a CNN, que faz o pré-processamento e entrega os vetores de atributos prontos para os classificadores.

4.2.1 Classificadores sem hiperparâmetros ajustados

Quando utilizamos os classificadores com os vetores de descritores sem hiperparâmetros ajustados, significa que estamos utilizando os algoritmos desses classificadores no modo *default* em que eles são apresentados na biblioteca *scikit-learn*.

4.2.2 Classificadores com hiperparâmetros ajustados

Após os classificadores operarem sem hiperparâmetros ajustados, utilizamos rotinas em Python que avaliam o melhor desempenho de cada classificador trabalhando com hiperparâmetros ajustados dentro de valores escolhidos para essa verificação de desempenho.

Feita essa avaliação, o código informa qual a melhor acurácia obtida pelos classificadores após os ajustes realizados em seus hiperparâmetros. A única excessão ao ajuste de hiperparâmetros é o classificador NB, que não possui hiperparâmetros a serem ajustados.

Na Tabela 2 mostraremos quais foram os hiperparâmetros ajustados em cada classificador.

Tabela 2 – Hiperparâmetros ajustados nos classificadores.

Classificador	Hiperparâmetro	Aplicação do hiperparâmetro
RLOG	random_state	Controla a forma como a randomização dos dados ocorre no classificador.
	max_iter	Determina o número máximo de iterações a serem realizadas.
KNN	n_neighbors	Estabelece o número de vizinhos a serem utilizados.
FA	n_estimators	Determina o número de árvores na floresta.
MLP	alpha	Hiperparâmetro de regularização com penalidade L2.
	max_iter	Determina o número máximo de iterações a serem realizadas.

Classificador	Hiperparâmetro	Aplicação do hiperparâmetro
AB	n_estimators	Estabelece o número máximo de estimadores para que o estímulo seja encerrado. Em caso de ajuste perfeito o processo de aprendizagem é interrompido precocemente.
SVC	*kernel*	Especifica o tipo de *kernel* a ser usado no algoritmo.
	C	Hiperparâmetro de regularização, com penalidade L2 quadrática.
AD	max_deph	Determina a profundidade máxima da árvore.
RLIN	normalize	Ajusta a normalização dos regressores.

4.2.3 Votação

Após serem treinados com os vetores de atributos de treino, que são retirados da última camada da CNN, a classificação final é validada com os vetores de atributos de validação, também extraídos da última camada da CNN.

Essa validação permite que conheçamos a classe fornecida pela saída de cada classificador do *ensemble* para cada amostra de validação. Sendo assim, na abordagem conhecida por votação, a classe final será aquela que foi atribuída pelo maior número de classificadores.

4.2.4 Empilhamento

Com o classificador devidamente treinado e validado, com seus hiperparâmetros ajustados para a situação mais próxima possível do ótimo, se armazenarmos em um vetor $\mathbf{X}$ as previsões desse classificador com os dados de treino, e também armazenarmos em um vetor $\mathbf{Y}$ as previsões desse classificador ao analisar os dados de validação, é possível construir uma nova base de dados de treino e de validação.

Essa é a ideia básica do empilhamento, formar um conjunto de treinamento e validação, a partir das previsões dos classificadores, para novamente treinar e validar os classificadores a partir desses dados agrupados. Em nosso trabalho foram utilizados 9 classificadores no *ensemble*, mas, para simplificar, vamos exemplificar utilizando apenas três desses classificadores: KNN, SVC e AdaBoost.

Vamos ao exemplo: KNN é treinado e validado utilizando os vetores de *features* extraídos da CNN; estando com seus hiperparâmetros devidamente ajustados, KNN é novamente exposto aos dados de treino e suas previsões para esse conjunto de dados são armazenadas, por exemplo, no vetor **XKNN**, bem como suas previsões para o cojunto de validação são armazenadas no vetor **YKNN**. Feito isso, exatamente o mesmo processo é realizado com SVC e AdaBoost, gerando os vetores **XSVC**, **YSVC**, **XADA** e **YADA**, respectivamente.

O empilhamento (que em nosso trabalho chamamos de empilhamento total) é o procedimento de treinar e validar os classificadores com um conjunto de treino e validação, que em nosso exemplo chamaremos de **XTREINO** e **XVALIDAÇÃO**, em que **XTREINO** contém todas as previsões armazenadas em **XKNN**, **XSVC** e **XADA**, ou seja, em **XTREINO** ocorre o "empilhamento" dos dados de treino e, de forma análoga, **XVALIDAÇÃO** contém todas as previsões armazenadas em **YKNN**, **YSVC** e **YADA**.

O que chamamos de *empilhamento seletivo* possui uma pequena diferença em relação ao empilhamento total: em vez de armazenar todos os conjuntos de treinamento e validação de todos os classificadores do *ensemble*, no empilhamento seletivo nós armazenamos apenas os dados dos classificadores que obtiveram as melhores acurácias durante o processo de treinamento[8].

Em nosso trabalho, utilizamos os dados dos três "melhores" classificadores para formar o que no nosso exemplo chamamos de **XTREINO** e **XVALIDAÇÃO**. E não somente isso, mas os classificadores que participarão do treinamento e validação no empilhamento seletivo, serão aqueles que obtiveram os três melhores desempenhos de classificação.

Voltando ao nosso exemplo, imaginemos que queiramos armazenar os dados dos dois "melhores" classificadores. Digamos que, ao analisar suas melhores acurácias durante a fase de treino, os melhores classificadores tenham sido KNN e SVC.

Com isso, **XTREINO** e **XVALIDAÇÃO** conterão apenas os dados dos vetores **XKNN**, **XSVC**, **YKNN** e **YSVC**, respectivamente. Como AdaBoost teve um desempenho inferior, em comparação com KNN e SVC, os vetores **XADA** e **YADA** serão descartados, e apenas os classificadores KNN e SVC serão treinados e validados com **XTREINO** e **XVALIDAÇÃO**, respectivamente.

Essa ideia foi expandida para os 9 classificadores do *ensemble*, e apenas os três classificadores que obtiveram os melhores desempenhos na fase de treino contribuíram para o empilhamento dos dados.

[8] Em todos os métodos que utilizamos, existem as fases de treino e validação. A que estamos nos referindo neste momento é a fase de treino do método Empilhamento Total.

4.2.5 Mistura

A mistura (tanto a total quanto a seletiva) é um procedimento muito parecido com o empilhamento. Vamos utilizar o exemplo que fizemos quando tratamos do empilhamento para explicar a diferença entre mistura e empilhamento.

Na mistura, os vetores **XTREINO** e **XVALIDAÇÃO** conterão os dados dos vetores com as previsões dos classificadores e também os vetores de atributos fornecidos pela CNN que serviram de treinamento para os classificadores. Essa é a única diferença entre a *mistura* e o *empilhamento*.

5 Aplicações

> "A textura visual está intimamente relacionada à sensação tátil de uma superfície. Algumas palavras que usamos para descrever características das texturas visuais são retiradas da nossa experiência ao toque, por isso falamos em áspera, macia, lisa, firme, etc. Outras características referem-se principalmente ao seu significado visual: sem brilho, brilhante, opaca, transparente, metálica, etc." Leyser (2012, apud SCOTT, 1951)

As aplicações deste trabalho consistem em classificar texturas visuais contidas em cinco bancos de imagens que descreveremos a seguir. Quatro bancos de imagens contêm texturas visuais diversificadas, de elementos que encontramos com alguma facilidade no dia-a-dia, como por exemplo frutas, cascas de árvores ou prateleiras de um supermercado. Esses quatro banco de imagens servem para validar o objetivo teórico deste trabalho, mostrando que os métodos que utilizamos são válidos para a classificação de imagens de uso geral e em particular na aplicação médica.

A quinta base de dados merece um destaque especial pois as imagens são de lesões císticas que ocorrem na madíbula humana, e é o material que utilizamos para atingir o objetivo aplicado deste trabalho.

A seguir, faremos uma breve abordagem das principais características de cada um dos cinco bancos de imagens, porém, quando tratarmos da base de dados médicas, faremos um breve resumo do tipo de cisto ao qual as imagens se referem e do contexto médico envolvido.

5.1 Bancos de imagens diversificadas

5.1.1 Base de dados UIUC

Esta base de dados consiste de um conjunto de 1000 imagens não calibradas e não registradas de tamanho 640×480, contendo 25 texturas diferentes, e cada uma dessas texturas com 40 amostras. Está disponível no cojunto de imagens UIUC 2006 (LAZEBNIK; SCHMID; PONCE, 2005). A Figura 10 traz uma das texturas visuais da base de dados UIUC.

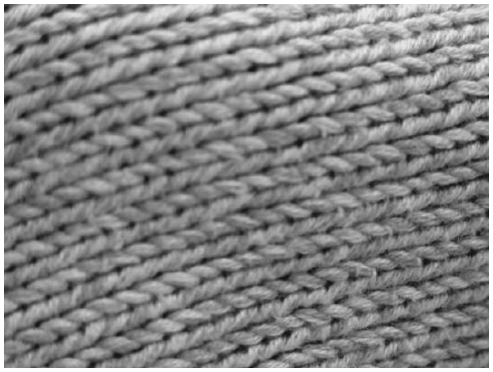


Figura 10 – Textura de UIUC.

5.1.2 Base de dados UMD

Esta base de dados é considerada de alta resolução, conforme descreve (XU; JI; FERMüLLER, 2009), sendo um conjunto de 1000 imagens não calibradas e não registradas de tamanho 1200 × 960, contendo 25 texturas diferentes, e com 40 amostras por textura. São disponibilizadas no cojunto de dados UMD 2006 (XU; JI; FERMüLLER, 2009). A Figura 11 traz uma das texturas visuais da base de dados UMD.

Figura 11 – Textura de UMD.

5.1.3 Base de dados KTH-TIPS-2b

Esta base de dados foi construído para complementar a base CUReT com variações de escala, conforme descreve (HAYMAN et al., 2004). Está dividida em quatro conjuntos de amostras, denominamos *samples* *a*, *b*, *c* e *d*. Cada conjunto contém 108 imagens, de tamanho 200 × 200, em cada uma das 11 texturas diferentes, totalizando 1188 imagens por conjunto. A Figura 12 traz uma das texturas visuais da base de dados KTH-TIPS-2b.

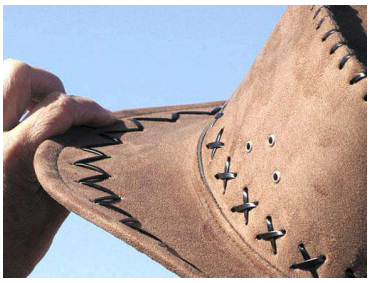
Figura 12 – Textura de KTH-TIPS-2b.

5.1.4 Base de dados FMD

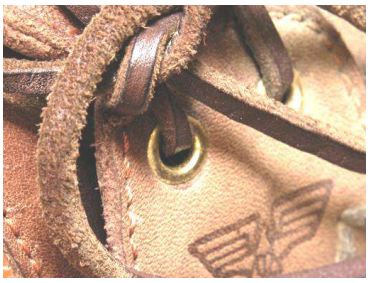
Dentre todas as bases de dados que utilizamos, essa é a mais heterogênea e complicada pois possui imagens agrupadas por similaridade de textura, mas com aspectos visuais bastante diferentes.

Contendo 1000 imagens de tamanho 512×384, divididas em 10 categorias, essa base foi criada originalmente para pesquisas médicas, com o intuito de verificar a velocidade de resposta do olho humano quando exposto a imagens sem um formato único, porém com texturas similares, conforme descreve (SHARAN; ROSENHOLTZ; ADELSON, 2009). A Figura 13 traz texturas visuais da base de dados FMD.

(a) Bola.



(b) Chapéu.



(c) Sapato.

Figura 13 – Texturas de FMD.

5.2 Imagens de Cistos Mandibulares

"Cistos são cavidades patológicas com conteúdo líquido ou semifluido e revestidas de tecido epitelial" Florindo, Bruno e Landini (2017, p.1).

Existem dois tipos importantes de cistos que surgem nas mandíbulas: o *cisto radicular* e o *queratocisto ondogênico* (OKC); os OKCs são ainda subdivididos em solitários (esporádicos) ou múltiplos (sincrônicos ou metacrônicos).

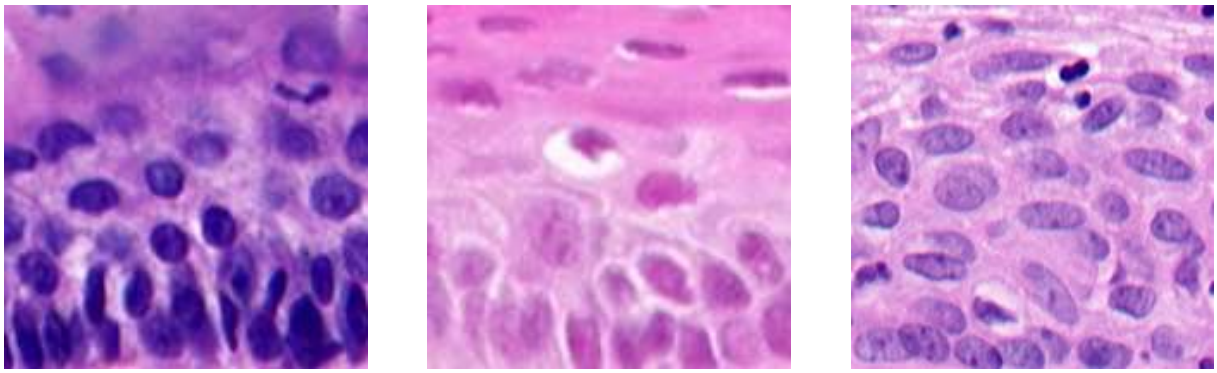
Os pesquisadores dos OKCs acreditam que este tipo de cisto benigno, mas localmente agressivo, se origina na lâmina dentária. A mucosa oral também desempenha importante papel na origem dos queratocistos, conforme (STOELINGA; PETERS, 1973).

Os cistos radiculares são mais comuns e "estão associados às raízes dos dentes com polpas não vitais, e apresentam crescimento lento" Florindo, Bruno e Landini (2017, p.1). Já os OKCs "não estão associados a doenças dentárias e têm algumas características comuns a neoplasias." Florindo, Bruno e Landini (2017, p.1)

5.2.1 Base de dados de cistos bucais

As imagens dos cistos madibulares que foram utilizadas neste trabalho são anônimas e foram classificadas por histolopatologistas, conforme descreve (FLORINDO; BRUNO; LANDINI, 2017).

A base de dados está dividida em três conjuntos de amostras: **K**, **S** e **R**; sendo **K** imagens dos OKCs esporádicos, **S** OKCs sidrômicos e **R** imagens dos cistos radiculares. Cada conjunto contém 1300 imagens de dimensões 179×179. A Figura 14 traz texturas visuais da base de dados CYST.



(a) OKC esporádico - tipo K. (b) OKC sindrômico - tipo S. (c) Cistos radiculares - tipo R.

Figura 14 – Texturas de cistos bucais.

6 Experimentos, Resultados Obtidos e Discussões

As bases de dados que vimos no Capítulo 5 foram utilizadas de formas diferentes e os algoritmos estudados rodaram sobre elas em quantidades de vezes diferentes, para se adaptar ao protocolo mais usado na literatura e assim viabilizar uma comparação quantitativa. Sendo assim, faremos abordagens distintas sobre esses experimentos que realizamos com as bases de imagens utilizadas.

6.1 Experimentos

Os experimentos que realizamos foram baseados na execução de um código escrito na liguagem Python e que seguiu basicamente a mesma estrutura, variando apenas as bases de dados a serem processadas[1].

Foram realizados 10 experimentos com cada uma dessas bases de dados, lembrando que a cada experimento as bases eram novamente e randomicamente organizadas em dados para treino e dados para validação.

6.1.1 Base de Dados KTH-TIPS-2b

Conforme descrevemos no Capítulo 5, essa base de dados está dividida em quatro conjuntos de amostras *a*, *b*, *c* e *d*. Usamos essa divisão das amostras considerando cada divisão como uma nova base de dados.

Na base de dados que denominamos KTH_A, as amostras do conjunto *a* foram utilizadas como dados de treino e as amostras dos conjuntos *b*, *c* e *d*, como dados de validação. Na base de dados denominado KTH_B, as amostras do conjunto *b* foram utilizadas como dados de treino e as amostras dos conjuntos *a*, *c* e *d* como dados de validação. Procedimento análogo foi realizado nos bancos de dados denominados KTH_C e KTH_D.

6.1.2 Bases de dados UIUC, UMD e FMD

As bases de dados UIUC, UMD e FMD foram aleatoriamente divididas em 50% das amostras para dados de treino e o restante para os dados de validação.

[1] A forma como as bases de dados foram utilizadas nos códigos será explicada nos tópicos a seguir.

