UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

MAURÍCIO KOENIGKAM SANTOS

REDES NEURAIS NA CLASSIFICAÇÃO DE IMAGENS DA MEMBRANA TIMPÂNICA

CAXIAS DO SUL
2018

MAURÍCIO KOENIGKAM SANTOS

REDES NEURAIS NA CLASSIFICAÇÃO DE IMAGENS DA MEMBRANA TIMPÂNICA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Tecnologias Digitais na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador:
Prof.ª Dra. Carine Geltrudes Webber

CAXIAS DO SUL
2018

MAURÍCIO KOENIGKAM SANTOS

REDES NEURAIS NA CLASSIFICAÇÃO DE IMAGENS DA MEMBRANA TIMPÂNICA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Tecnologias Digitais na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 11/12/2018

Banca Examinadora

_____
Prof.ª Dra. Carine Geltrudes Webber
Universidade de Caxias do Sul - UCS

_____
Prof.ª Dra. Elisa Boff
Universidade de Caxias do Sul - UCS

_____
Prof. Me. Marcelo Luís Fardo
Universidade de Caxias do Sul - UCS

À minha esposa Bruna e meu filho Gustavo.

"*Tudo o que temos de decidir é o que fazer com o tempo que nos é dado.*"
Gandalf, o Cinzento

RESUMO

As otites são o principal motivo de visita aos médicos por crianças na idade pré-escolar. No mundo, acredita-se que aproximadamente 740 milhões de pessoas por ano serão afetadas por otite média aguda ou otite média crônica supurada (LUNDBERG et al., 2017). A computação vem cada vez mais ajudando os profissionais na área da saúde no diagnóstico, prognóstico e tratamento das doenças. A classificação de imagens na área médica, principalmente pelas redes neurais de aprendizado profundo, é uma importante ferramenta no auxílio do diagnóstico de doenças, como a retinopatia diabética. A radiologia vem usando as redes neurais para ajudar a identificar tumores em exames de tomografia, por exemplo. A dermatologia também usa esses métodos para ajudar a interpretar imagens de manchas na pele, com o objetivo de identificar possíveis melanomas. Esses exemplos, como os de outras áreas da medicina, mostram o a importância que as redes neurais vêm tendo na classificação de imagens. Neste trabalho foram coletadas imagens do conduto auditivo externo humano por médico otorrinolaringologista, sendo que 133 destas foram submetidas ao processo de Extração de Conhecimento em Bancos de Dados. Foi construído um modelo de Rede Neural Convolucional para classificação destas imagens em três classes: "sem doença", "com doença"e "com cerumen". Durante o treinamento, os dados foram divididos em partições, e a acurácia média dos testes com todas as partições foi de 81.62%, sendo o melhor resultado de 91.26%. O modelo final foi submetido ao teste com 14 novas imagens, apresentando uma acurácia de 90.47%.

Palavras-chave: Inteligência Artificial. Aprendizado de Máquina. Rede Neural de Aprendizado Profundo. Rede Neural Convolucional. Imagens da Membrana Timpânica.

ABSTRACT

Otites are the main reason for preschoolers to visit a doctor. In the world, it is believed that approximately 740 million people per year will be affected by acute otitis media or suppurative chronic otitis media. Computers are increasingly helping health professionals to diagnose, prognosis and treat diseases. The interpretation of images in the medical field, mainly by deep learning neural networks, is an important tool in the aid of the diagnosis of diseases such as diabetic retinopathy. Radiology has been using neural networks to help identify tumors on CT scans, for example. Dermatology also uses these methods to help interpret images of skin blemishes to identify possible melanomas. These examples, like those in other areas of medicine, show the importance of neural networks in classifying images. In this work, images of the human external auditory canal were collected by otorhinolaryngologist, of which 133 were submitted to the Knowledge Discovery in Databases process. A Convolutional Neural Network model was constructed to classify these images into three classes: "without disease", "with disease" and "with cerumen". During training, the data was partitioned, and the average accuracy of all partitions was 81.62 %, the best result being 91.26 %. The final model was tested with 14 new images, presenting an accuracy of 90.47 %.

Keywords: Artificial Intelligence. Machine Learning. Deep Learning. Convolutional Neural network. Tympanic Membrane Images.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

IA	Inteligência artificial
AM	Aprendizado de máquina
RNAP	Rede neural de aprendizado profundo
MT	Membrana timpânica
GPU	Unidade de processamento gráfico
RNN	Rede neural recorrente
LSTM	Memória de longo prazo
GRU	Unidade recorrente bloqueada
CNN	Rede neural convolucional
DBN	Rede de crenças profundas
DSN	Rede de empilhamento profundo
RNM	Ressonância nuclear magnética
VM	Máquina virtual

SUMÁRIO

1 INTRODUÇÃO

A inteligência artificial (IA) pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente (LUGER, 2013). Com essa definição, o autor quis mostrar que o estudo da IA deve ser vinculado ao estudo das técnicas de programação. Isto leva à pesquisa nas áreas de representação dos dados, linguagens de programação e desenvolvimento de algoritmos. Porém esta definição também levanta alguns questionamentos quando se refere ao comportamento inteligente (LUGER, 2013), pois existe uma controvérsia sobre o que seria tal comportamento. Seria só o ser humano dotado de inteligência? O avanço nas pesquisas sobre IA vêm trazendo vários questionamentos sobre a inteligência humana, sobre como o cérebro humano funciona e como ele aprende com as experiências vividas. Como o ser humano desenvolve sua linguagem? E duas questões foram importantes para inspirar o desenvolvimento desse trabalho: Pode uma máquina aprender com suas experiências de modo semelhante ao ser humano? Que benefícios esse aprendizado de máquina poderia trazer para a humanidade?

O aprendizado de máquina (AM) é um campo de estudo dentro da IA. “Desde que o computador foi inventado, nos perguntamos se eles poderiam aprender. Se pudéssemos entender como programá-los para aprender - para melhorar automaticamente com a experiência - o impacto seria dramático.” (p.1 MITCHELL, 1997, tradução nossa). Além de ajudar na compreensão da inteligência humana, o AM poderia trazer vários benefícios no cotidiano do ser humano, como carros que dirigem sozinhos ou programas que detectam o uso fraudulento de cartões de crédito.

Observamos um crescente aumento no desenvolvimento de programas voltados para a área médica que utilizam o AM para tentar ajudar no diagnóstico, tratamento e prever o prognóstico de doenças. Um exemplo bem conhecido é o supercomputador Watson, da IBM, que ficou famoso por vencer o popular programa de TV de perguntas e respostas “Jeopardy”, em fevereiro de 2011. Atualmente, esse computador vem ajudando médicos na escolha do tratamento do câncer através de recomendações, após analisar inúmeros dados presentes em publicações médicas(IBM, 2015). Em um estudo duplo-cego, apresentado no San Antonio Breast Cancer Symposium em 2016, os médicos do Manipal Hospitals descobriram que o Watson estava de acordo com as recomendações do conselho de tumores em 90% dos casos de câncer de mama.

O trabalho proposto está direcionado para a classificação de imagens na área médica, com a finalidade de ajudar no exame físico e no diagnóstico de doenças do ouvido. O autor é médico especialista em otorrinolaringologia pela Associação Brasileira de Otorrinolaringologia e Cirurgia Cérvico-Facial(ABORL-CCF). O AM já vem sendo utilizado na classificação de imagens para auxiliar médicos nos diagnósticos de doenças. Um estudo obteve ótimos resultados ao analisar imagens da retina de seres humanos (exame de fundo de olho) com o objetivo de detectar lesões que ocorrem em pessoas com diabetes mellitus e que podem levar à cegueira (PIRES et al., 2015). Este autor utilizou um total de 1014 imagens do Hospital de Olhos de São

Paulo para treinar seu programa, e conseguiu aplicá-lo em um hospital na Austrália, obtendo uma sensibilidade de 100% e especificidade de 88.9% na detecção de lesões brilhantes na retina, por exemplo. Para isso, ele utilizou uma metodologia chamada *Bag of Visual Words*, que é um método utilizado na classificação computacional de imagens, tratando características da imagem como palavras e usando algoritmos de agrupamento ou com uma seleção aleatória.

Um outro estudo também importante que envolve a análise de imagens médicas por um programa de computador foi realizado por pesquisadores brasileiros (FORNACIALI; CARVALHO; BITTENCOURT, 2016). Nesse estudo, os pesquisadores procuraram comparar diferentes técnicas de aprendizado de máquina na classificação de imagens de lesões de pele, com o objetivo de diagnosticar o melanoma. Este é o câncer de pele que mais mata, porém seu prognóstico é muito bom se a lesão na pele for detectada de modo precoce. Nesse estudo, os autores viram que a metodologia do *Bag of Words* avançada atingiu uma acurácia de 84.6%, e um modelo baseado em redes neurais obteve a acurácia de 89.3%. Como vimos, existem diferentes técnicas sendo utilizadas na classificação de imagens médicas para o AM. Um algoritmo que vem sendo muito utilizado para esse objetivo se chama *Deep Neural Net* ou Rede Neural de Aprendizado Profundo (RNAP), que é um algoritmo que usa uma estrutura parecida com uma rede de neurônios, mas com muitas camadas que formam uma trama profunda e complexa (GOODFELLOW; BENGIO; COURVILLE, 2016). Uma pesquisa analisou 308 trabalhos que utilizaram esse algoritmo na classificação automática de imagens médicas (LITJENS et al., 2017). Observou-se o uso dessa técnica em diversas áreas da medicina como neurologia, oftalmologia (retina), pneumologia, patologia digital, tórax, cardiologia, abdome e musculoesquelético. O mais interessante foi o fato que de 308 trabalhos analisados, 242 foram publicados entre 2016 ou no primeiro mês de 2017, o que mostra que a RNAP é uma técnica que vem crescendo muito no uso do AM em análise de imagens, principalmente na área médica.

Este trabalho tem o objetivo de classificar, de modo automático, imagens da membrana timpânica humana, podendo assim ajudar no diagnóstico de otite média. “Otite média representa uma condição inflamatória dos espaços da orelha média e da mastoide, sem referência para sua etiologia ou patogênese.” (BAILEY; JOHNSON, 2006, p.1265, tradução nossa). Essa doença é o principal motivo de visita aos médicos por crianças na idade pré-escolar. Estudos realizados nos Estados Unidos mostram que 71% das crianças com até 3 anos de idade tiveram pelo menos uma otite média aguda, sendo que 33% tiveram três ou mais episódios (TEELE; KLEIN; ROSNER, 1980). Nesse mesmo país, o custo anual para o tratamento de otite média em crianças menores que 13 anos de idade é de 4 bilhões de dólares (BAILEY; JOHNSON, 2006). No mundo, acredita-se que aproximadamente 740 milhões de pessoas por ano serão afetadas por otite média aguda ou otite média crônica supurada, e aos 10 anos de idade pelo menos 90% das crianças terão sido acometidas por otite média com efusão (LUNDBERG et al., 2017). O diagnóstico é muito importante em crianças menores que 1 ano de idade, pois a chance de ocorrer otites de repetição nessas crianças é grande. Um estudo realizado por Kuruvilla et al. propôs uma classificação automática de dois tipos de otite (otite média aguda e otite média

com efusão) baseada na captura de imagens através da otoscopia (KURUVILLA; SHAIKH; KOVAČEVIĆ, 2013). Os autores utilizaram algumas técnicas para trabalhar as imagens, com o objetivo de eliminar obstáculos do conduto auditivo, eliminar problemas de luminosidade e identificar a membrana timpânica. Desse modo, puderam obter várias características da imagem da membrana (luz, opacidade, concavidade, posição do martelo, etc.) e usaram um algoritmo de vocabulário para fazer a classificação. Obtiveram uma acurácia de 89.9%, maior que à dos médicos não especialistas, segundo o estudo. O objetivo principal deste estudo é avaliar se os algoritmos de redes neurais são capazes de classificar imagens da membrana timpânica humana a fim de identificar se há doença na orelha média. É importante ressaltar que o ato de classificar imagens é diferente de realizar um diagnóstico médico. Este é realizado por profissionais, que além de analisarem a imagem da membrana timpânica através da otoscopia, obtêm todas as informações sobre a história clínica do paciente.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo principal deste trabalho é avaliar se os algoritmos de redes neurais são capazes de classificar imagens da membrana timpânica humana a fim de identificar se há doença na orelha média.

1.1.2 Objetivos Específicos

Os objetivos específicos são os seguintes:

- Identificar métodos de redes neurais de aprendizado profundo apropriados para classificação de imagens da membrana timpânica.
- Aplicar um algoritmo de redes neurais a fim de obter resultados referentes à classificação de imagens da membrana timpânica.
- Comparar os resultados obtidos com trabalhos relacionados na classificação de imagens médicas.

1.2 ORGANIZAÇÃO DO DOCUMENTO

O trabalho está estruturado em cinco capítulos. O primeiro apresenta a introdução e os objetivos do trabalho, desse modo o leitor pode saber suas principais características. O capítulo 2 explica a localização e função da membrana timpânica, e aponta os tipos de otites que causam alterações nessa estrutura. O capítulo 3 apresenta vários conceitos importantes relacionados

com as redes neurais, como o que é aprendizado de máquina, extração de conhecimento, mineração de dados, métodos para amostragem, redes neurais de aprendizado profundo com suas arquiteturas e suas aplicações na área médica. O capítulo 4 mostra a proposta de implementação de um modelo de rede neural para classificação de imagens da membrana timpânica. Por fim, o capítulo 5 apresenta como foi desenvolvido este modelo e os resultados obtidos pelo mesmo.

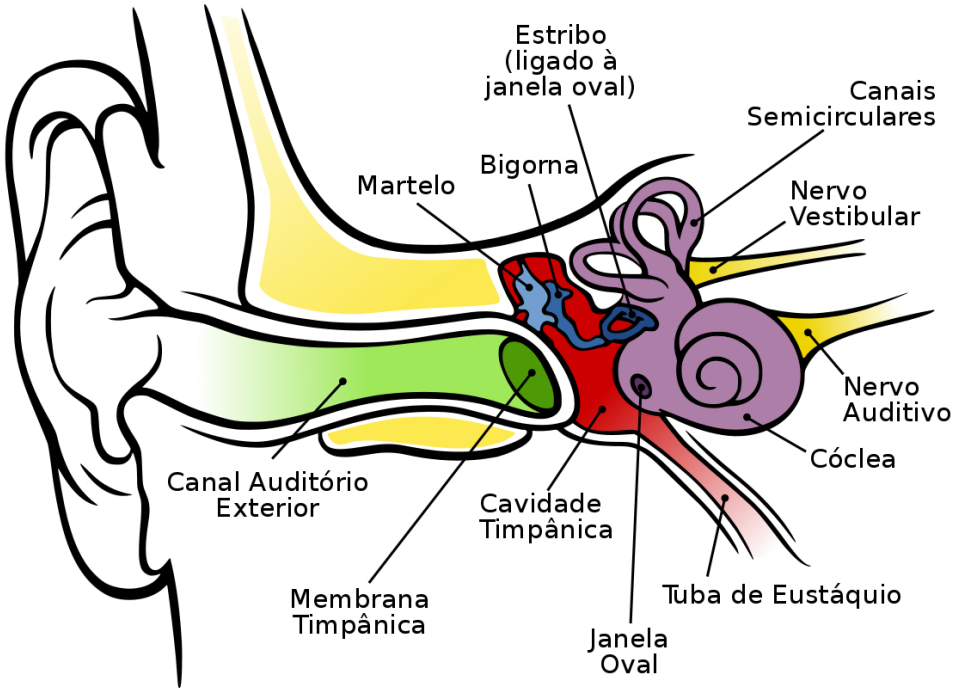
2 MEMBRANA TIMPÂNICA

Este trabalho irá conter algumas imagens da membrana timpânica (MT) humana. Por isso é importante informar onde se localiza essa estrutura, quais suas características, qual sua função no corpo humano e quais alterações acontecem durante uma otite média.

2.1 ANATOMIA DA ORELHA

A orelha é dividida em 3 partes: orelha externa, orelha média e orelha interna. A orelha externa é formada pelo pavilhão auricular e pelo meato acústico externo (ou conduto auditivo externo). Sua principal função é captar o som e direcioná-lo para a orelha média. A orelha média é um espaço no osso temporal que vai da MT (inclusive) até a orelha interna, veja a região vermelha na Figura 1. Ela contém 3 ossículos que vibram com a passagem do som, além de um canal (Tuba de Eustáquio) que tem a função de possibilitar a passagem de ar vindo do nariz. "A orelha média transmite energia acústica do conduto auditivo externo cheio de ar para a cóclea cheia de líquido"(p.1884 BAILEY; JOHNSON, 2006, tradução nossa). A orelha interna contém a cóclea, responsável pela audição e os vestíbulos e canais semicirculares, ligados ao equilíbrio.

Figura 1 – Anatomia da orelha

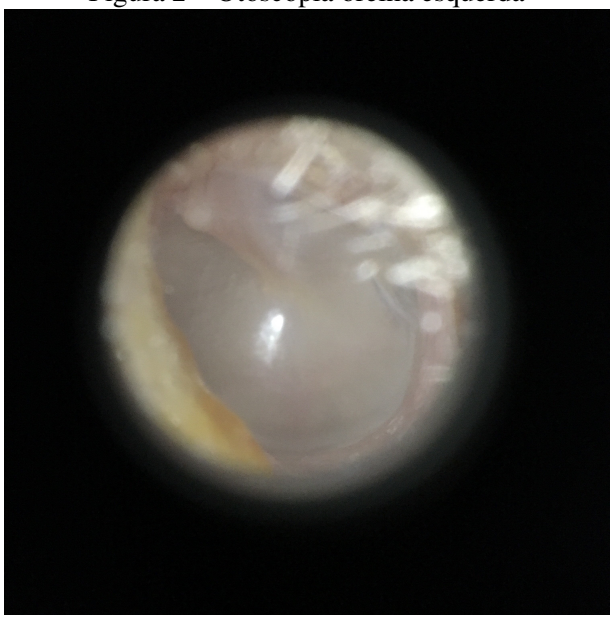


Fonte: Wikipédia

A MT separa o conduto auditivo externo da cavidade da orelha média. Quando essa

membrana é examinada através de uma otoscopia, ela se apresenta como translúcida, com brilho (geralmente um triângulo luminoso), com o martelo visível (podendo aparecer todos os 3 ossículos) e com vascularização. A Figura 2 apresenta o exemplo de uma otoscopia em uma orelha esquerda, no qual as características mencionadas podem ser observadas.

Figura 2 – Otoscopia orelha esquerda



Fonte: autor

2.2 OTITE MÉDIA

Otite média é uma condição inflamatória, que pode ou não ter causa infecciosa, que acomete o espaço da orelha média e, por isso, ocasiona alterações na MT. Essas alterações podem incluir hiperemia (vermelhidão), perda da transparência, abaulamentos, aumento da vascularização, perfuração, saída de secreção, entre outras. As otites podem ser divididas em agudas ou crônicas. Otite média aguda é a presença de secreção na orelha média associada à instalação rápida de sinais e sintomas de infecção aguda dessa região, como otalgia (dor de ouvido), febre e irritabilidade (BLUESTONE et al., 2002). Geralmente a MT fica com hiperemia, abaulada e pode até apresentar perfuração e saída de secreção (otite média aguda supurada). Otite média crônica é definida quando ocorre uma perfuração persistente da MT, geralmente mais de 3 meses. Pode ser simples, quando não há sinais de infecção, como secreção purulenta. Pode ser supurativa, quando apresenta secreção, tendo algumas causas para que isso ocorra, como infecção bacteriana ou colesteatoma, por exemplo. Otite média com efusão é uma inflamação crônica da orelha média em que uma coleção de líquido está presente na cavidade timpânica, porém há ausência de sinais e de sintomas de infecção aguda (BLUESTONE et al., 2002). Nesse caso a MT pode apresentar aumento da vascularização, retração e é possível ver o líquido na orelha média pela transparência da membrana. Essa patologia ocorre por falta de ventilação

da cavidade da orelha média, geralmente por disfunção da Tuba de Eustáquio, ou pode ocorrer como resolução da otite média aguda.

3 REDES NEURAIS DE APRENDIZADO PROFUNDO

Neste capítulo serão discutidos alguns conceitos com o objetivo de elucidar como funciona a RNAP, um mecanismo de AM que será utilizado no desenvolvimento de um programa para solucionar o problema proposto. Devemos estudar o que é AM e esclarecer alguns parâmetros importantes para seu funcionamento, mostrar seu processo de aprendizado e como analisar os seus resultados. Vamos mostrar como funcionam as RNAP, suas principais arquiteturas, quais são as ferramentas hoje disponíveis que facilitam sua utilização (pacotes, bibliotecas, plataformas), apontar como e para quê esse algoritmo vem sendo usado, principalmente na área de análise de imagens médicas.

3.1 APRENDIZADO DE MÁQUINA

Aprendizado de Máquina é um dos campos de estudo da IA. Os desenvolvedores que criam programas que utilizam IA procuram fazer com que esses programas consigam recriar a forma como a nossa inteligência funciona para resolver problemas, esses programas tentam imitar o raciocínio humano em diversos campos. AM é um campo que tenta imitar uma dessas capacidades do cérebro humano, que é aprender com a experiência. Mitchell define que "um programa de computador aprende com a experiência E em relação a alguma classe de tarefas T e medida de desempenho P, se seu desempenho em tarefas em T, que é medido por P, melhora com a experiência E." (p.2 MITCHELL, 1997, tradução nossa). O autor utiliza como exemplo um programa de computador que aprende a jogar Damas, em que a experiência (E) é jogar jogos de Damas contra si próprio, a medida de desempenho (P) é o número de vitórias e a tarefa (T) é jogar Damas. Isso significa que um programa seria capaz de melhorar seu desempenho com um treinamento, algo parecido com que acontece com o ser humano. Esse aprendizado que pode ocorrer em um programa mudou drasticamente os rumos da computação, pois abriram-se várias novas oportunidades para o uso dessas máquinas. Já existem programas que podem reconhecer a linguagem falada, podem prever a taxa de recuperação de pacientes com pneumonia, detectam o uso fraudulento de cartões de crédito baseado nos padrões do usuário, podem dirigir carros de modo automático e detectam e analisam imagens, inclusive na área da medicina. Para desenvolver esses programas, vários parâmetros tiveram que ser escolhidos, testados e retestados até que os resultados fossem satisfatórios. Em geral, para desenvolvermos um programa com a capacidade de aprender devemos identificar como realizar as três características apontadas por Mitchell: a tarefa a ser cumprida, a medida de desempenho do programa que deverá ser melhorado com o tempo e o treinamento, ou seja, como o programa vai ganhar experiência. Então deveremos seguir algumas etapas para escolher os melhores métodos ou algoritmos para que nosso programa possa resolver o problema proposto de modo mais eficiente possível.

