Matheus Leor Lopes de Lima Marrocos

Detecção automática de idade óssea através da radiografia de mão e punho utilizando Redes Neurais Convolucionais

Campina Grande, Brasil

15 de julho de 2019

Matheus Leor Lopes de Lima Marrocos

Detecção automática de idade óssea através da radiografia de mão e punho utilizando Redes Neurais Convolucionais

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Orientador: Edmar Candeia Gurjão, D.Sc.

Campina Grande, Brasil

15 de julho de 2019

Matheus Leor Lopes de Lima Marrocos

Detecção automática de idade óssea através da radiografia de mão e punho utilizando Redes Neurais Convolucionais

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Trabalho aprovado em: /07/2019

Edmar Candeia Gurjão, D.Sc.
Orientador

Luciana Ribeiro Veloso, D.Sc.
Convidado

Campina Grande, Brasil
15 de julho de 2019

Dedico este trabalho às minhas mães, Cleomar Marrocos e Germana Lopes, ao meu pai, Manuel Marrocos e à minha família.

Agradecimentos

Agradeço aos meus pais, pelo apoio e confiança em todos os momentos. Por me orientarem, mas sempre me deixando trilhar meu próprio caminho da maneira que me deixasse mais feliz.

Às minhas irmãs, Lígia, Aretuza e Ligiare, que sempre serviram de inspirações em ramos distintos para mim.

Aos meus amigos, Humberto, Flávio, Thiago, Michael, Niago, Eduardo, Stanley, Isaac, Yuri, Iara, Bruno, Patrícia, Emilly, Yago, e tantos outros que sempre estiveram comigo e tornaram essa jornada mais divertida.

Aos professores Gutemberg, Wamberto e Marcos, pelo apoio técnico e pessoal de sempre. Ao professor George e o pessoal do LIEC, pelas experiências e aprendizados. E em especial ao professor Edmar, por tentar inspirar os alunos nas aulas e por ter me orientado neste trabalho.

"In any moment of decision, the best thing you can do is the right thing. The worst thing you can do is nothing."

Theodore Roosevelt

Resumo

Este trabalho apresenta um resumo sobre os conceitos essenciais relacionados a inteligência artificial, considerando desde aspectos históricos, a explicações teóricas, concluindo com aplicação desses conceitos, aliado a outros aspectos de análise de dados, para a determinação automática da idade óssea de pacientes utilizando uma radiografia de mão e punho. Alguns procedimentos e ferramentas recorrentes dessa área, como o framework Keras e bibliotecas como Pandas, Matplotlib e Seaborn também serão mencionadas.

Palavras-chaves: Redes Profundas; Redes Neurais Convolucionais; Avaliação de Idade Ósses; Keras.

Abstract

This work presents a review about fundamental concepts related to artificial intelligence, beginning with a historical perspective, going through high level theoretical explanation, and ending with applications of those utilizing Keras framework, along with some other tools such as Pandas, Matplotlib and Seaborn.

Key-words: Deep Learning; Convolutional Neural Network; Bone Age Assessment; Keras.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CNN	*Convolutional Neural Network* (Rede Neural Convolucional)
GAN	*Generative Adversarial Networks*
IA	Inteligência Artificial
IO	Idade Óssea
NN	*Neural Network* (Rede Neural)
RSNA	Sociedade de Radiologia da América do Norte

Sumário

1 Introdução

Com o avanço da tecnologia de aquisição de imagem e diagnóstico, há uma diminuição no custo dos exames de radiografia, o que vem gerando um aumento do número de pacientes utilizando este tipo de serviço, tanto em instituições públicas, como privadas. Nesse meio, há um exame comum para pacientes em estágio de crescimento, que as vezes é demandado por questões prioritariamente médicas, outras vezes por pressão social ou parental, que é a radiografia de mão e punho. Essa é capaz de fornecer informações sobre a idade óssea do paciente, que correlacionada com sua idade cronológica pode facilitar o diagnóstico de diversas condições, como puberdade precoce, problemas de crescimento, etc (GILSANZ; RATIB, 2005). Sendo assim, nota-se que esse é um assunto relevante e, por conta da demanda médica por uma ferramenta mais precisa e que realize a análise dessas imagens com mais qualidade e rapidez, substituindo a imprecisão e atraso de um diagnóstico humano, faz-se necessário, mais uma vez, o uso da tecnologia para facilitar a vida humana, sendo, nesse caso, a tecnologia de processamento e classificação de imagens.

Esse problema pode ser melhor resolvido quando se utiliza o processamento de imagens e a aplicação de inteligência artificial na tomada de decisões. Sendo assim, inicialmente, deve-se coletar imagens de radiografias, então, essas são processadas e classificadas, de acordo com o banco de dados disponível e as referências médicas a respeito do tema, como o Atlas de Greulich e Pyle (GREULICH; PYLE, 1959). Por fim, realiza-se o treinamento de um sistema para que esse possa tomar decisões semelhantes às de um médico especializado.

Após o desenvolvimento do sistema, esse pode ser utilizado além dos ambientes puramente teóricos e acadêmicos. Por exemplo, pode ser implementado em um aplicativo (ou um site) para auxiliar os médicos no dia a dia com a identificação precisa da idade óssea, o que diminuiria as discrepâncias entre avaliações de médicos distintos (BULL et al., 1999).

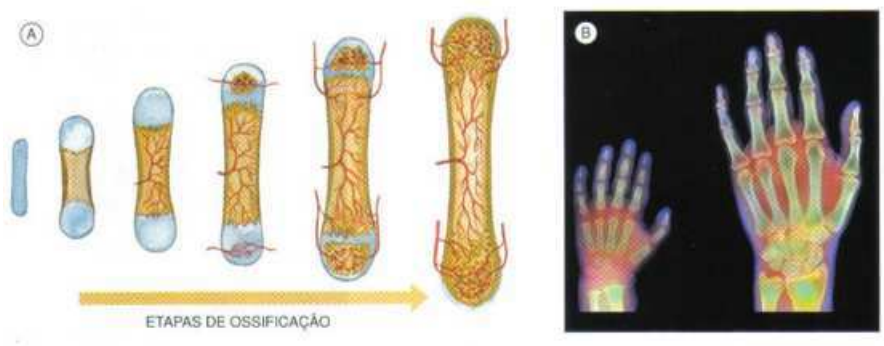
Com isso em mente, este trabalho tem dois grandes objetivos, que são, implementar um sistema capaz de identificar a idade óssea de um paciente utilizando uma radiografia de mão e punho e servir como material introdutório para fornecer uma base geral em termos de redes neurais e suas aplicações.

2 Considerações sobre radiografias de mão e punho e cálculo da idade óssea

A idade óssea é um método que fornece informações complementares para a avaliação de distúrbios de crescimento e da puberdade. Por ser uma medida que é diretamente correlacionada a fatores genéticos, ambientais e endócrinos, tem importância e aplicações diversas, sendo relevante em ramos como a nutrição, endocrinologia, odontologia (TOKUNAGA, 2013). Sendo assim, é relevante que se entenda como essa técnica de diagnóstico funciona e que essa seja otimizada com a utilização de tecnologia, visando obter diagnósticos com precisões mais próximas de 100%.

A idade óssea é calculada pela determinação dos centros de ossificação de um paciente em relação aos padrões cronológicos de crianças ditas normais (LONGUI, 1996). Essa informação pode ser obtida desde muito cedo, levando em conta que o desenvolvimento ósseo é percebido desde estágios iniciais da vida humana, onde ocorre a ossificação endocondral (dentro da cartilagem), em que há substituição do tecido cartilaginoso por tecido ósseo ao longo do tempo. Um bebê já possui um esqueleto bastante ossificado, no entanto, a extremidade de vários dos seus ossos mantêm regiões cartilaginosas, permitindo o crescimento, que se dá principalmente durante a puberdade (AMABIS; RODRIGUES, 2016). Na Figura 1 é vista uma ilustração do processo de ossificação de maneira geral e também é mostrada uma comparação entre uma mão de uma criança e a de um adulto, onde se observa que nesta segunda os ossos estão juntos, o que indica processo de ossificação (substituição da cartilagem) avançado.

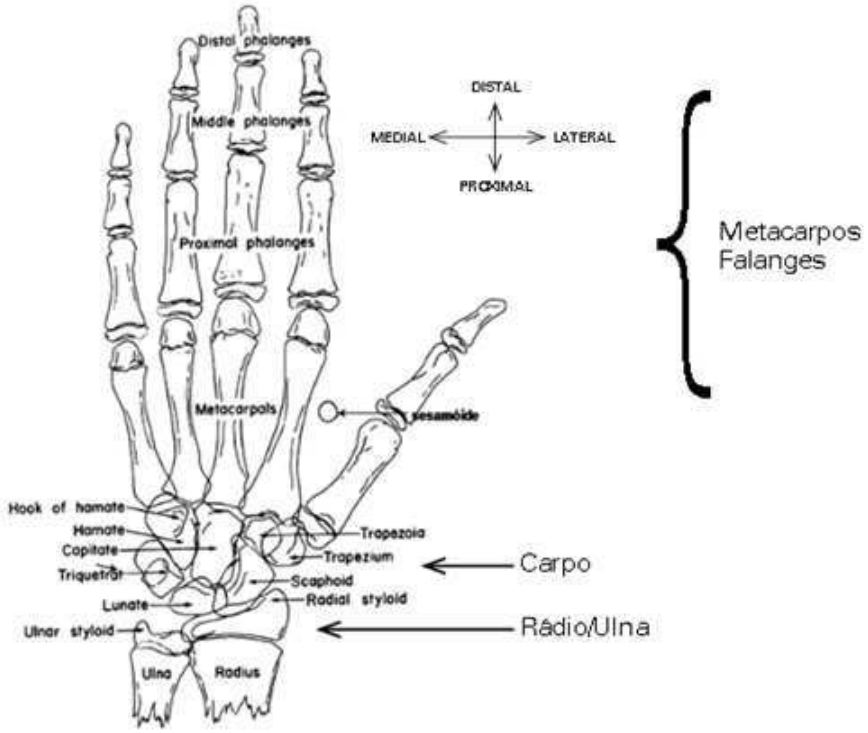
Figura 1 – Etapas de ossificação e comparação de radiografias da mão de um adulto e de uma criança



Fonte: (AMABIS; RODRIGUES, 2016)

Ao longo do tempo, diversos estudos foram feitos e notou-se que era possível fazer a mensuração da idade óssea através da radiografia de mão e punho (LONGUI, 1996). Por convenção, esse exame é feito na mão esquerda, que na maior parte da população é a mão menos propensa a sofrer lesões (KUPERMAN RAPHAEL LIBERATORE, 2007), levando em conta que a maioria da população é destra. Para realizar a radiografia, o paciente deve posicionar o seu dedo polegar em um ângulo de trinta graus com o dedo indicador e o aparelho de raio X deve estar posicionado na altura do III metacarpo e distante 75 cm da mão, A Figura 2 mostra as posições de ossos na mão humana, para que se tenha melhor noção do que está sendo descrito.

Figura 2 – Núcleos de ossificação dos ossos da mão esquerda



Fonte: (KUPERMAN RAPHAEL LIBERATORE, 2007)

Os métodos mais utilizados para a análise da idade óssea são o de Greulich-Pyle e o de Tanner-Whitehouse (LONGUI, 1996). Esses foram desenvolvidos com amostras de populações distintas e em épocas diferentes, mas são utilizados até hoje em regiões distintas do mundo, o que também é uma das causas claras de perca de precisão em diagnósticos.

O método de Greulich-Pyle foi baseado em um estudo feito na década de 30, em que 6879 radiografias de mãos e punho de indivíduos normais, de classe média alta, do sexo masculino ou feminino, foram avaliadas. Os indivíduos possuíam idades entre 3 meses

e 17 anos, de modo que as radiografias foram feitas de 3 em 3 meses no primeiro ano, de 6 em 6 até os cinco anos e anualmente em crianças maiores. Com esses dados, foi feita a determinação do correspondente entre a idade cronológica e a idade óssea de acordo com o sexo. Esses resultados estão apresentados na Figura 3, que é uma captura de tela do livro de Greulich e Pyle (GREULICH; PYLE, 1959).