6.1.3 Base de dados CYST

A base de dados CYST está dividida pelos tipos de cistos mandibulares que descrevemos no Capítulo 5, ou seja, amostras do tipo K, R e S.

Para essa base de dados fizemos dois tipos de experimentos diferentes. O primeiro usando os três tipos de amostras ao mesmo tempo. E o segundo usando apenas as amostras do tipo K e S, já que esta é a tarefa mais desafiadora nesta base (FLORINDO; BRUNO; LANDINI, 2017).

Por isso, consideramos a base de dados CYST como aquela composta pelas amostras do tipo K, R e S e consideramos a base de dados CYST_KS como sendo aquela composta pelas amostras do tipo K e S.

Para ambos os experimentos, as amostras foram divididas aleatoriamente em 50% para treino e 50% para validação e o código foi rodado 10 vezes para cada tipo de experimento, sendo registrado de forma similar ao uso das bases UIUC, UMD e FMD.

6.2 Resultados Obtidos

A seguir mostraremos a média de acurácias, com seus respectivos desvios padrões, obtidos pela CNN e pelo *ensemble* de classificadores nos métodos que utilizamos, em cada base de dados. Ressalte-se que aqui o termo "acurácia" corresponde à proporção de imagens classificadas corretamente, de modo que uma acurácia com valor $1,0$ é o cenário ideal (e raramente atingido) de 100% de acerto.

Para os métodos que utilizamos, considere as seguintes abreviações:

- SHA: Sem Hiperparâmetros Ajustados
- CHA: Com Hiperparâmetros Ajustados
- EmpTot: Empilhamento Total
- EmpSel: Empilhamento Seletivo
- MisTot: Mistura Total
- MisSel: Mistura Seletiva
- Vot: Votação

6.2.1 Resultados obtidos na base CYST

Na base de dados CYST a CNN obteve média de acurácia de 0,994 com desvio padrão de 0,002. A Tabela 3 traz a média de acurácias do *ensemble* na base CYST.

Tabela 3 – Média de acurácias do *ensemble* na base CYST.

Método	Classificador	Média de acurácias
SHA	KNN	0,994±0,002
	SVC	0,994±0,002
	AD	0,954±0,008
	FA	0,960±0,007
	MLP	0,994±0,002
	AB	0,970±0,012
	NB	0,988±0,002
	RLOG	0,994±0,002
	RLIN	0,754±0,015
CHA	KNN	0,993±0,003
	SVC	0,993±0,002
	AD	0,952±0,012
	FA	0,979±0,035
	MLP	0,994±0,002
	AB	0,969±0,013
	NB	0,983±0,002
	RLOG	0,994±0,002
	RLIN	0,755±0,015
EmpTot	KNN	0,991±0,002
	SVC	0,990±0,002
	AD	0,988±0,016
	FA	0,993±0,001
	MLP	0,990±0,003
	AB	0,989±0,010
	NB	0,993±0,002
	RLOG	0,990±0,003
	RLIN	0,965±0,015
EmpSel	KNN	0,497±0,002
	SVC	0,397±0,002
	AD	0,895±0,002
	FA	0,895±0,002
	MLP	0,496±0,002
	AB	0,696±0,002

Método	Classificador	Média de acurácias
	NB	0,894±0,002
	RLOG	0,397±0,001
	RLIN	–
MisTot	KNN	0,994±0,001
	SVC	0,994±0,001
	AD	0,994±0,002
	FA	0,981±0,004
	MLP	0,995±0,002
	AB	0,994±0,002
	NB	0,994±0,002
	RLOG	0,995±0,002
	RLIN	0,972±0,010
MisSel	KNN	0,895±0,001
	SVC	0,895±0,002
	AD	0,795±0,002
	FA	0,098±0,001
	MLP	0,995±0,002
	AB	0,895±0,002
	NB	0,597±0,002
	RLOG	0,995±0,002
	RLIN	–
Vot		0,992±0,002

Na Tabela 3, o classificador RLIN aparece sem acurácia nos métodos EmpSel e MisSel. Isso quer dizer que não houve, em nenhum experimento, qualquer valor de acurácia para esse classificador nesses métodos. Isto se deve ao fato de o empilhamento seletivo utilizar no treinamento e, consequentemente, na validação da classificação das imagens, somente os classificadores que obtiveram as três melhores acurácias de treino no método EmpTot.

Diante disso, a Tabela 3 está nos informando que RLIN não obteve nos métodos EmpSel e MisSel, em nenhum experimento com a base CYST, uma das três melhores acurácias dentre todas as acurácias do *ensemble* nos métodos EmpTot e MisTot, e por isso não participou do empilhamento seletivo e da mistura seletiva. Situações análogas serão reportadas nas tabelas que se seguem e devem ser interpretadas dessa mesma forma.

6.2.2 Resultados obtidos na base CYST_KS

Na base de dados CYST_KS a CNN obteve média de acurácia de 0,993 com desvio padrão de 0,002. A Tabela 4 traz a média de acurácias do *ensemble* na base CYST_KS.

Tabela 4 – Média de acurácias do *ensemble* na base CYST_KS.

Método	Classificador	Média de acurácias
SHA	KNN	0,985±0,004
	SVC	0,987±0,002
	AD	0,958±0,009
	FA	0,966±0,010
	MLP	0,988±0,003
	AB	0,982±0,004
	NB	0,968±0,009
	RLOG	0,988±0,003
	RLIN	0,390±0,097
CHA	KNN	0,987±0,005
	SVC	0,986±0,003
	AD	0,959±0,010
	FA	0,974±0,010
	MLP	0,988±0,002
	AB	0,985±0,005
	NB	0,968±0,010
	RLOG	0,988±0,003
	RLIN	0,391±0,097
EmpTot	KNN	0,988±0,002
	SVC	0,988±0,002
	AD	0,974±0,032
	FA	0,987±0,002
	MLP	0,988±0,002
	AB	0,976±0,021
	NB	0,988±0,003
	RLOG	0,989±0,003
	RLIN	0,953±0,011
EmpSel	KNN	0,890±0,003
	SVC	0,890±0,002
	AD	0,592±0,003
	FA	0,889±0,002
	MLP	0,989±0,002

Método	Classificador	Média de acurácias
	AB	0,789±0,004
	NB	0,890±0,003
	RLOG	0,890±0,002
	RLIN	0,192±0,003
MisTot	KNN	0,989±0,004
	SVC	0,989±0,003
	AD	0,987±0,004
	FA	0,976±0,010
	MLP	0,988±0,003
	AB	0,988±0,002
	NB	0,988±0,003
	RLOG	0,988±0,003
	RLIN	0,958±0,012
MisSel	KNN	0,988±0,003
	SVC	0,988±0,002
	AD	0,791±0,003
	FA	0,490±0,009
	MLP	0,889±0,003
	AB	0,889±0,003
	NB	0,889±0,002
	RLOG	0,988±0,002
	RLIN	–
Vot		0,988±0,004

6.2.3 Resultados obtidos na base FMD

Na base de dados FMD a CNN obteve média de acurácia de 0,838 com desvio padrão de 0,010. A Tabela 5 traz a média de acurácias do *ensemble* na base FMD.

Tabela 5 – Média de acurácias do *ensemble* na base FMD.

Método	Classificador	Média de acurácias
SHA	KNN	0,772±0,026
	SVC	0,825±0,019
	AD	0,425±0,046
	FA	0,540±0,018
	MLP	0,827±0,021
	AB	0,257±0,074

Método	Classificador	Média de acurácias
	NB	0,823±0,015
	RLOG	0,825±0,014
	RLIN	0,557±0,141
CHA	KNN	0,814±0,022
	SVC	0,813±0,018
	AD	0,541±0,030
	FA	0,770±0,032
	MLP	0,823±0,021
	AB	0,262±0,066
	NB	0,823±0,015
	RLOG	0,825±0,014
	RLIN	0,561±0,141
EmpTot	KNN	0,426±0,029
	SVC	0,444±0,027
	AD	0,628±0,103
	FA	0,797±0,027
	MLP	0,485±0,046
	AB	0,198±0,035
	NB	0,432±0,026
	RLOG	0,512±0,042
	RLIN	0,652±0,036
EmpSel	KNN	0,079±0,021
	SVC	–
	AD	0,735±0,014
	FA	0,824±0,014
	MLP	–
	AB	–
	NB	–
	RLOG	0,159±0,013
	RLIN	–
MisTot	KNN	0,826±0,024
	SVC	0,829±0,011
	AD	0,819±0,014
	FA	0,772±0,031
	MLP	0,829±0,017
	AB	0,275±0,035

Método	Classificador	Média de acurácias
	NB	0,811±0,015
	RLOG	0,826±0,013
	RLIN	0,638±0,056
MisSel	KNN	0,662±0,014
	SVC	0,829±0,011
	AD	0,328±0,014
	FA	–
	MLP	0,828±0,018
	AB	–
	NB	0,164±0,009
	RLOG	0,825±0,013
	RLIN	–
Vot		0,829±0,015

6.2.4 Resultados obtidos na base UMD

Na base de dados UMD a CNN obteve média de acurácia de 0,998 com desvio padrão de 0,002. A Tabela 6 traz a média de acurácias do *ensemble* na base UMD. A Tabela 6 traz a média de acurácias do *ensemble* na base UMD.

Tabela 6 – Média de acurácias do *ensemble* na base UMD.

Método	Classificador	Média de acurácias
SHA	KNN	0,992±0,005
	SVC	0,996±0,002
	AD	0,232±0,038
	FA	0,876±0,029
	MLP	0,995±0,003
	AB	0,131±0,046
	NB	0,968±0,007
	RLOG	0,996±0,003
	RLIN	0,809±0,029
CHA	KNN	0,996±0,002
	SVC	0,996±0,002
	AD	0,662±0,034
	FA	0,993±0,004
	MLP	0,995±0,004
	AB	0,140±0,030
	NB	0,968±0,008

Método	Classificador	Média de acurácias
	RLOG	0,996±0,030
	RLIN	0,809±0,029
EmpTot	KNN	0,797±0,036
	SVC	0,794±0,033
	AD	0,800±0,084
	FA	0,928±0,048
	MLP	0,731±0,054
	AB	0,512±0,073
	NB	0,787±0,032
	RLOG	0,741±0,046
	RLIN	0,995±0,004
EmpSel	KNN	0,291±0,009
	SVC	0,194±0,001
	AD	0,590±0,010
	FA	0,972±0,031
	MLP	–
	AB	–
	NB	–
	RLOG	–
	RLIN	0,696±0,004
MisTot	KNN	0,997±0,003
	SVC	0,997±0,002
	AD	0,820±0,084
	FA	0,991±0,004
	MLP	0,996±0,004
	AB	0,463±0,183
	NB	0,970±0,011
	RLOG	0,997±0,003
	RLIN	0,995±0,004
MisSel	KNN	0,897±0,003
	SVC	0,897±0,002
	AD	–
	FA	0,497±0,003
	MLP	0,994±0,003
	AB	–
	NB	0,196±0,009
	RLOG	0,996±0,003

Método	Classificador	Média de acurácias
	RLIN	–
Vot		0,996±0,003

6.2.5 Resultados obtidos na base UIUC

Na base de dados UIUC a CNN obteve média de acurácia de 0,991 com desvio padrão de 0,004. A Tabela 7 traz a média de acurácias do *ensemble* na base UIUC.

Tabela 7 – Média de acurácias do *ensemble* na base UIUC.

Método	Classificador	Média de acurácias
SHA	KNN	0,982±0,010
	SVC	0,991±0,007
	AD	0,264±0,040
	FA	0,774±0,026
	MLP	0,993±0,006
	AB	0,127±0,024
	NB	0,942±0,012
	RLOG	0,992±0,007
	RLIN	0,599±0,066
CHA	KNN	0,989±0,009
	SVC	0,991±0,007
	AD	0,647±0,069
	FA	0,972±0,015
	MLP	0,989±0,007
	AB	0,129±0,021
	NB	0,941±0,012
	RLOG	0,992±0,007
	RLIN	0,597±0,066
EmpTot	KNN	0,639±0,036
	SVC	0,639±0,033
	AD	0,787±0,060
	FA	0,902±0,065
	MLP	0,584±0,051
	AB	0,487±0,134
	NB	0,641±0,040
	RLOG	0,622±0,043
	RLIN	0,985±0,009

Método	Classificador	Média de acurácias
EmpSel	KNN	0,095±0,022
	SVC	0,095±0,001
	AD	0,961±0,011
	FA	0,953±0,039
	MLP	–
	AB	–
	NB	–
	RLOG	0,074±0,001
	RLIN	0,099±0,001
MisTot	KNN	0,990±0,007
	SVC	0,990±0,008
	AD	0,880±0,043
	FA	0,971±0,018
	MLP	0,989±0,008
	AB	0,544±0,136
	NB	0,940±0,020
	RLOG	0,992±0,008
	RLIN	0,983±0,008
MisSel	KNN	0,891±0,008
	SVC	0,891±0,008
	AD	–
	FA	0,295±0,010
	MLP	0,893±0,009
	AB	–
	NB	–
	RLOG	0,992±0,008
	RLIN	–
Vot		0,917±0,006

6.2.6 Resultados obtidos em KTH_A

Nesta base de dados, a CNN obteve média de acurácia de 0,713 com desvio padrão de 0,015. A Tabela 8 traz a média de acurácias do *ensemble* na base KTH-TIPS-2b, quando as amostras “a” são usadas como treino.

Tabela 8 – Média de acurácias do *ensemble* na base KTH-TIPS-2b - Usando amostras "a" como treino.

Método	Classificador	Média de acurácias
SHA	KNN	0,677±0,026
	SVC	0,688±0,021
	AD	0,307±0,048
	FA	0,576±0,013
	MLP	0,703±0,018
	AB	0,127±0,041
	NB	0,672±0,020
	RLOG	0,703±0,017
	RLIN	0,333±0,092
CHA	KNN	0,690±0,024
	SVC	0,688±0,020
	AD	0,483±0,018
	FA	0,691±0,019
	MLP	0,701±0,021
	AB	0,118±0,035
	NB	0,671±0,020
	RLOG	0,703±0,017
	RLIN	0,333±0,009
EmpTot	KNN	0,560±0,025
	SVC	0,561±0,024
	AD	0,624±0,044
	FA	0,681±0,029
	MLP	0,533±0,033
	AB	0,251±0,036
	NB	0,551±0,026
	RLOG	0,529±0,022
	RLIN	0,310±0,747
EmpSel	KNN	–
	SVC	0,465±0,023
	AD	0,610±0,022
	FA	0,704±0,016
	MLP	0,064±0,001
	AB	–
	NB	0,254±0,030
	RLOG	–

Método	Classificador	Média de acurácias
	RLIN	–
MisTot	KNN	0,695±0,025
	SVC	0,694±0,022
	AD	0,658±0,043
	FA	0,690±0,024
	MLP	0,699±0,018
	AB	0,281±0,022
	NB	0,655±0,033
	RLOG	0,705±0,019
	RLIN	0,489±0,069
MisSel	KNN	0,423±0,009
	SVC	0,351±0,014
	AD	0,726±0,001
	FA	0,419±0,022
	MLP	0,628±0,018
	AB	–
	NB	–
	RLOG	0,705±0,019
	RLIN	–
Vot		0,703±0,018

6.2.7 Resultados obtidos em KTH_B

Nesta base de dados, a CNN obteve média de acurácia de 0,727 com desvio padrão de 0,016. A Tabela 9 traz a média de acurácias do *ensemble* na base KTH-TIPS-2b, quando as amostras "b" são usadas como treino.

Tabela 9 – Média de acurácias do *ensemble* na base KTH-TIPS-2b - Usando amostras "b" como treino.

Método	Classificador	Média de acurácias
SHA	KNN	0,708±0,028
	SVC	0,706±0,024
	AD	0,402±0,049
	FA	0,553±0,029
	MLP	0,725±0,026
	AB	0,141±0,024
	NB	0,664±0,031

Método	Classificador	Média de acurácias
	RLOG	0,719±0,027
	RLIN	0,308±0,083
CHA	KNN	0,708±0,021
	SVC	0,706±0,024
	AD	0,488±0,025
	FA	0,703±0,034
	MLP	0,720±0,027
	AB	0,148±0,030
	NB	0,664±0,031
	RLOG	0,720±0,026
	RLIN	0,308±0,083
EmpTot	KNN	0,547±0,038
	SVC	0,549±0,044
	AD	0,619±0,061
	FA	0,698±0,030
	MLP	0,542±0,044
	AB	0,256±0,061
	NB	0,548±0,053
	RLOG	0,529±0,032
	RLIN	0,435±0,075
EmpSel	KNN	–
	SVC	0,135±0,018
	AD	0,624±0,038
	FA	0,712±0,030
	MLP	0,198±0,031
	AB	–
	NB	0,276±0,039
	RLOG	0,122±0,011
	RLIN	–
MisTot	KNN	0,718±0,025
	SVC	0,717±0,029
	AD	0,667±0,039
	FA	0,712±0,037
	MLP	0,724±0,028
	AB	0,291±0,059
	NB	0,670±0,048
	RLOG	0,722±0,029

Método	Classificador	Média de acurácias
	RLIN	0,442±0,087
MisSel	KNN	0,578±0,022
	SVC	0,508±0,027
	AD	–
	FA	0,362±0,032
	MLP	0,431±0,033
	AB	–
	NB	0,078±0,001
	RLOG	0,585±0,025
	RLIN	–
Vot		0,724±0,027

6.2.8 Resultados obtidos em KTH_C

Nesta base de dados, a CNN obteve média de acurácia de 0,701 com desvio padrão de 0,016. A Tabela 10 traz a média de acurácias do *ensemble* na base KTH-TIPS-2b, quando as amostras "c" são usadas como treino.