A primeira etapa para o desenvolvimento de um programa inteligente é escolher o tipo de treinamento que esse programa vai receber. O método de treinamento pode ser do tipo di-

reto, ou seja, a cada passo no treino o programa vai receber um *feedback* informando se aquele passo foi correto (MITCHELL, 1997). No exemplo do jogo de Damas, o programa vai receber a informação da jogada mais correta para todos os estados possíveis. Ou o método pode ser do tipo indireto. No caso do jogo citado, apenas dizemos quais são os movimentos possíveis, o programa vai jogar até o final e então indicamos se o jogo foi vitorioso ou não. Desse modo, o programa vai registrar que aqueles movimentos realizados levaram à vitória, por exemplo, e vai marcá-los positivamente. Com o passar dos treinos, ele vai saber quais são os movimentos mais eficientes baseado na taxa de vitórias ao realizá-los. Mas isso pode ocasionar alguns problemas, pois mesmo tendo realizados movimentos ótimos no início do jogo, o resultado final pode ser uma derrota, isso vai exigir um maior número de treinamentos para que o resultado seja satisfatório. Outro atributo do método de treinamento é o grau em que o aprendiz controla a sequência dos exemplos de treinamento. O aprendiz pode em cada estado do problema receber informações do professor dizendo qual é o passo correto. Ou o aprendiz pode apenas solicitar informações ao professor sobre estados complicados, em que o aluno está tendo dificuldades. Outra possibilidade é o aprendiz ter total controle sobre os estados, apenas sabendo no final se obteve sucesso ou não. Nesse caso, ele vai ter que decidir se em cada estado tenta uma nova forma de abordagem, ou tenta aprimorar uma abordagem anterior que julga ter tido sucesso. Um terceiro atributo importante no método de treinamento é o grau de semelhança dos exemplos de treinamento com os exemplos da avaliação de desempenho. No exemplo do jogo de Damas, o fato do programa estar treinando jogando contra si é um risco para o desempenho. Pois o propósito do programa seria enfrentar humanos, que certamente usarão estratégias diferentes da prevista pelo computador. Podemos classificar os algoritmos de treinamento em aprendizado supervisionado ou aprendizado não supervisionado. No supervisionado os exemplos de treinamento já foram previamente classificados por um professor. Nesse método, "o algoritmo de aprendizado sempre recebe a informação se uma ocorrência é um exemplo positivo ou negativo de um conceito-alvo"(LUGER, 2013). Redes Neurais são exemplos que usam o aprendizado supervisionado. No aprendizado não supervisionado os exemplos de treinamento não foram classificados, ou seja, o programa deverá procurar características nesses exemplos que possibilitem a classificação desses em categorias. Um exemplo desse tipo de algoritmo é o *Cluster*.

Outra etapa fundamental no AM é escolher uma função alvo. No exemplo do jogo de Damas, a função alvo deve escolher a melhor jogada dentre todas as jogadas possíveis em um estado. A dificuldade é como escolher essa função de modo que ela seja a mais eficiente possível. Mitchell sugere que uma função alvo ideal é difícil de se atingir, por isso devemos desenvolver uma função aproximada dessa ideal. Quanto mais próxima a nossa função for da ideal, mais eficiente ela será, assim como nosso programa. O jogo da Velha nos dá um exemplo de como usar essa função. Poderíamos dar um valor para cada jogada, sendo que esse valor se equivale ao número de possibilidades para uma jogada que o adversário ainda possui após a nossa jogada. Quanto menor o valor, menos jogadas restou para o adversário, melhor

foi a escolha do nosso programa. Esse método de adotar uma pontuação para a função alvo também pode ser utilizado no exemplo do jogo de Damas. Poderíamos dar um valor de 100, por exemplo, para uma jogada que resulte em vitória e um valor de -100 para uma jogada que leve à derrota. As jogadas anteriores serão definidas nas etapas seguintes.

A seguir, o desenvolvedor deverá escolher uma representação da função alvo. Uma rede neural é um exemplo dessa representação. No exemplo dado por Mitchell, o jogo de Damas, ele utiliza uma função linear usando seis parâmetros de dentro do jogo: número de peças pretas no tabuleiro; número de peças vermelhas; número de reis pretos; número de reis vermelhos; número de peças pretas que podem ser capturadas pelas vermelhas na próxima jogada; número de peças vermelhas que podem ser capturadas pelas pretas na próxima jogada. Cada um desses parâmetros seria multiplicado por um coeficiente, que determinaria a importância do parâmetro. Esses coeficientes são determinados na etapa seguinte, que é a escolha do algoritmo de aprendizado. No exemplo que estamos seguindo, esses valores vão depender da jogada seguinte, ou seja, vão recebendo valores retrógrados até chegar na jogada de vitória ou derrota.

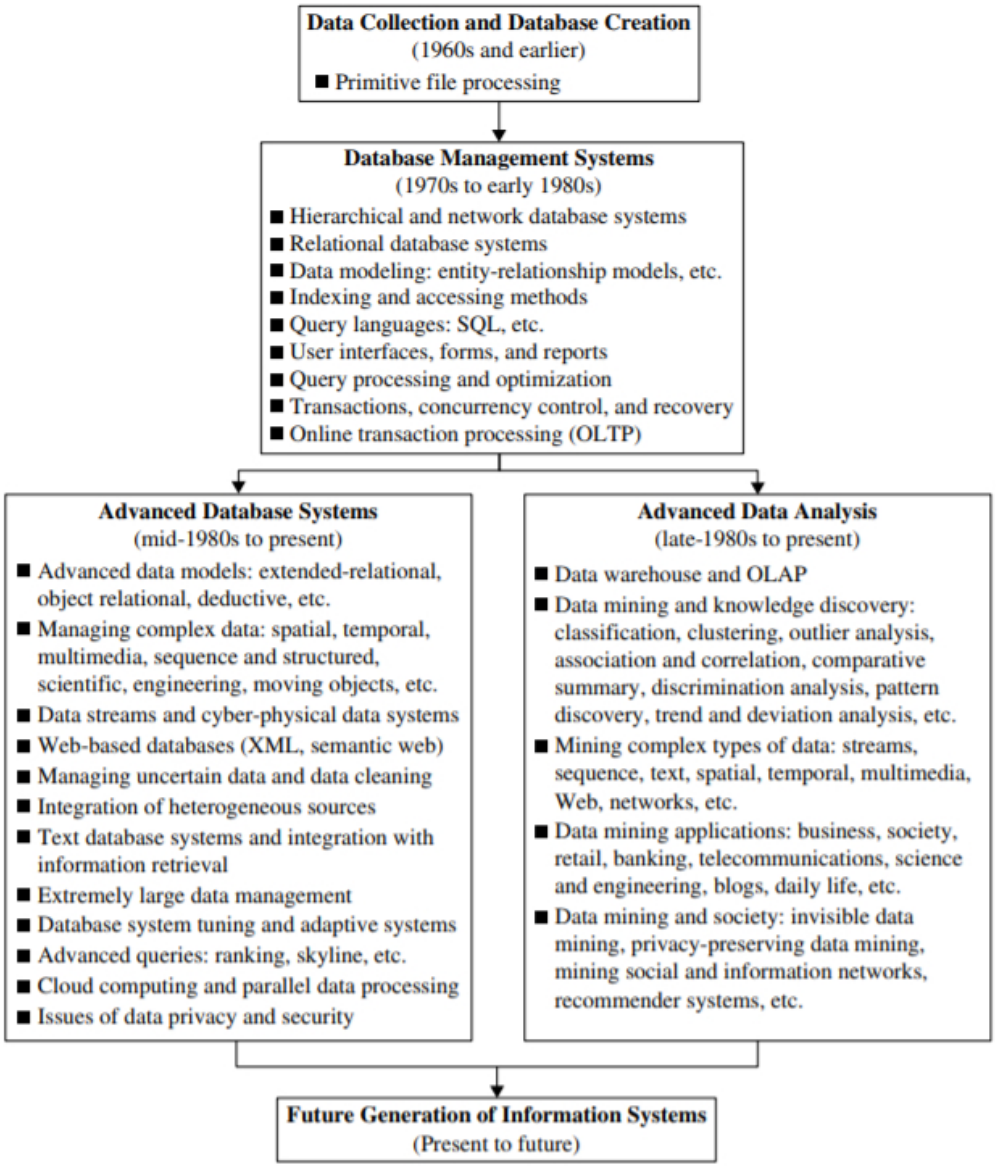
3.2 PROCESSO DE APRENDIZADO DE MÁQUINA

Para aprender e melhorar automaticamente com a experiência, os programas devem reconhecer padrões complexos e tomar decisões inteligentes com base em dados. Portanto o AM precisa analisar esses dados, descobrir e extrair conhecimento desses. Por isso precisamos entender como funciona o Processo de Aprendizado de Máquina, precisamos estudar como são tratados os dados que serão trabalhados pelo programa, pois a preparação desses dados é essencial para que o programa possa analisá-los com eficiência. Assim, vemos uma relação íntima entre o estudo do AM com a Mineração de Dados (*Data Mining*) e a Extração de Conhecimento (*Knowledge Discovery in Databases* ou KDD).

Em 1996, Fayyad et al. escreviam sobre a crescente preocupação de como conseguir trabalhar com o aumento da quantidade de dados que vinha circulando no mundo. Estava ocorrendo uma transição de como era feita a análise desses dados, deixando de ser realizado por indivíduos e passando para uma análise automática pelo computador. Fayyad cita o exemplo de um dos primeiros programas de sucesso nessa área, o Skicat, que analisava, classificava e catalogava imagens de objetos no espaço. Esse programa foi responsável por analisar 3 *terabytes* (10 a 12 *bytes*) de dados de imagens coletadas do *Second Palomar Observatory Sky Survey*, ação que não foi possível de ser realizada de modo eficiente por humanos ou por técnicas convencionais de computação, mas sim pelo método de Extração de Conhecimento. Hoje dizemos que vivemos na era da informação, porém não seria incorreto dizer que na verdade vivemos na era dos dados (Han, Kamber e Pei, 2012). *Terabytes* ou *pentabytes* podem ser acessados pelo nosso computador através da *World Wide Web*, empresas multinacionais manipulam centenas de milhões de transações por semana, práticas científicas e de engenharia produzem grandes quantidades de *pentabytes* de dados, a área médica produz grandes quantidades de dados em

prontuários médicos, monitorização de pacientes e em imagens médicas. Tudo isso vem aumentando significativamente o interesse no tratamento de grandes quantidades de dados, na mineração de dados e na extração de conhecimento. A mineração de dados pode ser vista como resultado da evolução natural da tecnologia da informação (HAN; KAMBER; PEI, 2012), por isso vemos um avanço significativo na tecnologia utilizada para essa mineração. Isso é apontado na Figura 3, onde vemos as principais tecnologias utilizadas desde a década de 60 até os dias atuais.

Figura 3 – A evolução da tecnologia do sistema de banco de dados

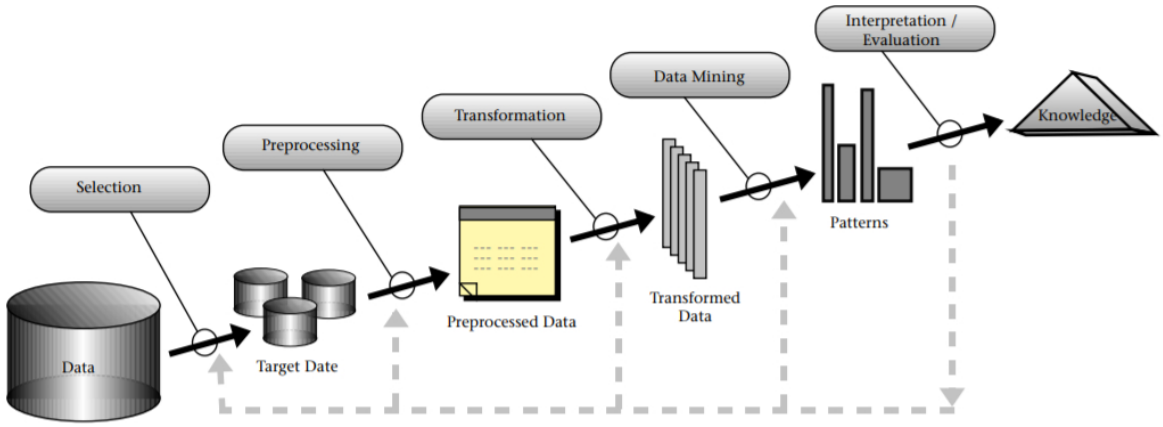


Fonte: (HAN; KAMBER; PEI, 2012)

Fayyad et al. definiram em 1996 a Extração de Conhecimento como "o processo, não trivial, de extração de informações implícitas, previamente desconhecidas e potencialmente úteis, a partir dos dados armazenados em um banco de dados". Ou seja, esse método é um conjunto de técnicas que visam tornar os dados compreensíveis para o programa, ele faz o ma-

peamento de dados de baixo nível e transforma em uma outra forma mais compacta, abstrata e útil (TEELE; KLEIN; ROSNER, 1996). Ele realiza isso através de alguns passos que envolvem a preparação de dados, busca de padrões, avaliação de conhecimento e refinamento, tudo repetido em várias iterações. Para os autores, a etapa de buscar padrões nos dados em análise é o que constitui o termo Mineração de Dados. É importante reforçar que o processo é iterativo, ou seja, são operações que se repetem de modo finito e cujos resultados de cada uma é dependente dos resultados das que a precedem. Para isso, o processo de busca de conhecimento contém uma série de passos: seleção, pré-processamento e limpeza, transformação, mineração de dados e interpretação/avaliação. Além disso, é um processo interativo, pois o programador pode intervir e controlar o curso das atividades. Os ciclos que os dados percorrem até virar informação podem ser vistos na Figura 4:

Figura 4 – Uma visão geral das etapas que compõem o processo do KDD



Fonte: (TEELE; KLEIN; ROSNER, 1996)

A primeira fase é a de Seleção de Dados, ela possui um impacto importante na qualidade do resultado final. Isso ocorre porque nessa fase é escolhido o conjunto de dados, que pertencente a um domínio, contendo todas as possíveis variáveis, também chamadas de características ou atributos, e registros, também chamados de casos ou observações, que farão parte da análise. Para o sucesso dessa fase, a escolha dos dados deve ser realizada por um especialista na área de conhecimento dos dados em questão. Esses dados podem vir para análise em diversos formatos e de diversas origens, como *data warehouses*, planilhas ou sistemas legados, tornando esse processo de seleção um processo complexo.

A segunda fase é a de Pré-processamento e Limpeza, aqui os dados deverão melhorar de qualidade para que os algoritmos de Mineração de Dados possam trabalhar com maior eficiência. Portanto, nessa etapa deverão ser eliminados os dados redundantes e inconsistentes, deverão ser recuperados os dados incompletos e serão identificados os possíveis dados discrepantes ao conjunto, chamados de *outliers*. Nessa fase também é importante recorrer à ajuda do

especialista no assunto, pois ele é capaz de diferenciar dados discrepantes de erros de digitação, por exemplo. Essa fase também é responsável por melhorar o desempenho do algoritmo de análise através da diminuição do número de variáveis envolvidas no processo, isso é realizado por métodos de redução ou transformação.

A próxima é a fase de Transformação dos Dados, ou seja, os dados devem ser armazenados e formatados de modo adequado aos algoritmos que serão aplicados. Eles podem ter se originado de diferentes sistemas operacionais e diferentes bancos de dados, desse modo apresentando formatos diferentes, que precisarão ser adequados. Essas fases de preparação dos dados são de extrema importância para que a próxima etapa, de Mineração de Dados, tenha sucesso. Steiner et al. mostraram que a análise exploratória dos dados preliminarmente ao uso das técnicas de mineração de dados proporcionara uma melhora significativa no desempenho de quatro das cinco técnicas testadas pelos autores na mineração. Isso enfatiza "a importância de se ter dados confiáveis e consistentes e, por conseguinte, dados multivariados explorados estatisticamente"(STEINER et al., 2006).

A quarta fase é a Mineração de Dados propriamente dita. Aqui serão encontrados os padrões nos dados tratados nas fases anteriores. "A mineração de dados, como usamos o termo, é a exploração e análise de grandes quantidades de dados para descobrir padrões e regras significativos."(BERRY; LINOFF, 1997, tradução nossa). Para isso, serão utilizadas técnicas que têm o objetivo de agrupar ou classificar os dados e, se possível, descobrir regras de associação entre eles (STEINER et al., 2006). As técnicas utilizadas derivam de estudos na área de ciências da computação, estatística e AM. Pode-se dividir a Mineração de Dados em dois tipos: direcionada e não direcionado. A primeira tenta categorizar ou explicar os dados para algum campo alvo, como renda, no caso do uso em economia. A segunda tenta achar padrões ou similaridades em grupos de dados sem um campo específico ou coleção de classes predefinido (BERRY; LINOFF, 1997). A mineração tem o objetivo de criar modelos, que são algoritmos ou um grupo de regras que relacionam um conjunto de dados com características similares, isso na tentativa de obter uma explicação, como por exemplo qual o perfil das pessoas que deixam de pagar o cartão de crédito (idade, renda, endereço, estado civil, etc). Alguns exemplos desses métodos são as Árvores de Decisão, Máquinas de Suporte de Vetores, Métodos Estatísticos, Redes Neurais, Algoritmos Genéticos e outros. A técnica mais adequada a ser escolhida vai variar de acordo com o tipo de dado que está sendo analisado e o objetivo do programa a ser desenvolvido ou os resultados que os pesquisadores desejam alcançar. Steiner et al., por exemplo, estudaram um banco de dados relacionado com um problema médico: doentes com icterícia (uma doença que deixa a pele e mucosas amareladas). Esse estudo observou que a árvore de decisão foi uma técnica que apresentou um percentual de erros relativamente alto, mas foi o único método que permitiu apontar para o usuário quais são os atributos que estão discriminando os padrões, ou seja, observaram que um exame chamado Bilirrubina Direta foi o atributo mais importante (raiz da árvore) na classificação da doença conforme o tipo de obstrução da via biliar, por cálculo ou por câncer. Outras técnicas, que obtiveram melhores resultados, como as redes neurais, po-

dem se tornar mais compreensíveis aos usuários se utilizarmos algum algoritmo de extração de regras.

A última fase é a de Interpretação e Avaliação, onde o conhecimento realmente vai ser descoberto de acordo com o objetivo dos pesquisadores. Estes podem usar o conhecimento para desenvolver novos programas, podem apenas documentar os padrões descobertos, ou até refutar conhecimentos descobertos previamente. Na próxima sessão será detalhada a avaliação dos resultados obtidos.

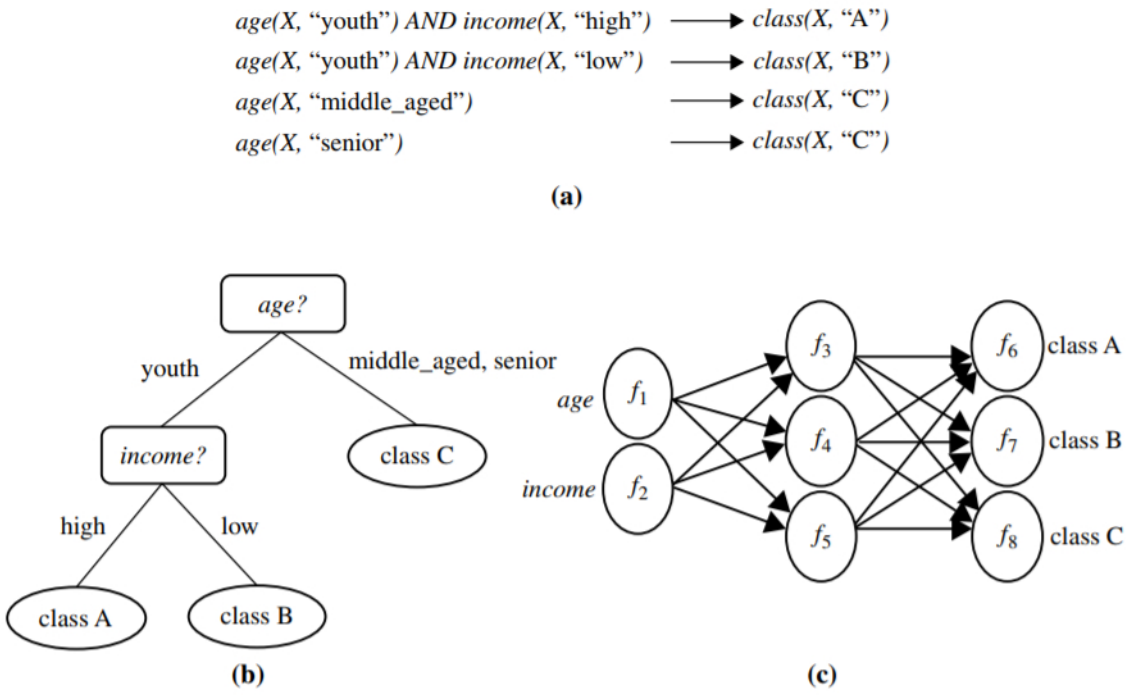
Existem alguns tipos de padrões que podem ser extraídos dos dados. Um tipo é a caracterização e discriminação, no qual as entradas de dados podem ser associadas com classes ou conceitos (HAN; KAMBER; PEI, 2012). Por exemplo, produtos em uma loja podem pertencer à classe de computadores ou à classe de cosméticos, e os clientes dessa loja podem apresentar o conceito de *bigSpenders* (gastam em média mais de cem dólares em compras) ou *budgetSpenders* (gastam em média menos de cem dólares em compras). Outros tipos de padrões que podem ser estudados são a mineração de padrões frequentes, associações e correlações; análises de *clustering*; análises de *outlier*; regressão e classificação, que é o tipo de interesse desse estudo. Na classificação, os dados são divididos em classes ou conceitos através de um processo para achar o modelo ou função que possa descrever e separar essas categorias de dados. Para se chegar a esse modelo é necessário um processo de treinamento com dados cujas classes já são conhecidas. O nosso estudo tem o objetivo de classificar imagens da membrana timpânica humana em imagens de membranas doentes ou imagens de membranas sadias. Existem várias formas de se chegar à esses modelos, a Figura 5 mostra três dessas formas.

A rede neural tenta se assemelhar à rede de neurônios que formam o cérebro humano, são formadas por conjuntos de unidades de processamento que possuem conexões com diferentes pesos entre si. Esse método será detalhado mais adiante.

3.3 AVALIAÇÃO DOS RESULTADOS

O estudo apresentado representa um problema de classificação. Imagens coletadas de MT humanas deverão ser classificadas como imagens de membranas doentes, imagens de membranas sadias ou imagens com cerumen que não permitem a avaliação da MT. Essas imagens estarão em uma quantidade limitada, elas são apenas uma amostra de um número muito maior de imagens que existem no mundo real. Mas os desenvolvedores buscam demonstrar que essa amostra pode ser extrapolada para o mundo real, por isso devemos discutir os métodos de amostragem, como estimar o desempenho do algoritmo escolhido e como comparar esse desempenho entre dois algoritmos. Primeiramente nossas imagens deverão ser classificadas por médico especialista na área de otorrinolaringologia, ou seja, será o professor no aprendizado de máquina. Depois, de modo aleatório, uma parte dessas imagens será utilizada para o treinamento do programa, e outra parte será utilizada para a realização de testes para verificar a eficácia do processo. Durante a fase de testes, para cada classe, teremos um grupo de imagens

Figura 5 – Um modelo de classificação pode ser representado de várias formas: (a) regras IF-THEN, (b) uma árvore de decisão , ou (c) uma rede neural



Fonte: (HAN; KAMBER; PEI, 2012)

que serão classificadas de modo positivo. Uma porcentagem desse grupo também foi previamente classificada como positiva pelos especialistas, esses são exemplos Verdadeiros Positivos. Mas uma porcentagem desse mesmo grupo foi classificada negativamente pelos professores, ou seja, são exemplos Falso Positivos. O outro grupo de imagens, classificado nos testes como negativos, também apresentarão uma porcentagem de acertos, os Verdadeiros Negativos, e uma porcentagem de erros, os Falsos Negativos. Com esses conceitos podemos criar uma tabela que tem um papel importante na avaliação do desempenho do algoritmo de classificação, essa tabela recebe o nome de Matriz de Confusão, como pode ser visto na Tabela 1.