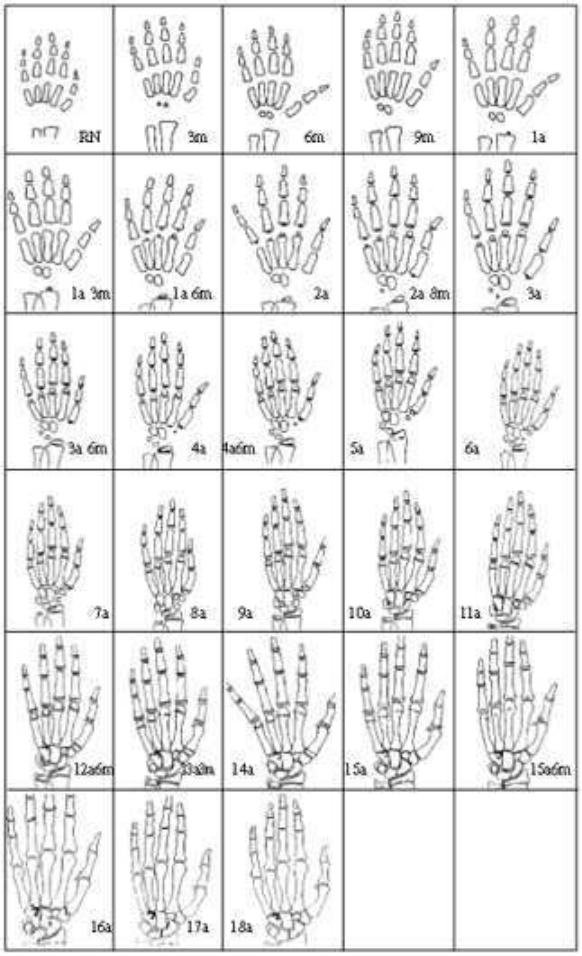
Figura 3 – Desvio-padrão da IO relacionado à idade cronológica e sexo

Idade cronológica (anos)	Desvio-padrão (meses) sexo feminino	Desvio-padrão (meses) sexo masculino
0,25	0,7	0,7
0,5	1,2	1,1
0,75	1,4	1,4
1,0	1,8	2,0
1,5	3,5	3,5
2,0	4,6	3,9
2,5	5,4	4,5
3,0	6,0	5,1
3,5	7,5	5,4
4,0	9,0	6,7
4,5	10,7	8,4
5,0	11,6	8,8
6,0	10,2	9,2
7,0	9,6	8,9
8,0	10,2	9,1
9,0	10,7	9,0
10,0	11,7	9,8
11,0	11,9	10,1
12,0	10,2	10,4
13,0	10,7	10,4
14,0	11,3	10,7
15,0	9,2	11,3
16,0	7,3	12,9
17,0	—	13,0

Fonte: (GREULICH; PYLE, 1959)

Como a idade óssea depende da forma dos núcleos de ossificação para que o diagnóstico (comparação entre radiologia obtida e valor de idade óssea correspondente) seja feito, usando esse método de observação pelo atlas, é preciso que se analise atentamente uma imagem e a compare com um conjunto de outras amostras normais. Além disso, em (GREULICH; PYLE, 1959) também é possível encontrar características relacionadas ao padrão de ossificação normal, por exemplo, com 3 meses, é esperado que haja aparecimento do captato e hamato na criança. Um exemplo das imagens de referência (apenas para o sexo masculino) apresentadas no atlas pode ser visto na Figura 4.

Figura 4 – Imagens de referência do atlas de Greulich e Pyle



Fonte: (GREULICH; PYLE, 1959)

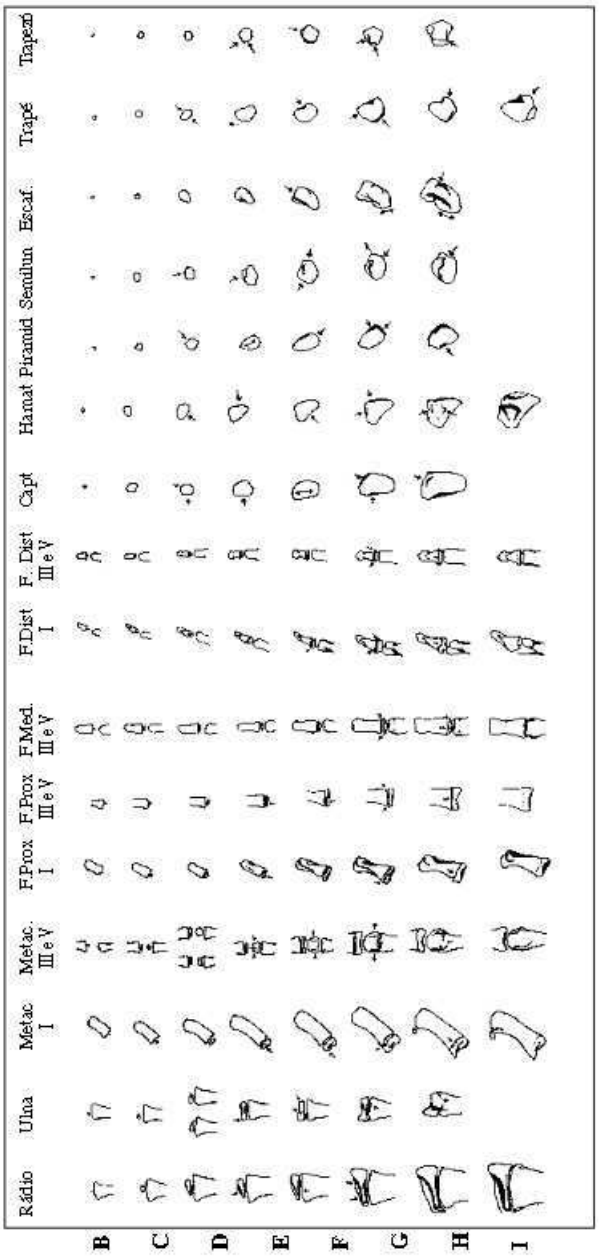
Outro método bastante utilizado é o de Tanner-Whitehouse (TANNER, 1983), que foi desenvolvido a partir de um estudo realizado na década de 50, em que 77000 radiografias de indivíduos de centros britânicos, de classe média e baixa, e do sexo masculino ou feminino foram avaliadas. Os indivíduos possuíam idades entre 6 meses e 21 anos. Diferenciando do método de Greulich-Pyle, neste, cada núcleo de ossificação é analisado individualmente e recebe uma pontuação relacionada. Ao fim, a pontuação total é comparada com pontuações esperadas. Além disso, nesse método também é feita a separação entre regiões, "TW20", que abrange 20 núcleos da mão e punho; "TW-carpo", que inclui a região carpal; TW-RUS, que compreende rádio, ulna e pequenos ossos (*short bones*). No entanto, apesar dessas regiões possuírem informações relevantes para análise de condições do paciente, se o objetivo do endocrinologista for a determinação da estatura final, deve-se sempre utilizar a região TW-RUS (LONGUI, 1996).

Levando em conta sua maior acurácia, sua característica de medir a idade óssea com precisão de décimos em décimos e sua capacidade de delimitar regiões de interesse,

com a utilização desse método pode-se realizar um acompanhamento contínuo do crescimento e pode-se fazer análises de patologias em uma abordagem mais local. Por ser mais preciso, mas mais trabalhoso, esse método já foi desenvolvido em programas de computadores e disponibilizado gratuitamente para a classe médica anteriormente (LONGUI, 1996).

De maneira resumida, há pontuações de A até I, onde A tem pontuação 0 e é o estágio em que o núcleo epifisário ainda não é visível e B até H (ou I em alguns casos) representam estágios subsequentes de maturação óssea, cada um com sua pontuação. A Figura 5 apresenta essas regiões de análise.

Figura 5 – Estágios de maturação dos núcleos epifisários (método TW2)



Fonte: (LONGUI, 1996)

Avalia-se os metacarpos, falanges proximais e distais no primeiro, terceiro e quinto dedos (I, III, V); as falanges médias no terceiro e quinto dedos (III e V) e, após a definição dos estágios de maturação (A até I), calcula-se sua pontuação correspondente total, que é a soma das pontuações individuais, e a idade óssea é obtida ao se observar um conjunto de tabelas.

Conforme mencionado anteriormente, com o conhecimento da idade óssea, pode-se fazer a previsão da altura final de um paciente, além de observar a possibilidade de

patologias correlacionadas a esse parâmetro quando em atraso ou avanço.

A idade óssea pode estar atrasada em relação ao padrão nas seguintes situações (KUPERMAN RAPHAEL LIBERATORE, 2007):

- Atraso de causa familiar
- Atraso constitucional do crescimento e da puberdade
- Hipotiroidismo
- Hipotituitarismo
- Desnutrição prolongada
- Doenças crônicas de modo geral
- Síndrome de Hurler
- Hipogonadismo
- Doença de Addison
- Uso crônico de corticoide exógeno ou hiperprodução endógena (síndrome de Cushing)

Já para o caso em que a idade óssea está em avanço, em geral se tem:

- Avanço de causa familiar
- Puberdade precoce central – idiopática, tumores hipotalâmicos/hiposifários
- Puberdade precoce periférica – carcinomas virilizantes da supra-renal, tumores de ovários e testiculares
- Síndrome adrenogenital (Hiperplasia adrenal congênita)
- Obesidade simples (exógena) associada à estatura elevada
- Adrenarca precoce
- Síndrome de McCune-Albright
- Hipertireoidismo

Prosseguindo, para fazer a determinação da idade cronológica, pode-se usar métodos baseados nas informações de Greulich-Pyle, Tanner-Whitehouse, ou outros. Um deles é o método de Bayley-Pinneau (LONGUI, 1996), que usa a idade óssea calculada através do método de Greulich-Pyle e prevê a estatura final considerando a hipótese de que a

idade óssea atual corresponde a uma fração do crescimento completo e que a evolução deste depende do fato da idade óssea estar atrasada, acelerada ou compatível com a idade cronológica correspondente. De maneira prática, determina-se a estatura final prevista através da divisão da idade óssea atual por um fator (Retardado, Normal ou Adiantado), conforme a captura de tela mostrada na Figura 6. Uma observação relevante deste método é que ele em geral superestima o crescimento final ou o retardo, por isso, outros métodos mais complexos foram desenvolvidos e são mais utilizados (LONGUI, 1996).

Figura 6 – Recorte da tabela de previsão de estatura de Bayley-Pinneau

	Meninas			Meninos		
IO	R	N	A	R	N	A
6-0	0,733	0,720		0,680		
6-3	0,742	0,729		0,690		
6-6	0,751	0,738		0,700		
6-9	0,763	0,751		0,709		
7-0	0,770	0,757	0,722	0,718	0,695	0,670
7-3	0,779	0,765	0,732	0,728	0,702	0,676
7-6	0,788	0,772	0,742	0,738	0,709	0,683
7-9	0,797	0,782	0,750	0,747	0,716	0,689
8-0	0,804	0,790	0,760	0,756	0,723	0,696
8-3	0,813	0,801	0,771	0,765	0,731	0,703
8-6	0,823	0,810	0,784	0,773	0,739	0,709
8-9	0,836	0,821	0,790	0,779	0,746	0,715
9-0	0,841	0,827	0,800	0,786	0,752	0,720
9-3	0,851	0,836	0,809	0,794	0,761	0,728
9-6	0,858	0,844	0,819	0,800	0,769	0,734
9-9	0,866	0,853	0,828	0,807	0,777	0,741
10-0	0,874	0,862	0,841	0,812	0,784	0,747
10-3	0,884	0,874	0,856	0,816	0,791	0,753
10-6	0,896	0,884	0,870	0,819	0,795	0,758
10-9	0,907	0,896	0,883	0,821	0,800	0,763
11-0	0,918	0,906	0,887	0,823	0,804	0,767
11-3	0,922	0,910	0,891	0,827	0,812	0,776

Fonte: (LONGUI, 1996)

Por fim, nota-se que ainda há um nível de empirismo e imprecisão nesses métodos, mas que a utilidade das radiografias de mão e punho é inquestionável. O que é mais uma justificativa deste trabalho.

3 Breve história da inteligência artificial e introdução a redes neurais

Apesar de parecer mágica para os leigos, inteligência artificial, um termo que foi cunhado por John McCarthy quando este realizou um workshop em 1956 sobre o tema (MCCARTHY et al., 1955), é apenas a aplicação direta da matemática, principalmente de conceitos relacionados à estatística, em um grande conjunto de dados de entrada, para descobrir (matematicamente) as características deste, e reproduzi-las (regressão) ou identificá-las (classificação) em novos conjuntos de dados semelhantes.

Nesta seção, uma breve história de inteligência artificial será apresentada, então, conceitos fundamentais para o entendimento de partes posteriores deste trabalho serão explanados da maneira mais simplificada possível, tentando se aproximar da ideia repetida exaustivamente por Andrew Ng (um dos maiores nomes de IA atualmente) de que a inteligência artificial tem que se tornar acessível para todos. Além disso, algumas nomenclaturas frequentemente usadas na área e breves considerações matemáticas serão apresentadas.

3.1 Avanços em inteligência artificial

Apesar de só estar aparecendo constantemente na mídia nesta década, inteligência artificial é um tema que vem sendo desenvolvido pelos cientistas e entusiastas há décadas. Em 1950, com artigo intitulado "programando um computador para jogar xadrez" (SHANNON, 1950), Claude Shannon já antevia aplicações e desenvolvimentos na área, propondo pela primeira vez uma rotina de computador (programa) para jogar xadrez. O artigo apresenta a ideia de utilizar a máquina para, de acordo com o estado atual, prever situações futuras e calcular qual deveria ser o próximo movimento. No entanto, ao ler esse artigo nota-se claramente as limitações da época, em que os computadores ainda estavam começando a se desenvolver. Por exemplo, o autor fala que para um jogo normal de xadrez, em cada jogada, haveria cerca de 30 movimentos legais possíveis, assim, considerando um movimento para branco e um para preto, tem-se cerca de 1000 possibilidades. Um jogo normal dura cerca de 40 movimentos, assim, haveriam cerca de 10^{120} variações para serem calculadas em relação à posição inicial. Uma máquina da época poderia calcular cerca de uma variação por micro-segundo, o que resultaria em cerca de 10^{90} anos para calcular o primeiro movimento.

Esse artigo de Shannon é interessante pois demonstra que a noção de máquinas fazendo predições, ganhando jogos, ou "pensando", não é nova, mas que a limitação, nesse

caso, de poder de processamento, sempre foi um problema nessa área.

Prosseguindo, também em 1950, Alan Turing, propôs no seu artigo intitulado "Computing Machinery and Intelligence" (MACHINERY, 1950), a ideia do "Jogo da Imitação", em que questionava se uma máquina poderia pensar. Nesse jogo, haveria, inicialmente, um homem, uma mulher e um interrogador. Aqueles responderiam questões em um papel, de modo que este tentaria descobrir de fato quem era o homem e quem era a mulher, sabendo que o homem estaria tentando imitar como uma mulher responderia. Então, Turing propós fazer um jogo semelhante, mas, utilizando a comparação entre um humano e uma máquina, para ver se esta seria capaz de enganar alguém.