Tabela 10 – Média de acurácias do *ensemble* na base KTH-TIPS-2b - Usando amostras "c" como treino.

Método	Classificador	Média de acurácias
SHA	KNN	0,684±0,020
	SVC	0,688±0,018
	AD	0,316±0,061
	FA	0,582±0,016
	MLP	0,693±0,015
	AB	0,129±0,035
	NB	0,669±0,015
	RLOG	0,692±0,015
	RLIN	0,408±0,065
CHA	KNN	0,688±0,021
	SVC	0,688±0,018
	AD	0,501±0,027
	FA	0,681±0,022
	MLP	0,692±0,015
	AB	0,133±0,035
	NB	0,669±0,015
	RLOG	0,692±0,015

Método	Classificador	Média de acurácias
	RLIN	0,408±0,065
EmpTot	KNN	0,576±0,027
	SVC	0,577±0,028
	AD	0,637±0,038
	FA	0,687±0,015
	MLP	0,571±0,033
	AB	0,223±0,045
	NB	0,579±0,029
	RLOG	0,564±0,021
	RLIN	0,359±0,141
EmpSel	KNN	0,063±0,001
	SVC	0,065±0,001
	AD	0,607±0,023
	FA	0,687±0,019
	MLP	0,201±0,008
	AB	–
	NB	0,322±0,021
	RLOG	0,065±0,001
	RLIN	–
MisTot	KNN	0,691±0,021
	SVC	0,691±0,018
	AD	0,653±0,024
	FA	0,680±0,029
	MLP	0,691±0,015
	AB	0,247±0,049
	NB	0,658±0,018
	RLOG	0,693±0,015
	RLIN	0,404±0,072
MisSel	KNN	0,483±0,018
	SVC	0,484±0,018
	AD	–
	FA	0,283±0,028
	MLP	0,275±0,018
	AB	–
	NB	–
	RLOG	0,693±0,015
	RLIN	–

Vot	0,694±0,015

6.2.9 Resultados obtidos em KTH_D

Nesta base de dados, a CNN obteve média de acurácia de 0,746 com desvio padrão de 0,025. A Tabela 11 traz a média de acurácias do *ensemble* na base KTH-TIPS-2b, quando as amostras "d" são usadas como treino.

Tabela 11 – Média de acurácias do *ensemble* na base KTH-TIPS-2b - Usando amostras "d" como treino.

Método	Classificador	Média de acurácias
SHA	KNN	0,726±0,020
	SVC	0,704±0,043
	AD	0,388±0,047
	FA	0,558±0,040
	MLP	0,745±0,028
	AB	0,185±0,048
	NB	0,667±0,041
	RLOG	0,736±0,032
	RLIN	0,173±0,181
CHA	KNN	0,711±0,017
	SVC	0,704±0,043
	AD	0,451±0,038
	FA	0,708±0,028
	MLP	0,738±0,032
	AB	0,196±0,047
	NB	0,667±0,042
	RLOG	0,736±0,032
	RLIN	0,173±0,181
EmpTot	KNN	0,529±0,025
	SVC	0,528±0,047
	AD	0,616±0,057
	FA	0,709±0,032
	MLP	0,518±0,038
	AB	0,244±0,058
	NB	0,521±0,036
	RLOG	0,501±0,036
	RLIN	0,439±0,071

Método	Classificador	Média de acurácias
EmpSel	KNN	–
	SVC	0,405±0,032
	AD	0,699±0,041
	FA	0,741±0,025
	MLP	0,073±0,001
	AB	–
	NB	0,135±0,022
	RLOG	0,064±0,001
	RLIN	–
MisTot	KNN	0,718±0,015
	SVC	0,712±0,039
	AD	0,709±0,031
	FA	0,714±0,029
	MLP	0,734±0,028
	AB	0,267±0,055
	NB	0,678±0,034
	RLOG	0,734±0,033
	RLIN	0,419±0,081
MisSel	KNN	0,291±0,015
	SVC	0,221±0,019
	AD	0,137±0,019
	FA	0,222±0,033
	MLP	0,733±0,029
	AB	–
	NB	–
	RLOG	0,667±0,025
	RLIN	–
Vot		0,732±0,025

6.3 Discussões

O código escrito em Python que implementa a CNN, bem como o que define a execução dos classificadores, é praticamente o mesmo em todos os experimentos que fizemos, mudando apenas a forma como as bases de dados são usadas como entrada no algoritmo, conforme descrevemos na Seção 6.1. Apesar disso, devido às diferentes texturas das imagens, o comportamento da CNN e dos classificadores foi bastante heterogêneo, confirmando uma máxima bem conhecida em aprendizado de máquinas de que o classificador mais adequado

vai depender em grande medida do tipo de dado sendo analisado.

Na Tabela 12 mostramos a quantidade de vezes em que os classificadores obtiveram, na média, índice de acerto maior que os demais classificadores nos experimentos [2] realizados nas 6 bases de dados.

É importante observar que houve empate em alguns casos, como pode se ver na Tabela 3, onde os classificadores KNN, SVC, MLP e RLOG empataram em primeiro lugar no método SHA. Por isso, o somatório da Tabela 12 é maior que 54[3].

Tabela 12 – Número de vezes que os classificadores superaram, na média, os demais em índice de acerto.

Classificador	Número de vezes com maior média de acurácia
RLOG	25
MLP	20
FA	13
SVC	8
KNN	5
AD	3
RLIN	2
NB	1
AB	0

6.3.1 O desempenho da CNN

Na média, a CNN classifica melhor a maioria das bases de dados que utilizamos, e quando não obteve a melhor média de acurácias, a CNN obteve resultados muito próximos das maiores médias, portanto, a CNN tem um desempenho mais eficaz que o *ensemble*, e também com menor tempo de execução, como pode ser visto na Tabela 14. Mostramos na Tabela 13 a comparação da média de acurácia da CNN (na coluna CNN) *versus* o melhor desempenho médio dos classificadores.

[2] 10 experimentos em cada uma das bases: CYST, CYST_KS, FMD, UMD, UIUC; e 40 experimentos na base KTH-TIPS-2b: 10 experimentos para cada vez em que as amostras “a”, “b”, “c” e “d” foram utilizadas como dados de treino.

[3] Se não houvesse empates, como utilizamos 9 bases (se contarmos KTH_A, KTH_B, KTH_C e KTH_D como 4 bases oriundas da base KTH-TIPS-2b e a base CYST_KS como uma base oriunda da base CYST, teremos 9 bases: CYST, CYST_KS, FMD, UMD, UIUC, KTH_A, KTH_B, KTH_C e KTH_D) e 6 métodos, com isso haveria um total de 54 primeiros lugares.

Tabela 13 – Quadro comparativo CNN × *Ensemble*.

Base de Dados	CNN	Classificador	Método	Média do Classificador
FMD	0,838	SVC	MISTOT	0,829
		MLP	MISTOT	
		SVC	MISSEL	
CYST_KS	0,993	RLOG	EMPTOT	0,989
		MLP	EMPSEL	
		KNN	MISTOT	
		SVC	MISTOT	
CYST	0,994	MLP	MISTOT	0,995
		RLOG	MISTOT	
		MLP	MISSEL	
		RLOG	MISSEL	
UMD	0,998	KNN	MISTOT	0,997
		SVC	MISTOT	
		RLOG	MISTOT	
UIUC	0,991	MLP	SHA	0,993
KTH_A	0,713	AD	MISSEL	0,726
KTH_B	0,727	MLP	SHA	0,725
KTH_C	0,701	MLP	SHA	0,693
KTH_D	0,746	MLP	SHA	0,745

6.3.2 O desempenho do classificador RLOG

O classificador RLOG obteve, em maior número de vezes, a maior média de acurácias em comparação aos demais classificadores, em todas as bases e em todos os métodos que utlizamos[4]. Os métodos que funcionaram melhor com RLOG foram SHA e CHA, independente da base de dados.

No método SHA, RLOG obteve a maior média de acurácias[5] ou obteve a segunda maior média. Destacamos a média 0,996 obtida na base UMD, índice também alcançado pelo classificador SVC. Já no método CHA, RLOG só não foi 1° lugar na base KTH_D[6]. Portanto, os ajustes nos hiperparâmetros de RLOG[7] lhe renderam um desempenho ainda melhor em todas as bases de dados.

O empilhamento (total ou seletivo) diminuiu o rendimento de RLOG em todas

[4] Vide Tabela 12.
[5] Isto ocorreu nas bases CYST, CYST_KS, UMD e KTH_A conforme consta nas Tabelas 3, 4, 6 e 8.
[6] Seu índice foi de 0,736; 0,002 abaixo de MLP que obteve a maior média, conforme consta na Tabela 11.
[7] Vide Tabela 2.

as bases de dados e isso ficou mais evidente na base KTH_A, em que RLOG não participou de nenhum empilhamento seletivo.

O EmpTot fez o rendimento de RLOG cair muito na base CYST (0,397) e o fez descer para o 2° lugar em CYST_KS ao obter o índice 0,890. Nas demais bases, sua acurácia não passou de 0,200. Diante desses fatos, podemos afirmar que o empilhamento é um método que não traz benefícios em classificação de texturas especificamente para o classificador RLOG.

Em comparação com o empilhamento, a mistura (total ou seletiva) já trouxe bons resultados para RLOG. Na MisTot, RLOG só não obteve o maior índice de acurácia média nas bases CYST_KS, FMD e KTH_B. Apesar de ter obtido o 2° lugar em CYST_KS, seu melhor índice (0,988) para esse método foi nessa base de dados. Na MisSel, seu melhor rendimento também foi em CYST_KS, apesar de não haver mudança de resultados (0,988), ou seja, MisSel não melhorou nem piorou a classificação de RLOG nessa base.

Seu índice mais baixo em MisSel foi na base KTH_B (0,585), mas esse foi o melhor índice alcançado dentre todos os classificadores[8]. Diante desses dados, podemos afirmar que o método da mistura contribui para a classificação de texturas em se tratando do classificador RLOG.

6.3.3 O desempenho do classificador MLP

No método SHA, MLP só não obteve a maior média em acurácias na base de dados UMD, apesar de ter ficado com a segunda maior média, com o índice 0,995. Seu melhor rendimento no método SHA foi justamente na base UMD. Porém, nas bases de dados CYST e CYST_KS obteve 0,994 e 0,988 de acurácia média, respectivamente, o que lhe garantiu o primeiro lugar em índice de acurácia nessas bases.

Por esses motivos, podemos afirmar que o método SHA combinou muito bem com o algoritmo de MLP. O ajuste de hiperparâmetros no método CHA para o classiicador MLP o fez perder rendimento nas bases FMD, UMD, UIUC e KTH_A. Porém, nas demais bases, o método CHA garantiu ao MLP o 1° lugar em índice médio de acurácia, sendo 0,994 o maior índice médio obtido por esse classificador no método CHA, e esse índice foi alcançado na base CYST.

O empilhamento (total ou seletivo) proporcionou resultados bastante heterogêneos para MLP. Nas bases CYST e CYST_KS, o empilhamento trouxe rendimento médio acima de 0,980 nessas bases. Inclusive, em CYST_KS, MLP obteve com EmpSel índice médio de 0,989; 0,010 mais elevado que o segundo índice médio mais alto alcançado pelo classificador KNN (0,890).

[8] Vide Tabela 9.

Já nas demais bases de dados, o EmpSel não se mostrou competitivo para MLP (sem qualquer resultado obtido), nas bases FMD, UMD, UIUC e nas bases oriundas de KTH-TIPS-2b seu maior índice médio foi em KTH_B com 0,198, um valor muito mais baixo que o maior índice alcançado, o do classificador FA (0,712). O EmpTot lhe trouxe um rendimento melhor (acima de 0,500 na maioria das bases), destacando-se os índices 0,990 e 0,988 nas bases CYST e CYST_KS, respectivamente.

A mistura (total ou seletiva) funcionou muito melhor para MLP, se comparada ao empilhamento. A exceção ocorreu em KTH_C, em que MLP obteve índice 0,275 no método MisSel, sendo seu índice mais baixo na mistura dos dados. Seu melhor resultado na MisTot foi na base CYST ao alcançar índice médio de acertos de 0,995, garantindo-lhe, ao lado de RLOG, o 1° lugar em índices médios. Na base CYST, novamente, MLP alcançou seu maior índice médio (0,995) no método MisSel. Diante disso, podemos afirmar que a mistura funciona bem com o classificador MLP.

6.3.4 O desempenho do classificador FA

O método que melhor se adaptou ao classificador FA foi o empilhamento. No EmpTot, FA só não ficou com a melhor média das acurácias nas bases de dados: CYST, apesar de obter 0,987 de média de acurácias; UMD e UIUC, onde obteve média maior que 0,900, o que lhe garantiu 2° lugar em média de acurácias em relação aos demais classificadores.

Sem hiperparâmetros ajustados, FA obteve desempenho abaixo de 0,600 nas bases FMD e KTH-TIPS-2b. Seus melhores índices no método SHA ocorreram nas bases CYST e CYST_KS, ficando com média acima de 0,950.

O ajuste dos seus hiperparâmetros melhorou seu rendimento médio em todas as bases de dados. Em KTH-TIPS-2b, por exemplo, FA passou a obter médias acima de 0,680, em UMD e UIUC, que em SHA não passou de 0,880. Com o ajuste dos hiperparâmetros, seu índice nessas bases passou de 0,970. Nas bases CYST e CYST_KS seu desempenho, que já estava bom no método SHA, melhorou aproximadamente 2% no método CHA.

A mistura total, para o classificador FA, foi menos proveitosa que o empilhamento (total ou seletivo), porém lhe trouxe bons resultados: nas bases CYST, CYST_KS, UMD e UIUC, a MisTot lhe garantiu médias acima de 0,970; e nas bases FMD e KTH-TIPS-2b, médias acima de 0,690.

O pior método aplicado no classificador FA foi a MisSel. Em FMD, por exemplo, ele não participou de nenhuma mistura seletiva e seu pior rendimento médio foi na base KTH_D com índice de 0,222.

6.3.5 O desempenho do classificador SVC

O classificador SVC mostrou-se bastante estável em todas as bases de dados e métodos que utilizamos. Na maioria dos métodos e bases, ele não obteve o 1° lugar em desempenho, mas nem por isso obteve índices de acurácia longe dos melhores classificadores, e seu comportamento estável lhe forneceu uma vantagem sobre os demais classificadores.

No método SHA, seu melhor desempenho foi na base UMD, com índice de acurácia médio de 0,996, mas nas bases CYST, CYST_KS e UIUC ele também obteve índice médio de acurácia acima dos 0,990, tendo com isso um bom desempenho nesse método; mesmo porque, para as demais bases, seus índices médios não foram menores que 0,688.

Apesar do método CHA ter diminuído o rendimento de SVC em todas as bases, essa perda (quando ocorreu) não foi maior que 5%, por isso, mantemos a ideia de que SVC foi estável com ou sem hiperparâmetros ajustados.

O empilhamento total trouxe prejuízos no trabalho de classificação das texturas por parte do classificador SVC. Seu melhor rendimento foi na base CYST, com índice de 0,990 e seu rendimento mais baixo foi de 0,444 na base FMD.

O empilhamento seletivo foi ainda pior para SVC. Na base FMD ele não participou de nenhuma seleção em nenhum dos experimentos que realizamos e seu melhor desempenho nesse método foi na base CYST_KS, com índice médio de 0,890.

Apesar de não ter sido um bom método para SVC, o empilhamento não mudou a característica de SVC em ser estável, ou seja, não houve um desempenho excelente em uma base e um péssimo desempenho em outra. O empilhamento foi ruim para SVC em todas as bases.

Já a mistura foi vantajosa para SVC. Principalmnete a MisTot que só não lhe garantiu os primeiros lugares em índice médio de acurácia nas bases oriundas de KTH-TIPS-2b. Seu melhor desempenho em MisTot foi na base UMD, com índice 0,997, apesar de ter índice acima de 0,829 nas bases CYST, CYST_KS, FMD e UIUC.