Tabela 1 – Matriz de Confusão

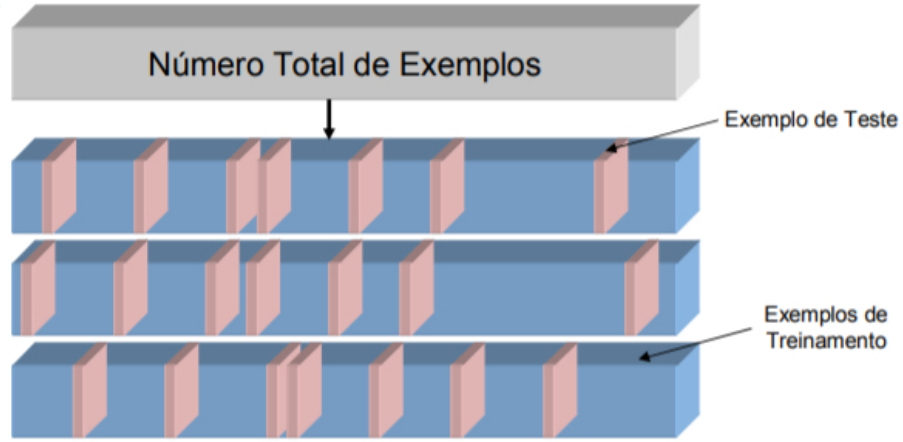
		PREDITO	
		Classe A	**Classe B**
VERDADEIRO	**Classe A**	VP	FN
	Classe B	FP	VN

Fonte: (HAN; KAMBER; PEI, 2012)

Em um problema de classificação, devemos decidir o que fazer com as amostras que foram coletadas, quantas amostras vão ser utilizadas para o treinamento e quantas vão para os testes. Por isso devemos estudar os métodos de amostragem conhecidos e adaptar nosso problema ao melhor método, testes com diferentes métodos podem ser feitos. O método de amostragem de Ressubstituição utiliza o mesmo conjunto de exemplos para realizar o treinamento e os testes, sendo mais indicado em casos em que há poucas amostras disponíveis. Este método possui uma estimativa altamente otimista da precisão, assim podemos dizer que este estimador fornece uma medida aparente (BARANAUSKAS, 2012). No caso de alguns algoritmos que classificam corretamente todos os exemplos, como o caso de árvores de decisão sem poda, a estimativa de precisão pode chegar até 100%. Esse bom desempenho no conjunto de treinamento em geral não se estende a conjuntos independentes de teste, por isso foram criados outros métodos de amostragem que serão discutidos a seguir.

O método *Holdout* divide as amostras em uma porcentagem fixa de exemplos (n) para o treinamento e uma quantidade n-1 para os testes. Geralmente destina-se 2/3 para o treinamento e 1/3 para os testes, mas estes valores não são obrigatórios. O melhor é realizar o método várias vezes, utilizando diferentes grupos de amostras para o treinamento, e utilizar uma média dos resultados. Desse modo obtemos uma estimativa média do *Holdout* e melhoramos sua eficácia, como mostra a Figura 6.

Figura 6 – Holdout

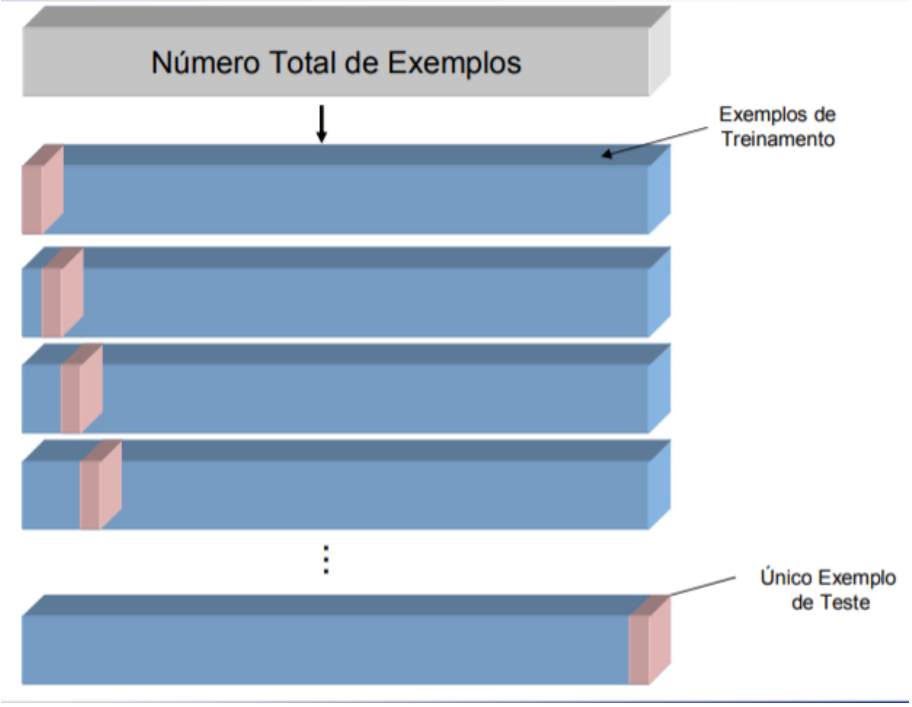


Fonte: (BARANAUSKAS, 2012)

Um outro método é o *Leave-one-out*. Como o próprio nome sugere, apenas um exemplo dentro da amostra será utilizado para teste, o restante vai ser utilizado para o treinamento. Esse procedimento é repetido um número de vezes igual ao número de amostras, sendo que cada vez uma amostra diferente é utilizada para o teste. O erro é calculado pela média dos erros em cada teste. Como se pode imaginar, isso tem um custo computacional alto se o número de amostras for muito grande, vide a Figura 7.

No método de *Cross-Validation* as amostras são divididas de modo aleatório em (r) partições, mutuamente exclusivas e de tamanho aproximadamente igual, como mostra a Fi-

Figura 7 – Leave-one-out



Fonte: (BARANAUSKAS, 2012)

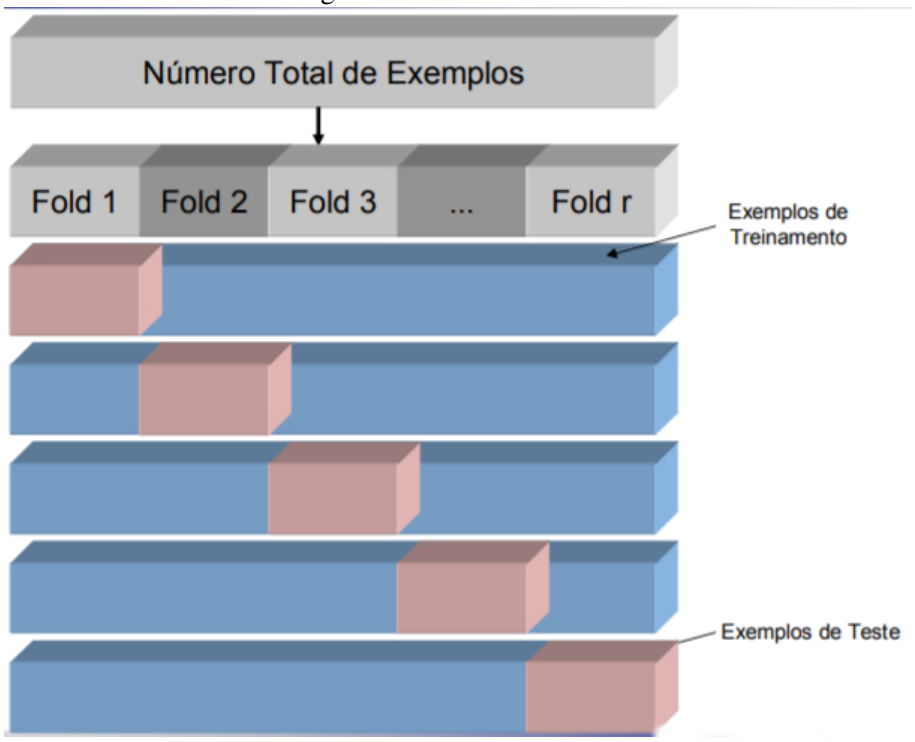
gura 8. Então o método vai ser realizado (r) vezes, sendo que em cada procedimento uma das partições vai ser separada para os testes e as outras servirão para o treinamento. O erro na *Cross-Validation* é a média dos erros calculados em cada um dos procedimentos. Uma desvantagem desse teste é o compartilhamento de uma porcentagem de exemplos para treinamento, sendo que essa porcentagem aumenta proporcionalmente com número de partições. Porém, uma vantagem é que ele tem um custo computacional menor que o *Leave-one-out*, e uma taxa de erro melhor que no *Holdout*.

Pode ser interessante separar alguns exemplos para ajustes de parâmetros e outros para o teste de validação. Como no caso das Redes Neurais em que pode ser desejável alterar o número de neurônios, tipo de função de ativação ou o número de camadas. Por exemplo, pode-se utilizar o método *Holdout* inicialmente, e depois utilizar o método *Cross-Validation* para realizar os ajustes necessários, como mostra a Figura 9.

3.4 REDES NEURAIS DE APRENDIZADO PROFUNDO

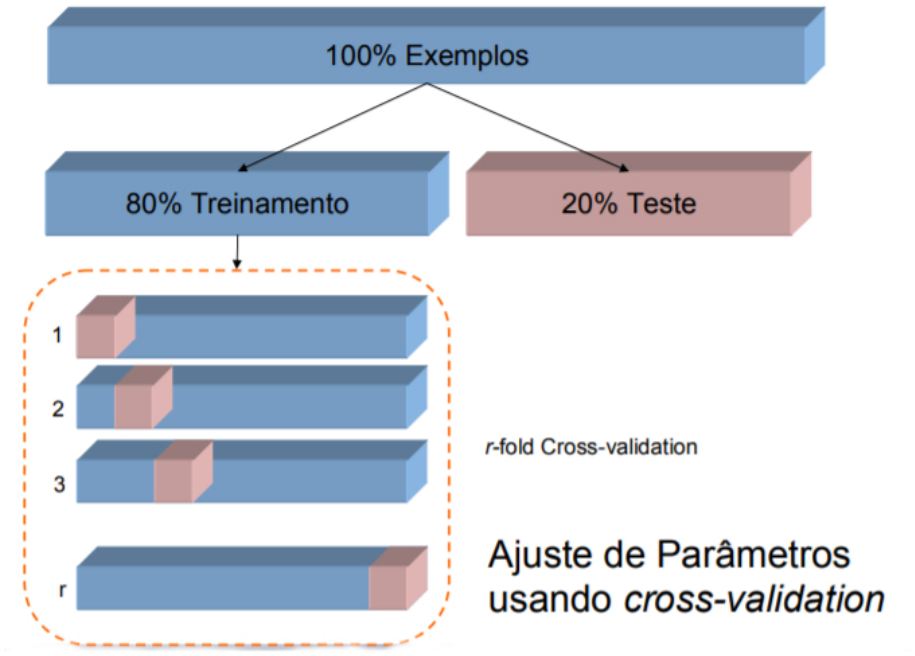
As redes neurais também são conhecidas como Processamento Paralelo Distribuído ou como Sistemas Conexionistas. Nesse sistema o processamento é focado em números, não em símbolos. Os padrões de um domínio são codificados como vetores numéricos, assim como as conexões entre os componentes. A transformação de padrões também é resultado de operações numéricas (LUGER, 2013). O desenvolvimento das redes neurais, como o próprio nome sugere, teve como inspiração o estudo do funcionamento das redes de neurônios do cérebro humano.

Figura 8 – Cross-Validation



Fonte: (BARANAUSKAS, 2012)

Figura 9 – Ajuste de Parâmetros com Cross-Validation



Fonte: (BARANAUSKAS, 2012)

Esses neurônios são unidades simples (células), mas com possibilidade de várias interconexões com outros neurônios. Essas unidades recebem uma quantidade de dados (input ou entrada), possivelmente vindo de outros neurônios, e produzem a saída de um único dado (output), que

provavelmente vai ser a entrada em outro neurônio. A rede neural no cérebro humano é surpreendente, acredita-se que participam dessa rede aproximadamente 10^{11} neurônios, e que cada um se conecta em média com mais 10^4 outros neurônios (MITCHELL, 1997). A atividade neural pode ser estimulada ou inibida pela conexão com outros neurônios. O que mais surpreende é a velocidade da capacidade de resolver problemas que o ser humano tem, sendo que a velocidade mais rápida conhecida de passagem de estímulo de um neurônio para outro é na ordem de 10^{-3} segundos. É uma velocidade baixa se comparado com velocidade de troca em um computador, na ordem de 10^{-10} segundos. O mais interessante é que uma pessoa demora 0,1 segundos para visualmente reconhecer sua mãe, portanto calcula-se que houve uma propagação de estímulo neural entre apenas algumas centenas de neurônios. Por isso é correto pensar que no cérebro humano ocorrem processamentos em paralelo, fato que os estudiosos tentam repetir no desenvolvimento das RNAP. A Tabela 2 mostra algumas tarefas para as quais a abordagem neural/conexionista é bem adequada.

Tabela 2 – Tarefas adequadas à abordagem neural/conexionista

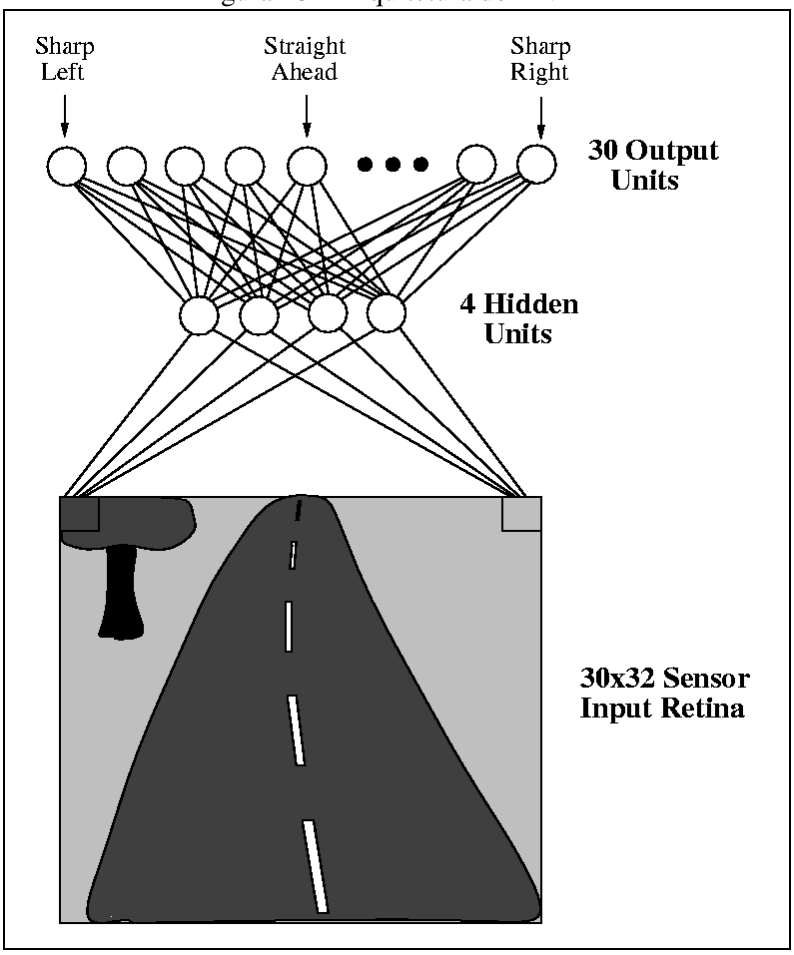
Classificação:	Decide a categoria ou grupo ao qual pertence um valor de entrada.
Reconhecimento de padrões:	Identifica a estrutura ou os padrões nos dados.
Evocação de memória:	Inclui o problema da memória endereçável por conteúdo.
Predição:	Identifica, por exemplo, doenças a partir de sintomas, causas a partir de efeitos.
Otimização:	Encontra a “melhor” organização de restrições.
Filtragem de ruído:	Separa o sinal de ruído de fundo, retirando os componentes irrelevantes de um sinal.

Fonte: (LUGER, 2013)

Um exemplo que pode ilustrar bem o funcionamento das redes neurais foi o desenvolvimento de um programa para aprender a dirigir um veículo automotivo em vias públicas. O ALVINN foi criado em 1993 e conseguiu guiar um veículo por uma distância de 90 milhas em uma estrada pública, à uma velocidade de 70 milhas por hora. Esse sistema funcionava através de uma rede neural, no qual a entrada era uma grade de 30 x 32 de intensidade de pixels, captada por uma câmera. A saída final da rede neural era formada por 30 unidades que iriam sugerir a direção do volante, conforme ilustra a Figura 10.

O modelo mais simples de rede neural é chamado de *Perceptron*, ele é um modelo de camada única, pois possui apenas uma camada além da camada de entrada. Ele apresenta

Figura 10 – Arquitetura do Alvinn

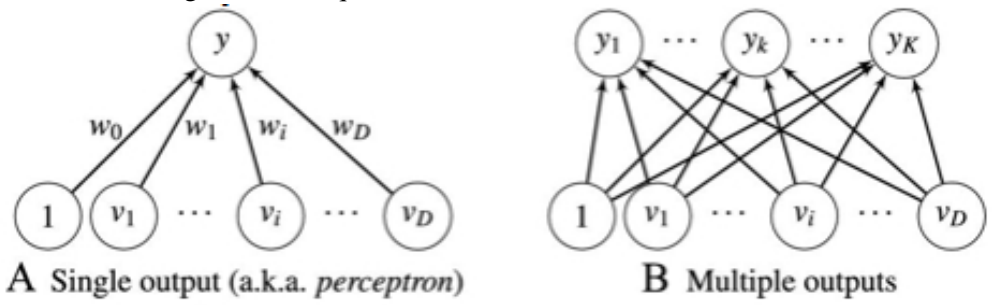


Fonte: (MITCHELL, 1997)

um vector de dados de entrada, calcula uma combinação linear desses valores e retorna uma saída de valor 1 se a soma desses valores for maior que um determinado limiar, ou retorna 0 se a soma for menor que esse limiar. Os valores de entrada são multiplicados (ponderados) por constantes (pesos). Essas constantes recebem um valor inicial, mas vão sendo modificadas de acordo com o aprendizado, pois elas representam a importância de cada dado de entrada e são o fator mais importante em um problema de classificação. Esse é um modelo de aprendizado supervisionado, e conforme o aprendizado vai ocorrendo, as constantes vão sendo modificadas e o erro vai diminuindo. Se temos valores de entrada *x* e pesos *w*, o *Perceptron* apresenta os seguintes valores de saída: 1 se w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n > 0 ; -1 se esse valor for <=0. Vemos que o limiar na verdade é determinado pelo w_0, que também é chamado de viés. A Figura 11 mostra a arquitetura da rede neural de camada única, sendo que a imagem "A"representa a arquitetura do *Perceptron*.

O *Perceptron* é um classificador linear, ou seja, os problemas solucionados por ele devem ser linearmente separáveis, de um lado da separação ficam os resultados 1 e do outro lado os resultados -1. Na Figura 12 vemos na imagem (a) um grupo de exemplos que podem ser separados por uma reta (ou plano), e na Figura (b) vemos um grupo de exemplos que não podem

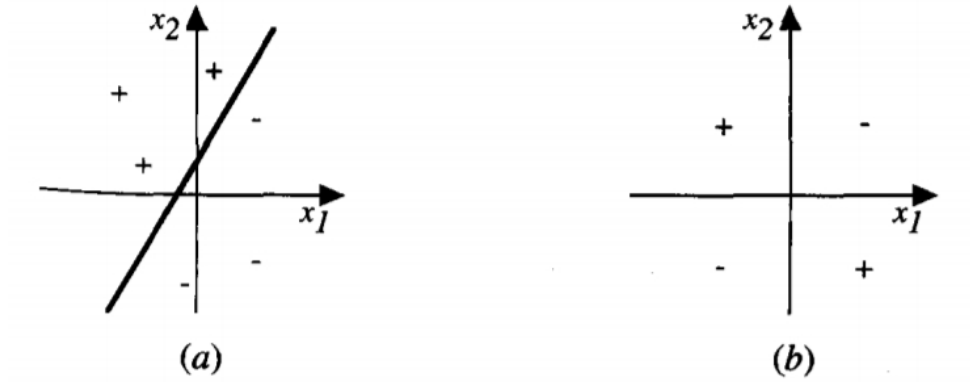
Figura 11 – Arquitetura de Rede Neural de camada única



Fonte: (ZHOU; GREENSPAN; SHEN, 2017)

ser separados por uma reta, portanto não serão corretamente classificados pelo *Perceptron*.

Figura 12 – Superfície de decisão representada por um Perceptron com 2 entradas



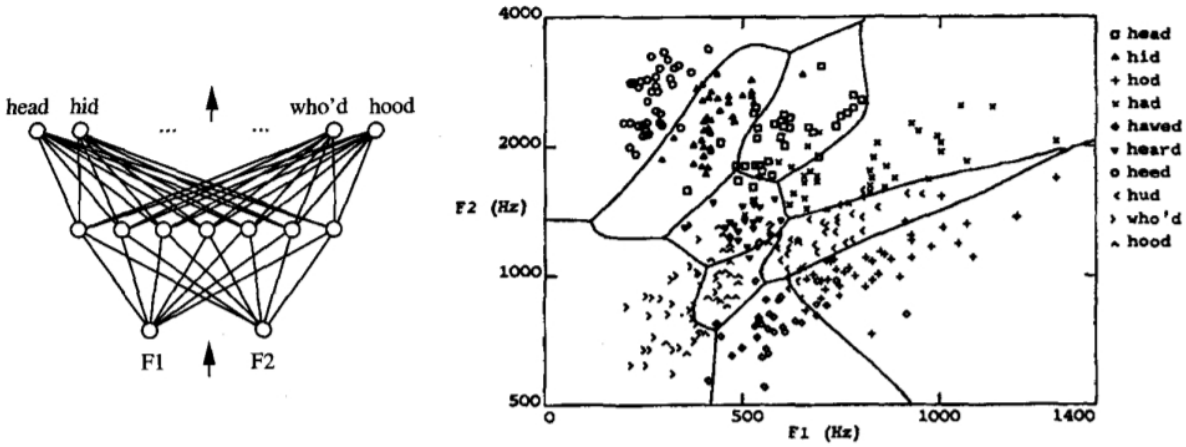
Fonte: (MITCHELL, 1997)

A limitação do *Perceptron* em apenas poder separar de modo linear um problema de classificação pode ser contornado pelo acréscimo de camadas "escondidas", ou seja, pelo uso de modelos de redes neurais de múltiplas camadas. Na Figura 13 podemos ver um exemplo de um algoritmo para reconhecimento da fala que usa um modelo de múltiplas camadas, vemos na imagem da direita a separação de uma superfície de modo não linear.

Um modelo popular que usa múltiplas camadas é o algoritmo de Retropropagação, que é capaz de expressar uma grande variedade de superfícies de decisão não lineares. Nesse modelo, os neurônios organizados em camadas passam suas ativações apenas para neurônios da camada seguinte. Porém, esse método pode espalhar erros de um modo progressivo e complexo para as camadas consecutivas. Para solucionar esse problema, o método realiza uma retropropagação do erro pelas camadas ocultas, como se fosse passado para trás uma cobrança da culpa das camadas anteriores pelo erro. A Figura 14 ilustra o modelo de Retropropagação com uma camada oculta.