Ainda na década de 50, Arthur Samuel, um cientista de computação que trabalhava na IBM, desenvolveu um programa, Game of Checkers, que era capaz de jogar xadrez e aprender. Já nessa época, o termo "*Machine Learning*" era usado(SAMUEL, 1960). Também nessa década, outros termos já eram populares, como se pode ver na proposta de projeto de pesquisa de 1955 escrita por John McCarthy, Marvin L Minsky, Nathaniel Rochester e Claude Shannon, em que já se falava de redes neurais e melhoria-própria (self-improvement) em máquinas, nesta época, o nome inteligência artificial foi cunhado por McCarthy, que é, além de tudo, autor da LISP, uma linguagem de programação que foi muito importante na história de inteligência artificial.

Nos anos 60, foram desenvolvidos diversos programas e algoritmos para resolver problemas matemáticos, como o SAINT (Symbolic Automatic INTegrator) de James Slagle, que resolvia problemas de integração simbólica. Além disso, a inteligência das máquinas já começava a ser usada em robôs, como o Shakey Robô, desenvolvido pelo time de Charles Rosen, que foi o primeiro robô móvel de propósito geral e o Unimate, de George Devol, que foi o primeiro robô industrial a trabalhar em uma linha de produção (REYNOSO, 2019).

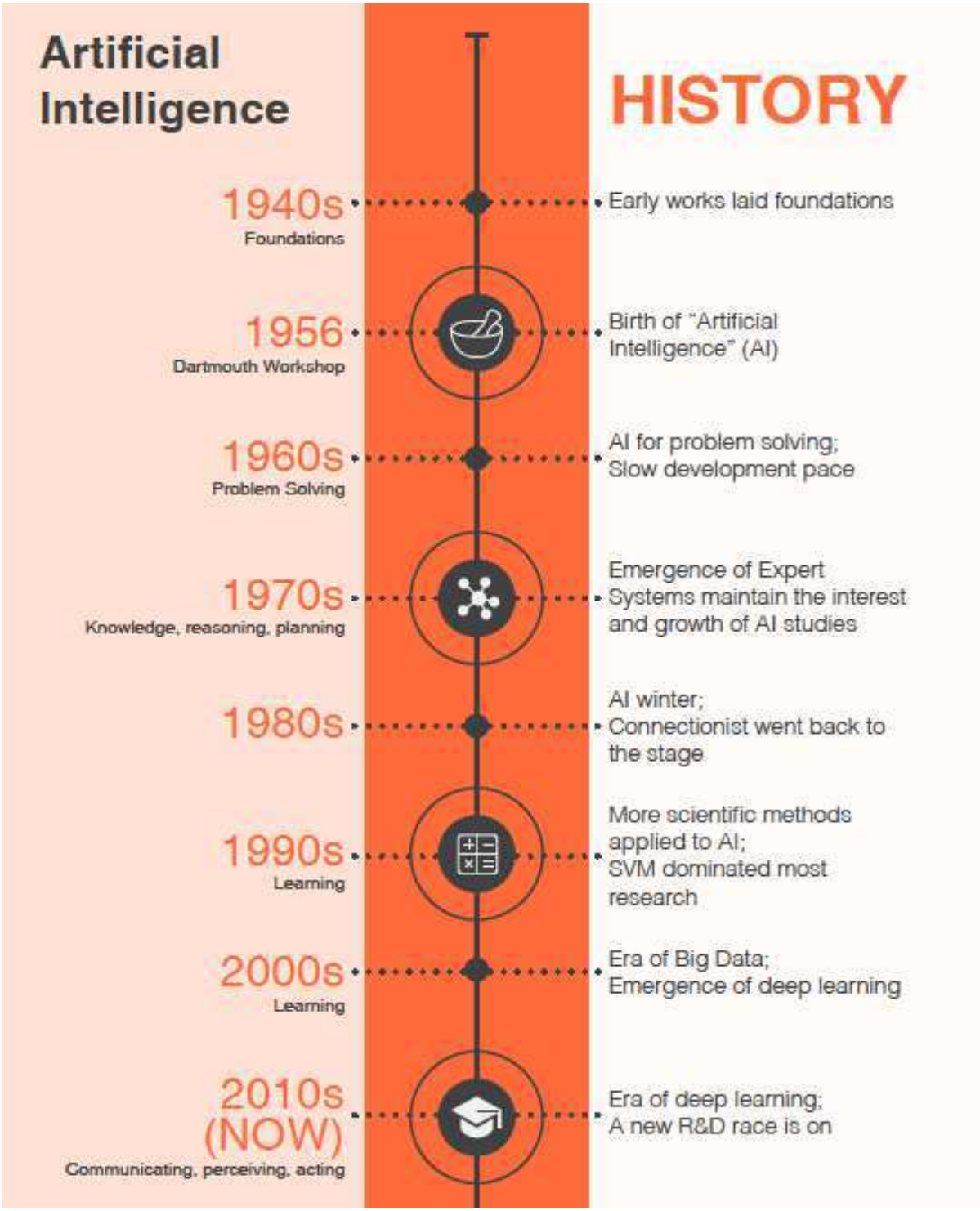
Os avanços em inteligência artificial continuaram, mas houveram dificuldades e diminuição de financiamento, pelo fato de que as aplicações de inteligência artificial precisavam de muitos dados e muito poder computacional para época, fazendo com que os cientistas e desenvolvedores não fossem capazes de cumprir promessas feitas anteriormente. Isso é visto claramente no relatório de James Lighthill (LIGHTHILL, 1973), que reportou o estado da pesquisa e resultados dessa área para o Conselho Científico Britânico. Por causa disso, entre o final da década de 70 e parte da década de 1990, a pesquisa em inteligência artificial viveu o que se chama de "*AI Winter*", relacionado ao congelamento dos financiamentos e diminuição da excitação pela área pelos governantes e outras instituições.

Os desenvolvimentos na área tiveram um grande crescimento a partir dos anos 90, levando em conta que os computadores estavam muito mais potentes e que a evolução dos algoritmos continuava. Uma prova disso é o Deep Blue, computador desenvolvido pela

IBM, que foi o primeiro sistema a vencer um jogo de xadrez contra um campeão mundial (HSU, 1999).

Então, houve continuação dos avanços de pesquisas, tanto em meios acadêmicos, como industriais, e hoje, com a qualidade dos computadores pessoais e crescimento da comunidade open-source, a indústria de inteligência artificial cresce ainda mais rápido. Por fim, a história de IA continua de uma maneira promissora, levando em conta que hoje a maioria das grandes empresas investe nisso, por exemplo, segundo o relatório da Fortune Eight (VANIAN, 2019), com participantes de 300 empresas, mais de 50% dos participantes da pesquisa pretendem realizar um investimento mínimo de 51 mil dólares em AI, enquanto que 13% disseram que têm dinheiro para despesas com AI entre 251 e 500 mil e 5% disseram que estão gastando mais de 5 milhões. Além disso, também há números expressivos em termos de investimentos em startups, por exemplo, o grupo SenseTime da China, que trabalha com reconhecimento de imagem e outros serviços, conseguiu arrecadar mais de 1 bilhão de dólares no ano passado. A Figura 7 traz um resumo de alguns dos principais fatos na história da IA.

Figura 7 – Uma breve revisão da história da inteligência artificial



Fonte: (SYNCED, 2018)

3.2 O que é aprendizado de máquina, inteligência artificial e redes neurais profundas

Inteligência artificial é focada em criar uma máquina que pensa/age como um humano em um conjunto pequeno de tarefas específicas (*Narrow AI*) ou em diversas situações possíveis (*General AI*). Um exemplo de *Narrow AI* é uma máquina que pode jogar bem um jogo, como o Go, o que é representado na Figura 8 e, um exemplo de *General AI* é o que os desenvolvedores da Hanson Robotics estão tentando atingir com a robô Sophia, mostrada na Figura 9.

Figura 8 – Google AlphaGo contra o campeão mundial de GO, Lee SeDOL

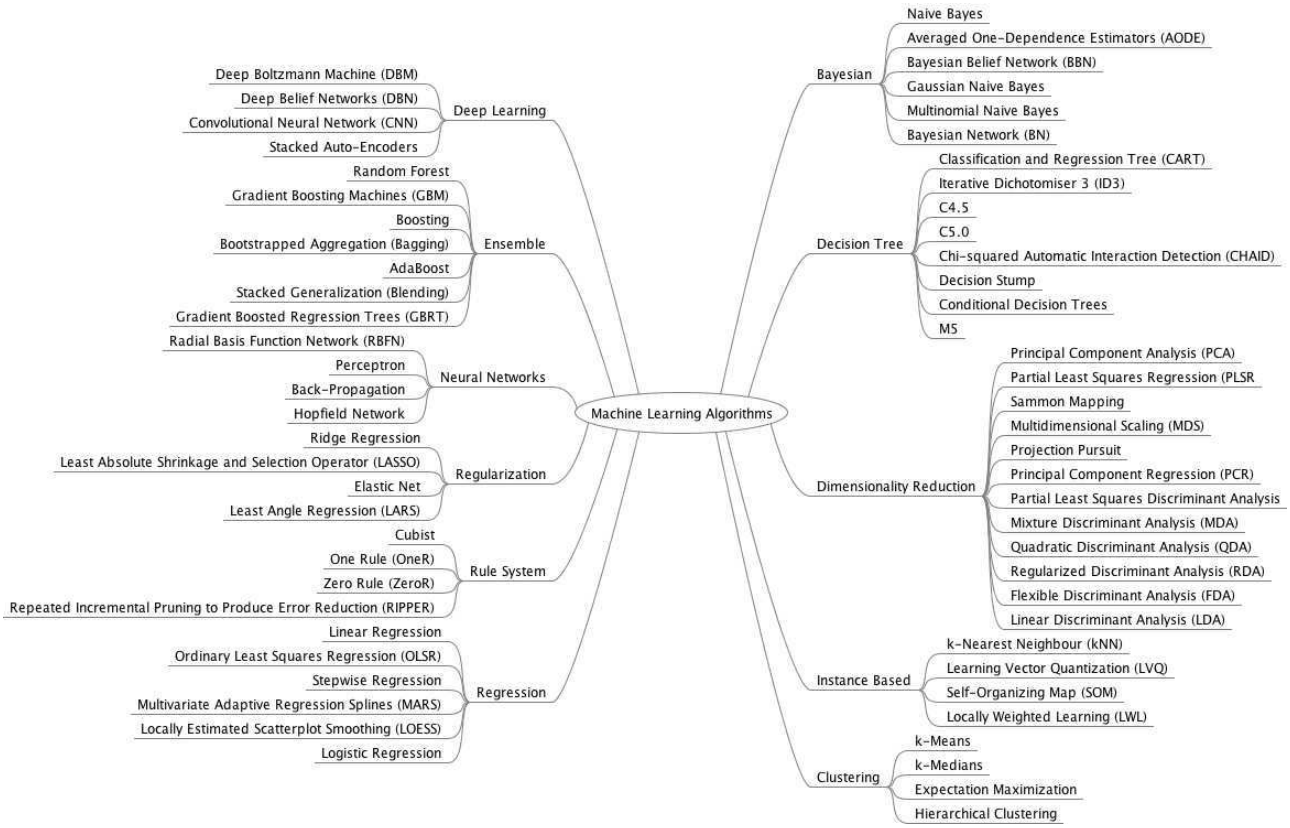
Fonte: Google Deep Mind

Figura 9 – Robô Sophia

Fonte: Hanson Robotics

Já *Machine Learning*, é um subramo de AI que consiste em aplicar ideias para que se tenha máquinas aprendendo por conta própria através de dados e experiências. Segundo Tom M. Mitchel, "diz-se que um computador aprende por experiência, 'E', em respeito a uma classe de tarefa, 'T', e medida de desempenho, 'P', se o desempenho em T, medido por P, aumenta com a experiência (E)". Alguns subconjuntos populares de *machine learning* são vistos na Figura 10.

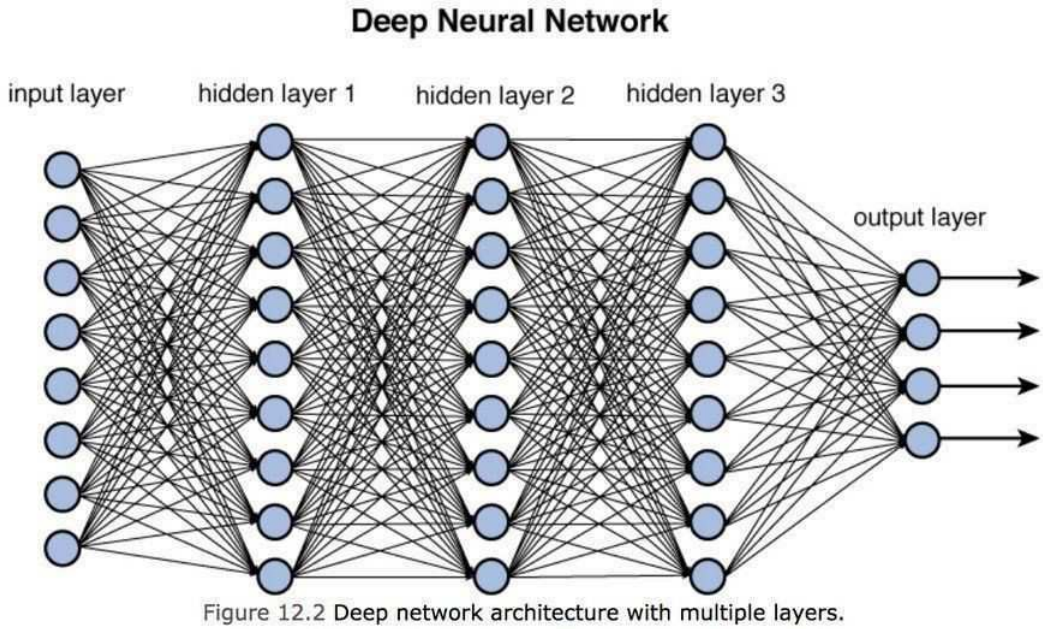
Figura 10 – Mapa de algoritmos de aprendizado de máquina



Fonte: (BROWNLEE, 2013)

Por fim, *Deep Learning* é uma técnica para implementação de *Machine Learning* que surgiu como um sub-ramo de redes neurais artificiais, que são estruturas de aprendizado que visam simular a atuação de neurônios humanos, e são formadas por grandes quantidade de camadas, com neurônios que irão aprender parâmetros e ativar ou não de acordo com certas características (será explicado em mais detalhes adiante). A Figura 11 ilustra como é a estrutura de uma *Deep Neural Network* (Rede Neural Artificial Profunda).