A MisSel fez SVC perder rendimento, mas não houve, em nenhuma base de dados, um experimento que ele não tenha participado da seleção. Seu melhor rendimento foi de 0,988 na base CYST_KS e seu pior rendimento foi na base KTH_D com o índice médio de 0,221.

6.3.6 O desempenho do classificador KNN

O método que melhor se adaptou ao classificador KNN foi MisTot, em que ele conseguiu suas melhores médias de acurácia. Na base CYST_KS, KNN atingiu o 1° lugar em média de acurácias no método MisTot com o índice 0,989, e também antingiu o 1°

lugar em UMD com o índice de 0,997. Nas bases CYST e UIUC também obteve índices acima de 0,990.

Seu desempenho no método MisTot nas bases FMD e KTH-TIPS-2b variou de 0,695 a 0,826, mostrando que esse método trouxe bons resultados ao classificador KNN. No método MisSel, o rendimento de KNN caiu, apesar desse classificador ter paticipado da seleção em pelo menos um experimento em cada base. O melhor desempenho de KNN em MisSel foi de 0,988 na base CYST_KS, o que lhe garantiu o 1° lugar nessa base para esse método. Porém, seu rendimento mais baixo foi em KTH_D, com o índice de 0,291 nesta base.

Seguido da MisTot, o EmpTot trouxe bons resultados para KNN, principalmente nas bases CYST e CYST_KS, onde alcançou índices médios de acurácia de 0,991 e 0,988, respectivamente. Seu pior desempenho no método MisTot foi na base FMD com o índice de 0,426. Nas bases UMD e KTH-TIPS-2b seus índices médios variaram entre 0,797 e 0,529, o que em comparação com os demais classificadores foi um redimento mediano.

O método EmpSel foi o que trouxe os piores resultados para KNN, a ponto de nas bases KTH_A, KTH_B e KTH_D ele não participar de nenhuma seleção. Mesmo ocupando o 2° lugar em médias de acurácia na base CYST_KS, com o índice de 0,890, na base CYST seu desempenho foi bastante baixo, com média de 0,497. Nas bases FMD, UMD, UIUC e KTH_C seus índices variaram de 0,063 a 0,291, o que é mais uma evidência de que EmpSel não rendeu bons resultados para KNN.

O ajuste de hiperparâmetros, de uma maneira geral, trouxe melhoras no rendimento de KNN da ordem de 5%, apesar de o método SHA ter lhe trazido bons resultados na classificação das texturas. Na base CYST houve uma pequena perda de rendimento do método SHA (0,994) para o método CHA (0,993), porém nas demais bases de dados houve ganho, e o mais significativo foi na base UMD, em que o KNN passou de 0,825 do método SHA para 0,996 no método CHA. Seus piores desempenhos nos métodos SHA e CHA não geraram índices extremamente baixos e ocorreram na mesma base de dados, KTH_A: 0,677 com o método SHA e 0,690 com o método CHA.

6.3.7 O desempenho do classificador AD

O classificador AD teve baixo rendimento na maioria dos métodos e na maioria das bases de dados. Na base KTH-TIPS-2b, AD obteve seus menores índices em comparação à maioria dos métodos. A excessão foi no método MisTot, em que AD obteve bons resultados para todas as bases.

Sem os ajustes nos hiperparâmetros, AD só obteve índices altos nas bases CYST (0,954) e CYST_KS (0,958). Nas demais bases, nesse método, AD não passou de 0,430 na média das acurácias.

Com o ajuste dos hiperparâmetros, o rendimento de AD caiu mais ainda. Como exemplo, destacamos seu desempenho nas bases CYST e CYST_KS, que foi de 0,952 e 0,959, respectivamente. Seu índice mais baixo nesse método foi de 0,451 na base de dados KTH_D.

O classificador AD não teve um comportamento estável nos métodos EmpTot e EmpSel. Para a maioria das bases (CYST, CYST_KS, UMD, KTH_A e KTH_C) houve piora no rendimento, se compararmos EmpTot com EmpSel. Porém, nas bases FMD, UIUC, KTH_B e KTH_D houve melhora no rendimento quando AD utilizou os dados empilhados seletivamente, em vez de utilizar todos os dados empilhados.

Seu melhor resultado no método EmpTot foi na base CYST, com índice de 0,988, e em EmpSel seu melhor rendimento foi em UIUC, com 0,961 de índice médio de acurácia. A variação mais relevante no rendimento de AD foi na base CYST_KS, pois no método EmpTot o classificador obteve média de 0,974 e no método EmpSel obteve 0,592, uma queda de mais de 60% em rendimento médio.

No método MisTot, nas bases CYST e CYST_KS, o classificador AD obteve seus maiores índices médios: 0,994 e 0,987, respectivamente. Nas bases FMD, UMD e UIUC, AD obteve médias próximas (0,819; 0,820; e 0,880, respectivamente) no método MisTot, mostrando alguma estabilidade de comportamento nesse método.

Nas bases oriundas de KTH-TIPS-2b, AD também mostrou certa regularidade, e obteve médias próximas das melhores médias obtidas no método MisTot, variando seus índices médios entre 0,653 e 0,709. Apesar dos bons resultados obtidos na MisTot, a MisSel não trouxe vantagens para o classificador AD, principalmente nas bases UMD e UIUC, onde ele não participou de nenhuma seleção. As excessões foram nas bases CYST, CYST_KS e KTH_A, onde AD obteve índices médios de 0,795, 0,791 e 0,726, respectivamente.

6.3.8 O desempenho do classificador RLIN

O classificador RLIN obteve baixo rendimento em quase todas as bases de dados e métodos que usamos. As excessões ficaram por conta do EmpTot e da MisTot para todas as bases, excluindo-se KTH-TIPS-2b.

Nos métodos SHA e CHA, RLIN apresentou certa regularidade em seus desempenhos, ou seja, o ajuste dos hiperparâmetros não melhorou as classificações de RLIN. Seu melhor índice foi de 0,809 tanto em SHA quanto em CHA. E seu índice mais baixo ocorreu na base KTH_D com o índice 0,173, tanto em SHA quanto em CHA.

O EmpTot foi o melhor método aplicado no classificador RLIN para todas as bases de dados, pois ele conseguiu médias acima de 0,950 nas bases CYST, CYST_KS, UMD e UIUC. Na base FMD seu índice foi de 0,652, que ficou atrás apenas do classificador FA no método EmpTot. Nas bases oriundas de KTH-TIPS-2b, no método EmpTot seu

desempenho foi baixo, variando de 0,310 até 0,439 os seus índices médios de acurácia.

A MisTot também trouxe bons resultados para RLIN nas bases CYST (0,995), CYST_KS (0,958), UMD (0,995) e UIUC (0,998); um redimento mediano na base FMD (0,638) e baixo rendimento na base KTH-TIPS-2b, onde seus índices médios variaram de 0,404 até 0,489.

O EmpSel trouxe resultados ruins para RLIN, principalmente nas bases CYST, FMD e KTH-TIPS-2b, pois nessas bases ele não participou de nenhuma seleção. Seu rendimento foi baixíssimo nas bases CYST_KS (0,192) e UIUC (0,099) e mediano em UMD (0,696).

O pior método aplicado no classificador RLIN foi o MisSel pois não houve em nenhum experimento, para nenhuma base, a seleção de suas classificações para participar da MisSel e, por isso, não obteve nenhuma média de acurácia nesse método.

6.3.9 O desempenho do classificador NB

O classificador NB conseguiu bons resultados na maioria dos métodos que utilizamos, excessão apenas aos métodos EmpSel e MisSel. No método SHA, NB conseguiu a segunda melhor média de acurácias com o índice 0,988 na base CYST, e também obteve índices maiores que 0,940 nas bases CYST_KS, UMD e UIUC.

Seu desempenho na base FMD no método SHA foi 0,004 menor que a maior média, ou seja, ele ficou muito próximo do melhor classificador para esse método, nessa base. Na base KTH-TIPS-2b, seu desempenho no método SHA foi mediano e regular, variando seus índices de 0,664 a 0,672.

O ajuste de hiperparâmetros ou manteve os índices do método SHA ou provocou uma pequena perda da ordem de 2% no desempenho de NB, portanto não foi um método vantajoso para as classificações de NB. O EmpTot fez o rendimento de NB cair na maioria das bases, excessão apenas às bases CYST (0,993) e CYST_KS (0,988), onde seus índices aumentaram em comparação ao método SHA. Sua maior queda de rendimento, comparando EmpTot com SHA, foi na base FMD onde NB saiu do índice 0,823 no método SHA para 0,423 no método EmpTot, uma queda de mais de 50%.

O EmpSel agravou mais ainda a perda de rendimento de NB no empilhamento; nas bases CYST e CYST_KS a queda foi da ordem de 9%, comparando EmpSel com EmpTot; nas bases FMD, UMD e UIUC o classificador NB não participou de nenhuma seleção, portanto não obteve média de acurácia nesse método para essas bases.

Na base KTH-TIPS-2b, no método EmpSel seus índices variaram de 0,135 a 0,322, comprovando que esse método não foi eficiente para a classificação de texturas pelo classificador NB.

A MisTot forneceu para NB seus melhores resultados em relação a todos os métodos que utilizamos. Seu maior índice médio (0,994) em todas as bases e em todos os métodos foi obtido em CYST, nesse método.

Nas bases CYST_KS, UMD e UIUC seus índices médios de acurácia foram maiores que 0,940 e em FMD ele obteve a média de 0,811, ou seja, 0,18 a menos que a maior média de acurácias obtida pelo classificador SVC.

Na base KTH-TIPS-2b, NB ficou cerca de 9% abaixo da melhor média de acurácias no método MisTot, mostrando mais uma vez que esse método foi o melhor método empregado nesse classificador.

Apesar dos bons resultados da MisTot em NB, a MisSel foi ruim para NB; nas bases UIUC, KTH_A, KTH_C e KTH_D, o classificador não obteve nenhuma média de acurácias por não participar de nenhuma seleção em nenhum dos experimentos que realizamos.

Na base CYST_KS seu desempenho caiu bastante, mas ainda assim foi relevante, com o índice 0,889 no método MisSel. Porém, na base CYST, seu índice que em em MisTot havia sido o melhor desse classificador, foi o pior para essa base com o índice de 0,597. Na base KTH-TIPS-2b, NB só não teve índice zero de acurácias na base KTH_B onde obteve 0,078 de média de acurácias, mostrando mais uma vez que MisSel foi o pior método aplicado no classificador NB.

6.3.10 O desempenho do classificador AB

O classificador AB obteve baixos índices médios de acurácia na maioria dos métodos que utilizamos. As bases de dados onde o classificador obteve resultados relevantes foram CYST e CYST_KS.

No método SHA, AB obteve índices acima de 0,970 nas bases CYST e CYST_KS, e melhorou seu desempenho em CHA na base CYST_KS, obtendo seu melhor índice médio (0,985) em todas as bases. Infelizmente, na base CYST, no método CHA, AB teve uma pequena perda (caiu 0,001), mas ainda conseguiu um resultado relevante (0,969).

Nas demais bases de dados, AB obteve índices que variaram de 0,127 a 0,257 no método SHA, e variaram de 0,118 a 0,262 no método CHA, mostrando que não obteve bons resultados nas outras bases.

O EmpTot melhorou o rendimento de AB na base CYST, se comparado ao método SHA, da ordem de 1%. Apesar de não ser um grande aumento, AB conseguiu nesse método e nessa base seu melhor índice médio (0,989).

Para as demais bases, AB não teve resultados relevantes, variando de 0,223 a

0,512, exceção apenas à base CYST_KS, onde obteve índice médio de 0,976, menor que nos métodos SHA e CHA, mas ainda um resultado considerado bom.

O EmpSel diminuiu os índices de AB em comparação ao método EmpTot nas bases CYST e CYST_KS e nas demais bases AB não obteve, em nenhum experimento, média de acurácias. Com isso, podemos concluir que esse método foi ineficaz para as classificações de AB.

A MisTot foi o melhor método aplicado ao classificador AB nas bases CYST e CYST_KS, pois o índice 0,994 conseguido em CYST foi a melhor média de AB em todas as bases e a média 0,988 na base CYST_KS ficou 0,001 abaixo da melhor média em acurácias nesse método.

Já nas demais bases, o desempenho de AB foi baixo, variando de 0,247 até 0,544; com isso, podemos concluir que MisTot não forneceu vantagens ao classificador AB nessas bases de dados.

A MisSel foi, juntamente com o EmpSel, o pior método aplicado ao classificador AB, pois diminuiu seu rendimento em CYST (0,895) e CYST_KS (0,889) e também não o fez gerar qualquer média de acurácias nas demais bases de dados.

6.3.11 O desempenho da Votação

A votação do *ensemble* de classificadores em linhas gerais perde em desempenho para todos os métodos que utilizamos nesse trabalho. Essa perda, contudo, foi de no máximo 7% em relação ao desempenho ou da CNN ou dos demais métodos aplicados nos classificadores.

Seu melhor rendimento foi na base UMD, com o índice 0,996, apesar de obter altos índices médios nas bases CYST (0,992) e CYST_KS (0,988). E seu menor índice médio de acurácias (0,694) foi na base KTH_C, menos de 1% menor que a melhor acurácia nessa base de dados conseguida pela CNN, conforme consta na Tabela 13.

6.3.12 Tempo médio de execução dos métodos

Cada método que utilizamos obteve um tempo médio de execução, dependendo da base de dados onde esse método é empregado. Na Tabela 14, constam os tempos médios de execução (em minutos e segundos) de cada método por base de dados.

Tabela 14 – Tempo médio em minutos e segundos da excução dos métodos por base de dados.

Base de Dados	CNN	SHA	CHA	Vot	EmpTot	EmpSel	MisTot	MisSel
CYST	16:12	16:25	27:21	27:33	28:18	28:19	37:16	37:27
FMD	07:10	07:17	15:03	15:09	19:11	19:12	22:43	22:46
KTH_A	28:28	28:45	40:49	41:02	48:30	48:34	52:52	53:02
KTH_B	23:05	23:21	34:24	34:36	41:28	41:32	45:06	45:14
KTH_C	22:20	22:36	32:32	32:43	38:48	38:51	42:54	43:00
KTH_D	19:36	19:52	31:02	31:14	37:16	37:18	42:08	42:17
CYST_KS	10:30	10:38	17:24	17:32	18:05	18:06	23:26	23:32
UMD	18:24	18:32	27:16	27:24	32:51	32:51	40:52	40:59
UIUC	07:51	08:00	15:45	15:51	21:10	51:10	29:36	29:45

A Tabela 14 mostra que a CNN, em todas as bases de dados, teve o menor tempo de execução em comparação aos demais métodos. Mas há um motivo para isso, a CNN fornece os vetores de descritores aos classificadores, então, o tempo de execução dos classificadores precisa ser acrescido do tempo de execução da CNN, pois os vetores de descritores são aqueles que foram armazenados no melhor desempenho de treino da CNN, como não sabemos previamente qual será a melhor época de treinamento da CNN, precisamos computar todo o processamento da CNN para o tempo de execução dos classificadores, e por isso, eles sempre perdem, em princípio, para a CNN em tempo de execução.

A CNN obteve seus menores tempos de execução nas bases de dados FMD e UIUC, conforme podemos ver na Tabela 14, e nessas bases sua média de acurácias foi de 0,838 e 0,991, conforme está registrado na Tabela 13. Podemos então perceber que nessas bases de dados, além de ter uma rápida execução de seu processamento, a CNN obteve alto índice de acurácia em comparação aos demais métodos que utilizamos, mostrando que, apesar das dificuldades inerentes a esses bancos de dados, a CNN é capaz de executar a tarefa de classificação com baixo custo computacional e ao mesmo tempo alta precisão.

Outro fato importante registrado na Tabela 14 é que na base de dados CYST_KS, onde há um trabalho mais complexo para a classificação das imagens dos queratocistos, uma vez que se trata apenas de imagens do tipo odontogênico, a CNN trabalhou mais rápido e com melhor desempenho em termos de acurácia do que todos os demais métodos, conforme consta na Tabela 13. Portanto, para o caso mais difícil de classificação dos cistos mandibulares, a CNN é também mais rápida e mais precisa em classificar esse tipo de imagem médica.

O método SHA teve seu melhor desempenho em média de acurácias com o classificador MLP na base de dados UIUC, conforme podemos constatar na Tabela 13. Conforme está registrado na Seção 6.3.3, o classificador MLP obteve seus melhores resultados no método SHA e na base de dados UMD, e na Tabela 14 podemos constatar que além de obter os melhores índices de acurácia, o tempo de execução de MLP no método SHA na base de dados UMD está oito segundos acima do tempo de execução da CNN, mostrando que seu desempenho médio nessa base de dados possui baixo custo operacional e alto índice de acurácias, o que nos leva a concluir que para esse método o classificador MLP obteve o melhor resultado para o trabalho de classificação de texturas visuais, se comparado aos demais classificadores.