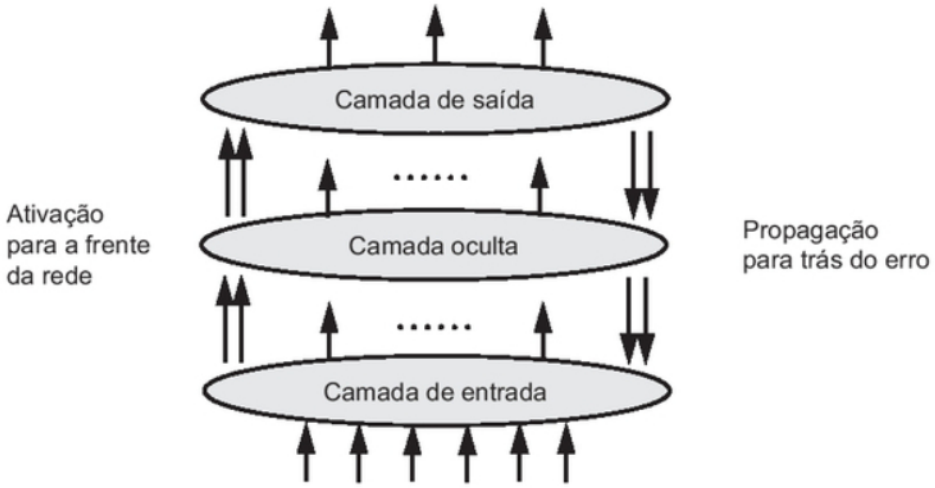
Um exemplo interessante de uma rede neural que usa Retropropagação foi implantado em 1987 na NETtalk, esse programa foi usado para aprender a pronunciar as palavras de um texto em inglês. Isso é uma tarefa difícil para um programa de computador, pois a pronuncia de

Figura 13 – Regiões de decisão de uma rede multicamadas



Fonte: (MITCHELL, 1997)

Figura 14 – Retropropagação em uma rede conexionista com uma camada oculta

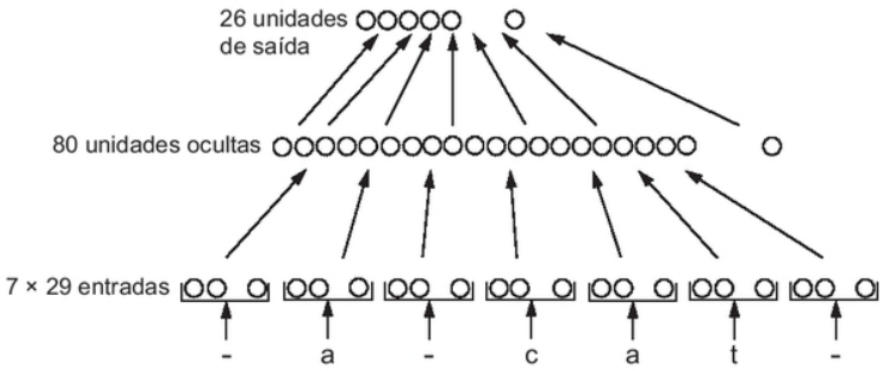


Fonte: (LUGER, 2013)

uma única letra depende do seu contexto e das letras vizinhas (LUGER, 2013). Sua arquitetura consiste em 3 camadas de unidades, sendo a camada de entrada formada por 7 grupos (7 caracteres de um texto) de 29 unidades (uma para cada letra do alfabeto, 3 para pontuação e espaço). A camada oculta é formada por 80 unidades e a camada de saída formada por 26 unidades que representam 21 características da articulação humana mais 5 para codificar a entonação e fronteiras silábicas. Essa arquitetura é representada na Figura 15.

A NETtalk recebia 7 caracteres para tentar pronunciar o caractere central, o professor comparava sua pronuncia com a pronuncia correta, então o programa ajustava seus pesos usando a retropropagação. Com esse exemplo, os pesquisadores conseguiram observar alguns comportamentos interessantes das redes neurais, que podem ser comparados com o aprendizado

Figura 15 – Topologia da rede de NETtalk



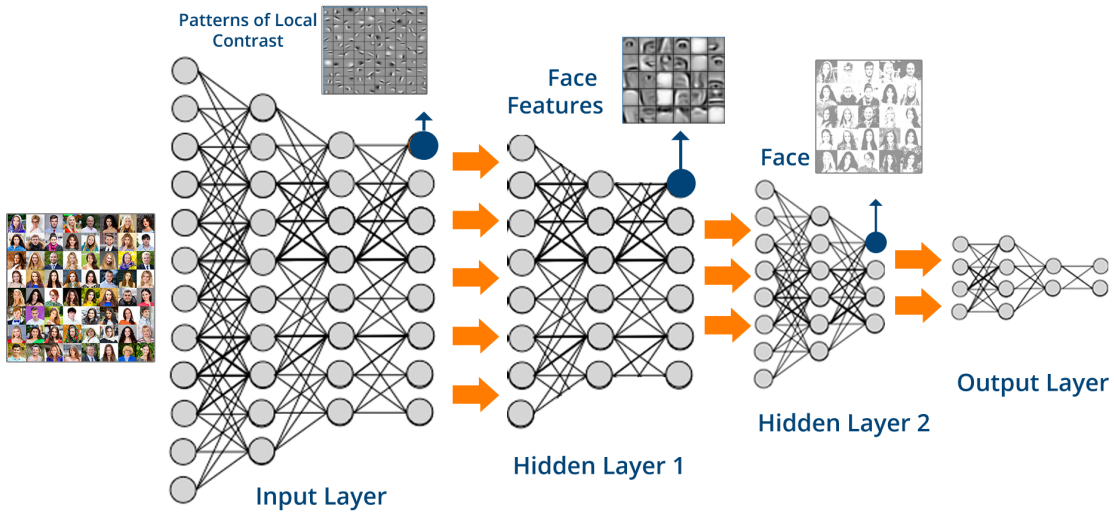
Fonte: (LUGER, 2013)

do ser humano. A medida que a rede vai aprendendo a pronunciar mais palavras, ela também pronuncia melhor novas palavras. Ao danificar intencionalmente uma rede, alterando os pesos já aprendidos, ela apresentou um grau de reaprendizado bastante eficiente. Também se observou o papel importante da camada oculta na generalização de uma rede neural, fato que faz com que ocorra no aprendizado um certo grau de abstração, fundamental para que o AM se aproxime do aprendizado humano.

As RNAP são redes neurais que apresentam várias camadas de neurônios, e vêm sendo muito utilizadas no AM para resolver problemas práticos da vida real. Isso ocorre por causa das altas taxas de sucesso no aprendizado do reconhecimento de letras do alfabeto na escrita manual, da fala humana e no aprendizado do reconhecimento facial (MITCHELL, 1997). A RNAP é bastante eficiente para resolver problemas no qual os dados de treinamento possuem muito "ruído", como o que ocorre em imagens de câmeras ou sons de microfones. A Figura 16 ilustra uma arquitetura usada no reconhecimento facial, ela retrata bem o uso de várias camadas ocultas.

As arquiteturas conexionistas existem há mais de 70 anos, porém o fato que contribuiu para o desenvolvimento de várias arquiteturas de RNAP recentemente foi a melhora do desempenho computacional, pois o treinamento de redes neurais com muitas camadas pode se tornar impraticável em termos de requisito computacional. Além disso, foram criadas as Unidades de Processamento Gráfico (GPU), que podem conter de mil a quatro mil núcleos de processamento de dados especializados, sendo que os processadores comuns costumam ter de 4 a 24 núcleos de processamentos gerais. Quanto maior o número de núcleos de processamento, mais neurônios podem ser trabalhados em paralelo, fato fundamental para o desempenho de redes neurais com muitas camadas. Podemos citar 5 arquiteturas de aprendizado profundo que se tornaram bastante populares: Redes Neurais Recorrentes (*recurrent neural networks* ou RNNs), Memória de Longo Prazo (*long short-term memory* ou LSTM)/Unidade Recorrente Bloqueada (*gated recurrent unit* ou GRU), Redes Neurais Convolucionais (*convolutional neural networks*

Figura 16 – Rede Neural de Aprendizado Profundo usada para reconhecimento facial



Fonte: Edureka

ou CNNs), Redes de Crenças Profundas (*deep belief networks* ou DBN) e Redes de Empilhamento Profundo (*deep stacking networks* ou DSNs) (JONES, 2017).

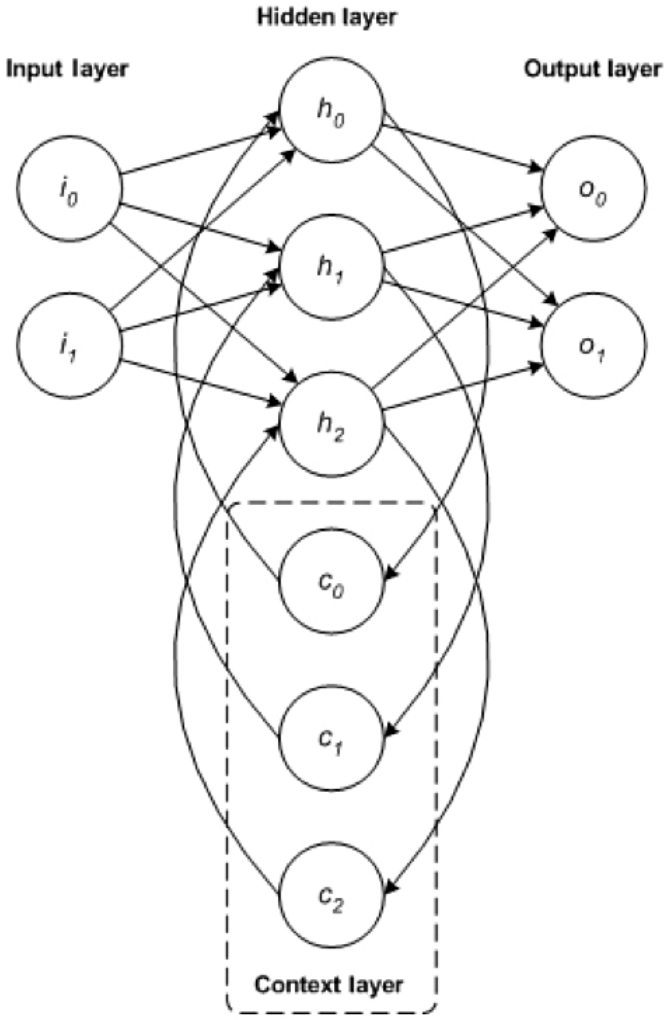
As Redes Neurais Recorrentes são redes de multicamadas que apresentam conexões de *feedback* que podem aparecer em qualquer camada, até mesmo a camada de saída também pode ter *feedback* para a própria camada. Esse fato permite que as RNNs mantenham a memória das entradas antigas e dos problemas do modelo no momento. Essa arquitetura é representada na Figura 17.

Memória de Longo Prazo, representada na Figura 18, é uma arquitetura bastante utilizada na atualidade, é encontrada em smartphones. Foi utilizada pela IBM no supercomputador Watson para reconhecimento de discurso de conversação de definição de marco. Essa arquitetura possui células de memória que armazenam dados que julgam importantes e fazem isso por um período variado de tempo. Elas possuem 3 portas: entrada, saída (quando a informação armazenada vai ser utilizada) e a porta de esquecimento (quando a informação vai ser desprezada e a célula poderá ficar livre para novos dados). Essas portas também são controladas por pesos.

Em 2014 essas células de memória foram simplificadas eliminando-se a porta de saída, passando a se chamar Unidade Recorrente Bloqueada. Isso eliminou alguns pesos e tornou a arquitetura mais rápida. As duas portas que restaram foram modificadas para porta de atualização (*update*) e porta de reconFiguração (*reset*), como mostra a Figura 19.

As Redes de Crenças Profundas são redes neurais geralmente profundas. As camadas trabalham em duplas, elas funcionam como uma máquina *Boltzmann* restrita. Essa máquina é uma rede estocástica constituída por duas camadas: uma visível e outra oculta. A camada de unidades visíveis representa os dados observados e está conectada à camada oculta, que por sua vez, deverá aprender a extrair características desses dados, portanto responsável pela abstração

Figura 17 – Redes Neurais Recorrentes



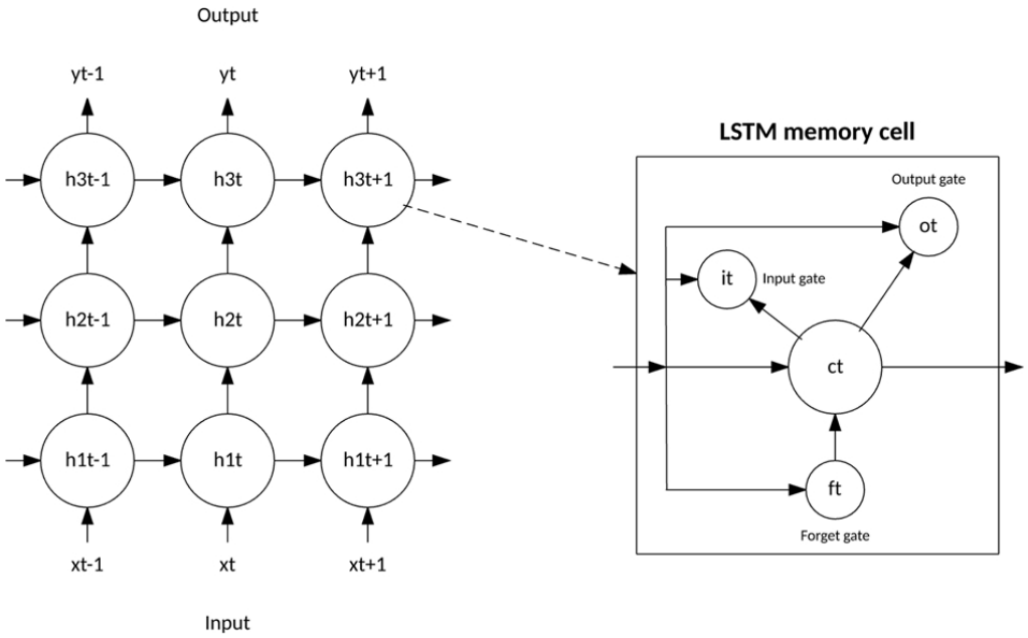
Fonte: (JONES, 2017)

dos dados. Na arquitetura DBN o treinamento ocorre em duas etapas: pré-treinamento sem supervisão e ajuste preciso supervisionado. Esta arquitetura é ilustrada na Figura 20.

A Rede de Empilhamento Profundo é uma arquitetura de rede neural profunda que surgiu para tentar resolver o problema da dificuldade de treinamento desse tipo de rede. Ela é na verdade um conjunto profundo de redes individuais, cada uma com suas próprias camadas ocultas, assim o problema geral é dividido em problemas individuais. Cada conjunto de redes forma um módulo que é constituído por uma camada de entrada, uma oculta e uma de saída, como mostra a Figura 21. Esses módulos são empilhados e podem trabalhar em paralelo, melhorando significativamente o desempenho da arquitetura.

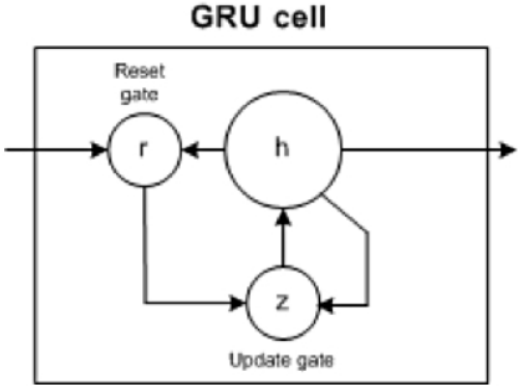
A Rede Neural Convolucional é uma rede de aprendizado profundo que vem sendo muito utilizada para classificação de imagens, por isso será mais detalhada a seguir. Seu desenvolvimento foi inspirado no funcionamento do córtex visual primário no cérebro, onde células simples detectam a existência de linhas e limites nas imagens captadas pela retina. Já foi mencionado que nas redes neurais a entrada dos dados costuma ser realizada através de um vetor

Figura 18 – Memória de Longo Prazo



Fonte: (JONES, 2017)

Figura 19 – Unidade Recorrente Bloqueada

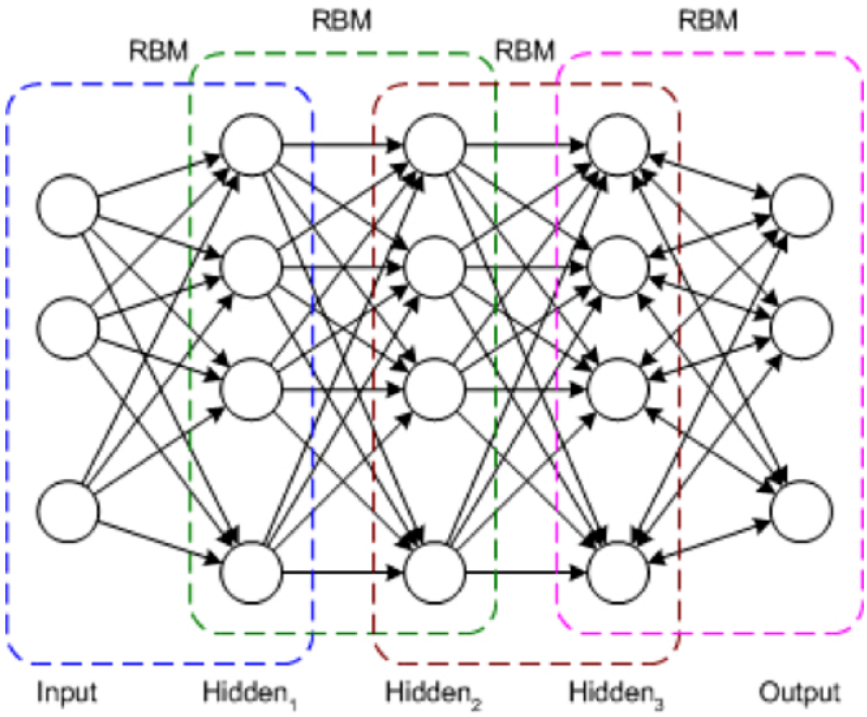


Fonte: (JONES, 2017)

de valores. Porém, no caso de imagens, essa entrada de dados se torna impraticável na forma de um vetor. Por exemplo, uma imagem em 2D que apresenta um tamanho de 200 x 200 teria uma camada de entrada com 40.000 células. Se a próxima camada (oculta) tiver 20.000 células, teremos uma matriz de pesos para a entrada com um total de 800 milhões de valores (40.000 x 20.000 = 800 milhões). Esses valores vão ficando cada vez mais complexos nas camadas seguintes. Além disso, fica difícil representar a estrutura dos pixels, com sua estrutura espacial e relações de vizinhança através de um vetor (ZHOU; GREENSPAN; SHEN, 2017). A Figura 22 ilustra como uma imagem pode ser capturada e transcrita para uma matriz.

Para solucionar o problema de desempenho citado, a CNN utiliza camadas convolucionais intercaladas com camadas de subamostragem (*subsampling*) ou também conhecidas como

Figura 20 – Redes de Crenças Profundas



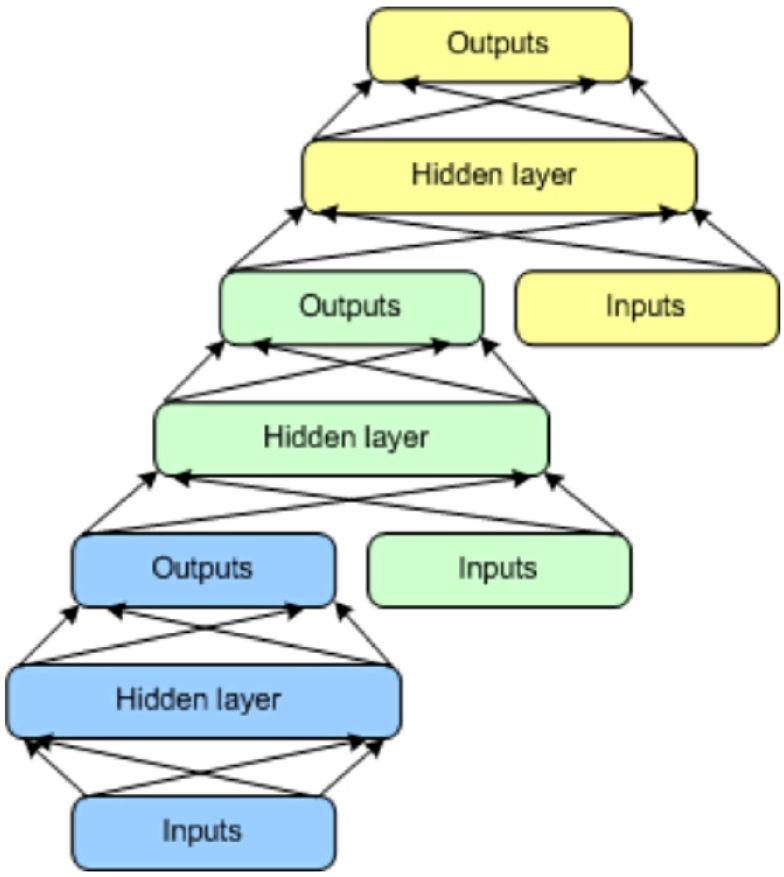
Fonte: (JONES, 2017)

camadas de agregação (*pooling*) e camadas inteiramente conectadas, como mostra a Figura 23.

As camadas convolucionais realizam operações entre matrizes utilizando uma matriz chamada de *Kernel*, que funciona como uma matriz de pesos ou um filtro. Nessas camadas o sistema vai aprender os filtros de tamanhos menores (11x11, por exemplo) em vez de aprender uma matriz de tamanho grande (40.000 x 20.000, como no exemplo já citado) (ZHOU; GREENSPAN; SHEN, 2017). Esses filtros vão trabalhando as imagens e retirando características, então eles podem utilizar a técnica da retropropagação nos pesos. Além disso as conexões compartilham esses pesos, o que reduz significativamente o número de parâmetros para ser computado. Cada camada convolucional é seguida de uma camada de subamostragem, com o objetivo de diminuir progressivamente o tamanho das matrizes, o número de parâmetros e por consequência a demanda computacional. Para realizar isso, essa camada utiliza o número máximo da matriz, média, mediana, entre outros. A Figura 24 mostra a redução da matriz pelas camadas de convolução e subamostragem (*pooling*):

A camada de subamostragem da imagem anterior usa o tipo mais comum de redução de escala, o número máximo. Uma região de tamanho n x n pode ser substituída por apenas o seu valor máximo. Isso vai diminuir o tamanho da ativação da próxima camada em uma dimensão de n^2, e vai pegar a maior ativação em uma determinada região fornecendo um pequeno grau de invariância espacial. Isto é análogo ao que acontece em outro tipo de células do cérebro, as complexas do córtex visual primário. As redes profundas utilizam uma operação não-linear após cada camada de ativação por simoid [0 -> 1], tanh [-1 -> 1] ou RELU (Rectified Linear

Figura 21 – Redes de Empilhamento Profundo

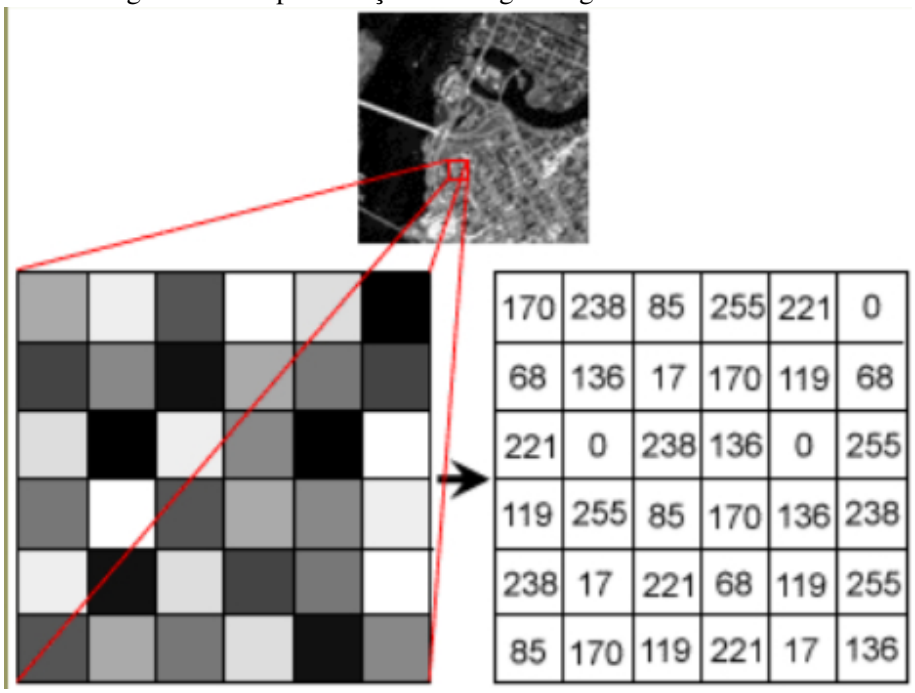


Fonte: (JONES, 2017)

Unit) para garantir a abstração do sistema (ZHOU; GREENSPAN; SHEN, 2017). A arquitetura da Rede Neural Convolucional é completada pelas camadas inteiramente conectadas que, como sugere o nome, possuem conexão entre todos os neurônios. Essa camada é importante para a classificação da imagem. Na Figura 25 podemos ver o exemplo de uma arquitetura que classifica a imagem de um carro.