Figura 11 – Arquitetura geral de uma rede neural profunda



Fonte: (MAHAPATRA, 2018)

3.3 O desenvolvimento de redes neurais profundas

O termo *Deep Learning* surgiu em 1986, sendo proposto por Rina Dechter em 1986 (DECHTER, 1986), mas, antes disso, alguns desenvolvimentos importantes para essa arquitetura já haviam sido feitos. Por exemplo, em 1958 (ROSENBLATT, 1957) Frank Rosenblatt inventou o algoritmo do perceptron, visando criar uma máquina em um hardware específico ("Mark 1 Perceptron") para reconhecimento de imagem. Esta, funcionava com 400 fotosensores que eram conectados aleatoriamente a "neurônios" e tinham seus pesos estabelecidos em potenciômetros, sendo ajustados durante o treinamento por motores elétricos. Antes disso, o termo "redes neurais" já tinha começado a ficar popular com o artigo de McCulloch e Pitt (MCCULLOCH; PITTS, 1943), em que já se tinha ideia de que a mente humana (analisando neurônios) funcionava a partir de uma "lógica proporcional", onde pesos de importância eram estabelecidos nos dados de entrada e a partir disso, se essa entrada com seu peso fosse suficiente para ultrapassar um limiar, o neurônio possuiria certo comportamento ativo.

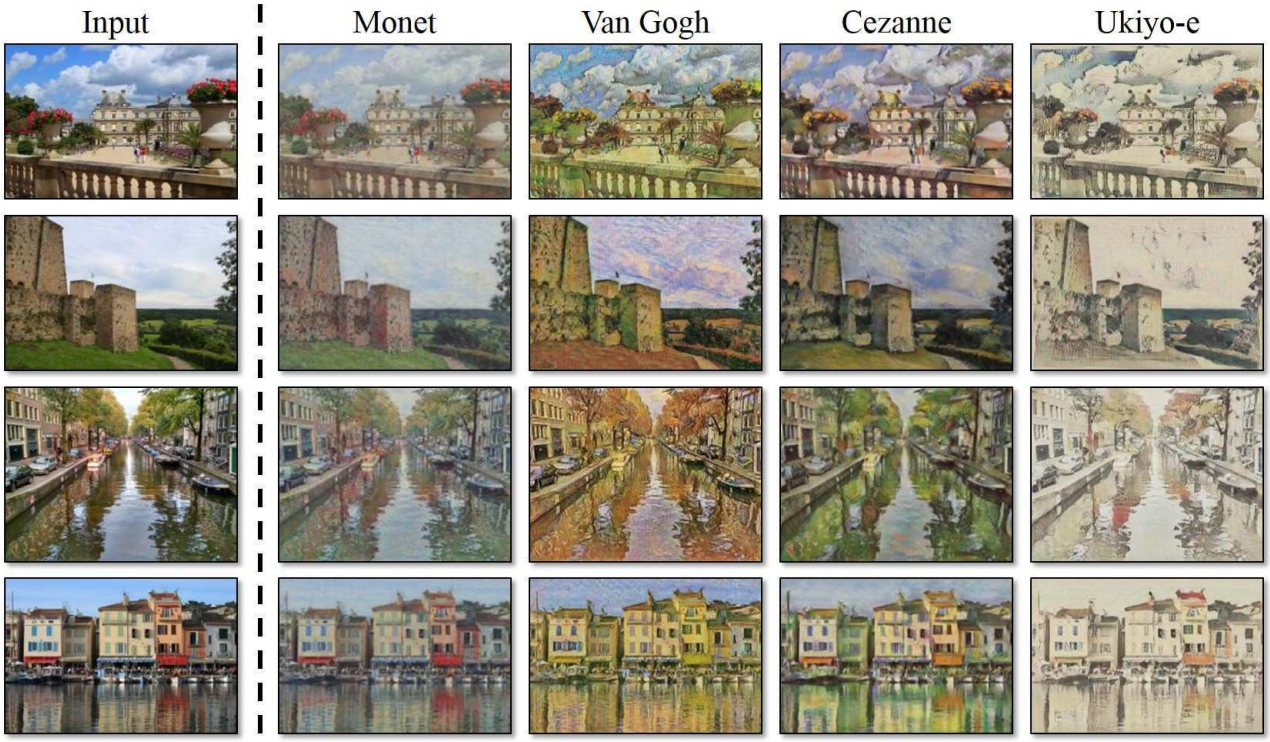
Em 1975, P. J. Werbos propôs em sua tese de doutorado (WERBOS, 1974) um processo de treinar redes neurais artificiais usando retropropagação (backpropagation) de erros, técnica extremamente importante no treino de redes neurais hoje em dia. Perto dessa data, em 1980, Kunihiko Fukushima propôs o "Neocognitron" (FUKUSHIMA, 1980), que consistia em uma rede neural que se organizava por conta própria através de aprendizado

para conseguir detectar padrões visuais baseado em similaridades geométricas de formas observadas, sem se preocupar com alterações em suas posições, baseada na teoria de Gestalt, que descreve como o ser humano percebe elementos visuais maiores através da soma de pequenas partes (WERTHEIMER, 1938).

Hinton, uma das figuras mais conhecidas no ramo de aprendizado profundo, também foi responsável por diversas contribuições na área, por exemplo, em 1985 inventou a máquina de Boltzman (ACKLEY; HINTON; SEJNOWSKI, 1985), que foi uma das primeiras redes neurais capaz de aprender representações internas (que significa representar informações do mundo real de uma maneira em que computadores podem usar para resolvar problemas complexos como diagnósticos médicos). Além disso, ao longo do tempo, esse pesquisador contribuiu (e ainda contribuiu) com diversos artigos relevantes, como (RUMELHART; HINTON; WILLIAMS, 1986), (HINTON et al., 2012), entre outros.

Hoje em dia os avanços continuam, com novos desenvolvimentos como as *Generative Adversarial Neural Networks* de Ian Godfellow (GOODFELLOW et al., 2014), que são redes capazes de fazer tarefas incríveis como gerar equivalentes de pinturas em estilos famosos a partir de fotos, conforme é visto na Figura 12.

Figura 12 – Aplicação de redes generativas adversárias

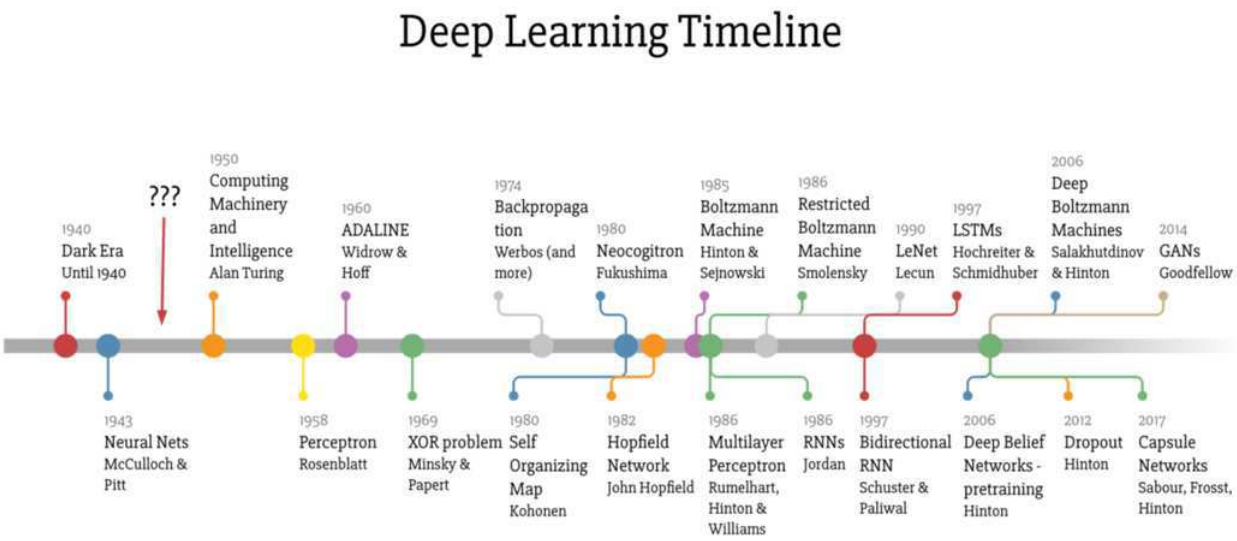


Fonte: (ZHU et al., 2017)

Por fim, uma linha do tempo contendo alguns avanços relevantes para o desenvol-

vimento de redes neurais artificiais é apresentada na Figura 13.

Figura 13 – Linha do tempo do desenvolvimento de redes profundas



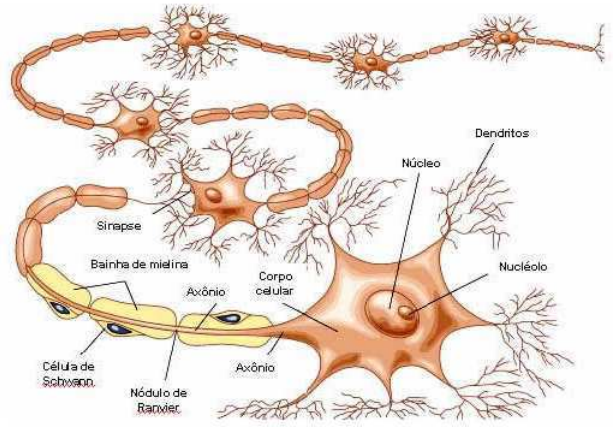
Fonte: (VáZQUEZ, 2018)

4 Redes Neurais Profundas

4.1 Explicação com elevada abstração sobre redes neurais profundas

Redes neurais são, como o nome sugere, baseado nas estruturas de funcionamento da mente humana, elas possuem nós que são análogos aos neurônios, que possuem, de maneira simplificada, três principais funcões: receber sinais, realizar a soma ponderada das entradas e processar informação, comunicar sinais às células alvo (outros neurônios, músculos ou glândulas). A Figura 14 mostra o axônio (transmissor), o dendrito (receptor, entrada do sinal) e o corpo de neurônios (processamento) conectados (por sinapses).

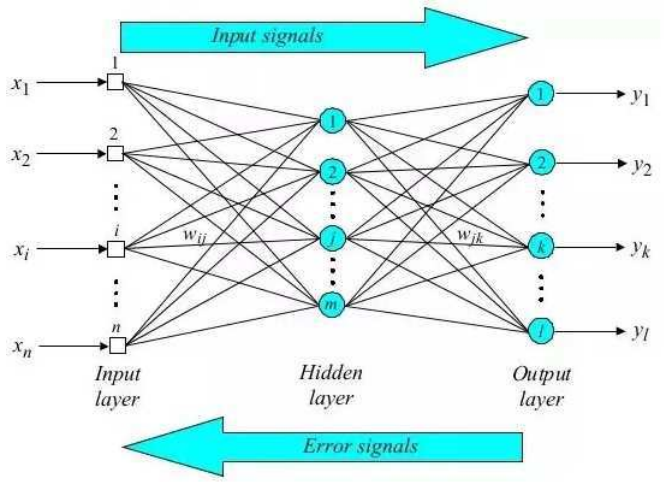
Figura 14 – Estrutura de funcionamento de um neurônio



Fonte: (AMABIS; RODRIGUES, 2016)

Para explicar como uma rede neural funciona, será utilizado o modelo com três camadas totalmente conectadas que usam retro propagação, ilustrado na Figura 15.