É importante notar que, nas imagens médicas, a diferença de tempo de execução entre a CNN e o método SHA com o classificador MLP foi de apenas três segundos para a base CYST e de oito segundos para a base CYST_KS. Já a MLP nessas bases para o método SHA obteve índices médios de acurácia de 0,994 e 0,988, respectivamente, mostrando que para as imagens médicas seu desempenho é também muito próximo da CNN, tanto em velocidade de processamento quanto em média de acurácias.

A Tabela 14 também nos mostra que o método CHA é computacionalmente caro, pois no caso de KTH_A, por exemplo, houve um aumento de mais de 40% no tempo de execução do código e não houve um ganho em médias de acurácia que justificasse esse custo operacional, exceto para o classificador RLOG, que obteve seus melhores índices de acurácia nesse método, conforme está registrado na Seção 6.3.2. Mas, de acordo com a Tabela 13, o método CHA não obteve a maior média em acurácia em nenhuma base de dados e para nenhum classificador, portanto, o método CHA aumenta em até 40% o tempo médio para a classificação das texturas visuais e não fornece um ganho em acurácias que justifique esse gasto de tempo.

Em contrapartida, o método MisSel, que possui o maior tempo médio de execução em comparação aos demais métodos que utilizamos, em alguns casos, como nas bases FMD, CYST e KTH-TIPS-2b, obteve altos índices de acurácia, principalmente em KTH-TIPS-2b utilizando as amostras “b” como dados para treino, pois nesse caso MisSel superou a CNN, apesar de consumir quase o dobro do tempo de execução do código para realizar as classificações das texturas visuais. Portanto, para esse método, é importante avaliar se compensa gastar mais tempo de execução do código para obter um maior índice médio de acurácias.

6.4 Matrizes de Confusão

As matrizes de confusão mostram a proporção de acerto de um classificador mediante uma determinada classe. Por exemplo, digamos que nosso experimento tenha três

classes de imagens: X, Y e Z. Haverá na matriz de confusão linhas e colunas contendo as classes X, Y e Z, e no cruzamento da linha X com coluna X, a matriz informa a proporção de vezes que o classificador identificou que a imagem era da classe X sendo a imagem realmente da classe X; no cruzamento da linha Y com a coluna Z, a matriz informará a proporção de vezes que o classificador identificou que a imagem era da classe Y sendo a imagem, na verdade, da classe Z; e assim sucessivamente.

A seguir mostraremos a matriz de confusão do(s) classificador(es) que obteve(ram) o melhor desempenho nas bases de dados que utilizamos, conforme a Tabela 13. Nestas figuras, células com valor zero estão vazias, para facilitar a visualização geral dos números nas demais células.

6.4.1 Matrizes de Confusão da Base CYST

Na base de dados CYST os classificadores MLP e RLOG obtiveram o melhor desempenho médio em comparação com os demais classificadores, com o índice de acurácia de 0,995[9]. Nas Figuras 15, 16, 17 e 18 veremos as matrizes de confusão dos classificadores que obtiveram as maiores acurácias na Base CYST.

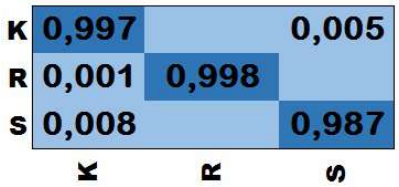


Figura 15 – Matriz de Confusão - CYST - MLP - MisTot.

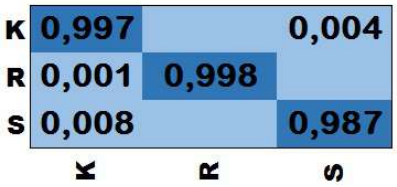


Figura 16 – Matriz de Confusão - CYST - RLOG - MisTot.

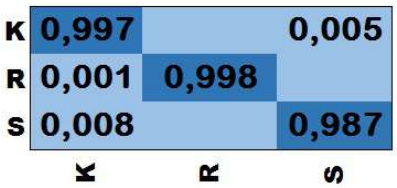


Figura 17 – Matriz de Confusão - CYST - MLP - MisSel.

[9] Vide Tabela 13.

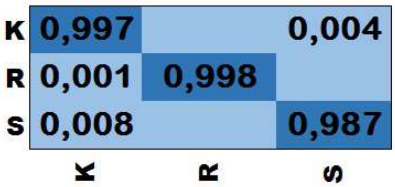


Figura 18 – Matriz de Confusão - CYST - RLOG - MisSel.

6.4.2 Matrizes de Confusão da Base FMD

Na base de dados FMD os classificadores MLP e SVC obtiveram o melhor desempenho médio em comparação com os demais classificadores, com o índice de acurácia de 0,829[10]. Nas Figuras 19, 20 e 21 veremos as matrizes de confusão dos classificadores que obtiveram as maiores acurácias na Base FMD.

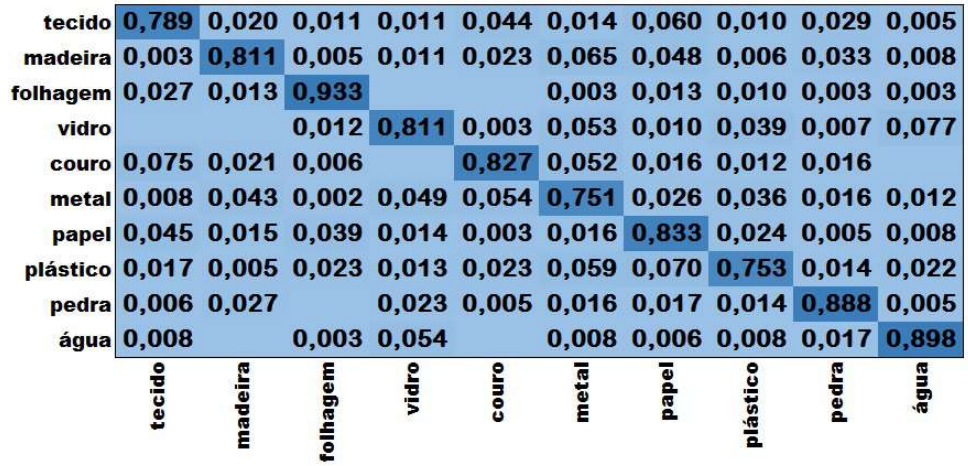


Figura 19 – Matriz de Confusão - FMD - SVC - MisTot.

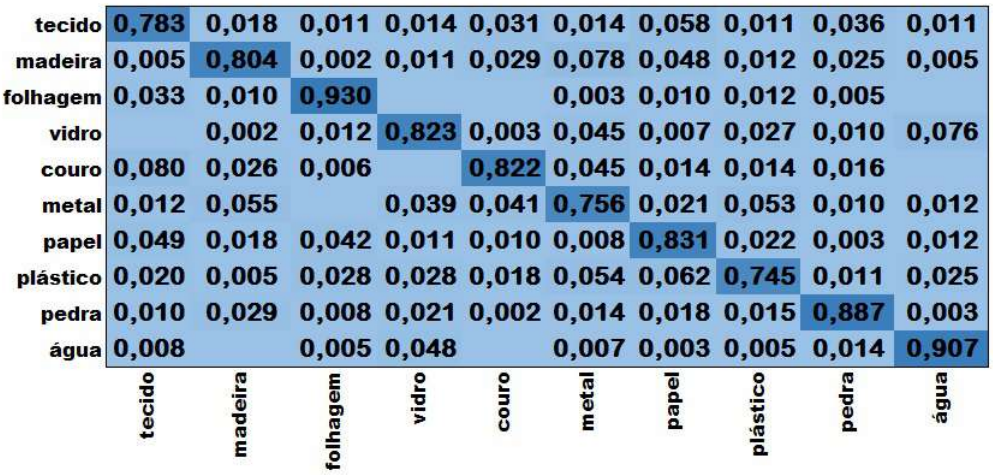


Figura 20 – Matriz de Confusão - FMD - MLP - MisTot.

[10] Vide Tabela 13.

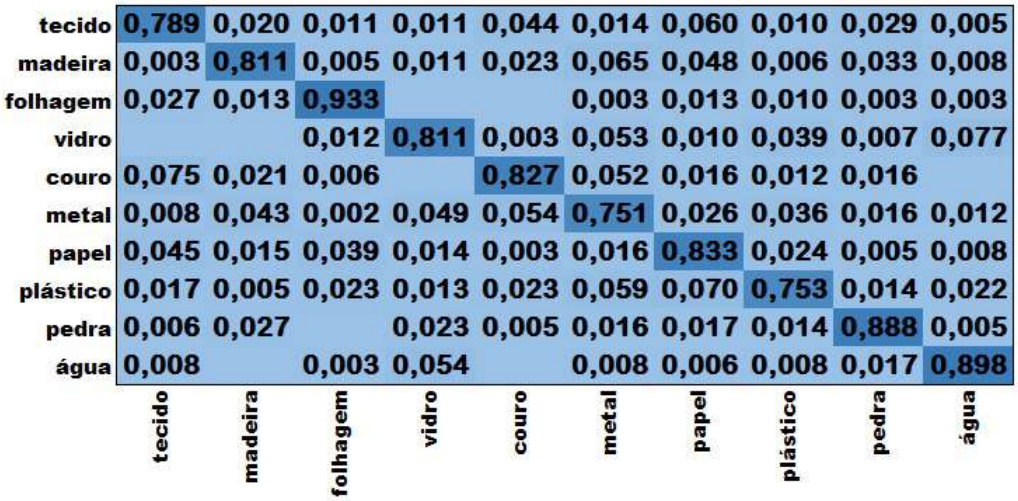


Figura 21 – Matriz de Confusão - FMD - SVC - MisSel.

6.4.3 Matrizes de Confusão da Base KTH_A

Na base de dados KTH_A o classificador AD obtive o melhor desempenho médio em comparação com os demais classificadores, com o índice de acurácia de 0,726[11]. Na Figura 22 veremos a matriz de confusão do classificador AD na Base KTH_A.

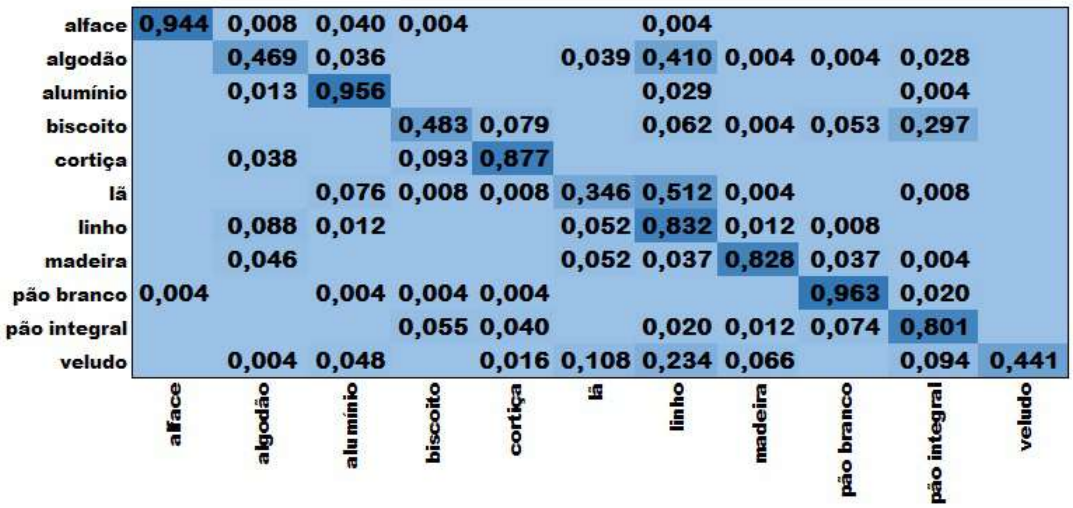


Figura 22 – Matriz de Confusão - KTH_A - AD - MisSel.

6.4.4 Matrizes de Confusão da Base KTH_B

Na base de dados KTH_B o classificador MLP obtive o melhor desempenho médio em comparação com os demais classificadores, com o índice de acurácia de 0,725[12]. Na Figura 23 veremos a matriz de confusão do classificador MLP na Base KTH_B.

[11] Vide Tabela 13.
[12] Vide Tabela 13.

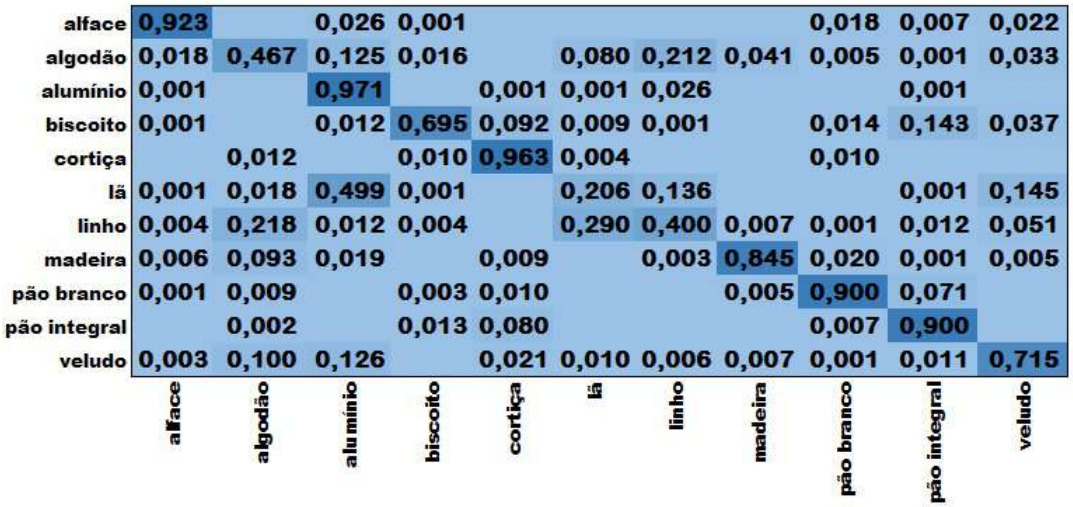


Figura 23 – Matriz de Confusão - KTH_B - MLP - SHA.

6.4.5 Matrizes de Confusão da Base KTH_C

Na base de dados KTH_C o classificador MLP obtive o melhor desempenho médio em comparação com os demais classificadores, com o índice de acurácia de 0,693[13]. Na Figura 24 veremos a matriz de confusão do classificador MLP na Base KTH_C.

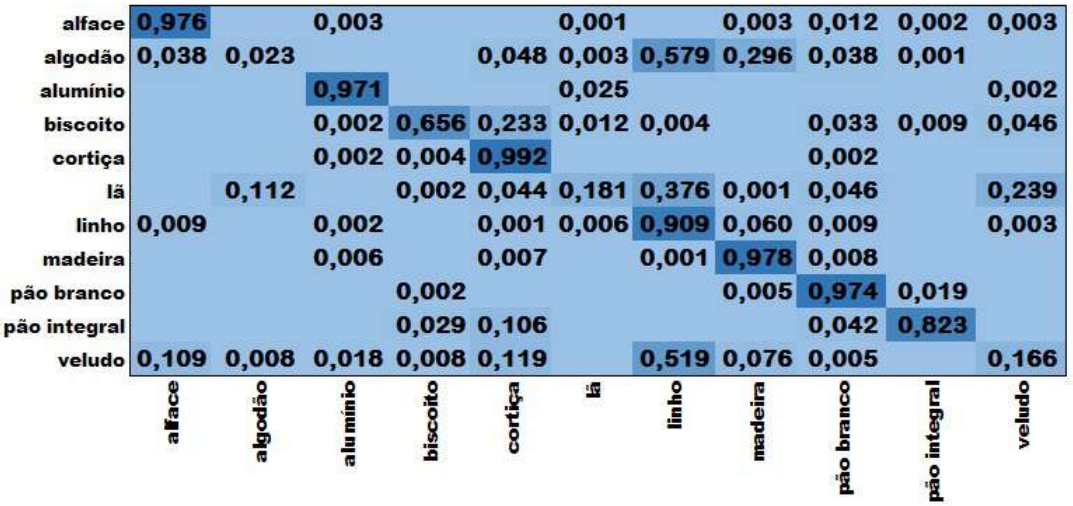


Figura 24 – Matriz de Confusão - KTH_C - MLP - SHA.

6.4.6 Matrizes de Confusão da Base KTH_D

Na base de dados KTH_D, o classificador MLP obtive o melhor desempenho médio em comparação com os demais classificadores, com o índice de acurácia de 0,745[14]. Na Figura 25 veremos a matriz de confusão do classificador MLP na Base KTH_D.