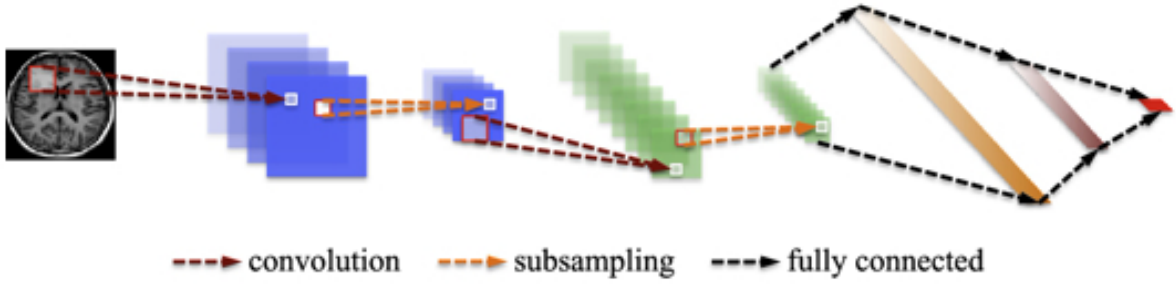
Durante muitos anos os pesquisadores tiveram dificuldade em desenvolver redes neurais com mais de 3 camadas, pois eles não tinham um algoritmo que pudesse treinar essa rede com eficiência. Hinton, Osindero, and Teh (2006) desenvolveram um método para um treinamento prévio, não supervisionado, das camadas das redes neurais. Isso possibilitou o desenvolvimento de redes com muitas camadas ocultas, caracterizando assim o que chamamos de RNAP. Exemplos modernos dessas redes são a VGGnet com 19 camadas e a GoogleNet com 22 camadas. Um problema para esse número crescente de camadas é a dificuldade para o treinamento, mesmo assim pesquisadores vêm desenvolvendo técnicas que permitem hoje o treinamento de redes de até 152 camadas (ZHOU; GREENSPAN; SHEN, 2017).

Figura 22 – Representação da imagem digital em uma matriz



Fonte: (HOCHULI, 2016)

Figura 23 – Arquitetura de uma Rede Neural Convolucional



Fonte: (ZHOU; GREENSPAN; SHEN, 2017)

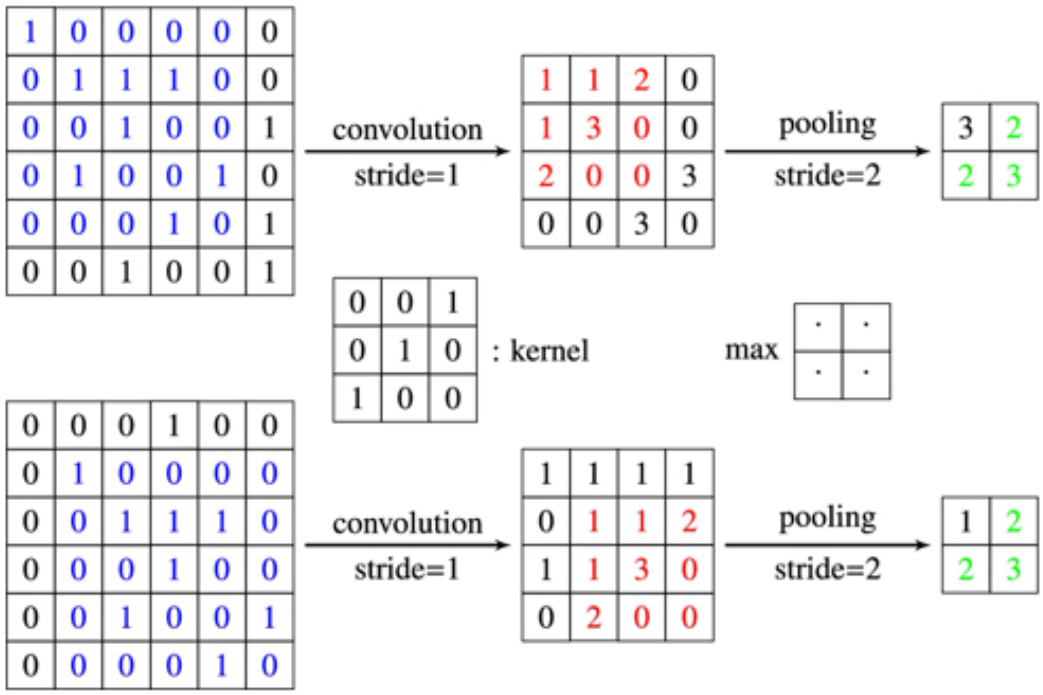
3.5 FERRAMENTAS PARA PROGRAMAÇÃO

A programação de algoritmos para RNAP pode ser um processo bastante complexo e vai fazer o desenvolvedor despender tempo. O sucesso desses algoritmos em várias áreas vem fazendo com que grupos de pesquisadores publiquem códigos, ferramentas e até mesmo modelos completos já treinados para algumas aplicações (ZHOU; GREENSPAN; SHEN, 2017). A seguir serão mostradas algumas ferramentas que facilitam o desenvolvimento de programas inteligentes e as pesquisas na área do aprendizado profundo.

Uma das estruturas de aprendizado profundo mais utilizadas é a Caffe[1]. Ela foi desenvolvida pela *Berkeley Vision* e *Learning Center* na Universidade da California, e atualmente

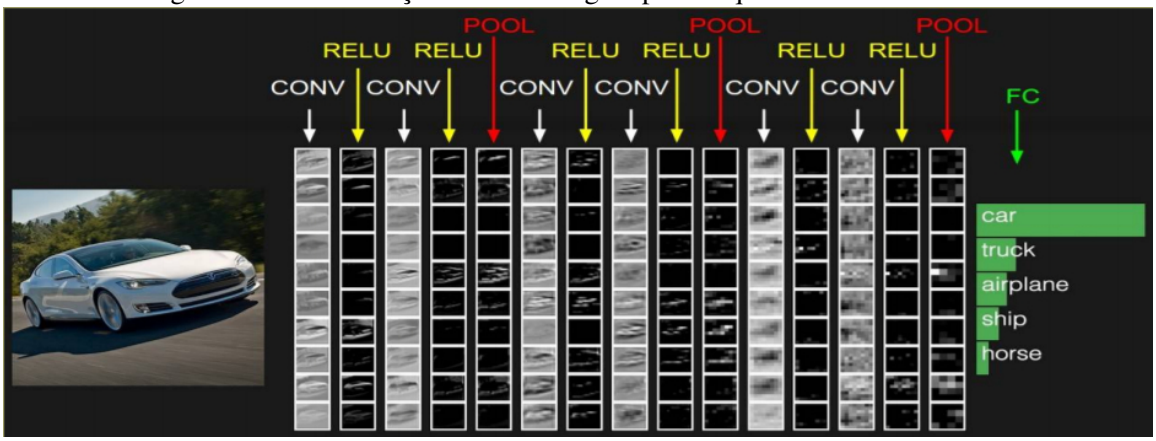
[1]https://github.com/BVLC/caffe

Figura 24 – Ação das camadas convolucional e subamostragem (*pooling*)



Fonte: (ZHOU; GREENSPAN; SHEN, 2017)

Figura 25 – Classificação de uma imagem pela Arquitetura Convolucional



Fonte: (HOCHULI, 2016)

vem sendo desenvolvida por uma comunidade de contribuidores, portanto é uma ferramenta livre (*open-source*). A Caffe oferece suporte a uma ampla variedade de arquiteturas de aprendizado profundo, incluindo a CNN e a LSTM. Por causa da arquitetura neural convolucional, a Caffe vem sendo muito utilizada na classificação de imagens e em outras aplicações relacionadas com a computação de visão. Ela oferece suporte para aceleração baseada em GPU com a biblioteca *NVIDIA CUDA Deep Neural Network*. Foi desenvolvida em C++, e trabalha com uma interface em Python e MATLAB para treinamento e execução de aprendizado profundo.

Outra ferramenta livre disponível é o Theano[2]. É uma biblioteca em Python que foi desenvolvida e disponibilizada em 2008 pelo *Montreal Institute for Learning Algorithms* da Universidade de Montreal. Possui uma boa integração com um pacote da linguagem Python, o NumPy, que permite trabalhar com arranjos, vetores e matrizes de N dimensões. Outras características importantes do Theano são: permite o uso de GPU, diferenciação simbólica eficiente, otimização com alta velocidade e estabilidade, geração dinâmica de código em C, teste de unidade extensivo e auto-verificação (ZHOU; GREENSPAN; SHEN, 2017).

Torch[3] é um *framework* livre, com um ótimo suporte para algoritmos de aprendizado de máquina. Ele também permite o uso de GPUs, a construção de redes neurais e o treinamento destas de modo eficiente. Uma desvantagem é que é uma ferramenta que depende da linguagem de programação Lua.

MatConvNet[4] é uma biblioteca do MATLAB, que é um *software* interativo de alto desempenho que trabalha com cálculo numérico, principalmente cálculo com matrizes, processamento de sinais e construção de gráficos. O MatConvNet foi inicialmente desenvolvido para trabalhar com CNNs, mas agora já é possível utilizá-lo em outras redes neurais de aprendizado profundo. Ele é bastante eficiente e simples de usar, além de fornecer modelos previamente treinados que são usados para classificação de imagens, por exemplo.

Deeplearning4j[5] é uma estrutura de aprendizado profundo popular que se concentra na tecnologia Java, utiliza o Eclipse para seu desenvolvimento, mas inclui interfaces de programação de aplicativos para outras linguagens, como Scala, Python e Clojure. É um *framework* de uso livre, liberada por meio da licença da Apache, trabalha com várias arquiteturas de aprendizado profundo como CNNs, RNNs e DBNs. Também apresenta suporte para estruturas de processamento de *big data* como o Apache Hadoop e Spark, além de suporte para o uso de GPUs através do CUDA (uma API destinada a computação paralela criada pela Nvidia). Vem sendo muito utilizada na detecção de fraude no setor financeiro, sistemas de recomendações, reconhecimento de imagem e detecção de intrusão na rede (segurança cibernética).

A TensorFlow[6] foi desenvolvida pela Google como uma biblioteca de software livre e tem sua origem na fonte fechada DistBelief, um programa de treinamento de redes neurais profundas construído pela Google. A ferramenta TensorFlow foi liberada por meio da licença Apache 2.0, e trabalha com várias arquiteturas de redes neurais como CNNs, RNNs, DBNs e outras. É mais utilizada na linguagem Python, porém também é possível trabalhar com C++, Java, Rust e Go. Apresenta suporte para trabalhar com Hadoop e também o CUDA para melhorar seu desempenho com as GPUs. Recentemente foi liberada uma pilha para Android, o TensorFlow Lite, que permite o desenvolvimento de aplicativos inteligentes para dispositivos móveis.

[2]http://deeplearning.net/software/theano/
[3]http://torch.ch/
[4]http://www.vlfeat.org/matconvnet/
[5]https://deeplearning4j.org/
[6]https://www.tensorflow.org/

O Microsoft Cognitive Toolkit[7], antigamente chamado de CNTK, é um kit de ferramentas de aprendizado profundo, que foi liberado pela Microsoft em 2015. Ele funciona como um gráfico direcionado no qual os nós de folha representam valores de entrada ou parâmetros de rede, enquanto outros nós representam operações de matriz em suas entradas. Com essa ferramenta é possível associar várias arquiteturas de aprendizado profundo como CNNs, RNNs, DBNs e outras. Além disso tem suporte para GPUs e servidores, aumentado significativamente o desempenho no treinamento das redes.

Outro *frameworks* utilizado em redes neurais de aprendizado profundo é o Chainer[8]. Ele suporta computação CUDA, também é executado em várias GPUs com pouco esforço. Trabalha com várias arquiteturas como Redes Neurais Pró-alimentadas (*Feed-forward Nets*), CNNs, RNNs e Redes Recursivas. A computação avançada pode incluir quaisquer instruções de fluxo de controle do Python, sem a falta de capacidade da retropropagação.

A DDL (*Distributed Deep Learning*)[9] é uma biblioteca livre, disponibilizada pela IBM, que se vincula aos principais *frameworks* de aprendizado profundo como o TensorFlow, Caffe, Torch e Chainer. A DDL pode ser usada para acelerar os algoritmos de aprendizado profundo sobre *clusters* de servidores e centenas de GPUs.

Weka[10] é um projeto desenvolvido pelo grupo de Aprendizado de Máquina da *University of Waikato* que visa disponibilizar uma coleção de algoritmos para facilitar as tarefas de mineração de dados. Ela possui ferramentas para pré-processamento, classificação, regressão, *clustering*, regras de associação e visualização. A base do programa utiliza a linguagem de programação Java.

3.6 APLICAÇÕES NA MEDICINA

As RNAP vêm sendo bastante usadas na medicina, tanto para analisar textos presentes em bancos de dados médicos (prontuários, revistas, jornais), como para ajudar na classificação de imagens (tomografias, exames de fundo de olho, eletrocardiogramas). Essas análises feitas por programas inteligentes vêm ajudado na realização de diagnósticos de doenças e nas decisões de qual o melhor tratamento. São vários os exemplos do uso do AM na área médica, a seguir serão mostrados alguns desses.

Um exemplo que vem sendo usado em um dos principais hospitais de tratamento de câncer na Índia é o supercomputador Watson da IBM, que ficou famoso por vencer o popular programa de perguntas e respostas da TV americana “Jeopardy”, em fevereiro de 2011. O Watson é um sistema de programação cognitiva, ou seja, ele possui várias APIs que usam o AM (algumas arquiteturas de aprendizado profundo) para interpretar imagens, textos e outros. No *Manipal Hospital*, na Índia, são tratados mais de 200 mil pacientes com câncer por ano

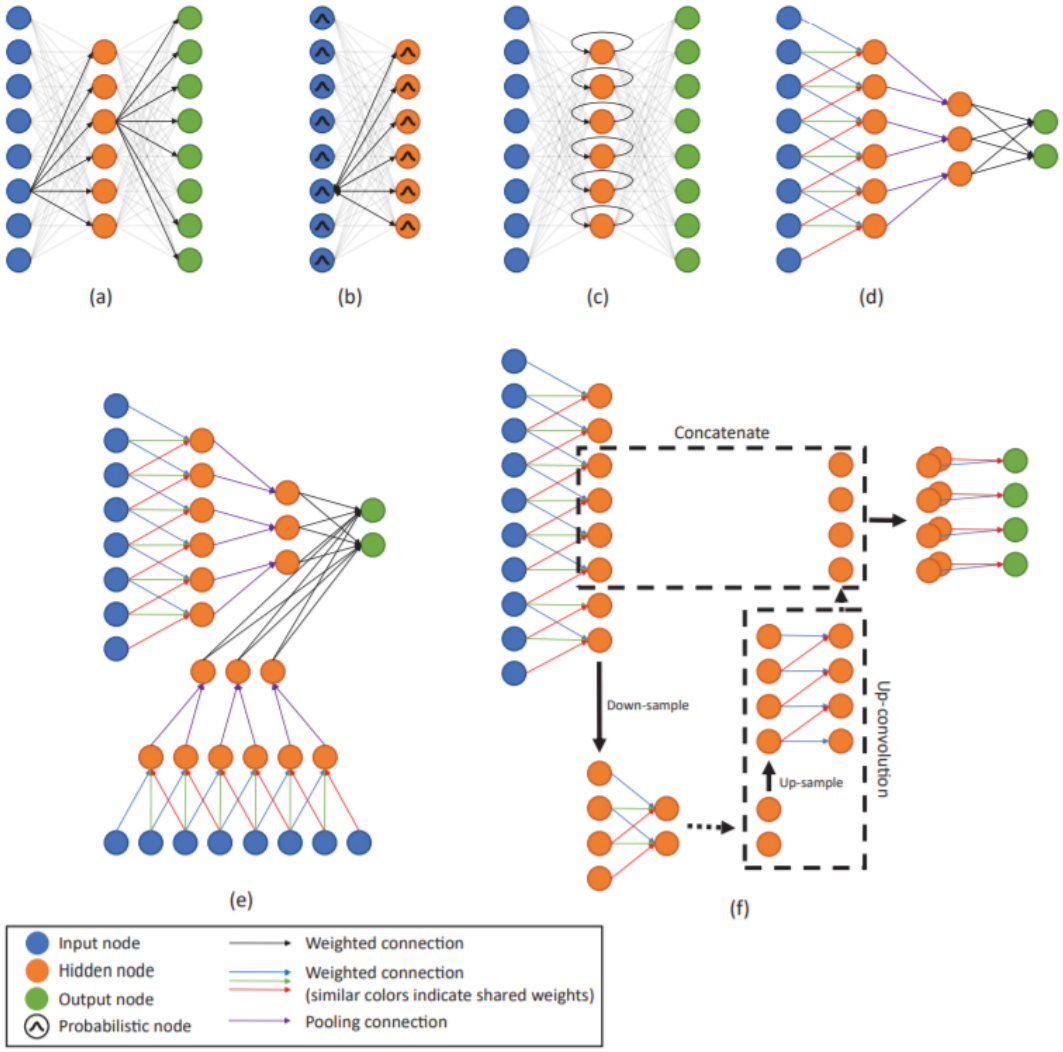
[7]https://github.com/Microsoft/CNTK
[8]https://chainer.org/
[9]https://dataplatform.ibm.com/docs/content/analyze-data/ml_dlaas_ibm_ddl.html
[10]https://www.cs.waikato.ac.nz/ml/weka/index.html

(IBM, 2015). Desde de 2015 o hospital vem utilizando o Watson para ajudar na escolha do tratamento do câncer de mama. O computador usa sua capacidade de interpretar textos, e até dezembro de 2015 já tinha analisado perto de 15 milhões de páginas de conteúdo médico, incluindo mais de 200 livros e 300 revistas médicas. Em um estudo duplo-cego, apresentado no *San Antonio Breast Cancer Symposium* em 2016, os médicos do *Manipal Hospital* descobriram que o Watson estava de acordo com as recomendações (para a escolha de um tratamento) do conselho de tumores em 90% dos casos de câncer de mama. Essa máquina tem acesso ao *PubMed*, uma biblioteca pública americana de publicações, e vem se aperfeiçoando cada vez mais com o passar dos anos. Existem outros usos para esse supercomputador na medicina, como monitorização 24 horas de pacientes, análise de riscos de infecção hospitalar, entre outros.

Com relação ao uso dos computadores para analisar imagens na área médica, pesquisadores já tentavam desde a década de 1970 analisar de modo automático imagens processando os pixels de baixa resolução (LITJENS et al., 2017). Depois as pesquisas passaram para técnicas com o uso de algoritmos não supervisionados, como análise do componente principal e métodos de *clustering*. Hoje os computadores evoluíram para o uso das redes neurais profundas que recebem como entrada imagens e têm como saída classificação de doenças, por exemplo. LITJENS G. et al estudaram 300 publicações que usavam redes de aprendizado profundo na área médica, assim puderam observar quais as arquiteturas mais populares, como esses métodos estão sendo utilizados e para quais áreas médicas eles estão sendo usados. O levantamento feito pelos autores mostraram que as publicações que envolvem aprendizado profundo e imagens em medicina vêm crescendo muito, dos 308 artigos analisados, 242 foram publicados em 2016 ou em janeiro de 2017. Sobre as arquiteturas, eles observaram que a CNN é a mais popular, sendo que no começo os pesquisadores estavam utilizando essa arquitetura realizando um pré-treinamento com métodos não-supervisionados. Depois passaram a utilizar essa mesma arquitetura, mas com treinamento de ponta-a-ponta para interpretar imagens médicas. Porém, foi observado que a RNN também vem ganhando popularidade nessa área de estudo. A Figura 26 foi retirada do artigo e mostra as arquiteturas mais utilizadas: a) *Auto-encoder*, b) Máquina Restrita de Boltzmann, c) Rede Neural Recorrente, d) Rede Neural Convolucional, e) Rede Neural Convolucional Multi-stream, f) *U-net*.

A classificação de imagens de exames é um dos objetivos do uso das RNAP na medicina. Como entrada temos imagens de um exame de fundo de olho, por exemplo. E como saída temos a classificação se há ou não doença, como a diabetes. Outro tipo de classificação é a de um objeto ou lesão, no qual temos uma imagem de entrada como uma tomografia de tórax, e a classificação de uma lesão em duas ou mais classes, como a classificação de um nódulo pulmonar. Esse é um tipo mais complexo de classificação, pois pode precisar de mais informações sobre o aparecimento da lesão. Para solucionar esses problemas, autores utilizaram associações de arquiteturas de redes neurais ou *multi-stream architectures* (LITJENS et al., 2017). Outra utilidade para as redes neurais foi a detecção de órgãos ou regiões em imagens. Isto tem um papel importante no planejamento de terapias ou intervenções, como cirurgias. O uso da CNN

Figura 26 – Arquiteturas utilizadas na análise de imagens médicas

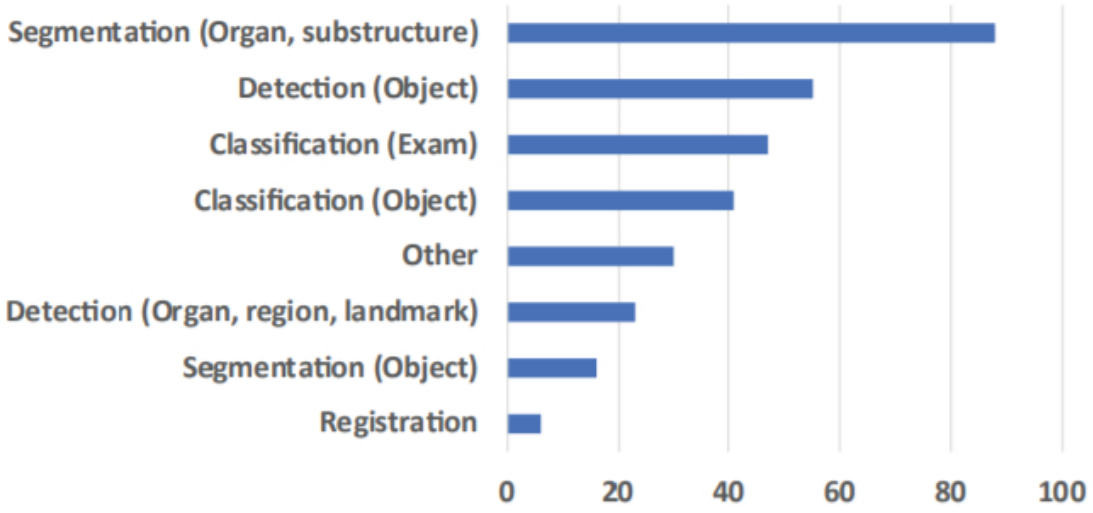


Fonte: (LITJENS et al., 2017)

tem sido bastante satisfatório na classificação dessas imagens, que geralmente são em 3D. Também é utilizada a detecção de objetos ou lesões em um espaço de uma imagem. Essa detecção busca encontrar nódulos, por exemplo, em uma tomografia de tórax. CNN já é uma arquitetura que foi usada em 1995 para detecção de nódulos em imagens de rx, e hoje ainda é a técnica mais utilizada para esse objetivo. Há uma diferença importante entre detectar ou classificar um objeto, e um ponto chave é que, como cada pixel é classificado, geralmente o equilíbrio da classe é direcionado severamente para a classe sem objeto em uma conFiguração de treinamento. A segmentação de órgãos ou estruturas usa as redes neurais para detectar os limites de um órgão, podendo-se calcular o volume do coração, por exemplo. Esse é o objetivo da maioria dos trabalhos analisados por LITJENS G. et al. Um tipo específico de CNN é o mais popular para esse propósito, a *U-net*. Porém alguns pesquisadores já vêm utilizando a RNN para a segmentação de imagens médicas. Existe também a segmentação de objetos ou lesões. Outro uso das redes neurais é para o registro de imagens na área médica, por exemplo para alinhamento espacial.