Figura 15 – Rede neural totalmente conectada e com três camadas



Fonte: (KING, 2016)

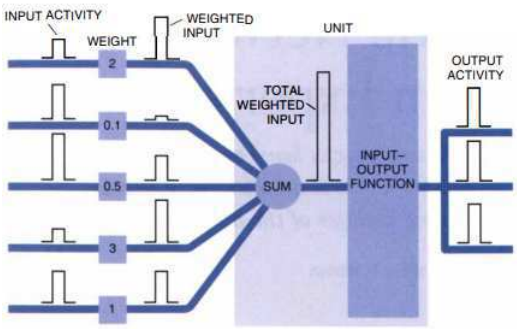
Nesse caso, a rede tem três camadas, a primeira é chamada de entrada, a última de saída e entre elas há a chamada camada oculta (*hidden layer*). Em cada camada, há nós, que foram desenvolvidos usando a analogia do funcionamento do neurônio (explicado anteriormente). Esses nós recebem entradas multiplicadas de acordo com um certo peso, que reflete a sua importância para o nó, soma esses valores e adiciona um viés (*bias*). Então, essa informação é transferida a uma função de ativação, em geral não linear, que determinará qual será a saída daquele neurônio (ou seja, fará uma transformação não linear nos valores que receber), em geral, essa saída será um valor entre 0 (menos ativo) e 1 (muito ativo), de modo que, a seguir, é apresentada uma equação, onde "Y"seria a saída do neurônio, "A"representaria a função de ativação, "w"seriam os pesos, "b"o viés e "X"as entradas. Aquele valor, Y, então, é transformado em entrada para outros neurônios ou saída final (se estiver na última camada). Essa passagem de informação é conhecida como Propagação para Frente (*forward propagation*).

$$Y = A(\sum W_{ij} * X + b) \tag{4.1}$$

As conexões entre nós são análogas às sinapses na mente humana e possuem pesos, conforme mencionado anteriormente, que determinam a relevância de determinada informação para um determinado nó. Esses pesos são chamados de parâmetros e, junto do viés, são modificados durante o treinamento da máquina. Outra definição nesse sentido é a de hiperparâmetros (*hyperparameters*), que são parâmetros fixados antes do processo de treinamento e dizem respeito a arquitetura da rede, alguns exemplos de hiperparâmetros, citados em (RADHAKRISHNAN, 2017), são: número de camadas ou profundidade da rede; taxa de aprendizagem da rede (*learning rate*); característica de *dropout*; pesos inici-

ais da rede; função de ativação; número de épocas, tamanho do *batch*. A imagem a seguir ilustra o que foi explicado sobre entradas (*input activity*), pesos (*weight*), nós (*sum*) e a função de ativação (*input-output function*).

Figura 16 – Esquema de Funcionamento de um nó em uma rede neural



Fonte: (HINTON, 1992)

Para exemplificar o funcionamento da rede e do treinamento, pode-se considerar um exemplo para classificar uma imagem entre 4 categorias de animais distintas. A entrada da rede (os "x" em 15) poderiam representar, por exemplo, o valor de um pixel em escala de cinza de uma imagem com 100x100 pixels. Nesse caso, haveriam 100x100 = 10000 entradas, cada uma com um valor de pixel entre 0 e 255. As saídas (os "y" na figura 15) representariam os animais possíveis, nesse caso, para 4 tipos de animais, haveriam 4 saídas (nós) possíveis, que, nesse caso, usando regressão logística, indicariam uma medida de quanto a entrada é, para rede, provável de ser um dos 4 animais. No meio, haveriam camadas ocultas, que serviriam para detectar características relevantes dos dados (fariam transformações matemáticas nas entradas).

Para treinar a rede, seria apresentada uma imagem de um dos quatro animais, então, essa imagem seria multiplicada pelos pesos (wij) e esses valores iriam (como entrada) para a camada oculta, que, de acordo com sua configuração, geraria saídas Y que iriam acionar os nós finais (de saída). O nó final mais ativo (de maior valor Y) representaria o que a rede acha que é aquela imagem de animal apresentada. Imagine que a rede detecta cachorro (nó de saída 1), gato (nó de saída 2), rato (nó de saída 3) e cavalo (nó de saída 4), se apresentado um cachorro para ela e ela tiver seu terceiro nó (rato) com maior valor Y em relação aos outros, significa que há um erro de detecção. A rede observa isso e tenta minimizar esse erro ao ajustar os parâmetros (pesos) para que na próxima vez que um cachorro aparecer, os valores (Y) dos seus nós sejam o mais próximo possível de 1, 0, 0, 0, o que representaria 100% de certeza que é um cachorro. Para isso, a rede faz cálculos visando minimizar uma função de custo. Esse processo se repete até que ela consiga classificar corretamente novas imagens (não vistas anteriormente) desses 4 animais.

4.2 Aprendizado supervisionado, não supervisionado e por reforço

Com o avanço da história relacionada a inteligência artificial, nota-se que frequentemente se menciona aprendizado, que é a ideia de fazer uma máquina se comportar como um ser humano, ou seja, adquirindo experiências e reagindo ao futuro. Nesse sentido, o aprendizado de máquina é, em geral, dividido em algumas subcategorias.

Primeiro, há a aprendizagem supervisionada, que consiste em aprender a mapear dados de entrada a saídas esperadas, chamados de "anotações", que em geral são dados que foram rotulado por humanos. Um exemplo deste tipo de aprendizado é o processo de classificação de imagens, onde, em geral, se tem um conjunto de imagens com algum rótulo e esses dados são fornecidos à máquina para "ensiná-la" a observar as características das imagens de acordo com seu rótulo, de modo que, ao receber uma nova imagem no futuro, a máquina avaliaria suas características de acordo com o que aprendeu no processo de treinamento e classificaria qual seria o rótulo (provável) dessa nova imagem. A figura 17 trás um exemplo do tipo de anotações que são fornecidas à máquina para esse caso.

Figura 17 – Imagens e seus rótulos

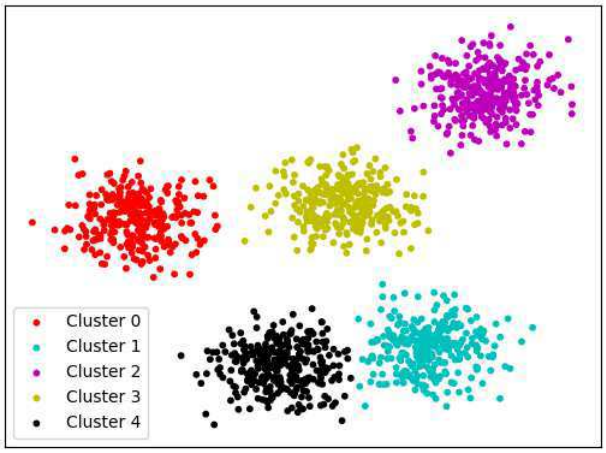
Nome	Label
Imagem 1	Dexter
Imagem 2	Pia
Imagem 3	Universidade

Fonte: Autor

Segundo, há o aprendizado sem supervisão, onde os dados que são fornecidos ao computador não possuem rótulo, mas ele consegue, mesmo assim, tirar informações relevantes. É usado para propósitos de visualização (organização), compressão e remoção de ruídos em dados. É extremamente utilizado no ramo de análise de dados e também serve para outros problemas de aprendizado, por exemplo, para ajudar a observar os dados antes de executar a rotulação para um problema de aprendizado supervisionado. Um exemplo conhecido de aprendizado sem supervisão é o de agrupamento e de redução de dimensões de dados. A Figura 18 ilustra o que seria um exemplo de agrupamento, em que

dados parecidos ficariam próximos.

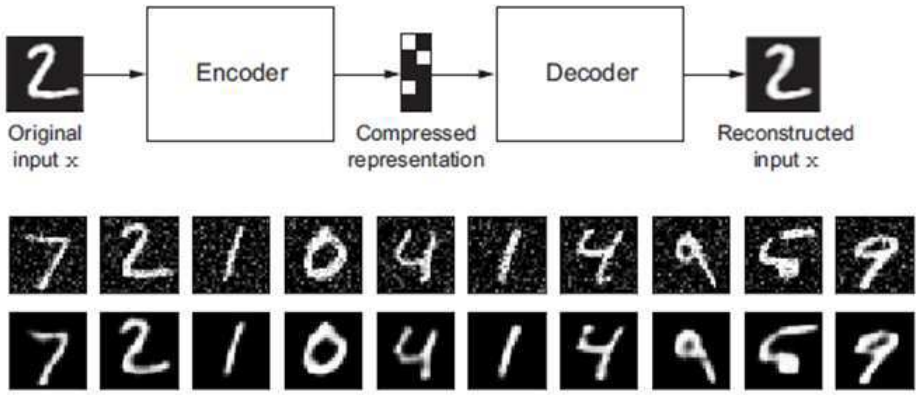
Figura 18 – Agrupamento (cluster) de dados



Fonte:(SEIF, 2018)

Prosseguindo, há uma subcategoria da aprendizagem supervisionada que é relevante o suficiente para ser mencionada de maneira separada, que é o aprendizado supervisionado por conta própria (CHOLLET, 2018), onde a máquina não precisa de um humano rotulando (ou seja, fazendo anotações) (n)os dados, mas de alguma maneira, por exemplo, euristicamente, essas anotações são geradas e usadas. Um exemplo desse tipo de rede relativo a esse tipo de aprendizado são os autoencoders, que possuem a característica do número de neurônios de entrada ser igual ao de saída, mas possuem um achatamento no meio da rede neural, o que faz com que esta seja forçada a representar a entrada com menos características, mas ainda semelhante à saída, o que pode ser usado para compressão de imagens, para tirar ruído (já que a rede receberia uma entrada ruidosa e tentaria representá-la com uma saída com as características principais, ou seja, sem ruído) e até para geração de imagens. Um exemplo de *autoencoder* é encontrado na Figura 19.

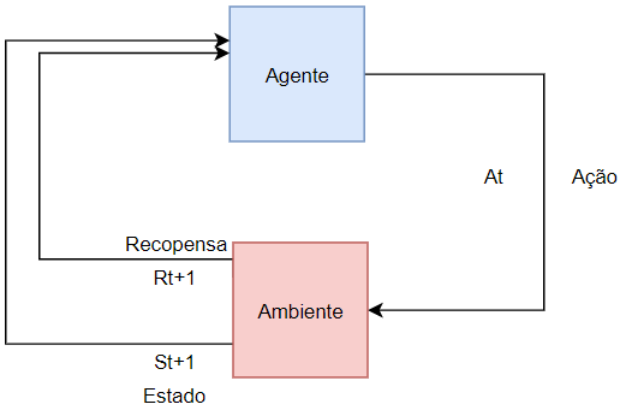
Figura 19 – Exemplo de funcionamento de um autoencoder



Fonte: (CHOLLET, 2018)

Por fim, há o aprendizado por reforço, onde não há um conjunto de dados rotulados onde se deve treinar a rede, mas há um ambiente com estados que a máquina consegue compreender, por exemplo, um jogo de Pacman, em que a máquina consegue entender situações de pontuação. Então, o treino se dá a partir de uma malha fechada em que a máquina vai tomando ações e recebendo recompensas por elas, de modo que seu objetivo final é aprender a tomar um conjunto de ações, de acordo com o estado atual, visando obter sempre a maior recompensa possível. Esse tipo de aprendizado tem aplicações em ramos distintos que envolvam ações sequenciais, como robótica, logística e em jogos. A Figura 20 ilustra a malha fechada relacionada a como o aprendizado por reforço funciona. Para mais informações a respeito dessa técnica, que ainda está em estágio de testes (CHOLLET, 2018), pode-se visitar o site do Google DeepMind e observar aplicações interessantes como a de treinar redes para jogos de Atari.

Figura 20 – Esquema de aprendizado por reforço



Fonte: Autor

4.3 Conceitos e parâmetros essenciais

Para que se entenda o que será desenvolvido neste trabalho, é preciso que se tenha pelo menos uma ideia do que são alguns termos relevantes no ramo de inteligência artificial. Esses termos serão explicados a seguir.

4.3.1 Regressão e Classificação

Os problemas relacionados a redes neurais, em geral, são divididos em duas categorias: regressão ou classificação. A primeira consiste em problemas que tentam prever uma quantidade de acordo com uma entrada. Um exemplo simples de regressão é o desenvolvimento de uma equação que determina o quanto uma mola se deforma de acordo com a força aplicada nela (lei de Hooke). Ou seja, observa-se o comportamento da mola e se faz um ajuste de curva para obter uma função que modela esse comportamento (equivalente ao treino da rede) e, a partir de uma nova entrada, que nesse caso seria a deformação, é obtida a saída (quantidade) correspondente, que nesse caso seria a força. Já a segunda categoria, diz respeito a problemas que visam prever a categoria (ou categorias) de determinada entrada e já foi citada com o exemplo dos quatro animais.

4.3.2 Validação, treino e conjunto de teste

Para que a rede treine é preciso que se tenha dados, já que o aprendizado se baseia em um algoritmo iterativo de atualização de pesos. Nesse sentido, as amostras que são reservadas para treino são usualmente chamadas de conjunto de dados de treino (*training set*).

No entanto, como o modelo é feito para ser usado também em dados diferentes daqueles em que foi treinado, é importante que seja observado o processo de ajuste dos parâmetros sintonizados em conjuntos de dados de validação (*validation set*.

Por fim, há um conjunto final para testes, em que se usa o modelo para fazer predições e se observa se elas foram adequadas. Uma regra comumente usada em grandes conjuntos de dados é usar 80 % dos dados para treinamento, 16 % para validação e 4 % para treinamento.

4.3.3 Número de camadas

O número de camadas se refere ao quanto a rede será profunda. Em geral, quanto maior a quantidade de camadas, mais complexos poderão ser os modelos que a rede usa para mapear da entrada para saída. Para tarefas simples, ou com poucos dados, em geral se usa duas camadas ocultas. Para tarefas maiores, usa-se mais camadas e observa-se se

o desempenho da rede está aumentando, processo que pode ser feito manualmente ou através do uso de algoritmos de otimização, como algoritmos genéticos.

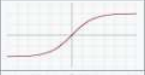
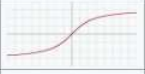
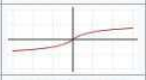
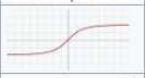
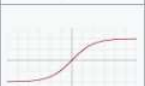
4.3.4 Função de ativação

A função de ativação, conforme mencionada anteriormente, é, em geral, uma função não linear responsável por determinar qual será a saída do nó de acordo com as entradas vezes os pesos, somado ao viés do nó. Ou seja, considerando que a função é “f”, que os pesos que saiem de i (camada anterior) para a camada j (em análise) são "wij"e que o viés é "b", a função de ativação será "f(x)", com "x"definido por:

$$x = (\sum w_{ij} + b) \tag{4.2}$$

Uma lista com as funções de ativaçao e a transformação que elas fazem em x é encontrada a seguir.