[13] Vide Tabela 13.
[14] Vide Tabela 13.

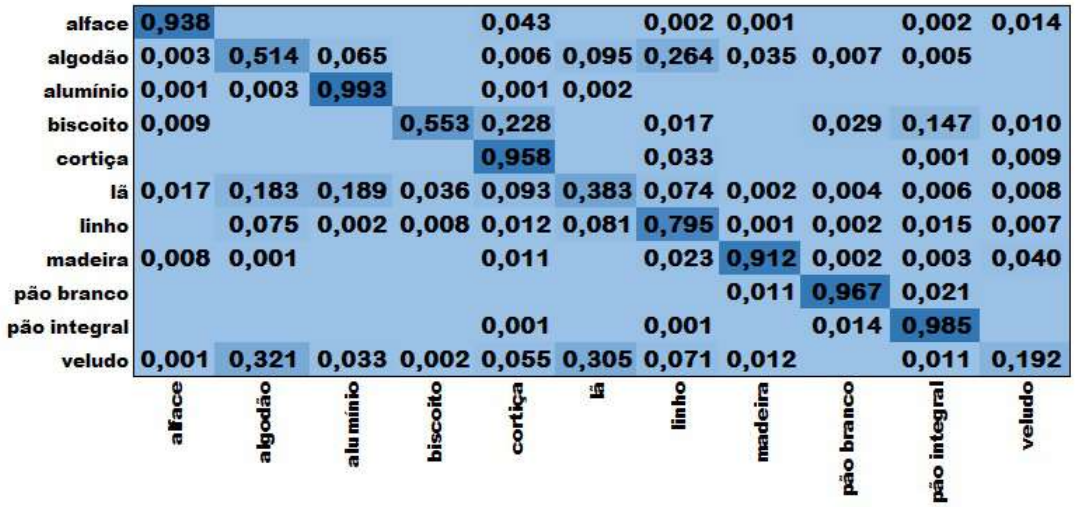


Figura 25 – Matriz de Confusão - KTH_D - MLP - SHA.

6.4.7 Matrizes de Confusão da Base CYST_KS

Na base de dados CYST_KS, os classificadores RLOG, MLP, KNN e SVC obtiveram o melhor desempenho médio em comparação com os demais classificadores, com o índice de acurácia de 0,989[15]. Nas Figuras 26, 27, 28 e 29 veremos as matrizes de confusão dos classificadores que obtiveram as maiores acurácias na Base CYST_KS.

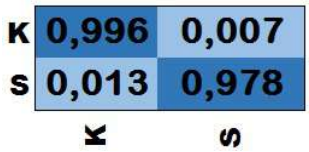


Figura 26 – Matriz de Confusão - CYST_KS - RLOG - EmpTot.

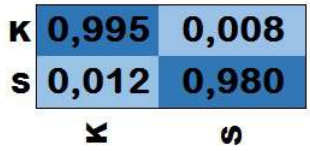


Figura 27 – Matriz de Confusão - CYST_KS - MLP - EmpSel.

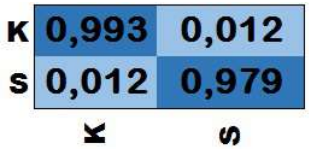


Figura 28 – Matriz de Confusão - CYST_KS - KNN - MisTot.

[15] Vide Tabela 13.

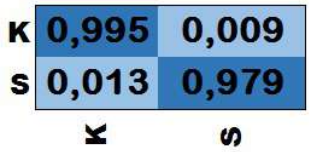


Figura 29 – Matriz de Confusão - CYST_KS - SVC - MisTot.

6.4.8 Matrizes de Confusão da Base UMD

Na base de dados UMD, os classificadores KNN, RLOG e SVC obtiveram o melhor desempenho médio em comparação com os demais classificadores, com o índice de acurácia de 0,997[16]. Podemos perceber em todos estes casos que, exceto na diagonal principal dessa matriz, todos os demais valores se aproximam muito de zero. Isto porque na base de dados UMD os classificadores aqui ilustrados obtiveram um índice médio de acurácia já muito próximo do acerto absoluto. Nas Figuras 30, 31 e 32 veremos as matrizes de confusão dos classificadores que obtiveram as maiores acurácias na Base UMD.

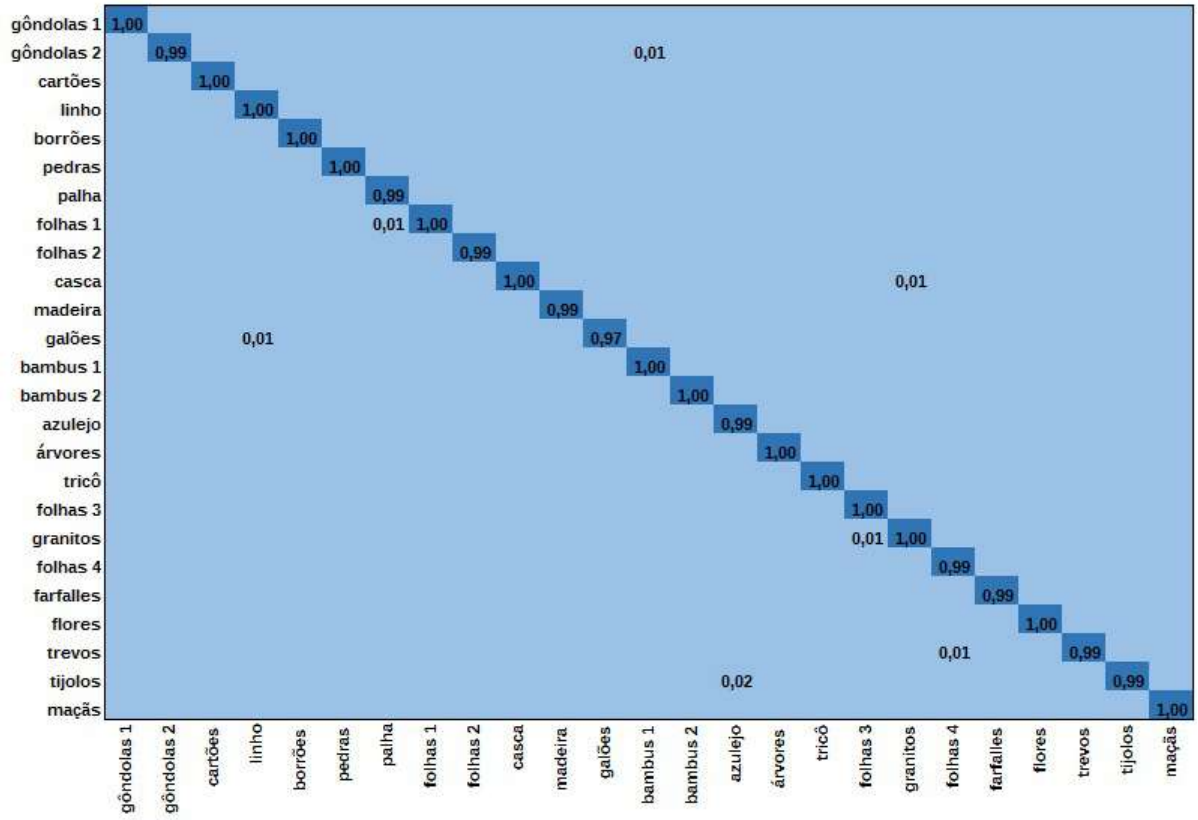


Figura 30 – Matriz de Confusão - UMD - SVC - MisTot.

[16] Vide Tabela 13.

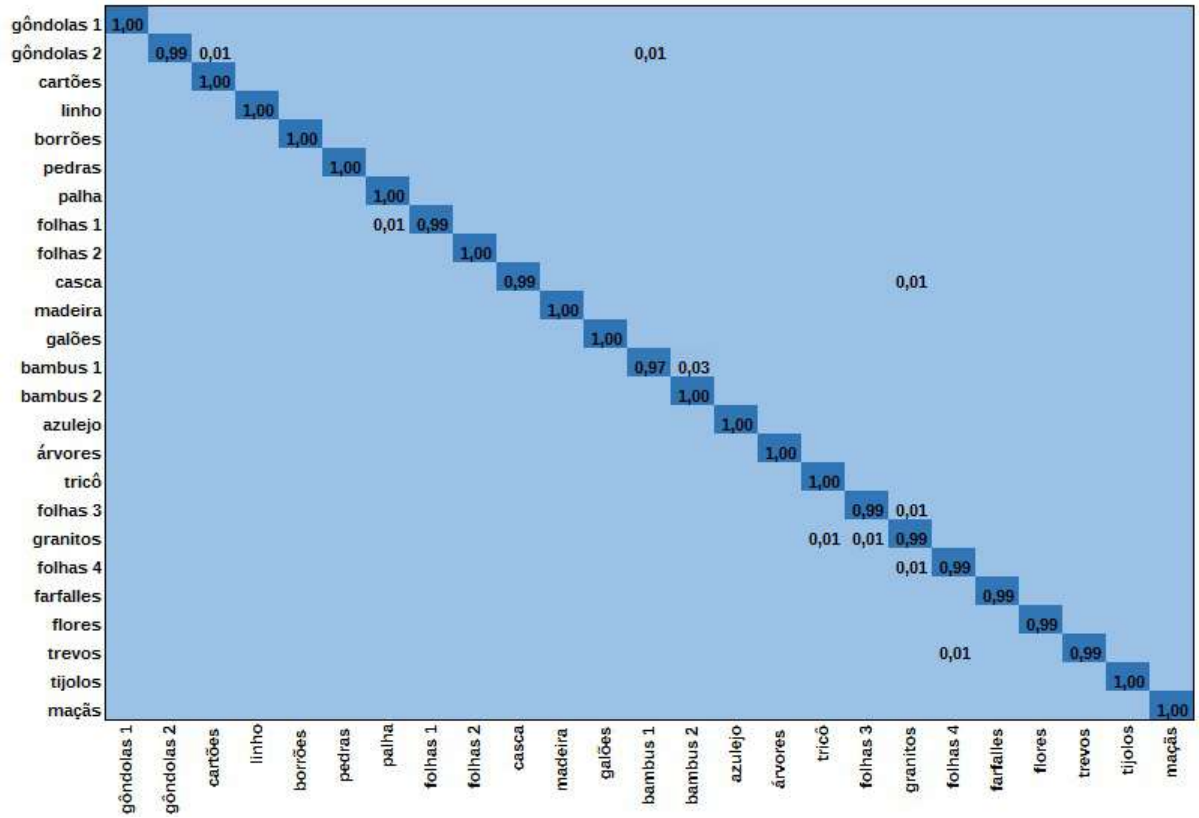


Figura 31 – Matriz de Confusão - UMD - RLOG - MisTot.

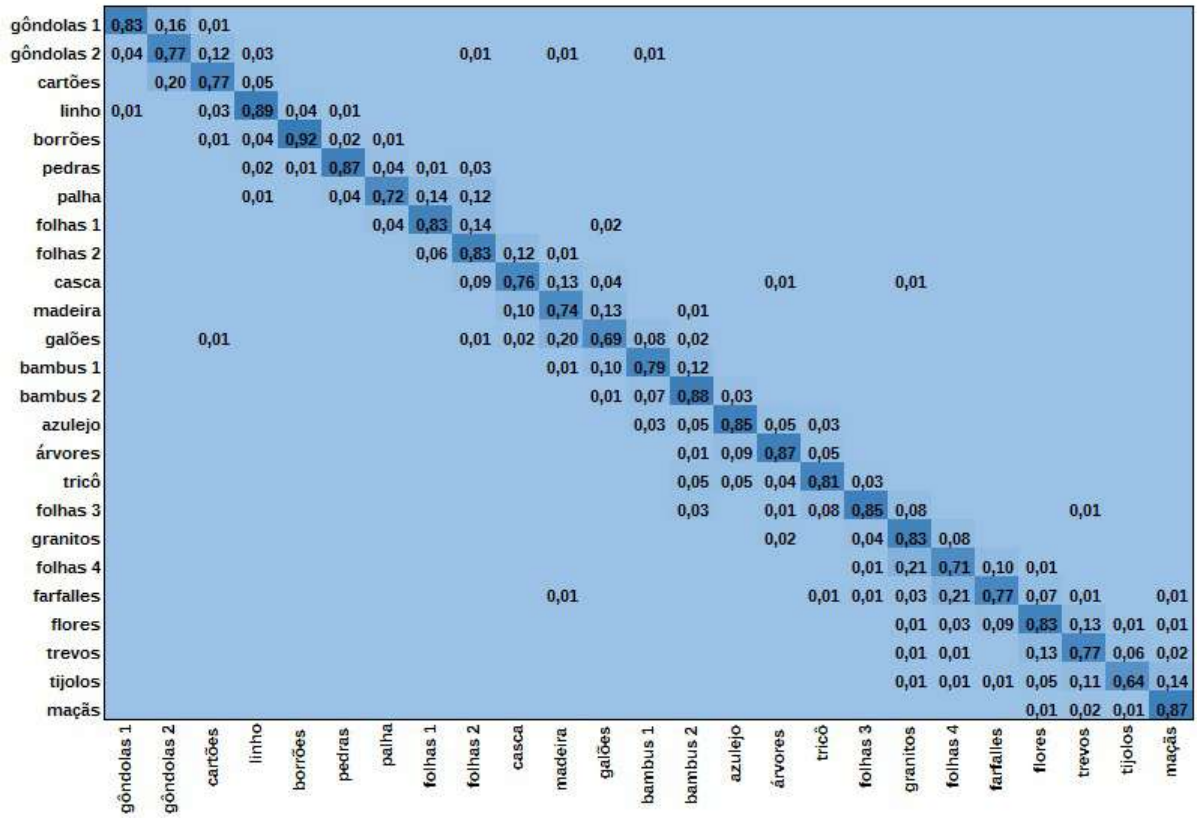


Figura 32 – Matriz de Confusão - UMD - KNN - EmpTot.

6.4.9 Matrizes de Confusão da Base UIUC

Novamente, podemos perceber na Figura 33 que, exceto na diagonal principal dessa matriz, todos os demais valores se aproximam muito de zero. Isto porque na base de dados UIUC o classificador MLP obteve um índice médio de acurácia de 0,993[17], já muito próximo portanto do acerto absoluto. Na Figura 33 veremos a matriz de confusão do classificador MLP na Base UIUC.

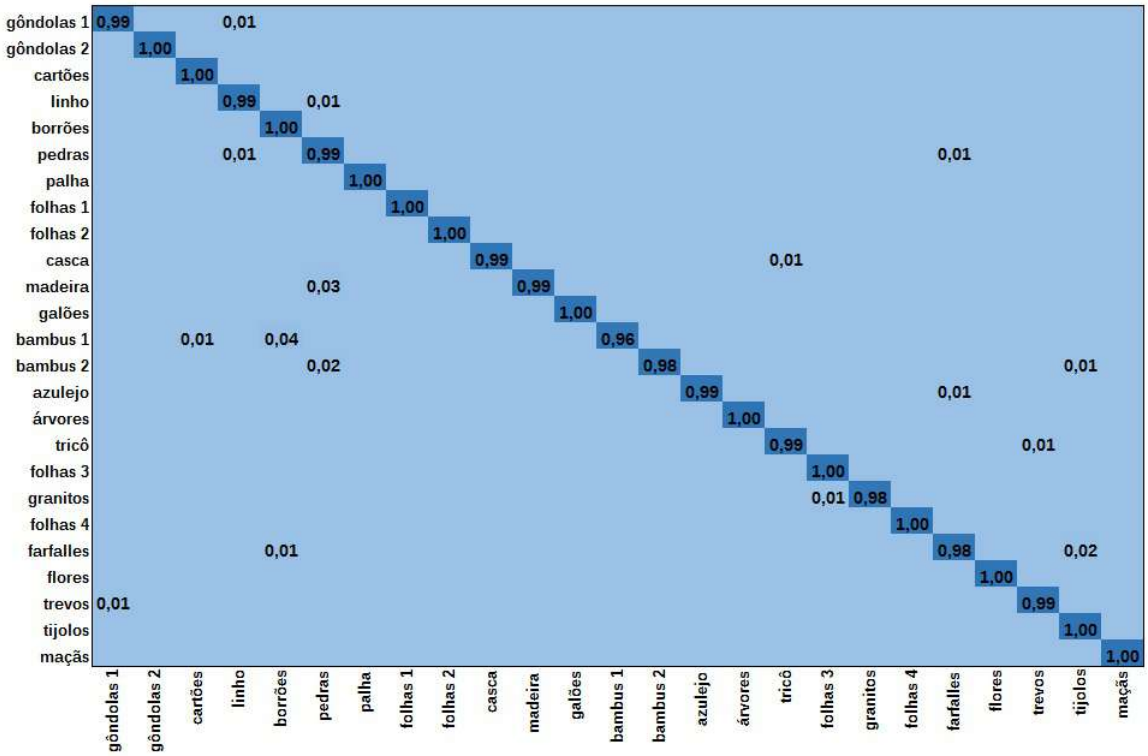


Figura 33 – Matriz de Confusão - UIUC - MLP - SHA.