Isso é utilizado como técnica para tratamento de imagens. Por exemplo, estimar uma medida de similaridade para duas imagens para orientar uma estratégia de otimização iterativa ou para prever diretamente os parâmetros de transformação usando redes de regressão profunda. Outra utilidade é a recuperação de imagem baseada em conteúdo, no qual há uma descoberta de conhecimento em grandes quantidades de dados como o objetivo de identificar imagens. A CNN é bastante utilizada por causa da sua capacidade de aprender recursos ricos em vários níveis de abstração. A Figura 27 mostra a quantidade desses objetivos encontrados nos 308 artigos estudados por LITJENS G. et al.

Figura 27 – Objetivos de estudos com RNAP na análise de imagens médicas



Fonte: (LITJENS et al., 2017)

Algumas áreas da medicina vêm sendo objeto de estudo dos desenvolvedores de redes neurais profundas. Na neurologia, existem estudos voltados para a classificação automática da Doença de Alzheimer, na segmentação do tecido cerebral e de estruturas anatômicas, como o hipocampo. Há também o importante papel na detecção e segmentação de lesões, como tumores ou micro hemorragias. A grande maioria das pesquisas nessa área são feitas em imagens 3D de ressonância nuclear magnética (RNM) do cérebro, apesar da maioria dos trabalhos estudados por LITJENS G. et al. terem usado a técnica de dividir a imagem 3D em imagens 2D, para melhor desempenho. Em competições que visam analisar o desempenho das arquiteturas de redes neurais, como o Desafio de Segmentação de Lesão de Esclerose Múltipla Longitudinal de 2015 ou o Desafio de Segmentação de Lesão de Acidente Vascular Cerebral Isquêmico de 2015, as equipes com as melhores posições utilizaram a arquitetura CNN.

Na área da oftalmologia, as redes neurais recentemente vêm sendo utilizadas na classificação de imagens de fundo de olho, principalmente. Várias aplicações são utilizadas: segmentação de estruturas anatômicas, segmentação e detecção de anormalidades retinianas, diagnóstico de doenças oculares e avaliação da qualidade da imagem. Mas o grande destaque vai para a detecção de retinopatia diabética, uma complicação comum em pacientes com diabe-

tes. Em uma competição em 2015 organizada pela Kaggle (uma plataforma que visa organizar competições em mineração de dados) foram analisadas 35 mil imagens de fundo de olho. A maioria dos 661 times que disputaram o torneio usaram a CNN, sendo que 4 times obtiveram um desempenho melhor que o ser humano usando a CNN Ponto-a-Ponto.

Na análise de imagens torácicas de radiografia e tomografia computadorizada, a detecção, caracterização e classificação de nódulos é a aplicação mais comumente abordada. A radiografia de tórax é o exame radiológico mais comum, e pesquisadores utilizam a CNN para interpretar as imagens deste exame e a RNN para analisar os textos nos laudos. Na detecção de nódulos nas imagens de tomografia, a CNN recebe o destaque.

A patologia digital é um campo de pesquisa promissor, lâminas estão sendo digitalizadas e podem ser estudadas por computadores. Essas lâminas após serem digitalizadas, permitem a criação de uma grande imagem única, uma tecnologia que vem sendo chamada de *whole-slide images*. Isso permitiu que pesquisadores em aprendizado profundo pudessem desenvolver aplicações para essa área. Dentre essas aplicações estão a detecção, segmentação e classificação de núcleos; segmentação de grandes órgãos e detecção e classificação de doenças. Novamente a arquitetura que fica em destaque é a CNN.

Na ginecologia deve-se destacar o uso das redes neurais na detecção do câncer de mama. A grande maioria dos estudos usam a mamografia como fornecedora de imagens. Os estudos focam na detecção e classificação de massas tumorais, na detecção e classificação de micro calcificações e na pontuação do risco de câncer de mama baseado nas imagens. No estudo de LITJENS G. et al (2017), apenas um trabalho estudou o uso de aprendizado profundo em imagens de RNM. Nessa área, vários métodos são utilizados para o aprendizado de máquina, como o aprendizado semi-supervisionado, aprendizado fracamente supervisionado, aprendizado por transferência, CNN e associações destes métodos.

Na cardiologia, as redes neurais estão sendo mais utilizadas na análise de imagens de RNM, e a tarefa mais comum é a segmentação do ventrículo esquerdo. Mas outras utilidades estão presentes, como avaliação da qualidade da imagem, pontuação automatizada de calcificação e rastreamento da linha central coronariana. A arquitetura mais usada foi a CNN, porém uma competição importante, a *2015 Kaggle Data Science Bowl*, teve o uso da *U-net* entre os líderes além da CNN convencional. Existe também um crescente interesse no uso do aprendizado profundo na análise de eletrocardiogramas, pois as técnicas mais tradicionais de aprendizado de máquina não conseguiram extrair as características importantes desse exame (PYAKILLYA; KAZACHENKO; MIKHAILOVSKY, 2017), fato que foi possível com o uso da CNN.

Na região abdominal, a maioria dos trabalhos visa a localização e segmentação de órgãos, como fígado, rins, bexiga e pâncreas. Algumas pesquisas se relacionam com a segmentação de tumores no fígado. As imagens mais comuns são provenientes de RNM para a próstata e tomografia para os outros órgãos. Em competições que analisam o desempenho de arquiteturas, apenas após 2016 os métodos mais modernos, como a CNN e *U-net*, conseguiram as primeiras posições.

Outras áreas da medicina também podem ser citadas, como a musculoesqueléticas, na qual imagens foram analisadas por algoritmos de aprendizagem profunda para segmentação e identificação de anormalidades ósseas, articulares e de partes moles. Isto ocorreu em diversas modalidades de imagem. Na dermatologia, a CNN vem sendo utilizada para analisar imagens de dermatoscopia (lesões de pele), com o objetivo de detectar o câncer de pele.

3.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Existe uma relação íntima entre Extração de Conhecimento e o Aprendizado de Máquina. Para poder aprender, um computador assim como o ser humano, deve primeiro organizar as informações recebidas (dados) e prepará-las para poderem ser analisadas e interpretadas. Essa análise vai ser feita através dos algoritmos de aprendizado, como a RNAP. Os dados que entram nessa rede recebem pesos, que marcam a sua importância para a solução do problema proposto. Conforme o algoritmo vai sendo processado, através de várias iterações, os pesos vão sendo recalculados, esse é o processo de aprendizado. As Redes Neurais usam um tipo de aprendizado supervisionado, e precisam de uma parcela das amostras para o treinamento e uma parcela para os testes. Existem técnicas que visam separar essas amostras de um modo que os resultados obtidos possam refletir o mundo que as amostras pretendem representar. Um exemplo eficiente de um desses métodos é o *Cross-Validation*. As RNAP são muito eficientes para problemas de classificação, principalmente relacionados com a análise de imagens. Vemos o crescente uso do aprendizado profundo na área da medicina, em particular a arquitetura CNN está sendo muito estudada na análise de imagens médicas, pois vem demonstrado ser muito eficiente nesta tarefa.

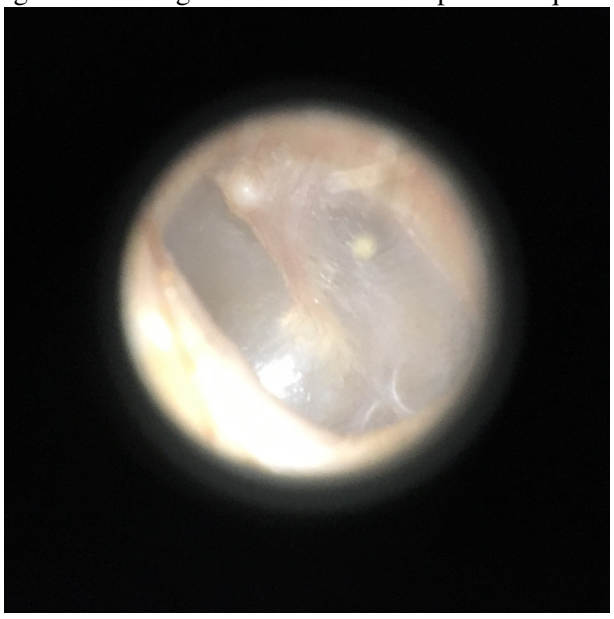
A grande dificuldade das RNAP é o treinamento, que pode ser muito custoso para o computador. Porém, hoje vemos algumas técnicas que solucionam bem esse problema. Plataformas e bibliotecas já incorporam essas técnicas, como por exemplo o uso de GPUs, que são muito mais eficientes que um processador comum, pois permitem o processamento em paralelo em vários núcleos. Isso é fundamental para o treinamento eficiente das redes neurais profundas, pois vários neurônios podem ser treinados ao mesmo tempo. Essa eficiência melhorou tanto, que hoje temos o exemplo de uma biblioteca disponível para dispositivo móvel, o TensorFlow Lite da Google. Essa é uma ferramenta interessante para nossa proposta de trabalho, pois caso o AM consiga classificar de modo eficiente as imagens da MT, no futuro poderá ser criado um mecanismo para dispositivos móveis capaz de capturar as imagens do canal do ouvido e poder classificar se há doença na orelha média (otite). Um estudo realizado em 2017 por Lundberg T. et al. procurou mostrar a taxa de acerto do diagnóstico de otites por médicos generalistas usando um otoscópio comum e depois usando vídeo-otoscopia. No primeiro caso o acerto foi de 89% e no segundo o acerto máximo foi de 94%. Essas taxas podem ser usadas como comparação para medir o desempenho do programa proposto pelo nosso trabalho.

4 DESENVOLVIMENTO DE UMA REDE NEURAL PARA CLASSIFICAÇÃO DE IMAGENS DA MEMBRANA TIMPÂNICA

Esse trabalho se propõe a construir um modelo de RNAP que possa classificar as imagens da MT humana em com ou sem doença. Alguns passos devem ser realizados conforme a teoria da Extração de Conhecimento, estes serão detalhados mais adiante.

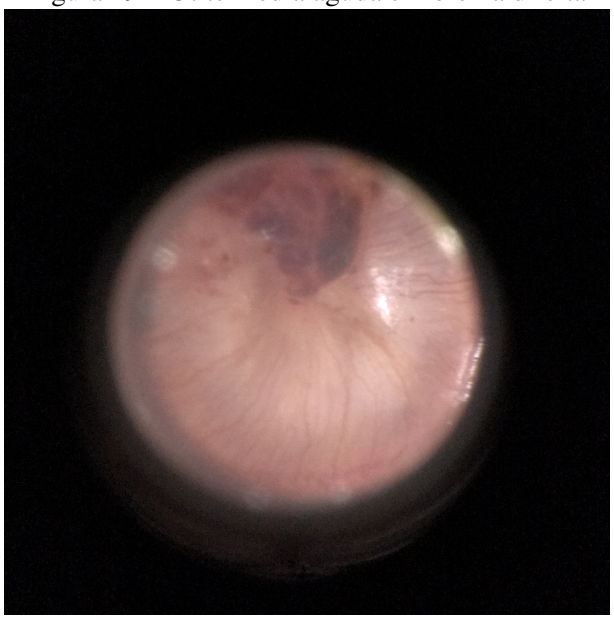
No futuro, uma possível utilidade para o modelo proposto seria a possibilidade de criar um dispositivo que possa ser posicionado no início do conduto auditivo humano e consiga capturar a imagem da MT para que esta possa ser classificada por um *software*. Porém, nem sempre o conduto auditivo permite que a MT seja vista, ele pode estar obstruído por *cerumen*, por exemplo. Por isso, vamos utilizar uma terceira classificação para as imagens, a de obstrução do conduto auditivo. Portanto, deverão ser apresentadas ao AM imagens do conduto auditivo humano no qual possa ser vista a MT saudável, a MT com doença ou não possa ser vista a MT devido à obstrução do conduto auditivo externo. Com isso, durante o treinamento em nossa RNAP teremos ao todo 3 classes: "membrana timpânica sem doença", "membrana timpânica com doença"e "conduto auditivo com cerumen". A Figura 28 é um exemplo de uma imagem capturada de um conduto auditivo esquerdo, no qual é possível identificar e classificar a MT como saudável. Essa membrana pode ser descrita como íntegra (não contém perfuração), transparente, com brilho, com vascularização preservada, sem retrações ou abaulamentos e com o martelo (um dos ossículos da orelha média) em sua posição habitual. Exemplos de uma imagem da classe *com doença* e da classe *com cerumen* podem ser vistos nas Figuras 29 e 30, respectivamente.

Figura 28 – Imagem da membrana timpânica esquerda



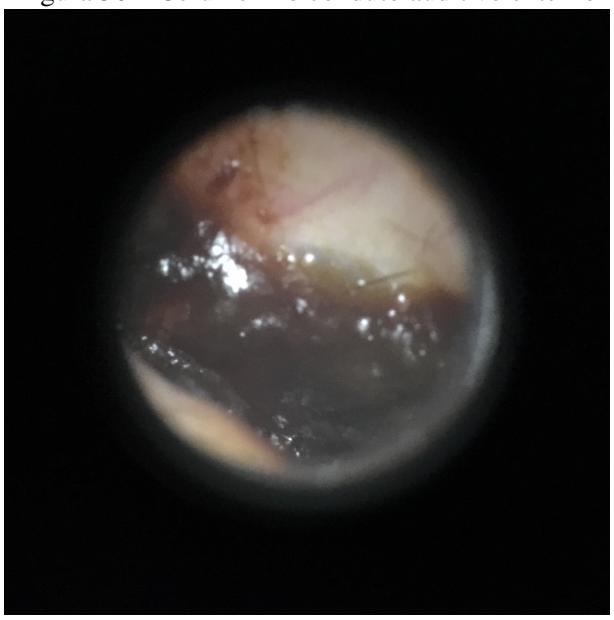
Fonte: Autor

Figura 29 – Otite média aguda em orelha direita



Fonte: Autor

Figura 30 – Cerumen no conduto auditivo externo



Fonte: Autor

4.1 COLETA DE DADOS

Os dados utilizados para o treinamento e testes com nossa rede neural são imagens do conduto auditivo externo humano, como já foi citado. Essas imagens foram coletadas através de um dispositivo chamado *Oto for Clinicians* da empresa *Cellscope*[1], os arquivos são salvos no formato JPG. Esse dispositivo é acoplado somente a alguns modelos de *smartphones* da

[1] https://www.cellscope.com/

empresa Apple (*iPhone*). Ele utiliza a câmera e iluminação destes aparelhos, por isso o *Oto for Clinicians* só se adpta ao *iPhone* e não às outras marcas. Deste modo, ele consegue capturar a imagem através de um espéculo de ouvido, como mostra a Figura 31, retirada do website da *Cellscope*. O aplicativo utilizado na captura das imagens se chama *CellScopeLite* e é fornecido pela empresa que desenvolveu o dispositivo.

Figura 31 – Oto for Clinicians, da Cellscope

Fonte: Cellscope

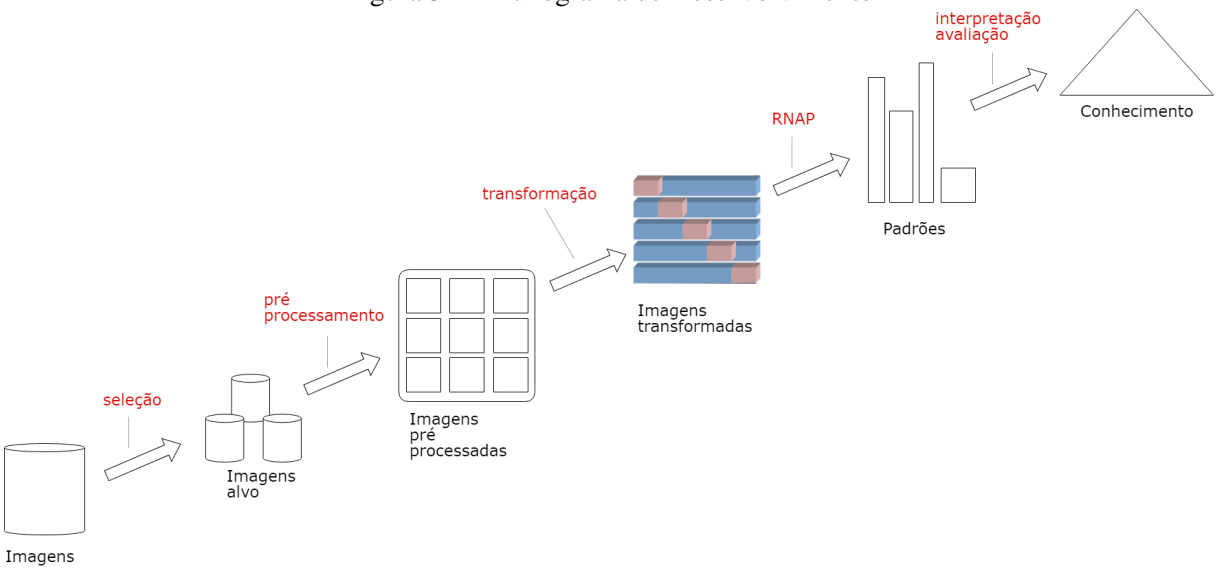
Os arquivos apresentam um tamanho próximo de 1 Mb, sendo o menor com 625 Kb e o maior com 1.48 Mb. As imagens têm as dimensões de 3024 x 3024 pixels, a resolução de 72 x 72 DPI (pontos por polegada) e a representação de cores em sRGB. Elas foram coletadas pelo autor do trabalho, que é médico especialista em otorrinolaringologia pela ABORL-CCF (Associação Brasileira de Otorrinolaringologia e Cirurgia-Cérvico Facial). Durante a coleta, primeiramente foi realizada uma otoscopia com um otoscópio pneumático da marca *Welch Allyn* e, junto com as informações clínicas do paciente, foi realizado o diagnóstico em relação a orelha externa e média. Isto foi feito para facilitar a classificação da imagem da MT previamente, pois a proposta deste trabalho é realizar um treinamento supervisionado com as redes neurais. Foram coletadas ao todo 208 imagens do conduto auditivo externo humano.

4.2 VISÃO GERAL DO SOFTWARE

O fluxograma de desenvolvimento do *software* é mostrado na Figura 32.

Na primeira etapa, imagens do conduto auditivo externo humano serão selecionadas pelo autor deste trabalho. Na segunda etapa serão eliminadas imagens repetidas, danificadas ou inconsistentes, esta será a fase de pré-processamento dos dados. Na terceira etapa, serão verificados os formatos dos arquivos das imagens, para estes sejam compatíveis com o algoritmo

Figura 32 – Fluxograma de Desenvolvimento



Fonte: Autor

selecionado para o AM. A quarta etapa será dividida em 2 fases: treinamento e testes. As imagens serão divididas em amostras para estas fases de acordo com o método de *Cross-Validation*. Na fase de treinamento, os dados serão trabalhadas pelo algoritmo da CNN, que deverá encontrar padrões e classificar as imagens. Então o algoritmo será avaliado na fase de testes, na qual poderemos obter uma taxa de erros. Na última etapa, os resultados serão avaliados e o desempenho do algoritmo poderá ser comparado com a literatura. Para facilitar esse processo usaremos o TensorFlow, um *framework* usado para treinar redes neurais, de fácil manipulação.

4.3 MANIPULAÇÃO DAS IMAGENS

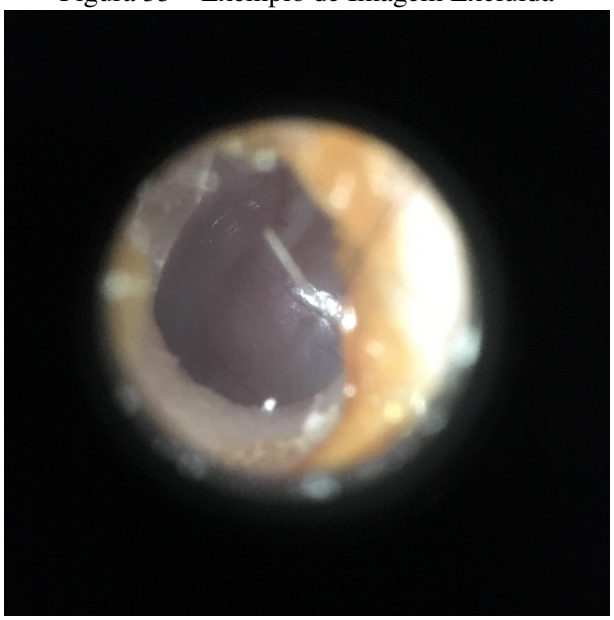
Como foi visto no processo de Extração de Conhecimento, existem etapas anteriores à mineração de dados, que srão mostradas a seguir.

4.3.1 Seleção

Em outro momento, foi realizada nova análise das imagens pelo autor do trabalho, com o objetivo de verificar a nitidez da MT. Esta etapa foi necessária após as primeiras tentativas de treinamento da rede. Nessas tentativas foi verificado uma acurácia abaixo de 50%, além disso a rede classificava quase todas as imagens da classe *com doença* como *sem doença*. Ao verificar as imagens foi visto que muitas apresentavam uma baixa nitidez da MT, por isso foi necessário excluí-las. Observou-se que as imagens excluídas foram aquelas que tiveram problemas como foco ruim e iluminação insuficiente. Também foram excluídas as imagens realizadas com um espéculo pequeno, pois houve uma diferença importante na luminosidade das imagens e tam-

bém uma diferença na área preta do espéculo. Na Figura 33 podemos ver um exemplo de uma imagem excluída, um pequeno cerumen na entrada do conduto auditivo dificultou a iluminação da MT, não permitindo a classificação desta imagem. Assim, foram selecionadas um total de 133 imagens, sendo 56 classificadas com *sem doença*, 34 como *com doença* e 43 como *com cerumen* (não sendo possível ver a MT).