Figura 21 – Lista com alguns exemplos de função custo

Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
ArSinH		$f(x) = \sinh^{-1}(x) = \ln\left(x + \sqrt{x^2+1}\right)$	$f'(x) = \frac{1}{\sqrt{x^2+1}}$	$(-\infty, \infty)$
ElliotSig[8][9][10] Softsign[11][12]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$	$(-1, 1)$
Inverse square root unit (ISRU)[13]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3$	$\left(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}}\right)$
Inverse square root linear unit (ISRLU)[13]		$f(x) = \begin{cases} \frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$\left(-\frac{1}{\sqrt{\alpha}}, \infty\right)$
Square Nonlinearity (SQNL)[10]		$f(x) = \begin{cases} 1 & : x > 2.0 \\ x - \frac{x^2}{4} & : 0 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & : -2.0 \leq x < 0 \\ -1 & : x < -2.0 \end{cases}$	$f'(x) = 1 \mp \frac{x}{2}$	$(-1, 1)$
Rectified linear unit (ReLU)[14]		$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$	$[0, \infty)$

Fonte: https://en.wikipedia.org/wiki/Activation$_{f}unction$

Em geral, utiliza-se "ReLU"em camadas ocultas e sigmoide em camadas de saída, sendo recomendado utilizar "Softmax"(variação da sigmoide) nos últimos nós em tarefas de classificação com múltiplas saídas.

4.3.5 *Batch* e seu tamanho

Batch é um hiperparâmetro que indica quantas amostras passarão pela rede (ou seja, terão seus erros de predição calculados) antes de se atualizar os parâmetros internos do modelo. Afetando o quanto de amostras que passam de uma vez tem efeito no algoritmo de minimização e também na velocidade de treino da rede. Em geral, é comum na literatura o uso de "mini-batch" entre 32 e 512 amostras por vez, a depender do problema e quantidade de dados disponível, em geral recomenda-se o uso de 32 para iniciar (KESKAR et al., 2016). Também é importante saber que, se o conjunto de dados não for dividido igualmente pelo número definido no *batch*, o último *batch* que passar pela rede possuirá tamanho reduzido.

No Keras, *framework* utilizado neste trabalho e que será explicado posteriormente, é preciso definir o tamanho do *batch* de treino, segundo o que foi explicado acima. Além disso, define-se o tamanho do batch de validação e teste, que só indica o quanto de imagens serão carregadas de uma vez na memória computador, o que não tem nenhum efeito prático nos valores calculados.

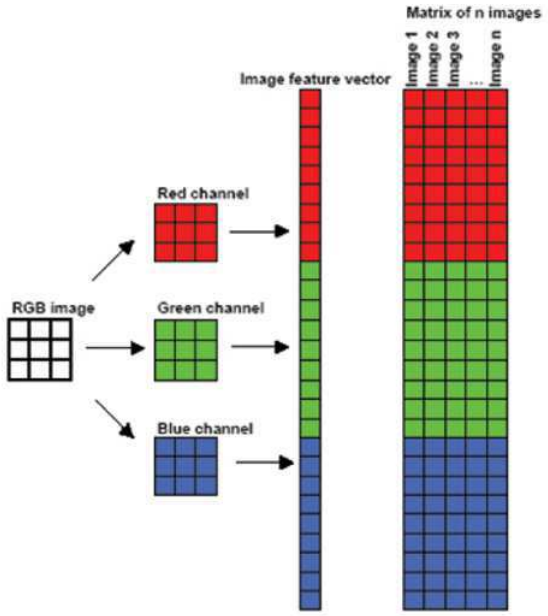
4.3.6 Épocas

Uma época é quando todo o conjunto de dados passa pela rede pelo menos uma vez. Ou seja, se o tamanho do batch for 32 e a rede tiver 3200 amostras, haveriam 100 iterações (passos por época) para que se completasse uma época. Em geral, o número de épocas usadas para treinar um modelo é determinado automaticamente ou manualmente. Manualmente, usa-se um gráfico de épocas versus acurácia e observa-se quando o plateau foi atingido.

4.3.7 Considerações sobre formatos de imagens

Imagens nada mais são do que conjunto de pixels e, nesse sentido, em geral são divididas em duas categorias, RGB (Red Green Blue) ou Grayscale (nível de cinza). É importante saber como são os formatos dessas categorias pelo fato de que eles são usados para definir os parâmetros de entrada na rede. Para uma imagem de 100x100 pixels, em RGB, haveriam 100x100x3 = 30000 pixels de entrada. Já em Grayscale, haveriam 100x100x1 = 10000. A Figura 22 ilustra como uma imagem RGB é "desdobrada" em pixels.

Figura 22 – Desdobramento de imagem RGB em vetor de pixels



Fonte: (PRABHU, 2018)

4.3.8 *Overfitting* e *Underfitting*

Overfitting é quando o modelo se ajusta em excesso ao conjunto de dados que foram usados para treino, deixando de funcionar adequadamente para dados novos. Em termos estatísticos, *overfitting* acontece quando o algoritmo de mapeamento captura, além das características essenciais, o ruído dos dados (CAI, 2014). Já *underfitting*, é quando o modelo não consegue capturar as características principais dos dados de entrada, necessárias para computar as saídas, por exemplo, a classificação adequada de uma determinada entrada.

Em geral, para se evitar *overfitting*, usa-se *dropout*, remoção de camadas, modificação de hiperparâmetros, adiciona-se regularização L1 e L2 (CHOLLET, 2018). Já para *underfitting*, em geral, a solução usada é aumentar a complexidade da arquitetura do modelo ou conseguir mais dados através de data augmentation.

4.3.9 *Fine-tuning*

O ramo de desenvolvimento de modelos de redes neurais está sempre com novidades e muitas das descobertas são movidas a competições. Sendo assim, há diversas soluções disponíveis que já foram testadas e que conseguem realizar tarefas que são recorrentes, por exemplo, extrair características de animais. Assim, um conceito relevante nesse ramo é o de *"fine-tuning"*, que consiste em não começar um modelo do zero, mas usar alguma

estrutura inicial, em geral um dos modelos de alguma dessas competições, como as da Imagenet, e ajustá-lo para o problema em questão.

Assim, o procedimento comum para criação de um modelo novo começa ao replicar camadas de modelos já estabelecidos na literatura, “congelar” o treinamento dessas camadas replicadas (que já foram treinadas e conseguem capturar informações relevantes de determinado tipo de dados), diminuir a taxa de aprendizado do modelo novo (em relação à que o modelo importado utilizava), adicionar novas camadas de acordo com o problema e ajustar os parâmetros destas através do treinamento.

4.3.10 Taxa de aprendizado

É uma medida que determina o quanto os parâmetros da rede serão modificados em cada atualização. Considerando gradiente descendente, onde os parâmetros buscam um mínimo e movem-se através de uma curva, em geral usa-se valores baixos de taxa de aprendizado para evitar oscilações grandes.

4.3.11 Atualização de parâmetros

Os parâmetros aprendidos são mudados conforme as equações (4.3 e 4.4). Cada valor do peso e do viés é atualizado, sabendo que “C” é a função custo, "w"um dos pesos daquele nó e "b"o viés daquele nó. Considerando o algoritmo de gradiente descendente, esse método de atualização funciona pelo fato da derivada ser uma forma de “mover” um parâmetro (para frente ou para trás) em uma curva, o que pode ser usado para achar valores mínimos da função custo (que é o objetivo).

$$W_{novo} = W_{antigo} + \frac{\partial C}{\partial W} \tag{4.3}$$

$$B_{novo} = B_{antigo} + \frac{\partial C}{\partial B} \tag{4.4}$$

Considerando um exemplo com função custo quadrática, dada por $C = (y-a)^2/2$, onde "y"é a saída desejada e "a"é a saída obtida. Lembrando que "a"é a função de ativação, "f", recebendo "z", definido como a soma de um conjunto de pesos (w) multiplicados por entradas (x), somados de um viés (b). Assim, a derivada parcial de C será, pela regra da cadeia:

$$\frac{\partial C}{\partial W} = (a-y)f'(z)x \tag{4.5}$$

$$\frac{\partial C}{\partial B} = (a-y)f'(z) \tag{4.6}$$

4.3.12 Métricas

Métricas são escolhas que o desenvolvedor faz para observar a qualidade do seu modelo. Em modelos em que se quer fazer uma classificação e observar se ela foi certa ou não, em geral se utiliza da métrica de "acurácia".

Para o caso do Keras, a acurácia é calculada ao se checar se a classe prevista é a mesma que a classe real. Isso é feito ao compaar o índice da classe com maior valor (prevista) e o índice da classe real, retornando 0 ou 1. Então, soma-se o total de acertos e divide-se pelo total de previsões feitas. Para validação, calcula-se com todos os dados de uma vez, e, para treino, calcula-se a acurácia por *batch* e depois se faz a média desses valores para uma época (YUAN, 2018).

4.3.13 Função custo

É uma quantidade calculada a cada *batch* que passa pela rede e indica o quanto o modelo está tendo sucesso na tarefa, por exemplo, de classificar imagens. Se a cada passo do treino tanto o custo do treino quanto o de validação diminuírem, significa que há uma melhoria no modelo. Em geral, para problemas de classificação com múltiplas classes, usa-se a função custo de entropia cruzada. Para o caso em que os neurônios de saídas usam função sigmoid, uma boa escolha é utilizar entropia cruzada (cross-entropy) (NIELSEN, 2018). Sabendo que "y" são os valores corretos para as saídas e "a" os valores previstos (nós de saída) pela rede, essa função custo será calculada, para cada amostra, por:

$$C = \frac{-1}{n} \sum_{x} \sum_{j} [y_j ln a_j^L + (1 - y_j) ln(1 - a_j^L)] \tag{4.7}$$

Então, esses valores são somados e divididos pela quantidade total do *batch* e isso é o que determina o parâmetro "loss", normalmente encontrado em *frameworks* de inteligência artificial. Para o caso da validação, esse cálculo é feito com o conjunto todo de dados. Uma consideração adicional relevante, em termos do *framework* Keras, é que existe o argumento de função "categorical crossentropy", que é usado para quando as classes foram codificadas usando números binários ("one-hot encoded") e sparse categorical crossentropy, que é usado para quando as classes estão como números inteiros (LIN, 2018).

4.3.14 Regularização

Regularização é uma técnica para diminuir overfitting que consiste em tentar diminuir a complexidade do modelo, o que afeta sua capacidade de generalização. Na prática, a regularização adiciona um termo (soma) à função custo para penalizar a rede por alta

complexidade. Essa complexidade, em geral, acontece quando a rede possui pesos (wij) com valores elevados. Em regularização L2, adiciona-se à função custo o termo abaixo, sabendo que λ será o parâmetro da regularização, "m", o número de entradas, e "n", o número de camadas:

$$C(custo) = C + L_2 L_2 = (\sum_{j=1}^{n} ||w^j||^2)\frac{\lambda}{2m} \tag{4.8}$$

Ao observar esse termo, percebe-se que ele forçará o modelo a ajustar os pesos, diminuindo os valores destes, de forma que algumas camadas teriam pesos tão pequenos que pareceriam inexistentes, ou seja, o modelo ficaria (aproximadamente) com menos termos e menor complexidade.

Outra técnica interessante de regularização é o *dropout*, que consiste em "apagar" alguns neurônios durante parte do treino, o que diminui a chance de *overfitting* da rede (HINTON et al., 2012).

4.3.15 Normalização, centralização e padronização

Normalização consiste em transformar os valores mínimos e máximos de cada entrada pertencente a uma amostra em uma escala entre 0 e 1, que é a escala recomendada para treinamento (SARLE, 2002). No caso de imagens, divide-se os valores de cada pixel por 255. Já a padronização, em caso de imagens, consiste em escalonar os valores dos pixels de modo que tenham média zero e variância unitária. Esta padronização consiste em diminuir o valor do pixel pela média e dividir pela variância, o que pode ser feito por imagem (tirando a média e variância dela) ou por conjunto de dados (usando a média e variância do conjunto de dados). No Keras, além de fazer a normalização antes de inserir os dados na rede, também é possível fazer a normalização por camada.

5 Redes Neurais Convolucionais e outros conceitos

Para a construção de redes neurais, como já mencionado anteriormente, usam-se camadas. Cada tipo de camada tem sua estrutura e é utilizada de acordo com o tipo de problema que se pretende resolver. Em geral, para problemas de classificação de imagens, é comum a utilização de redes neurais convolucionais e por isso esta terá foco nesta seção.

5.1 Camada Densa

É o tipo de camada mais simples, tem todas as entradas e todas as saídas conectadas. Faz uma transformação nas entradas que recebe através da função de ativação. Um exemplo de camada densa é encontrado na Figura 11.

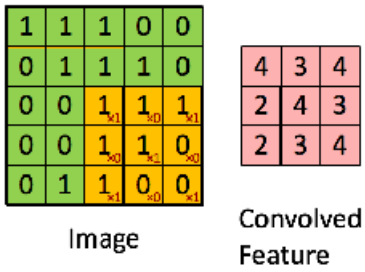
5.2 Camada de Achatamento (*Flatten*)

É uma camada que transforma dados de entrada de múltiplas dimensões em dados de saída de uma dimensão. Em geral são usadas depois de camadas convolucionais para gerar as entradas de uma dimensão das camadas totalmente conectadas (densas).

5.3 Camada de Convolução

Primeiro, lembra-se que imagens são apenas pixels, números em uma matriz, com valores entre 0 e 255, de modo que, quando unidos, representam características. Um conjunto desses números pode ser multiplicado, modificando os valores de seus pixels e, assim, amplificando ou diminuindo uma característica. Essa operação é chamada de filtragem.