6.5 Comparação com a literatura

As bases de dados KTH-TIPS-2b, FMD, UIUC e UMD têm sido objeto de estudo em vários outros trabalhos na literatura. Na Tabela 15[18], mostraremos os principais resultados que obtivemos e também os resultados obtidos por esses outros trabalhos.

Tabela 15 – Comparação de acurácias (%) em diferentes métodos para as bases de dados: KTH-TIPS-2b (HAYMAN et al., 2004), FMD (SHARAN; ROSENHOLTZ; ADELSON, 2009), UIUC (LAZEBNIK; SCHMID; PONCE, 2005), UMD (XU; JI; FERMüLLER, 2009).

Método	KTH	FMD	UIUC	UMD

[17] Vide Tabela 13.

[18] Para mantermos homogeneidade com os valores encontrados na literatura, estamos mostrando a acurácia em porcentagem na Tabela 15.

MFS (XU; JI; FERMüLLER, 2009)	-	-	92.74	93.93
VZ-MR8 (VARMA; ZISSERMAN, 2005)	46.3	22.1	92.94	-
LBP (OJALA; PIETIKäINEN; MäENPää, 2002)	50.5	-	88.36	96.15
VZ-Joint (VARMA; ZISSERMAN, 2009)	53.3	23.8	78.4	-
BSIF (KANNALA; RAHTU, 2012)	54.3	-	73.39	96.10
C.Liu et al.-aLDA (Liu et al., 2010)	-	44.60	-	-
LPQ (RAHTU et al., 2012)	54.4	-	-	-
CLBP (GUO; ZHANG; ZHANG, 2010)	57.3	43.6	95.75	98.62
MS-LBP-HF-KlSVM (SULC; MATAS, 2013)	-	-	96.4	-
PLS (Quan et al., 2014)	-	-	96.57	98.99
MRELBP (Liu et al., 2016)	-	-	96.88	-
SIFT+LLC (CIMPOI et al., 2014)	57.6	50.4	96.3	98.4
SIFT + KCB (CIMPOI et al., 2014)	58.3	45.1	91.4	98.0
SIFT + BoVW (CIMPOI et al., 2014)	58.4	49.5	96.1	98.1
LBP_{riu2}/VAR (OJALA; PIETIKäINEN; MäENPää, 2002)	58.52	-	84.44	95.88
$RandNet_{riu2}$ (NNC) (Chan et al., 2015)	-	-	48.20	87.40
$PCANet_{riu2}$ (NNC) (Chan et al., 2015)	-	-	49.80	85.67
PCANet (NNC) (Chan et al., 2015)	59.43	-	57.70	90.50
RandNet (NNC) (Chan et al., 2015)	60.67	-	56.57	90.87
SIFT + VLAD (CIMPOI et al., 2014)	63.1	52.6	96.5	99.3
OBIFs (TIMOFTE; GOOL, 2012)	-	55.78	-	-
ScatNet (NNC) (BRUNA; MALLAT, 2013)	63.66	-	88.64	93.36
OBIFs INNC (TIMOFTE; GOOL, 2012)	66.26	-	-	-
SIFT+IFV (CIMPOI et al., 2014)	69.3	58.2	97.0	99.2
Xu et al.-OTF (XU et al., 2009)	-	-	97.40	98.49

Método	KTH	FMD	UIUC	UMD
FV-CNN AlexNet (CIMPOI et al., 2016)	69.7	67.7	99.2	99.7
DeCAF (CIMPOI et al., 2014)	70.7	60.7	94.2	96.4
SIFT + FV (CIMPOI et al., 2016)	70.8	59.8	-	-
FC-CNN VGGM (CIMPOI et al., 2016)	71	70.3	94.5	97.2
FC-CNN AlexNet (CIMPOI et al., 2016)	71.5	64.8	91.1	95.9
(H+L)(S+R) (LAZEBNIK; SCHMID; PONCE, 2005)	-	-	97.02	96.95
Joint (VARMA; ZISSERMAN, 2009)	-	-	97.83	-
OTF (XU et al., 2012)	-	-	98.14	98.84
Zhang et. al (ZHANG et al., 2006)	-	-	98.3	-
WMFS (XU et al., 2010)	-	-	98.6	98.7
Varma and Ray (Varma; Ray, 2007)	-	-	98.76	-
BIF (CROSIER; GRIFFIN, 2008)	-	-	98.8	-
BIFs SRC (TIMOFTE; GOOL, 2012)	-	-	99.01	99.54
Scattering + multiscale train (SIFRE; MALLAT, 2013)	-	-	99.4	99.7
FC-CNN + FV-CNN AlexNet (CIMPOI et al., 2016)	72.1	71.4	99.3	99.7
FV-CNN VGGM (CIMPOI et al., 2016)	73.3	73.5	99.6	99.9
FC-CNN + FV-CNN VGGM (CIMPOI et al., 2016)	73.9	76.6	99.6	99.8
FC-CNN VGGVD (CIMPOI et al., 2016)	75.4	77.4	97.0	97.7
Deep-TEN Multi (ZHANG; XUE; DANA, 2017)	-	78.8	-	-
Ffirst (SULC; MATAS, 2015)	76.0	-	99.3	-
IFV + DeCAF (CIMPOI et al., 2014)	76.2	65.5	99.0	99.5
VisGraphNet (FLORINDO et al., 2021)	77.5	77.3	98.0	98.4
CNN	72.7	83.8	99.1	99.8
SHA	74.5	82.7	99.3	99.6
CHA	73.8	82.5	99.2	99.6
EmpTot	70.9	79.7	98.5	99.5
EmpSel	74.1	82.4	96.1	97.2
MisTot	73.4	82.9	99.2	99.7
MisSel	73.3	82.9	99.2	99.6
Vot	73.2	82.9	91.7	99.6

De um modo geral, nota-se aqui que a metodologia desenvolvida neste trabalho se mostrou competitiva frente ao estado-da-arte.

Particularmente, na base FMD, a solução estudada teve desempenho bem acima do reportado anteriormente. Esta é uma base de dados bastante desafiadora, contendo imagens que, embora semanticamente representem objetos relacionados, possuem uma ampla gama de variação em seus atributos visuais, por exemplo, tecidos dos mais variados tipos de materiais, cores e padrões, mas todos eles devendo ser identificados pelo algorimo como "tecido". Assim, o bom desempenho alcançado nesta base pode ser considerado notável tendo em conta a relativa simplicidade e pouca exigência computacional dos algoritmos aqui descritos.

Já na KTH-TIPS-2b, notamos um desempenho um pouco pior, mas ainda assim superando várias soluções mais complexas e caras computacionalmente, como algumas combinações de FC e FV-CNN (*Fisher vectors*) em (CIMPOI et al., 2016).

Aqui, a competitividade com o estado-da-arte em bases de *benchmark* é mais um indício do quanto os algoritmos aqui estudados e do quanto *ensembles* em um contexto geral podem ainda ser de grande valia como uma forma rápida e direta de melhorar a performance no reconhecimento de imagens de texturas.

A Tabela 16 traz a média de acurácias nos métodos que utilizamos para o banco de imagens CYST [19]. Essa tabela também traz as médias de acurácias do trabalho realizado por Florindo *et al.* (FLORINDO; BRUNO; LANDINI, 2017) sobre aquela base.

Tabela 16 – Comparação de acurácias em diferentes métodos para as base de dados: CYST e CYST_KS (SHARAN; ROSENHOLTZ; ADELSON, 2009).

Método	CYST	CYST_KS
CNN	99.4	99.3
SHA	99.4	98.8
CHA	99.4	98.8
EmpTot	99.3	98.9
EmpSel	89.5	98.9
MisTot	99.5	98.9
MisSel	99.5	98.8
Vot	99.2	98.8
Florindo *et al.* (FLORINDO; BRUNO; LANDINI, 2017)	72.0	68.0

[19] Onde está especificado CYST, usamos as classes K, R e S; onde está especificado CYST_KS, usamos apenas as classes K e S.

Observando os resultados da Tabela 16, concluímos que em todos os experimentos, a acurácia teve uma média maior que 98%, o que comparado aos resultados obtidos por Florindo *et al.* é um resultado muito acima do esperado, se comparado ao trabalho da referência.

Esses resultados nos dão base para afirmar que nossos experimentos obtiveram o sucesso desejado, e a técnica de utilizar a CNN como classificadora e provedora de descritores para um *ensemble* de classificadores na análise de imagens de cistos mandibulares pode gerar aplicações para o diagnóstico por imagens, nos casos de suspeita dos tipos de cistos mandibulares que estudamos.

7 Considerações Finais

Este trabalho teve um objetivo principal teórico e outro aplicado. O objetivo teórico era o de mostrar o funcionamento de um *ensemble* de classificadores com descritores fornecidos por uma CNN. O objetivo aplicado foi investigar o desempenho da CNN e do *ensemble* de classificadores usando vetores de descritores da CNN na classificação de imagens médicas.

Para atingir esses objetivos, usamos uma combinação de técnicas de aprendizado de máquina e de redes neurais (em particular, redes neurais convolucionais) e *ensembles* de classificadores para comparar os resultados de classificação de imagens de texturas, com aplicação particular em imagens de queratocistos bucais. Foi utilizada uma rede neural convolucional (CNN) baseada na arquitetura ResNet-18. O modelo foi validado na classificação de quatro bases de dados de texturas de *benchmark*: FMD, UMD, UIUC e KTH-TIPS-2b; e na aplicação em imagens médicas (base de dados CYST)[1].

Além de fornecer vetores de descritores aos *ensembles* de classificadores[2], a metodologia aqui desenvolvida realizou a classificação das imagens dessas bases de dados, obtendo assim seu próprio conjunto de índices de acurácia. Os vetores descritores fornecidos pela CNN foram utilizados pelos classificadores em sete métodos (Sem Hiperparâmetros Ajustados, Com Hiperparâmetros Ajustados, Empilhamento Total, Empilhamento Seletivo, Mistura Total, Mistura Seletiva e Votação), que se diferenciam pela forma como os classificadores estão operando e pela forma como os dados são utilizados por esses classificadores.

De um modo geral, embora já houvesse a expectativa de que a CNN e o *emsemble* de classificadores tivesse um bom desempenho na classificação de imagens de texturas em geral, incluindo o caso das imagens médicas de queratocistos, podemos considerar que os resultados mostraram-se acima da expectativa inicial. As Tabelas 15 e 16 mostram como os resultados de outros trabalhos obtiveram, em grande maioria, menores valores de acurácia na classificação das imagens que utilizamos, se comparados aos nossos resultados. Utilizar a CNN para fornecer os vetores de descritores aos classificadores mostrou-se um procedimento muito bem sucedido e com promissoras vantagens para fins práticos, conforme pode ser visto na Tabela 13. Com isso, podemos concluir que os objetivos teórico e aplicado desse trabalho foram atingidos.

Os bons resultados alcançados levam a uma constatação interessante: a de que técnicas já conhecidas do aprendizado de máquinas clássico ("pré-*deep learning*") podem

[1] Vide Capítulo 4.

[2] Os classificadores do *ensemble* foram assim nominados: MLP, RLOG, KNN, SVC, FA, AB, NB, AD e RLIN.

ainda contribuir muito com as arquiteturas profundas modernas. E o uso de *ensembles* é um exemplo nesta direção.

Observamos que a combinação de descritores profundos usando diferentes paradigmas trouxe ganhos de acurácia nos modelos de classificação investigados neste projeto. Em geral, estes resultados abrem perspectivas interessantes para o quanto essas combinações do clássico e do moderno em *deep learning* ainda precisam ser melhor exploradas.

Utilizar as bases de dados KTH-TIPS-2b, UIUC, UMD e FMD, com o intuito de comparar os resultados do algoritmo com outros trabalhos já consolidados, fomentou também uma maior confiabilidade nos resultados que obtivemos, uma vez que nossos resultados nessas bases de dados não fugiram do padrão de outras abordagens[3], mostrando que as acurácias obtidas com a base de dados CYST[4] têm a confiabilidade que desejávamos e a metodologia analisada tem potencial para trazer bons resultados ao ser utilizada com outras imagens de queratocistos e imagens médicas em geral.

Os equipamentos de geração de imagens médicas não envolvem apenas circuitos de alta complexidade eletrônica, mas também apresentam um refinado *software* de controle e operação do equipamento. Utilizar um modelo computacional "inteligente" como o que propomos neste trabalho nesses equipamentos pode diminuir o erro de diagnóstico dos queratocistos, aumentando as chances de um tratamento mais eficaz dessa odonto-patologia.

Porém, nosso *software* não possui interface para fins comerciais, então seriam necessários vários ajustes no código para ser utilizado por um profissional de saúde, por exemplo, e também o uso de plataformas de programação mais adaptadas ao contexto do usuário, para ser utilizado por um profissional não acostumado com plataformas de mais baixo nível como as usadas aqui para a implementação em Python. Mas, com esses ajustes, vemos como promissor o uso das técnicas que utilizamos no ambiente de mundo real do médico.

Cabe salientar, também, que utilizamos em todos os experimentos a mesma arquitetura de CNN, que é a grande geradora dos vetores de descritores para os classificadores. Podem ser feitos outros estudos mudando, por exemplo, a arquitetura da rede neural convolucional, utilizando uma ResNet-34 ou uma ResNet-50, por exemplo.

Para gerar o código utilizado neste trabalho, foi necessário realizar programação procedural e também orientada a objetos na linguagem Python. Para isso, foi necessária uma ampla pesquisa sobre essa linguagem, pois existem vários *frameworks*[5] projetados para aprendizado de máquina e redes neurais. Do mesmo modo, houve a necessidade da

[3] Vide Tabela 15.

[4] Vide Tabela 16.

[5] Um *framework* é um pacote da linguaguem de programação que fornece ferramentas para facilitar o desenvolvimento de um projeto.

compreensão dos processos matemáticos das redes neurais, em especial das redes convolucionais, para que os ajustes na rede fossem feitos sem prejudicar a lógica computacional. Todos esses fatores trouxeram uma abordagem da Matemática Aplicada Computacional que corroboraram para um trabalho com ganhos de aprendizado teórico e prático, tanto na Ciência da Computação quanto na Matemática Aplicada.

Por fim, este trabalho, por ter sido gerado em uma linguagem de programação de código aberto (Python) e por ter rodado em uma IDE aberta ao desenvolvimento científico (Google Colab), mostrou como a IA pode se desenvolver no meio acadêmico usando, entre outros recursos, as plataformas livres.

Referências

7GRAUS. *Dicio - Dicionário Online de Português*. 2020. Disponível em: <https://www.dicio.com.br/estomatologia/>. Cited on page 15.

ABDULHAY, E.; ELAMARAN, V.; CHANDRASEKAR, M.; BALAJI, V. S.; NARASIMHAN, K. Automated diagnosis of epilepsy from eeg signals using ensemble learning approach. *Pattern Recognition Letters*, v. 139, p. 174–181, 2020. Cited on page 16.

ACKLEY, D. H.; HINTON, G. E.; SEJNOWSKI, T. J. A learning algorithm for boltzmann machines. *Cognitive Science*, v. 9, p. 147–169, 1985. Cited on page 29.

ALBOY, R. G. *Técnicas de reconhecimento de imagem para incorporação em ferramentas de auxílio a deficientes visuais*. 83 p. Dissertação (Mestrado) — Universidade Federal de São Carlos, São Carlos, 2019. Cited on page 52.

AMARI, S. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, EC-16, p. 299 – 307, 1967. Cited on page 29.

ARAUJO, G. M. *Algoritmo para reconhecimento de características faciais baseado em filtros de correlação*. 78 p. Dissertação (Mestrado) — UFRJ/COOPE/Programa de Engenharia Elétrica, Rio de Janeiro, 2010. Cited on page 52.

BISNETO, C. R. de B. *Reconhecimento de objetos utilizando redes neurais artificiais e geometria fractal*. 92 p. Dissertação (Mestrado) — SENAI - CIMATEC, Salvador, 2011. Cited on page 52.

BITTENCOURT, H. R.; CLARKE, R. T. Um classificador baseado na discriminação logística: vantagens e desvantagens. *Anais X SBSR*, p. 1217–1223, 2001. Cited on page 52.

BOTTOU, L.; BOUSQUET, O. The tradeoffs of large scale learning. *in Neural Information Processing Systems*, v. 20, p. 161–168, 2007. Cited on page 45.

BRUNA, J.; MALLAT, S. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 8, p. 1872–1886, 2013. Cited on page 99.