Figura 33 – Exemplo de Imagem Excluída



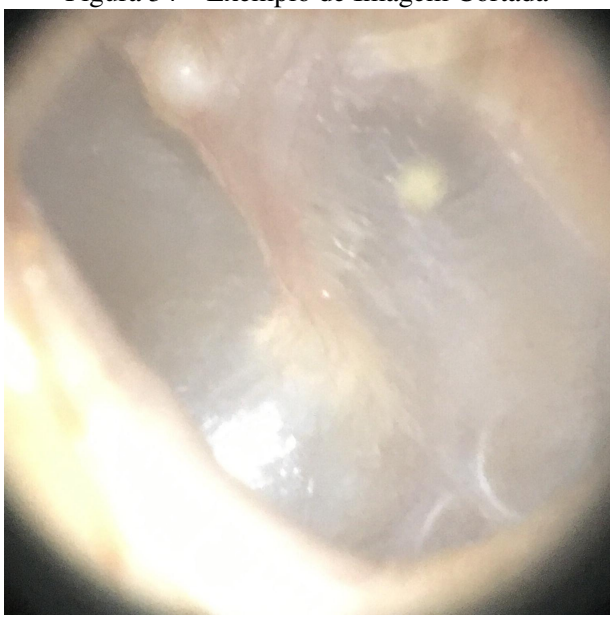
Fonte: Autor

4.3.2 Pré-processamento

Nesta etapa, foi realizado uma diminuição das dimensões da imagem através do corte de pixels periféricos. Podemos observar na Figura 28, por exemplo, que as imagens coletadas apresentam uma região de coloração preta, que correspondem ao espéculo do dispositivo que capta as imagens da otoscopia. Essa região corresponde a quase 50% das dimensões da imagem e poderia gerar um custo computacional desnecessário durante o treinamento da rede neural. Portanto, optou-se por utilizar um quadrado central da imagem, que corresponde a 50% dos pixels, eliminando assim grande parte da região preta. Parte dos pixels correspondentes a imagem da MT também foi perdida em alguns casos, porém a extrema periferia destas imagens não é importante para a classificação proposta neste trabalho. Para este processo, foi utilizada uma biblioteca para Python chamada Pil (criada por Fredrik Lundh e colaboradores), que é utilizada na manipulação de imagens. Esta biblioteca apresenta uma função chamada "crop", na qual recebe um conjunto de quatro números que delimitam um quadrado para ser retirado de uma imagem. Os 2 primeiros números são as coordenadas do canto superior esquerdo, e os outros 2 números são as coordenadas do canto inferior direito. Através desta biblioteca, foi possível trazer as imagens de 3024 x 3024 pixels para a linguagem Python, e elas puderam ser

representadas por matrizes com 3024 valores de altura e 3024 de largura, isto para cada uma das cores no sistema sRGB. Portanto, cada imagem com as dimensões citadas corresponde a uma matriz com 27.433.728 valores (3024 x 3024 x 3). Depois foram utilizados apenas os 50% centrais de cada imagem, passando um vetor com os valores 756, 756, 2268 e 2268 para a função "crop". Isto delimita um quadrado central, e reduz as dimensões de cada imagem para 1512 x 1512 (50% do original), o que corresponde a uma matriz com 6.858.432 valores (uma redução de 75% nestes valores). Na Figura 34 podemos ver o exemplo de como ficou cortada a imagem da Figura 28.

Figura 34 – Exemplo de Imagem Cortada



Fonte: Autor

Quanto maior for a quantidade de imagens para o treinamento, melhor será o aprendizado de uma CNN. Por exemplo, existem pacotes de imagens disponíveis na internet para auxiliar no aprendizado de estudantes nesta área, alguns até já são disponibilizados por bibliotecas como o TensorFlow. O *"MNIST"* é um pacote consagrado de imagens de letras escritas por humanos, ele contém 55 mil imagens para treinamento e 10 mil para testes. Outro pacote também popular é o *"CIFAR-10 dataset"*, que contém 50 mil imagens (carros, aviões, pássaros, gatos, por exemplo) para treinamento e 10 mil imagens para teste. Portanto, 133 é um número de imagens que dificultou os testes com as redes neurais. Os treinamentos com redes mais simples (com poucas camadas) apresentaram uma acurácia muito baixa (abaixo de 60%), e as redes mais complexas também obtiveram uma acurácia abaixo do valor citado, devido a um fenômeno chamado de *overfitting*, que será explicado mais adiante. Para tentar solucionar este problema foi aumentado o número de imagens totais através da rotação e espelhamento destas. Cada imagem originou um total de 6 novas imagens: rotações de 90, 180, 270 graus; espelhamento direita-esquerda; espelhamento em cima-embaixo e espelhamento diagonal. Deste modo, foram obtidas 931 imagens (133 multiplicado por 7).

4.3.3 Transformação

A CNN trabalha realizando operações entre matrizes. Estas são formadas por números que correspondem a cada pixel da imagem. Por exemplo, uma imagem na escala de cinza, que apresenta as dimensões de 32 por 32 pixels, pode ser representada por uma matriz com 32 linhas por 32 colunas. São 1024 números, sendo que cada número representa uma quantidade de branco que vai de 0 a 255. Se a imagem for colorida utilizando-se o método sRGB, teremos uma matriz tridimensional, pois teremos 32 linhas e 32 colunas para representar cada quantidade de cor vermelha, verde e azul. Ou seja, no exemplo citado serão 3072 números entre 0 e 255.

Quando é criado um modelo para treinamento da CNN, é preciso especificar o formato da camada de entrada. Ou seja, após planejar a arquitetura do modelo, precisamos transformar as imagens em um conjunto de números, sejam eles na forma de uma matriz bidimensional ou tridimensional. As imagens coletadas para este estudo apresentavam um tamanho de 3024 por 3024 pixels, e o sistema de cores sRGB. Porém, estas imagens foram cortadas em um tamanho de 1512 por 1512 pixels na fase de pré-processamento. Na fase de transformação, as imagens foram redimensionadas para um tamanho conforme o planejamento do modelo a ser treinado. Isto foi realizado com o auxílio da biblioteca de manipulação de imagens para Python, a Pil. Esta ferramenta apresenta uma função chamada *resize*, que recebe uma dupla de valores: largura e altura em pixels. Assim, esta função vai retornar uma cópia da imagem dentro das dimensões especificadas. Para transformar a imagem em uma matriz de números, utilizamos uma outra biblioteca chamada Numpy, que facilita o trabalho com vetores e matrizes. Esta ferramenta possui uma função chamada *array*, que recebe um objeto (podendo ser uma imagem) e retorna um vetor. No caso de uma imagem de 1512 por 1512 pixels, no sistema sRGB, a função *array* vai retornar uma matriz tridimensional de formato (1512,1512,3). Esta matriz já pode ser usada na camada de entrada do modelo da CNN.

4.4 TREINAMENTO DA CNN

Os treinamentos foram realizados através do Google Cloud Platform, utilizando um serviço que disponibiliza máquinas virtuais (VM) na internet. A VM utilizada neste serviço tinha as seguintes características: 8 vCPUs, 30 GB de memória principal, 1 GPU NVIDIA Tesla P100.

4.4.1 Parâmetros de treinamento

Na Seção 3.4 foi explicada com detalhes a estrutura da CNN, ela é formada por camadas convolucionais, de agregação (*pooling*) e camadas inteiramente conectadas (ou camada densa). Esta rede neural faz operações com matrizes, por isso a entrada de dados é na forma de matriz, assim como a conFiguração de alguns parâmetros.

Neste trabalho, foi utilizado uma API (Interface de Programação de Aplicativos) chamada Keras para facilitar a visualização do modelo da CNN e permitir a rápida troca de parâmetros para os vários treinamentos realizados. O Keras[2] é uma API de redes neurais de alto nível, escrita em Python e capaz de rodar em cima do CNTK, Theano ou TensorFlow, como foi no caso do presente estudo. Ela foi escolhida devido sua interface de fácil manuseio, sua capacidade de trabalhar com as CNNs e com CPU ou GPU. Nesta API, um modelo de rede neural pode ser atribuído a uma variável, facilitando sua manipulação como adição ou remoção de camadas, alteração de parâmetros, salvamento e resgate de modelos. Na Figura 35 vemos o exemplo da interface do melhor modelo criado para este projeto.

Figura 35 – Modelo com melhor desempenho

```
In [17]: model = Sequential()

In [18]: model.add(Conv2D(filters=8, kernel_size=(16,16), input_shape=(1512,1512,3), strides=(2,2),
                          padding='valid', activation='relu'))
         model.add(Conv2D(filters=16, kernel_size=(12,12), strides=(2,2), padding='valid', activation='relu'))
         model.add(MaxPool2D(pool_size=(2,2), strides=(2,2), padding='valid'))

In [19]: model.add(Conv2D(filters=32, kernel_size=(8,8), strides=(2,2), padding='valid', activation='relu'))
         model.add(Conv2D(filters=48, kernel_size=(4,4), strides=(2,2), padding='valid', activation='relu'))
         model.add(MaxPool2D(pool_size=(2,2), padding='valid'))

In [20]: model.add(Conv2D(filters=64, kernel_size=(2,2), strides=(2,2), padding='valid', activation='relu'))
         model.add(MaxPool2D(pool_size=(2,2), padding='valid'))

In [21]: model.add(Dropout(rate=0.5))
         model.add(Flatten())
         model.add(Dense(2024, activation='relu'))
         model.add(Dropout(rate=0.5))
         model.add(Dense(3, activation='softmax'))
         optimizer = Adam(lr=0.0001)
```

Fonte: Autor

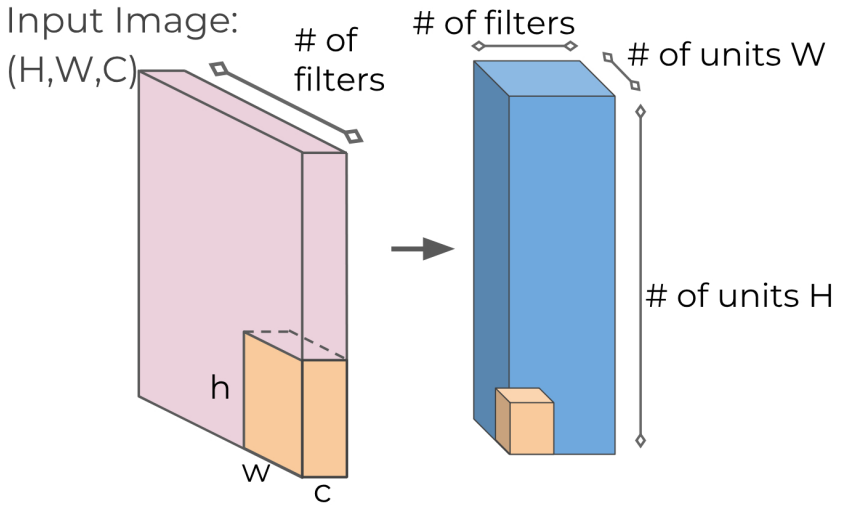
Na Figura 35 também podemos observar uma estrutura em células, com um fundo mais amigável do que o do Prompt de Comando do Windows, por exemplo. Esta estrutura é do Jupyter Notebook, um aplicativo web de código aberto que permite organizar a programação na linguagem Python, entre outras utilidades. Na primeira célula podemos observar a criação de um modelo sequencial do Keras e sua atribuição à variável *model*. Na próxima célula vemos a adição, na primeira linha, de uma camada convolucional bidimensional, ou seja, para trabalhar com *Kernels* de duas dimensões.

O primeiro parâmetro dessa camada convolucional é o número de filtros (*filters*), que corresponde a 8 neste exemplo. Um dos fundamentos da CNN é tentar trabalhar com determinadas características da imagem(PORTILLA, 2017). Por isso, cada filtro vai isolar uma característica, como os contornos pretos de uma Figura por exemplo. Então a quantidade de filtros em cada camada representa o número de características que o modelo vai tentar trabalhar. Realizar o treinamento utilizando filtros, trabalhando com partes da imagem em vez de toda ela, representa uma vantagem da CNN em termos de desempenho computacional. O se-

[2]https://keras.io/

gundo parâmetro é o tamanho do filtro (*kernel_size*). Na Figura 35 vemos os valores de 16 de altura por 16 de largura na primeira linha da célula 18. Como foi visto, a camada de entrada é uma estrutura tridimensional, composta por (altura X largura X cores) da imagem. Com os dois primeiros parâmetros da camada convolucional, podemos delimitar blocos que serão trabalhados nesta estrutura inicial, como pode ser visto na Figura 36. Através do uso das camadas convolucionais, de agregação e inteiramente conectadas estaremos montando a arquitetura de uma CNN conforme já foi ilustrado na Figura 23.

Figura 36 – Estrutura em blocos da CNN



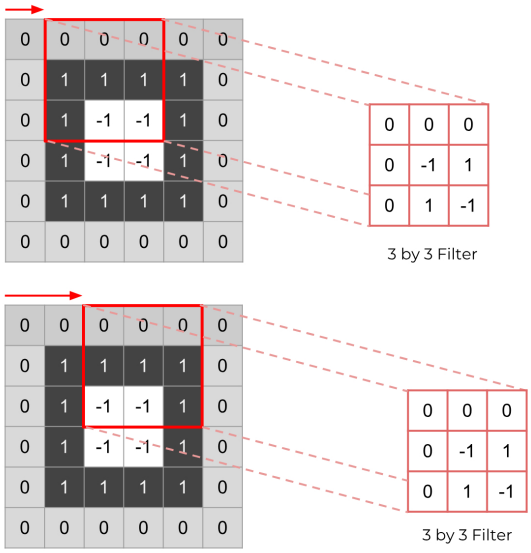
Fonte: (PORTILLA, 2017)

O parâmetro chamado *input_shape* aparece apenas na primeira camada e se refere a estrutura da camada de entrada, sendo os valores correspondentes à altura, largura e sistema de cores das imagens (1 para escala de cinza, 3 para o sistema sRGB, por exemplo). Depois deve-se pensar no deslocamento que o bloco de filtro vai fazer após trabalhar com cada parte da imagem inicial. Este parâmetro se chama *strides*, no exemplo temos um tamanho de 2 de altura por 2 de largura como sendo o deslocamento. Na Figura 37 pode-se ver um exemplo com *stride* de 1.

Outro parâmetro importante se chama *padding* e se refere à opção de envolver a matriz sendo analisada com números 0, por exemplo. Isto pode ser visto na Figura 37, nas linhas e colunas representadas em cinza. O *padding* é útil para que o filtro não saia das dimensões da matriz ao se deslocar.

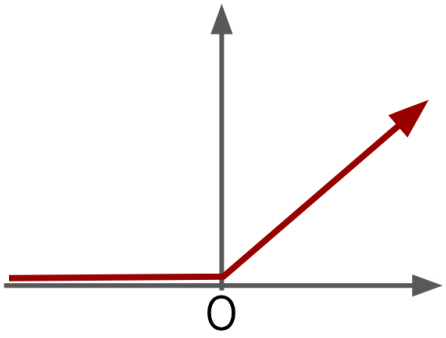
Como foi explicado na Seção 3.4, as redes neurais profundas utilizam uma operação não-linear como função de ativação. Este parâmetro também deve ser passado em cada camada, no exemplo da Figura 35 foi escolhido uma função chamada ReLU para o parâmetro *activation*. ReLU significa *Rectified Linear Unit*, ela retorna o valor zero para todos os valores menores ou iguais a zero, ou retorna o próprio valor quando estes são positivos. Os valores de saída da função ReLU podem ser representados no gráfico mostrado na Figura 38.

Figura 37 – Stride de 1



Fonte: (PORTILLA, 2017)

Figura 38 – Gráfico da função ReLU



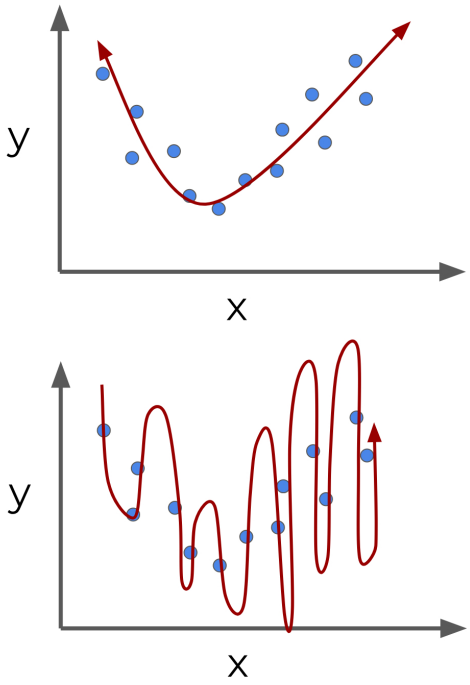
Fonte: (PORTILLA, 2017)

O modelo da Figura 35 apresenta as duas primeiras camadas como sendo camadas convolucionais, seguidas de uma camada de agregação. Esta camada é representada no Keras pela classe *MaxPool2D*, é realizado uma diminuição das matrizes resultantes da camada anterior utilizando o método do número máximo. Devemos escolher a altura e largura da matriz de agregação através do parâmetro *poll_size*, além de poder escolher o *strides* e o *padding*.

Observamos outros parâmetros na célula 21 da Figura 35. Na primeira linha, foi optado por utilizar um mecanismo chamado *dropout*. Este é método utilizado nas redes neurais para evitar o *overfitting* durante o treinamento da rede. *Dropout* consiste em remover de modo randômico neurônios durante o treinamento, para que a rede não fique dependendo de nenhum neurônio específico. No exemplo seguido, o *dropout* foi utilizado em dois momentos, com uma taxa de remoção de neurônios de 50%. *Overfitting* é um fenômeno que pode ocorrer no treinamento das RNAP, e sua chance de ocorrer aumenta com o número crescente de camadas e parâmetros da rede, e diminui com um maior número de dados para treinamento. Na Figura

39 podemos observar no primeiro gráfico um modelo (linha em vermelho) que representa bem um conjunto de dados (pontos azuis). Porém, no segundo gráfico podemos ver um modelo que passa por todos os dados do treinamento, mas que provavelmente não vai representar novos dados que podem ser inseridos no gráfico. Deste modo, teremos durante o treinamento uma acurácia alta, mas teremos este valor baixo quando utilizarmos os dados de teste. Este é um exemplo de *overfitting*.

Figura 39 – Exemplo de overfitting



Fonte: (PORTILLA, 2017)

Na última linha do modelo seguido, podemos ver um parâmetro importante da rede. Devemos escolher o *optimizer*, que corresponde ao método de como a rede neural vai ajustar seus pesos para o aprendizado. A rede neural deve tentar minimizar o erro, para que os pesos possam refletir em um modelo que represente da melhor maneira possível os dados. A CNN é uma rede de aprendizado supervisionado, que vai realizar o *backpropagation* para ir corrigindo o erro. A velocidade com que a rede vai aprendendo pode ser regulada pela taxa de aprendizado, que possui o valor de 0.0001 no exemplo. Porém, podemos observar que o *optimizer* não utiliza esta taxa de maneira simples, ou seja, está taxa não é constante durante todo treinamento. Ele utiliza um método chamado *Adam*, que tem seu nome derivado de *adaptive moment estimation*. Este método foi desenvolvido por Diederik Kingma da *OpenAI* e Jimmy Ba da *University of Toronto* em 2015(KINGMA; LEI BA, 2015).

Também devemos escolher um método para o cálculo do erro, ou seja, precisamos de uma Função de Custo. Esta vai calcular o quão longe os pesos estão do ideal, calculando assim o desempenho de cada neurônio. Um exemplo é uma função chamada *Cross Entropy*: C = (-1/n) $\sum$ (y*ln(a) + (1-y)*ln(1-a). Ela é eficiente pois quanto mais longe o valor está do ideal, mais

rápido o neurônio vai aprender(PORTILLA, 2017).

Para diminuir o custo computacional, pode-se dividir os dados do treinamento em lotes, que são chamados de *batches*. Quanto menor for o tamanho do *batch* menor será o custo computacional, porém maior será a chance deste valor não ser representativo do tamanho das amostras a serem treinadas, diminuindo a acurácia do treinamento.

4.4.2 Modelo com melhor desempenho

Foram testados vários modelos de CNNs para se obter um resultado satisfatório. Dentre os modelos consagrados, foram testados adaptações do AlexNet e do GoogLeNet, este também conhecido como *Inception V1*. Foram feitas adaptações para que essas redes pudessem ser construídas de modo sequencial na linguagem Python, utilizando o TensorFlow. A AlexNet, por exemplo, tem sua arquitetura original feita para ser treinada de modo paralelo em dois processadores(KRIZHEVSKY; SUTSKEVER; HINTON, 2012). O modelo criado com a estrutura da AlexNet pode ser observado na Figura 41. Ele apresenta um total de 28.081.754 parâmetros, isto devido ao número de camadas e filtros. O modelo que obteve os melhores resultados é o representado na Figura 35, e apresenta um total de 3.340.875 parâmetros, além da seguinte sequência de camadas: 2 convolucionais, 1 agregação, 2 convolucionais, 1 agregação, 1 convolucional, 1 agregação e 2 inteiramente conectadas (densas). O sumário da adaptação da Alexnet pode ser visto na Figura 40, e o do modelo com melhor desempenho está na Figura 42. Podemos observar que o melhor modelo apresenta uma quantidade menor de filtros, e trabalha com uma matriz de entrada de (1512,1512,3), enquanto o da AlexNet tem a entrada de (224,224,3). Isto significa que foram obtidos melhores resultados trabalhando imagens com maior resolução e utilizando-se uma menor quantidade de características destas imagens. Na próxima seção veremos os resultados.

4.5 RESULTADOS

As 931 imagens foram divididas em 10 partições, cada uma contendo aproximadamente o mesmo número de imagens de cada classe. Foram realizados 10 treinamentos com o modelo escolhido, sendo que 9 partições foram utilizadas para treinamento e 1 partição para testes, variando-se sempre a partição teste. O conjunto de dados, contendo todos os *batches*, foi trabalhado 50 vezes pela rede, este é o parâmetro chamado de *epochs*. Cada *batch* possui um total de 50 imagens (na forma de matriz numérica). A evolução de um dos treinamentos pode ser vista na Figura 43, assim como alguns parâmetros de entrada. Este foi o treinamento que obteve melhores resultados, no qual a partição 10 foi separada para testes.

Nestes testes foi obtido um erro de 0.2724 e uma acurácia de 91.26%, a Tabela 3 mostra a matriz de confusão deste treinamento. Quando analisamos cada classe, temos os seguintes valores verdadeiro positivos de 87.80%, 74.19% e 100% para as classes *sem doença*, *com doença*

Figura 40 – Sumário da adaptação da AlexNet

```
In [25]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 54, 54, 96)        34944
_________________________________________________________________
activation_1 (Activation)    (None, 54, 54, 96)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 27, 27, 96)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 27, 27, 96)        384
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 17, 17, 256)       2973952
_________________________________________________________________
activation_2 (Activation)    (None, 17, 17, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 6, 384)         885120
_________________________________________________________________
activation_3 (Activation)    (None, 6, 6, 384)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 384)         1536
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 4, 384)         1327488
_________________________________________________________________
activation_4 (Activation)    (None, 4, 4, 384)         0
_________________________________________________________________
batch_normalization_4 (Batch (None, 4, 4, 384)         1536
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 2, 256)         884992
_________________________________________________________________
activation_5 (Activation)    (None, 2, 2, 256)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
batch_normalization_5 (Batch (None, 1, 1, 256)         1024
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              1052672
_________________________________________________________________
activation_6 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
batch_normalization_6 (Batch (None, 4096)              16384
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              16781312
_________________________________________________________________
activation_7 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
batch_normalization_7 (Batch (None, 4096)              16384
_________________________________________________________________
dense_3 (Dense)              (None, 1000)              4097000
_________________________________________________________________
activation_8 (Activation)    (None, 1000)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1000)              0
_________________________________________________________________
batch_normalization_8 (Batch (None, 1000)              4000
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 2002
_________________________________________________________________
activation_9 (Activation)    (None, 2)                 0
=================================================================
Total params: 28,081,754
Trainable params: 28,060,618
Non-trainable params: 21,136
```

Fonte: Autor

e *com cerumen*, respectivamente.