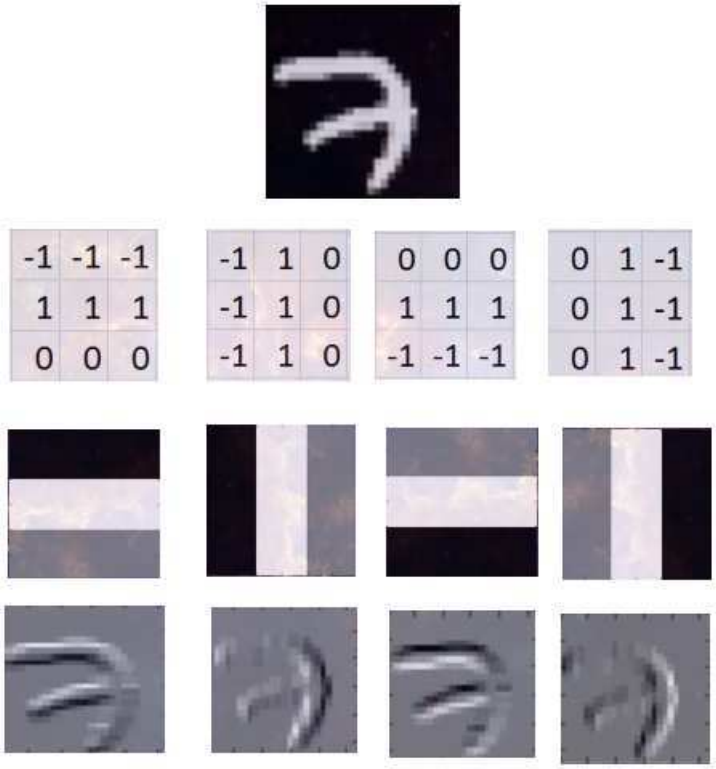
Figura 23 – Atuação da convolução



Fonte: (PRABHU, 2018)

É justamente isso que uma rede neural convolucional faz, ela filtra (extrai) características de uma imagem e usa essa informação para poder classificar melhor uma entrada. Um exemplo que ilustra melhor isso está apresentado na Figura 24, onde quatro filtros diferentes são aplicados a uma entrada e acentuam ou amenizam bordas, respectivamente, horizontal no topo, vertical na esquerda, horizontal embaixo e vertical na direita da imagem.

Figura 24 – Retirada de característica (feature) pelo filtro

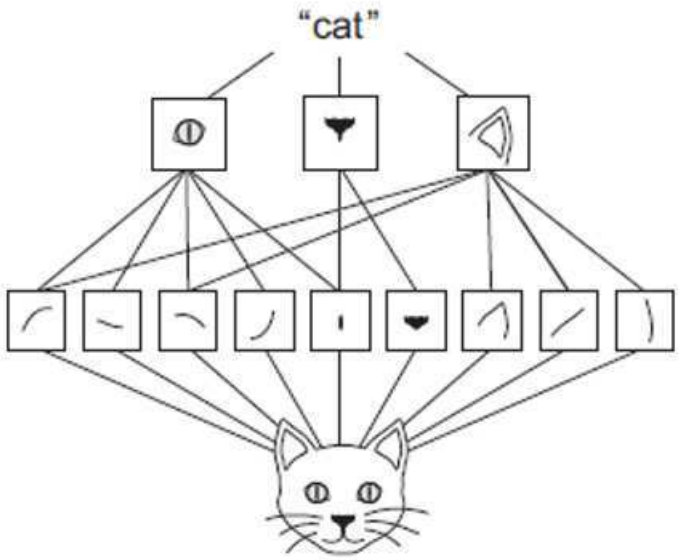


Fonte: https://www.youtube.com/watch?v=YRhxdVk$_sIs$

Uma característica interessante nesse processo é que a medida que a rede aprende

adequadamente a detectar uma característica relevante, ela será capaz de observá-la independentemente da translação na imagem. Outra característica importante é o fato de que camadas convolucionais podem se unir para detectar de partes menores a um todo, conforme é ilustrado na Figura 25.

Figura 25 – Reconhecimento de padrões pela rede



Fonte: (CHOLLET, 2018)

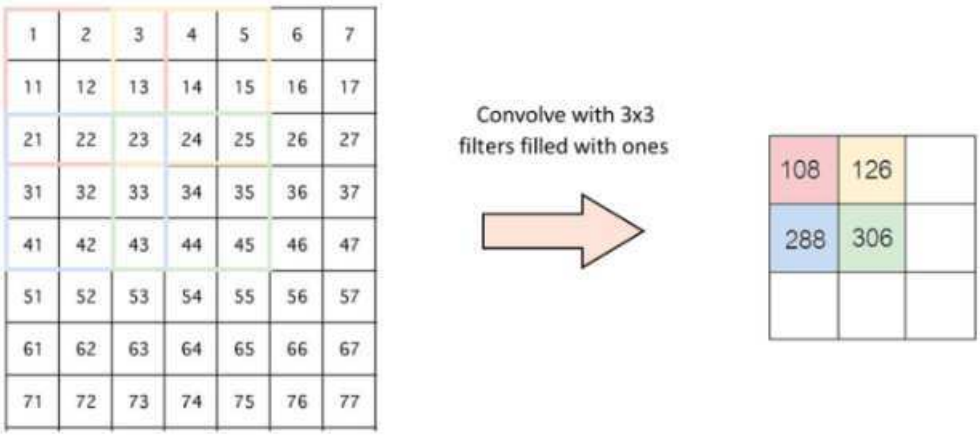
5.4 *Padding*

É uma operação que consiste em adicionar zeros na matriz reduzida por alguma operação, como a convolução, visando evitar que as dimensões de uma imagem propagada sejam diminuídas.

5.5 *Stride*

Stride é um parâmetro das redes convolucionais que indica como a rede moverá ao longo da imagem, se será de 1 em 1 pixel, 2 em 2, etc. A Figura 26 exemplifica esse funcionamento.

Figura 26 – Visualização do *Stride*

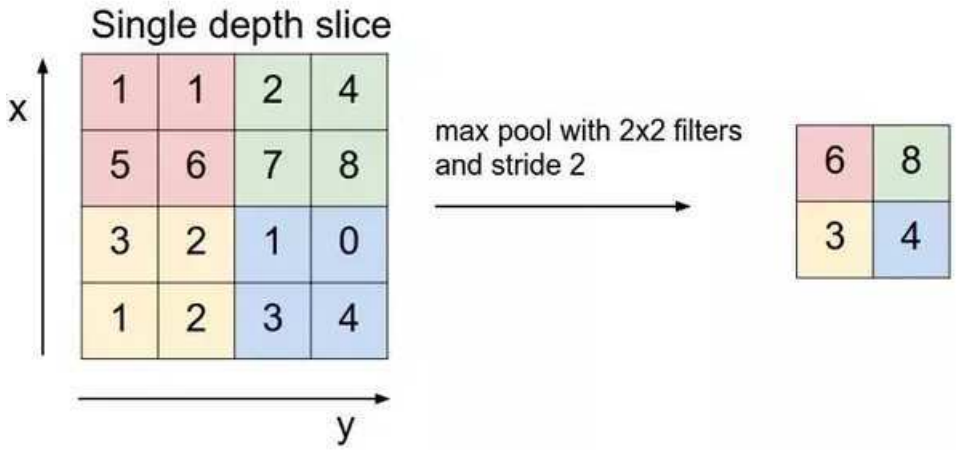


Fonte: (PRABHU, 2018)

5.6 *Pooling*

Pooling consiste em reduzir o número de características da imagem ao agrupá-la em pedaços, mas mantendo suas principais características. Podem ser de diferentes tipos, como pooling de média, de soma ou de máximo. Na Figura 27 está ilustrado o pooling de máximo com stride 2 e filtro de dimensões 2x2, ou seja, move-se o filtro de 2 em 2 pixels e tira-se o valor máximo do que está sendo observado.

Figura 27 – Atuação da camada de *max pooling*

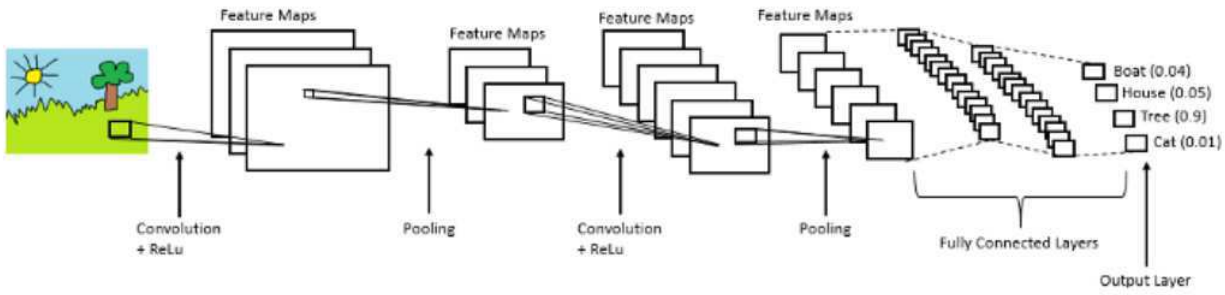


Fonte: (PRABHU, 2018)

5.7 Projeto de uma rede usando redes convolucionais

Para o projeto usando redes convolucionais, em geral, a abordagem que se usa é a de adicionar camadas convolucionais até que se extraia características relevantes o suficiente. Então, essas características, em inglês, *feature maps*, são então transformadas em vetores de uma dimensão usando uma camada *Flatten*. Por fim, adiciona-se camadas densas até que se tenha processamento suficiente desses dados e classificação adequada (sem *underfitting*). Se ocorrer *overfitting*, conforme mencionado anteriormente, usa-se técnicas de regularização ou diminui os parâmetros e camadas usadas. Um exemplo dessa ideia está ilustrado na Figura 28.

Figura 28 – Projeto utilizando redes convolucionais



Fonte: (PRABHU, 2018)

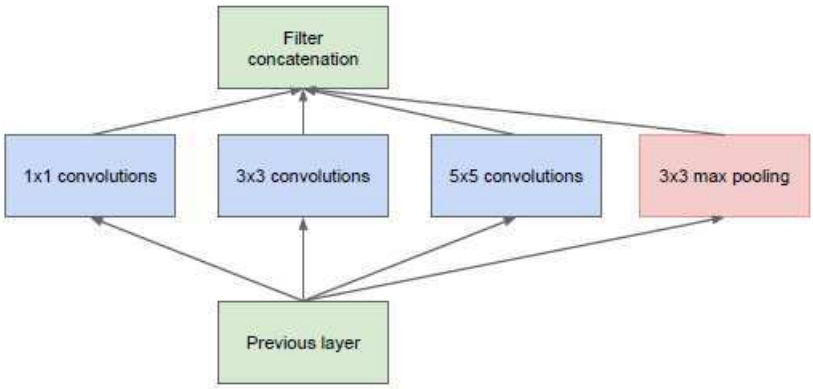
5.8 Concatenação e adição

Outros conceitos importantes em redes neurais são o de concatenação e adição. Adição consiste em juntar camadas de mesma dimensão através da soma de seus pesos. É útil quando se quer interpretar uma camada como um resíduo, correção de outra, como se fosse uma complementação da informação. Já a concatenação, consiste em unir camadas paralelamente, ou seja, se uma camada tinha dimensões de 30x100x100 e a outra tinha dimensões de 60x100x100, ao concatenar, será gerada uma camada com dimensões 90x100x100 e é usado quando as camadas precisam ser unidas mas não tinham muita relação entre si (SHIMAO, 2018).

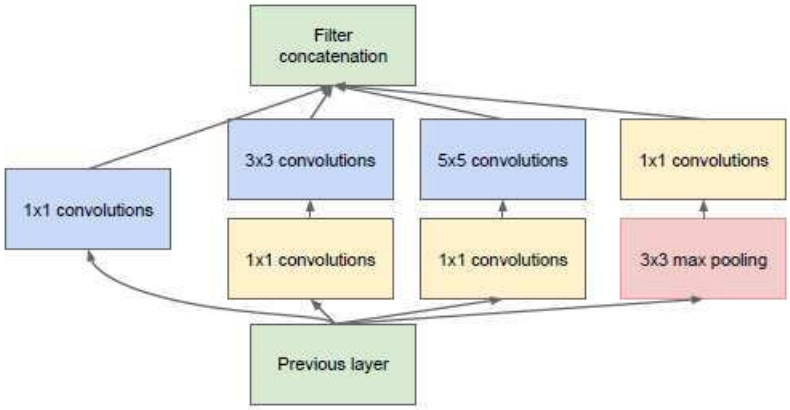
5.9 *Inception* e Inception V3

Inception é uma arquitetura que usa a ideia de usar camadas convolucionais em paralelo e concatená-las. O que aumenta a eficiência de extração de características e, com a concatenação, também se pode representar filtros maiores através do uso da soma de menores. A ideia geral está ilustrada na Figura 29.

Figura 29 – Explicação da camada de *Inception*



(a) Inception module, naïve version

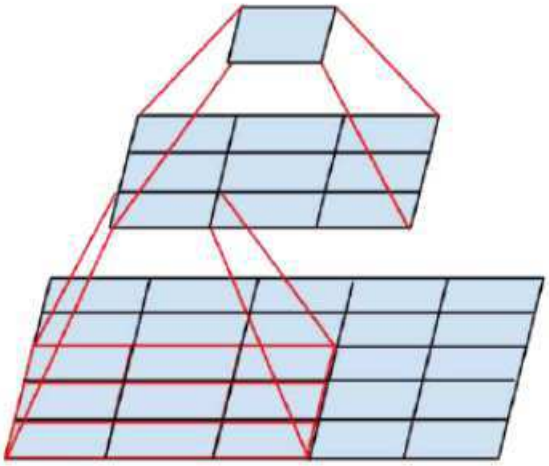


(b) Inception module with dimensionality reduction

Fonte: (SANDHU, 2017)

Para exemplificar o que foi dito, considere a Figura 30. Nela, observa-se que a utilização de uma camada com filtro 5x5 (25 parâmetros) foi substituída pelo uso de duas com filtro 3x3 (18 parâmetros), o que gera uma economia de 28% no número de parâmetros.