BRYSON, A. E. A gradient method for optimizing multi-stage allocation processes. In: *Proc. Harvard Univ. Symposium on digital computers and their applications*. [S.l.: s.n.], 1961. Cited on page 28.

CAJAL, S. Ramón y. Histologie du système nerveux de l'homme et des vertébrés. *Maloine*, Paris, 1909. Cited on page 27.

Chan, T.; Jia, K.; Gao, S.; Lu, J.; Zeng, Z.; Ma, Y. PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, v. 24, n. 12, p. 5017–5032, 2015. Cited on page 99.

CHELLAPILLA, K.; PURI, S.; SIMARD, P. High performance convolutional neural networks for document processing. In: *International Workshop on Frontiers in Handwriting Recognition*. [S.l.: s.n.], 2006. Cited on page 31.

CIMPOI, M.; MAJI, S.; KOKKINOS, I.; MOHAMED, S.; VEDALDI, A. Describing textures in the wild. In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2014. (CVPR '14), p. 3606–3613. Cited 2 times on pages 99 and 100.

CIMPOI, M.; MAJI, S.; KOKKINOS, I.; VEDALDI, A. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision*, v. 118, n. 1, p. 65–94, 2016. Cited 2 times on pages 100 and 101.

CIRESAN, D.; MEIER, U.; MASCI, J.; SCHMIDHUBER, J. A committee of neural networks for traffic sign classification. *IEEE, International Joint Conference on Neural Networks (IJCNN)*, p. 1918–1921, 2011. Cited on page 31.

CROSIER, M.; GRIFFIN, L. D. Texture classification with a dictionary of basic image features. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2008. p. 1–7. Cited on page 100.

DAMASCENO, G. S. de S. *Reconhecimento Facial com variações de iluminação utilizando PCA e modificações da DCT associadas aos classificadores GMM, Naive Bayes e KNN*. 73 p. Dissertação (Mestrado) — Universidade Estadual do Ceará, Fortaleza, 2017. Cited on page 52.

DAVIES, R. W.; MORRIS, B. J. *Molecular Biology of the Neuron - Second Edition*. New York: Oxford University Press, 2004. ISBN 0-19-850998-7. Cited 2 times on pages 27 and 33.

FLORINDO, J. B.; BRUNO, O. M.; LANDINI, G. Morphological classification of odontogenic keratocysts using bouligand-minkowski fractal descriptors. *Computers in Biology and Medicine*, v. 81, p. 1 – 10, 2017. Cited 4 times on pages 59, 60, 62, and 101.

FLORINDO, J. B.; LEE, Y.-S.; JUN, K.; JEON, G.; ALBERTINI, M. K. Visgraphnet: A complex network interpretation of convolutional neural features. *Information Sciences*, v. 543, p. 296 – 308, 2021. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025520307131>. Cited on page 100.

FLORINDO, J. B.; SIKORA, M. S.; PEREIRA, E. C.; BRUNO, O. M. Characterization of nanostructured material images using fractal descriptors. *Physica A: Statistical Mechanics and its Applications*, v. 392, n. 7, p. 1694 – 1701, 2013. Cited 3 times on pages 15, 17, and 20.

FRYDENLUND, A.; ERAMIAN, M.; DALEY, T. Automatic classification of four types of developmental odontogenic cysts. *Computerized Medical Imaging Graphics*, v. 38, p. 151–162, 2014. Cited on page 17.

FUKUSHIMA, K. Neural network model for a mechanism of pattern recognition unaffected by shift in position - neocognitron. *Trans. IECE*, J62-A, n. 10, p. 658–665, 1979. Cited 4 times on pages 28, 29, 30, and 31.

GROSSBERG, S. Some networks that can learn, remember, and reproduce any number of complicated space-time patterns, I. *Journal of Mathematics and Mechanics*, v. 19, p. 53–91, 1969. Cited on page 28.

GUO, Z.; ZHANG, L.; ZHANG, D. A completed modeling of local binary pattern operator for texture classification. *Trans. Img. Proc.*, IEEE Press, v. 19, n. 6, p. 1657–1663, 2010. Cited on page 99.

HADAMARD, J. *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. [S.l.]: Mémoires présentés par divers savants à l'Académie des sciences de l'Institut de France: Éxtrait. Imprimerie nationale, 1908. Cited on page 28.

HANIN, B. Which neural net architectures give rise to exploding and vanishing gradients? *32nd Conference on Neural Information Processing Systems (NIPS)*, v. 31, 2018. Cited on page 30.

HAYKIN, S. *Neural Networks - A Comprehensive Foudation*. New Jersey: Prentice-Hall, 1994. ISBN 0-13-895863-7. Cited on page 39.

_____. *Redes Neurais - Princípios e Prática*. São Paulo - SP: Bookman, 2008. ISBN 0-13-273350-1. Cited 8 times on pages 27, 28, 29, 33, 35, 36, 37, and 38.

HAYMAN, E.; CAPUTO, B.; FRITZ, M.; EKLUNDH, J.-O. On the significance of real-world conditions for material classification. In: PAJDLA, T.; MATAS, J. (Ed.). *Computer Vision - ECCV 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 253–266. Cited 4 times on pages 10, 19, 58, and 98.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, p. 770–778, 2016. Cited 2 times on pages 49 and 50.

_____. Identity mappings in deep residual networks. *Computer Vision – ECCV 2016. Lecture Notes in Computer Science.*, v. 9908, 2016. Cited on page 50.

HEBB, D. O. *The Organization of Behavior*. [S.l.]: Wiley, New York, 1949. Cited 2 times on pages 27 and 28.

HOCHREITER, S. *Untersuchungen zu dynamischen neuronalen Netzen*. [S.l.]: Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991. Cited on page 30.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9(8), p. 1735–1780, 1997. Cited on page 30.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v. 2(5), p. 359–366, 1989. Cited on page 29.

IDOETA, I. V.; CAPUANO, F. G. *Elementos de Eletrônica Digital*. São Paulo: Editora Érica Ltda, 1984. Cited on page 23.

IVAKHNENKO, A. G. The group method of data handling – a rival of the method of stochastic approximation. *Soviet Automatic Control*, p. 13(3):43–55, 1971. Cited on page 28.

_____. Polynomial theory of complex systems. *IEEE Transactions on Systems, Man and Cybernetics*, v. 4, p. 364–378, 1971. Cited on page 28.

IVAKHNENKO, A. G.; LAPA, V. G. *Cybernetic Predicting Devices*. [S.l.]: CCM Information Corporation, 1965. Cited on page 28.

IVAKHNENKO, A. G.; LAPA, V. G.; MCDONOUGH, R. N. Cybernetics and forecasting techniques. *American Elsevier - N.Y.*, 1967. Cited on page 28.

JI, S.; XU, W.; YANG, M.; YU, K. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 1, p. 221–231, 2013. Cited on page 31.

KANNALA, J.; RAHTU, E. Bsif: Binarized statistical image features. In: *ICPR*. [S.l.]: IEEE Computer Society, 2012. p. 1363–1366. Cited on page 99.

KELLEY, H. J. Gradient theory of optimal flight paths. *ARS Journal*, v. 30, n. 10, p. 947–954, 1960. Cited on page 28.

KOLMOGOROV, A. N. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Doklady Akademii. Nauk USSR,*, v. 114, p. 679–681, 1965. Cited on page 29.

KRAMER, I. R. H. Changing views on oral disease. *Proceedings of the Royal Society of Medicine*, v. 67, p. 271–276, 1974. Cited on page 15.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing system*, 2012. Cited on page 31.

LANDINI, G. Quantitative analysis of the epithelial lining architecture in radicular cysts and odontogenic in radicular cysts and odontogenic keratocysts. *Head & Face Medicine*, v. 4, 2006. Cited on page 17.

LAZEBNIK, S.; SCHMID, C.; PONCE, J. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 27, n. 8, p. 1265–1278, 2005. Cited 5 times on pages 10, 19, 57, 98, and 100.

LECUN, Y. Generalization and network design strategies. *Departament of Computer Science - University of Toronto*, CRG-TR-89-4, 1989. Cited on page 29.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, p. 436–444, 2015. Cited 3 times on pages 16, 32, and 35.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Back-propagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, 1989. Cited 4 times on pages 16, 29, 30, and 45.

LEYSER, M. *Texturas Visuais em Embalagens para Cosméticos: estudo de caos de um produto no mercado*. 126 p. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012. Cited on page 57.

LINNAINMAA, S. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. [S.l.]: Master's thesis, Univ. Helsinki, 1970. Cited on page 29.

Liu, C.; Sharan, L.; Adelson, E. H.; Rosenholtz, R. Exploring features in a bayesian framework for material recognition. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2010. p. 239–246. Cited on page 99.

Liu, L.; Lao, S.; Fieguth, P. W.; Guo, Y.; Wang, X.; Pietikäinen, M. Median robust extended local binary pattern for texture classification. *IEEE Transactions on Image Processing*, v. 25, n. 3, p. 1368–1381, 2016. Cited on page 99.

LUGER, G. F. *Inteligência artificial: estruturas e estratégias para a resolução de problemas complexos*. Porto Alegre: Bookman, 2004. ISBN 85-363-0396-4. Cited 5 times on pages 22, 23, 24, 25, and 26.

LUZ, N. B. da; SANTOS, D. J. dos; ANTUNES, A. F. B. Segmentação de imagens e classificação baseada em regras de conhecimento como novas abordagens para o mapeamento do uso da terra no estado do paraná. *Anais XIV Simpósio Brasileiro de Sensoriamento - INPE, Natal, Brasil*, p. 989–996, 2009. Cited on page 52.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. *Proceedings of the* 30^{th} *International Conference on Machine Learning*, Atlanta, Georgia, USA, v. 28, n. 1, 2013. Cited on page 35.

MATERKA, A.; STRZELECKI, M. Texture analysis methods - a review. *Poland: Institute of Eletronics, Technical University of Lodz*, 1998. Cited on page 15.

MCCULLOCH, W. S.; PITTS, W. H. A logical calculus of the ideas immanent in nervous activity. *Reprinted from the Bulletin of Mathematical Biophysics*, v. 5, 1943. Cited 3 times on pages 16, 27, and 34.

MEDINA-QUERO, J.; ZHANG, S.; NUGENT, C.; ESPINILLA, M. Ensemble classifier of long short-term memory with fuzzy temporal windows on binary sensors for activity recognition. *Expert System with Applications*, v. 114, p. 441–453, 2018. Cited on page 16.

MENEZES, J. M. P. de; RODRIGUES, F. I. B.; CARVALHO, R. R. M.; ANDRADE, B. H. S. Análise do reconhecimento de faces utilizando algoritmo de knn. *Congresso Técnico Científico da Engenharia e da Agronomia - CONTECC'2017*, p. 1–5, 2017. Cited on page 52.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. *Machine Learning: An Artificial Intelligence Approach*. USA: Morgan Kaufmann, 1986. ISBN 0-934613-09-5. Cited 2 times on pages 22 and 26.

MINSKY, M. L.; PAPERT, S. A. *Perceptrons: An Introduction to Computational Geometry*. [S.l.]: MIT Press, 1969. Cited on page 29.

OJALA, T.; PIETIKäINEN, M.; MäENPää, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 7, p. 971–987, 2002. Cited on page 99.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE, Trans. on Knowl. and Data Eng.*, v. 22, n. 10, p. 1345–1359, 2010. Cited on page 32.

Quan, Y.; Xu, Y.; Sun, Y.; Luo, Y. Lacunarity analysis on image patterns for texture classification. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 160–167. Cited on page 99.

RAHTU, E.; HEIKKILä, J.; OJANSIVU, V.; AHONEN, T. Local phase quantization for blur-insensitive image analysis. *Image Vision Comput.*, v. 30, n. 8, p. 501–512, 2012. Cited on page 99.

RANZATO, M. A.; HUANG, F.; BOUREAU, Y.; LECUN, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. *In Proc. computer vision and patten recognition conference*, IEE Press, p. 1–8, 2007. Cited on page 31.

ROSEMBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, Cornell Aeronautical Laboratory, p. 386–408, 1958. Cited 4 times on pages 28, 36, 37, and 38.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. *In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing*, v. 1, p. 318–362, 1986. Cited on page 29.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, v. 115.2, p. 211–252, 2015. Cited on page 32.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, v. 61, p. 85–117, 2015. Cited 2 times on pages 30 and 46.

SHARAN, L.; ROSENHOLTZ, R.; ADELSON, E. H. Material perceprion: What can you see in a brief glance? *Journal of Vision*, v. 9, n. 8, p. 784, 2009. Cited 5 times on pages 10, 19, 59, 98, and 101.

SHEAR, M.; SPEIGHT, P. M. *Cysts of the Oral and Maxillofacial Regions*. [S.l.: s.n.], 2007. ISBN 978-14051-4937-2. Cited on page 15.

SHEPHERD, A. J. *Second-Order Methods for Neural Networks – Fast and Reliable Methods for Multi-Layer Perceptrons*. [S.l.]: Springer, 1997. Cited on page 46.

SIFRE, L.; MALLAT, S. Rotation, scaling and deformation invariant scattering for texture discrimination. In: *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. USA: IEEE Computer Society, 2013. (CVPR '13), p. 1233–1240. Cited on page 100.

STOELINGA, P.; PETERS, J. A note on the origin of keratocysts of the jaws. *Int. J. Oral Surg*, v. 2, p. 37–44, 1973. Cited on page 60.

SULC, M.; MATAS, J. Kernel-mapped histograms of multi-scale lbps for tree bark recognition. In: . [S.l.: s.n.], 2013. p. 82–87. Cited on page 99.

_____. Fast features invariant to rotation and scale of texture. In: AGAPITO, L.; BRONSTEIN, M. M.; ROTHER, C. (Ed.). *Computer Vision - ECCV 2014 Workshops*. Cham: Springer International Publishing, 2015. p. 47–62. Cited on page 100.

THEPADE, S. D.; CHAUDHARI, P. R. Land usage identification with fusion of thepade sbtc and sauvola thresholding features of aerial images using ensemble of machine learning algorithms. *Applied Artificial Intelligence*, 2020. Cited on page 16.

TIMOFTE, R.; GOOL, L. V. A training-free classification framework for textures, writers, and materials. In: *BMVC*. [S.l.: s.n.], 2012. Cited 2 times on pages 99 and 100.

VAPNIK, V. *The Nature of Statistical Learning Theory*. [S.l.]: Springer, New York, 1995. Cited on page 30.

Varma, M.; Ray, D. Learning the discriminative power-invariance trade-off. In: *2007 IEEE 11th International Conference on Computer Vision*. [S.l.: s.n.], 2007. p. 1–8. Cited on page 100.

VARMA, M.; ZISSERMAN, A. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, v. 62, n. 1, p. 61–81, 2005. Cited on page 99.

_____. A statistical approach to material classification using image patch exemplars. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 31, n. 11, p. 2032–2047, 2009. Cited 2 times on pages 99 and 100.

WIDROW, B.; HOFF, M. Associative storage and retrieval of digital information in networks of adaptive neurons. *Biological Prototypes and Synthetic Systems*, v. 1, p. 160, 1962. Cited on page 28.

WIDROW, B.; HOFF, M. E. Adaptive switching circuits. *IRE WESCON Convention Record*, p. 96–104, 1960. Cited on page 36.

WIESEL, D. H.; HUBEL, T. N. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.*, v. 148, p. 574–591, 1959. Cited on page 28.

XU, Y.; HUANG, S.; JI, H.; FERMüLLER, C. Scale-space texture description on sift-like textons. *Comput. Vis. Image Underst.*, v. 116, n. 9, p. 999–1013, 2012. Cited on page 100.

XU, Y.; HUANG, S.-B.; JI, H.; FERMÜLLER, C. Combining powerful local and global statistics for texture description. In: IEEE. *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*. [S.l.]: IEEE, 2009. p. 573 – 580. Cited on page 99.

XU, Y.; JI, H.; FERMüLLER, C. Viewpoint invariant texture description using fractal analysis. *International Journal of Computer Vision*, v. 83, n. 1, p. 85–100, 2009. Cited 5 times on pages 10, 19, 58, 98, and 99.

XU, Y.; YANG, X.; LING, H.; JI, H. A new texture descriptor using multifractal analysis in multi-orientation wavelet pyramid. In: . [S.l.: s.n.], 2010. p. 161–168. Cited on page 100.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science, ECCV 2014*, v. 8689, p. 818–833, 2014. Cited on page 32.

ZHANG, H.; XUE, J.; DANA, K. Deep ten: Texture encoding network. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017. Cited on page 100.

ZHANG, J.; MARSZALEK, M.; LAZEBNIK, S.; SCHMID, C. Local features and kernels for classification of texture and object categories: A comprehensive study. In: *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*. USA: IEEE Computer Society, 2006. (CVPRW '06), p. 13. Cited on page 100.