Contando todos os 10 treinamentos, o erro médio foi de 0.5144 e a acurácia média

Figura 41 – Modelo da adaptação da AlexNet

```
In [15]: model = Sequential()

In [16]: model.add(Conv2D(filters=96, input_shape=(224,224,3), kernel_size=(11,11), strides=(4,4), padding='valid'))
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
         model.add(BatchNormalization())

In [17]: model.add(Conv2D(filters=256, kernel_size=(11,11), strides=(1,1), padding='valid'))
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
         model.add(BatchNormalization())

In [18]: model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid'))
         model.add(Activation('relu'))
         model.add(BatchNormalization())

In [19]: model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid'))
         model.add(Activation('relu'))
         model.add(BatchNormalization())

In [20]: model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='valid'))
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
         model.add(BatchNormalization())

In [21]: model.add(Flatten())
         model.add(Dense(4096, input_shape=(224*224*3,)))
         model.add(Activation('relu'))
         model.add(Dropout(0.4))
         model.add(BatchNormalization())

In [22]: model.add(Dense(4096))
         model.add(Activation('relu'))
         model.add(Dropout(0.4))
         model.add(BatchNormalization())

In [23]: model.add(Dense(1000))
         model.add(Activation('relu'))
         model.add(Dropout(0.4))
         model.add(BatchNormalization())

In [24]: model.add(Dense(2))
         model.add(Activation('softmax'))
```

Fonte: Autor

Tabela 3 – Matriz de confusão: partição teste de número 10

		Predito		
		sem doença	com doença	com cerumen
Verdadeiro	sem doença	36	4	1
	com doença	5	23	3
	com cerumen	0	0	31

Fonte: Autor

foi de 81.62%. A matriz de confusão gerada por todos estes treinamentos está na Tabela 4. Quando analisamos cada classe, temos os seguintes valores verdadeiro positivos de 88.60%, 54.58% e 90.00% para as classes *sem doença*, *com doença* e *com cerumen*, respectivamente. Podemos observar que a classe *com doença* obteve um número baixo de verdadeiros positivos quando analisamos todos os testes, porém este número é melhor quando vemos o teste feito com a partição 10. Isto pode ter ocorrido devido ao número baixo de imagens coletadas (133 imagens), principalmente das imagens com doença (25.56% das imagens).

Um total de 14 novas imagens foram testadas com o modelo de melhor desempenho, sendo que as classes *sem doença*, *com doença* e *com cerumen* apresentavam 5, 3 e 6 imagens respectivamente. Este teste com imagens inéditas ao programa apresentou um erro de 0.2287 e

Figura 42 – Sumário do modelo com melhor desempenho

```
In [22]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 749, 749, 8)       6152
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 369, 369, 16)      18448
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 184, 184, 16)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 89, 89, 32)        32800
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 43, 43, 48)        24624
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 21, 21, 48)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 10, 10, 64)        12352
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2024)              3240424
_________________________________________________________________
dropout_2 (Dropout)          (None, 2024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 6075
=================================================================
Total params: 3,340,875
Trainable params: 3,340,875
Non-trainable params: 0
```

Fonte: Autor

Tabela 4 – Matriz de confusão: todos treinamentos

		Predito		
		sem doença	com doença	com cerumen
Verdadeiro	sem doença	311	25	15
	com doença	42	113	52
	com cerumen	15	12	243

Fonte: Autor

uma acurácia de 90.47%. Vale ressaltar que estas imagens também foram coletadas pelo autor do trabalho, médico especialista, utilizando o mesmo método das imagens anteriores. Na Tabela 5 podemos ver a matriz de confusão gerada por este novo teste.

Figura 43 – Evolução do treinamento com melhor desempenho

```
In [23]: model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
         model.fit(x=X_train, y=Y_train, epochs=50, batch_size=50, shuffle=True)

Epoch 1/50
828/828 [==============================] - 62s 75ms/step - loss: 6.1298 - acc: 0.3406
Epoch 2/50
828/828 [==============================] - 52s 63ms/step - loss: 1.1986 - acc: 0.3804
Epoch 3/50
828/828 [==============================] - 52s 63ms/step - loss: 1.0935 - acc: 0.4118
Epoch 4/50
828/828 [==============================] - 53s 63ms/step - loss: 1.0921 - acc: 0.4287
Epoch 5/50
828/828 [==============================] - 52s 63ms/step - loss: 1.0745 - acc: 0.4384
Epoch 6/50
828/828 [==============================] - 53s 63ms/step - loss: 1.0136 - acc: 0.5314
Epoch 7/50
828/828 [==============================] - 52s 63ms/step - loss: 0.9729 - acc: 0.5314
Epoch 8/50
828/828 [==============================] - 52s 63ms/step - loss: 0.9101 - acc: 0.5954
Epoch 9/50
828/828 [==============================] - 52s 63ms/step - loss: 1.1039 - acc: 0.4167
Epoch 10/50
828/828 [==============================] - 53s 63ms/step - loss: 0.9682 - acc: 0.5254
Epoch 11/50
828/828 [==============================] - 52s 63ms/step - loss: 0.8218 - acc: 0.6618
Epoch 12/50
828/828 [==============================] - 53s 63ms/step - loss: 0.7613 - acc: 0.6812
Epoch 13/50
828/828 [==============================] - 52s 63ms/step - loss: 0.7268 - acc: 0.6957
Epoch 14/50
828/828 [==============================] - 52s 63ms/step - loss: 0.6504 - acc: 0.7271
Epoch 15/50
828/828 [==============================] - 53s 63ms/step - loss: 0.6378 - acc: 0.7186
Epoch 16/50
828/828 [==============================] - 53s 64ms/step - loss: 0.6475 - acc: 0.7355
Epoch 17/50
828/828 [==============================] - 52s 63ms/step - loss: 0.5788 - acc: 0.7512
Epoch 18/50
828/828 [==============================] - 53s 63ms/step - loss: 0.5289 - acc: 0.7802
Epoch 19/50
828/828 [==============================] - 53s 63ms/step - loss: 0.5972 - acc: 0.7391
Epoch 20/50
828/828 [==============================] - 52s 63ms/step - loss: 0.5428 - acc: 0.7754
Epoch 21/50
828/828 [==============================] - 52s 63ms/step - loss: 0.4852 - acc: 0.7995
Epoch 22/50
828/828 [==============================] - 53s 63ms/step - loss: 0.4809 - acc: 0.8019
Epoch 23/50
828/828 [==============================] - 52s 63ms/step - loss: 0.4696 - acc: 0.8056
Epoch 24/50
828/828 [==============================] - 53s 63ms/step - loss: 0.5123 - acc: 0.7729
Epoch 25/50
828/828 [==============================] - 52s 63ms/step - loss: 0.5162 - acc: 0.7790
Epoch 26/50
828/828 [==============================] - 52s 63ms/step - loss: 0.4439 - acc: 0.8213
Epoch 27/50
828/828 [==============================] - 53s 63ms/step - loss: 0.4425 - acc: 0.8285
Epoch 28/50
828/828 [==============================] - 53s 63ms/step - loss: 0.4736 - acc: 0.7947
Epoch 29/50
828/828 [==============================] - 53s 63ms/step - loss: 0.3803 - acc: 0.8406
Epoch 30/50
828/828 [==============================] - 53s 64ms/step - loss: 0.3948 - acc: 0.8551
Epoch 31/50
828/828 [==============================] - 53s 63ms/step - loss: 0.3848 - acc: 0.8551
Epoch 32/50
828/828 [==============================] - 52s 63ms/step - loss: 0.3227 - acc: 0.8889
Epoch 33/50
828/828 [==============================] - 52s 63ms/step - loss: 0.3708 - acc: 0.8454
Epoch 34/50
828/828 [==============================] - 53s 64ms/step - loss: 0.3383 - acc: 0.8563
Epoch 35/50
828/828 [==============================] - 53s 63ms/step - loss: 0.3366 - acc: 0.8756
Epoch 36/50
828/828 [==============================] - 53s 64ms/step - loss: 0.3342 - acc: 0.8720
Epoch 37/50
828/828 [==============================] - 52s 63ms/step - loss: 0.3293 - acc: 0.8671
Epoch 38/50
828/828 [==============================] - 52s 63ms/step - loss: 0.3573 - acc: 0.8611
Epoch 39/50
828/828 [==============================] - 52s 63ms/step - loss: 0.3039 - acc: 0.8889
Epoch 40/50
828/828 [==============================] - 53s 64ms/step - loss: 0.3322 - acc: 0.8708
Epoch 41/50
828/828 [==============================] - 53s 63ms/step - loss: 0.3302 - acc: 0.8708
Epoch 42/50
828/828 [==============================] - 53s 63ms/step - loss: 0.2866 - acc: 0.8961
Epoch 43/50
828/828 [==============================] - 53s 63ms/step - loss: 0.2973 - acc: 0.8756
Epoch 44/50
828/828 [==============================] - 52s 63ms/step - loss: 0.2934 - acc: 0.8768
Epoch 45/50
828/828 [==============================] - 53s 64ms/step - loss: 0.2554 - acc: 0.9106
Epoch 46/50
828/828 [==============================] - 52s 63ms/step - loss: 0.2822 - acc: 0.8986
Epoch 47/50
828/828 [==============================] - 53s 64ms/step - loss: 0.2372 - acc: 0.9130
Epoch 48/50
828/828 [==============================] - 53s 64ms/step - loss: 0.2378 - acc: 0.9022
Epoch 49/50
828/828 [==============================] - 53s 64ms/step - loss: 0.2083 - acc: 0.9191
Epoch 50/50
828/828 [==============================] - 52s 63ms/step - loss: 0.1743 - acc: 0.9372
```

Fonte: Autor

Tabela 5 – Matriz de confusão: imagens inéditas

		Predito		
		sem doença	com doença	com cerumen
Verdadeiro	sem doença	4	1	0
	com doença	0	2	1
	com cerumen	0	0	6

Fonte: Autor

5 CONCLUSÃO

Neste capítulo serão apresentados o resumo sobre o trabalho, a discussão dos resultados e nossa intenção para trabalhos futuros.

5.1 SÍNTESE DO TRABALHO

No mundo, acredita-se que aproximadamente 740 milhões de pessoas por ano serão afetadas por otite média aguda ou otite média crônica supurada. Estas doenças podem ser detectadas através de um exame de otoscopia, que tem o objetivo de analisar a aparência da membrana timpânica. A imagem da MT pode ser capturada por um otoscópio adaptado à câmera de um *smartphone*, como foi realizado no presente trabalho. Para que um computador possa classificar esta imagem, ele precisa passar por um processo de aprendizado de máquina. Para este fim, foi utilizado uma arquitetura de Rede Neural de Aprendizado Profundo chamada de Rede Neural Convolucional. Esta rede foi escolhida devido aos resultados superiores quando comparado com outras arquiteturas, principalmente nos estudos realizados na área médica(LITJENS et al., 2017). Ela apresenta uma arquitetura formada por três tipos principais de camadas: convolucional, agregação e camada inteiramente conectada. A camada de entrada da rede deve receber dados numéricos na forma de matriz. Uma imagem colorida no sistema sRGB deve ser transformada em uma matriz tridimensional, cujos números representam a quantidade de cada cor para cada pixel. A camada convolucional possui filtros (*kernel*), que funcionam como os pesos de uma rede neural, e vão ser realizadas operações entre matrizes para retirar características das imagens. A camada de agregação também realiza operações entre matrizes para diminuir o tamanho destas, podendo por exemplo calcular o número máximo de uma matriz. Isto tem o objetivo de reduzir o custo computacional. A camada final apresenta uma rede de neurônios totalmente conectada, ela é responsável pela classificação dos dados.

No presente estudo foram utilizadas 133 imagens do conduto auditivo humano para o processo de treinamento e teste, e mais 14 imagens foram separadas para testar a rede neural já treinada. As 133 imagens iniciais foram classificadas por médico especialista (o autor) em *sem doença* (56 imagens), *com doença* (34 imagens) e *com cerumen* (43 imagens). As 14 imagens separadas também foram classificadas em *sem doença* (5 imagens), *com doença* (3 imagens) e *com cerumen* (6 imagens). Mais imagens foram coletadas durante o trabalho, porém foram descartadas pelo autor devido à dificuldade de classificar a membrana timpânica. Isto ocorreu por causa de artefatos como falta de luminosidade e foco inadequado.

O trabalho com as imagens e treinamento da CNN foi realizado por uma máquina virtual do Google Cloud Plataform, que contém uma GPU NVIDIA Tesla P100. A programação foi feita em Python, com o auxílio de ferramentas como TensorFlow, Keras, Pil e NumPy. O número de imagens foi ampliado de 133 para 931 (multiplicado por 7) para ajudar a diminuir o efeito de *overfitting*. Isto foi realizado através dos processos de rotação e espelhamento destas.

Elas foram divididas em 10 partições, com o objetivo de serem realizados 10 treinamentos com os dados de 9 partições e testes com a partição restante, conforme o processo de Cross-Validation. Após os treinamentos, o melhor modelo foi escolhido, assim como os melhores pesos, e as 14 imagens separadas foram testadas.

5.2 RESULTADOS

O modelo de CNN com melhores resultados apresenta a seguinte sequência de camadas: 2 convolucionais, 1 agregação, 2 convolucionais, 1 agregação, 1 convolucional, 1 agregação e 2 inteiramente conectadas (densas). Ele apresenta uma quantidade de filtros por camada significantemente menor que modelos consagrados, como a AlexNet. Quando foram testados modelos com grandes quantidades de parâmetros e filtros, ocorreu o fenômeno de *overfitting*. Isto pode ser explicado pelo baixo número de imagens coletadas e também pelas características das imagens. Na clínica médica, os contornos ou traços da MT não são tão importantes para o diagnóstico de otite média. São mais importantes características como coloração, brilho e presença de secreção geralmente amarelada. Portanto, pode-se perceber uma diferença na classificação destas imagens em relação à classificação de imagens como carros, aviões, navios e até mesmo no reconhecimento facial, onde os contornos têm mais importância.

O melhor treinamento apresentou um erro de 0.2724 e uma acurácia de 91.26%. Observamos na matriz de confusão gerada os seguintes valores verdadeiro positivos de 87.80%, 74.19% e 100% para as classes *sem doença*, *com doença* e *com cerumen*, respectivamente. Contando todos os 10 treinamentos, o erro médio foi de 0.5144 e a acurácia média foi de 81.62%. Matriz de confusão gerada tem os seguintes valores verdadeiro positivos de 88.60%, 54.58% e 90.00% para as classes *sem doença*, *com doença* e *com cerumen*, respectivamente. Podemos observar que a maior dificuldade está na classificação das imagens "com doença", cujos falso negativos se distribuem entre as outras duas classes. Além disso, houve uma variação significativa da acurácia em cada treinamento, evidenciando a importância das imagens em cada partição. Isto pode ter ocorrido devido ao baixo número de imagens coletadas. Também é importante ressaltar que estas imagens foram coletadas por um médico especialista, com treinamento em realizar otoscopias. E mesmo assim uma quantidade de imagens foi descartada devido baixa nitidez da membrana timpânica. Após o término do treinamento, o melhor modelo foi testado com 14 imagens que foram previamente separadas para este fim. Este teste apresentou um erro de 0.2287 e uma acurácia de 90.47%.

Lundberg et al realizaram em 2017 um trabalho para avaliar a eficiência da vídeo-otoscopia no diagnóstico de doenças da orelha(LUNDBERG et al., 2017). Neste processo, os autores utilizaram como referência para o diagnóstico o exame feito por um otologista (médico especialista em orelhas) com mais de 35 anos de experiência e utilizando um otomicroscópio cirúrgico. Depois os mesmos pacientes (280 orelhas) foram examinados por médicos generalistas. Estes médicos obtiveram uma taxa de acerto de 89% utilizando um otoscópio comum

e uma taxa máxima de acerto de 94% com vídeo-otoscopia. Quando analisamos uma média de acurácia de 81.62%no treinamento das 10 partições (Cross-validation) vemos uma porcentagem inferior ao dos médicos generalistas no estudo citado. Porém, um dos treinamentos da CNN obteve uma acurácia de 91.26% nas imagens teste. Depois este modelo obteve 90.47% de acurácia no teste de 14 imagens inéditas. Mesmo sendo uma pequena quantidade de testes, os valores estão um pouco acima dos médicos generalistas em questão. Testes com mais imagens devem ser realizados. Um estudo em 2016(FORNACIALI; CARVALHO; BITTENCOURT, 2016) utilizou o método de *Bag of Words* e um modelo de rede neural para classificar imagens de lesões de pele e diagnosticar um tumor chamado melanoma. Na primeira técnica eles obtiveram uma acurácia de 84.6%, na segunda a acurácia foi de 89.3%. Em 2013, Kuruvilla et al(KURUVILLA; SHAIKH; KOVAČEVIĆ, 2013) fizeram um trabalho utilizando algumas técnicas para melhorar as imagens de otoscopias e poder classificar de modo automático dois tipos de otites: otite média aguda e otite média com efusão. Eles utilizaram um algoritmo de vocabulário que trabalha com características da MT, como luz, opacidade, concavidade, posição do martelo, etc. Obtiveram uma acurácia de 89.9%. Kuruvilla et al encontraram problemas relacionados com a obtenção das imagens, eles utilizaram técnicas com imagens para eliminar obstáculos do conduto auditivo, problemas de luminosidade e problemas para identificar a MT. Nosso estudo apenas eliminou as imagens que não apresentavam nitidez da MT.

5.3 TRABALHOS FUTUROS

A expectativa deste trabalho seria possibilitar a criação de um dispositivo que possa identificar otites em seres humanos e que possa ser utilizado por indivíduos não médicos. Para que isso possa ser desenvolvido, é importante ressaltar algumas dificuldades no trabalho atual. Primeiramente, as imagens foram coletadas por um médico especialista em otorrinolaringologia, portanto são imagens com um nível de qualidade provavelmente superior às que seriam obtidas por um dispositivo nas mãos de um indivíduo não médico. Além disso, os testes de treinamento com as redes neurais mostraram uma dificuldade inicial devido à baixa nitidez de algumas imagens. Estas apresentaram artefatos como diminuição da luminosidade e foco da região da membrana timpânica inadequado. Talvez este treinamento possa ser mais eficiente mesmo com qualquer imagem coletada, até mesmo por pessoas não profissionais, se forem utilizadas um número muito maior de imagens (500 imagens por classe, por exemplo). Portanto, o autor pretende continuar coletando imagens para aperfeiçoar o treinamento.

Outra possibilidade é que a coleta de imagens possa ser feita por não profissionais. Para isso, talvez seja necessário desenvolver um mecanismo que identifique a membrana timpânica, ou a impossibilidade de identificá-la, no momento da obtenção da imagem. Semelhante ao que ocorre em câmeras fotográficas que identificam um sorriso. Outra estratégia seria, no processo de coleta da imagem, a obtenção de uma sequência de imagens e a escolha da melhor, de maneira automática pelo software do mecanismo de coleta.

REFERÊNCIAS

BAILEY, B.; JOHNSON, J. T. **Head and neck surgery—otolaryngology.** [S.l.]: Lippincott Williams & Wilkins, 2006. v. 2.

BARANAUSKAS, J. A. **Métodos de amostragem e avaliação de algoritmos**. Notas de aula disponível em: <http://dcm.ffclrp.usp.br/ augusto/teaching/ami/AM-I-Metodos-Amostragem.pdf>. Acesso em: 03 junho 2018.

BERRY, M. J. A.; LINOFF, G. S. **Data mining techniques**: for marketing, sales, and customer support. [S.l.]: Wiley Computer Publishing, 1997.

BLUESTONE, C. D. et al. Definitions, terminology and classification of otitis media. **Annals of Otology, Rhinology & Laryngology**, [S.l.], v. 111, p. 08 – 18, 2002.

FORNACIALI, M.; CARVALHO, M.; BITTENCOURT, F. V. **Towards automated melanoma screening**: proper computer vision & reliable results. 2016.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT Press, 2016.

HAN, J.; KAMBER, M.; PEI, J. **Data mining concepts and techniques**. [S.l.]: Morgan Kaufmann publications, 2012.

HOCHULI, A. G. **Redes neurais convolucionais**. Notas de aula disponível em: <http://www.inf.ufpr.br/aghochuli/caffe/CNN_PPT.pdf>. Acesso em: 03 junho 2018.

IBM. **Manipal hospitals adopts watson for oncology to help physicians identify options for individualized, evidence-based cancer care across india**. IBM news release disponível em: <https://www-03.ibm.com/press/us/en/pressrelease/48189.wss>. Acesso em: 03 junho 2018.

JONES, M. T. **Arquiteturas de aprendizado profundo - o surgimento da inteligência artificial**. IBM developer works disponível em: <https://www.ibm.com/developerworks/br/library/cc-machine-learning-deep-learning-architectures/index.html>. Acesso em: 03 junho 2018.

KINGMA, D. P.; LEI BA, J. Adam: a method for stochastic optimization. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 3RD INTERNATIONAL CONFERENCE FOR LEARNING REPRESENTATIONS, 2015, San Diego, USA. **Anais...** [S.l.: s.n.], 2015.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in Neural Information Processing Systems**, [S.l.], p. 1097 – 1105, 2012.

KURUVILLA, A.; SHAIKH, N.; KOVAČEVIĆ, J. Automated diagnosis of otitis media: vocabulary and grammar. **International Journal of Biomedical Imaging**, [S.l.], p. 2845 – 2848, 2013.

LITJENS, G. et al. A survey on deep learning in medical image analysis. **Medical Image Analysis**, [S.l.], v. 42, p. 60 – 88, 2017.

LUGER, G. F. **Inteligência artificial**. [S.l.]: Pearson Education do Brasil, 2013.

LUNDBERG, T. et al. Diagnostic accuracy of a general practitioner with video-otoscopycollected by a health care facilitator compared to traditional otoscopy. **International Journal of Pediatric Otorhinolaryngology**, [S.l.], v. 99, p. 49 – 53, 2017.

MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-Hill, 1997.

PIRES, R. et al. Automated multi-lesion detection for referable diabetic retinopathy in indigenous health care. **PLOS ONE**, [S.l.], v. 10, n. 6, p. 1–12, 06 2015.

PORTILLA, J. **Complete guide to tensorflow for deep learning with python**. Notas de aula disponível em: <https://www.udemy.com/complete-guide-to-tensorflow-for-deep-learning-with-python/>. Acesso em: 12 novembro 2018.

PYAKILLYA, B.; KAZACHENKO, N.; MIKHAILOVSKY, N. Deep learning for ecg classification. **Journal of Physics: Conference Series**, [S.l.], v. 913, 2017.

STEINER, M. T. A. et al. Abordagem de um problema médico por meio do processo de kdd com ênfase à análise exploratória dos dados. **Gestão & Produção**, [S.l.], v. 13, p. 325 – 337, 05 2006.

TEELE, D. W.; KLEIN, J. O.; ROSNER, B. A. Epidemiology of otitis media in children. **The Annals of otology, rhinology & laryngology. Supplement.**, [S.l.], v. 89, p. 5–6, 1980.

TEELE, D. W.; KLEIN, J. O.; ROSNER, B. A. From data mining to knowledge discovery in databases. **AI Magazine**, [S.l.], v. 17, n. 3, 1996.

ZHOU, S. K.; GREENSPAN, H.; SHEN, D. **Deep learning for medical image analysis**. [S.l.]: Joe Hayton, 2017.