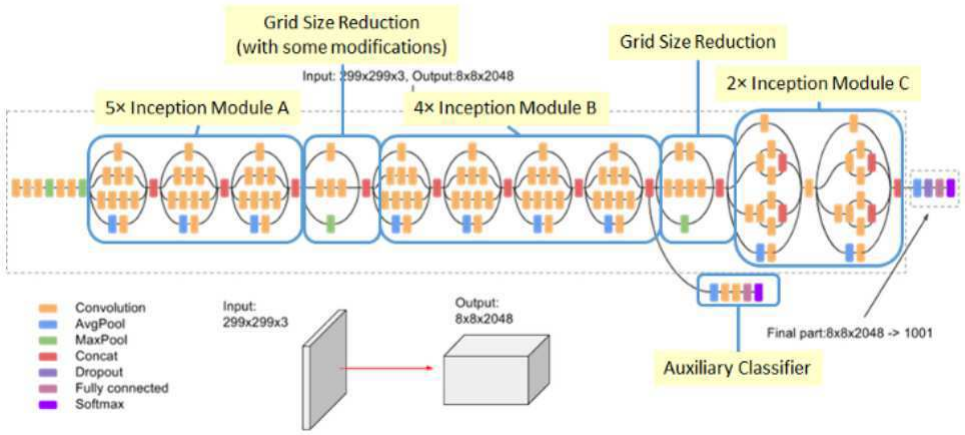
Figura 30 – Demonstração da substituição de filtros



Fonte:(RAJ, 2018)

Por fim, utilizando as ideias explicadas anteriormente e mais alguns ajustes, uma equipe do Google desenvolveu uma arquitetura que foi campeã numa competição da Imagenet em 2015. Essa arquitetura é representada na Figura 31.

Figura 31 – Inception V3



Fonte: (RAJ, 2018)

6 A competição e o padrão das arquiteturas

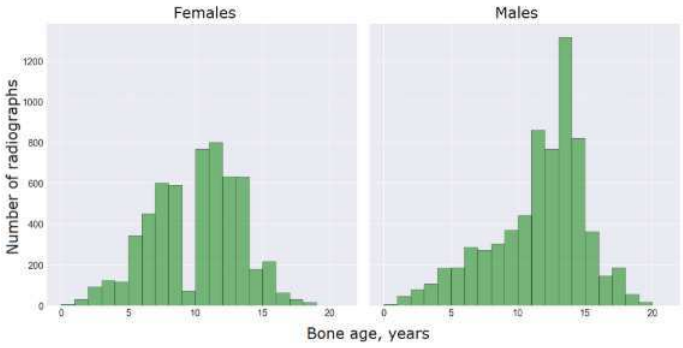
O professor da Universidade de Toronto, Geoffrey Hinton, que é um dos mais importantes pesquisadores de inteligência artificial, tem uma frase famosa que diz: “se você trabalha como um radiologista, você é como o coiote que já passou da beirada do penhasco, mas ainda não olhou para baixo, então não percebe que o chão está por baixo dele”, essa frase demonstra o potencial de transformação da inteligência artificial em ramos tradicionais do trabalho, como a medicina.

Nesse sentido, muitas instituições estão se dividindo entre duas abordagens, a primeira consiste em ignorar as mudanças da tecnologia e não se preparar para o futuro. A segunda, aceita as mudanças e, ao invés de rejeitá-las, tenta se adaptar, aprendendo e melhorando com elas.

Com essa visão, a Sociedade de Radiologia da América do Norte (RSNA) promoveu uma competição em inteligência artificial em 2017, visando descobrir os melhores algoritmos para predição de idade óssea em radiografias de mão e punho. Para essa competição, eles disponibilizaram 12612 imagens de treino, devidamente rotuladas, que consistiam em radiografias realizadas nos hospitais dos Estados Unidos, em pacientes crianças e adolescentes. Além disso, haviam 200 imagens para teste dos resultados.

O gráfico a seguir demonstra a distribuição desses dados.

Figura 32 – Distribuição dos dados de idade óssea entre indivíduos do sexo masculino e feminino



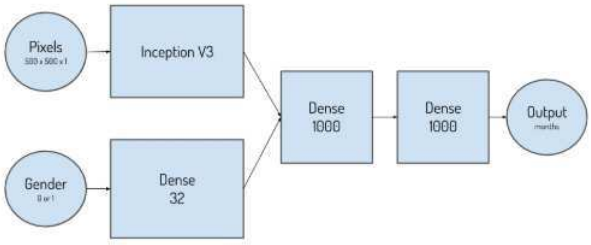
Fonte: Autor

O vencedor da competição utilizou uma rede que recebia as imagens ajustadas, em escala de 500x500 pixels, e a informação do sexo (masculino ou feminino). Como na época não possuíam dados para teste, dividiram as imagens entre 85% (10720) para treino

e 25% (1892) para validação. Como usaram Inception V3, que já tem normalização de dados dentro das camadas, decidiram não utilizar normalização adicional nos dados.

A arquitetura passava as imagens para um bloco contendo Inception V3, subtraindo a camada após a última concatenação. Então usava a camada Flatten nesses dados, o que gerava 100384 entradas que eram concatenadas com a outra parte de entrada, que consistia na informação se era masculino ou feminino aplicada a uma camada densa de 32 nós. Então, duas camadas densas de mil nós, com ativação usando ReLU, eram usadas para fazer o mapeamento desses dados. A imagem dessa arquitetura é demonstrada na Figura 33.

Figura 33 – Distribuição dos dados de idade óssea entre indivíduos do sexo masculino e feminino



Fonte: (CICERO; BILBILY, 2017)

Por fim, como havia um único nó com camada linear para fazer a determinação do mês da imagem de entrada. Usando a função de preprocessamento do Keras, utilizaram rotação de 20 degraus, translação horizontal/vertical de 20%, zoom de 20% e giro horizontal no conjunto de dados para treino, esses valores foram baseados em condições realistas de variação desse tipo de imagens. O modelo final foi treinado usando minibatch de tamanho 16 em 500 épocas, o que durou aproximadamente 50 horas. Além disso, modificaram a taxa de aprendizado para diminuir quando fosse atingido o plateau na validação. Para testar a validação, usaram um gerador modificado para criar 10 novas imagens, que eram usadas em cinco modelos diferentes, gerando o total de 50 predições. Então, foi feita a média dessas predições para que se chegasse ao valor final. Por fim, nos testes, conseguiram um resoltado de MAD (média absoluta da diferença) de 4.265 nas 200 imagens testadas.

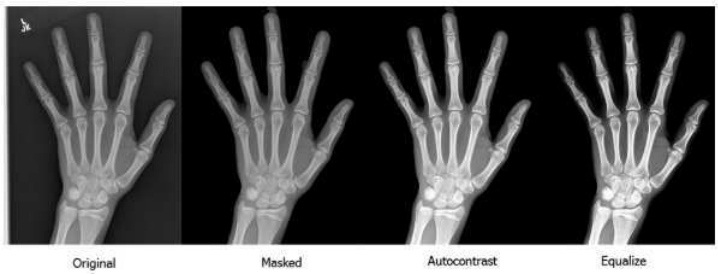
Revisando a literatura (em que os artigos estão listados na seção de referência deste documento) para observar soluções de problemas semelhantes, da mesma época, ou não, observa-se que as abordagens podem ser resumidas em cinco passos, citados a seguir:

- Definir o problema entre classificação ou regressão.

- Definir qual arquitetura base será usada. Em geral utilizaram Inception (ou variantes), VGG (nos artigos mais antigos) ou ResNET.
- Aplicação de estratégia de preprocessamento de dados.
- Aplicação de técnicas de regularização para evitar overfitting.

Outro artigo que é mais diferenciado nesse sentido e por isso vale a pena ser citado é o artigo (IGLOVIKOV et al., 2018), em que é adicionada a ideia de remoção do fundo da imagem e outros ruídos através da segmentação da mão utilizando a arquitetura U-Net e também são feitos preprocessamentos adicionais nas imagens. Os resultados desses pré-processamentos estão ilustrados na Figura 34.

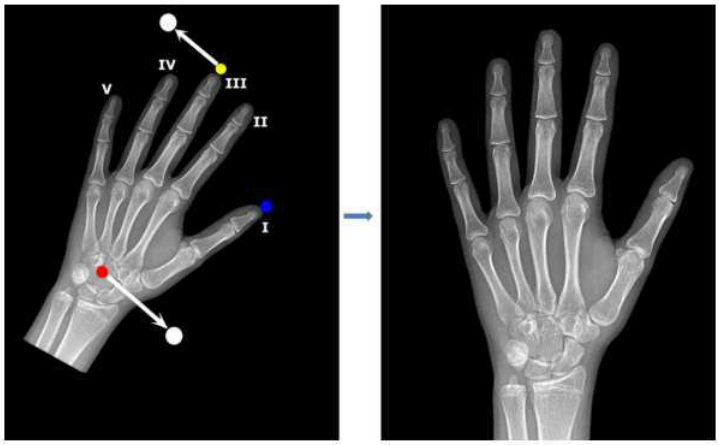
Figura 34 – Preprocessamento de imagens realizado



Fonte:(IGLOVIKOV et al., 2018)

Além disso, outra contribuição relevante desse artigo foi em relação ao ajuste de posição das imagens. Para isso, foi feita a segmentação de alguns pontos da mão e, a partir da posição "ideal" de onde eles deveriam estar em uma imagem, foram ajustados parâmetros de rotação e movimentação dos pixels. Essa ideia é compreendida pela Figura 35.

Figura 35 – Rotação e ajuste de imagem



Fonte: (IGLOVIKOV et al., 2018)

7 Implementação e aprendizados

Para atingir o objetivo, diversas considerações e problemas tiveram que ser resolvidos. Primeiro, foi feita a pesquisa para descobrir qual conjunto de dados utilizar e, conforme explicado na seção anterior, utilizou-se os dados da competição da RSNA.

Então, foi feito um estudo em relação a *frameworks* de desenvolvimento em redes neurais e, após observar Tensorflow, Caffe, PyTorch, entre outros, a conclusão que se obteve foi que era melhor resolver esse problema com o Keras, que é mais simples de utilizar e contêm funções de pré-processamento úteis.

Então, como não se tinha conhecimento nenhum sobre redes neurais, foi feito um estudo a respeito do tema e de como aplicá-lo na prática utilizando o Keras. Vários experimentos foram feitos e alguns resultados serão demonstrados nesta seção. Fazendo uma análise prévia sobre o trabalho, apesar dos resultados finais obtidos na implementação não serem estado da arte, nem nada muito diferente daqueles obtidos em competições, principalmente por conta das limitações de hardware e de tempo de desenvolvimento, há uma enorme utilidade e relevância nesse trabalho como ferramenta de resumo, ponto de partida, para quem deseja cumprir tarefa semelhante de começar do zero e tentar resolver um problema de aprendizado de máquina. Por conta disso, neste seção, serão abordadas algumas "dicas" que já foram explicadas em capítulos anteriores, mais algumas outras, e os resultados finais.

7.1 Características de um problema de Inteligência artificial

Levando em conta o que foi observado ao realizar este trabalho, percebe-se que o projeto de uma rede a partir do zero dificilmente obterá resultados excelentes em um conjunto de dados pequenos. Isso acontece pelo fato de que os projetos tradicionais usam como ponto de partida arquiteturas que já foram testadas em competições com milhões de imagens, ou seja, estas já são capazes de extrair diversos padrões de uma imagem, isso se dá pelo fato de seus parâmetros já terem sido treinados exaustivamente.

Sendo assim, para começar, deve-se selecionar uma arquitetura como base. Para isso, deve-se buscar problemas semelhantes na literatura, o que pode ser encontrado, por exemplo, ao buscar competições da Imagenet, Kaggle, ou de associações médicas.

Então, modifica-se uma arquitetura base a partir dos dados que se tem em mãos, por exemplo, a arquitetura campeã da competição da RSNA fez apenas uma simples modificação na Inception V3, adicionando informações sobre gênero das imagens avaliadas e modificando o final da rede para obter regressão linear.

Prosseguindo, deve-se unir as informações (se foram utilizadas camadas em paralelo) utilizando operações de adição ou concatenação, conforme explicado anteriormente.

Depois desse passo, testa-se o modelo. Se o resultado estiver com *underfitting*, adiciona-se camadas, aumenta-se os nós e as épocas de treinamento. Continuando com *underfitting*, realiza-se preprocessamento nos dados, semelhante ao que foi mencionado na seção anterior, tanto para melhorar a visão da rede em termos de predição, como para aumentar a quantidade de dados disponíveis ao gerar imagens artificiais. Se mesmo assim o *underfitting* continuar, deve-se mudar a arquitetura de uma maneira mais brusca ou rever se não foi feito algum procedimento incorreto na passagem dos dados.

7.2 Instalação e requisitos de processamento e internet

Uma consideração relevante a ser feita é em relação ao custo de hardware, espaço de armazenamento e necessidade de uma boa internet para realizar algum trabalho de qualidade nesta área. Isso acontece pelo fato de que é preciso grandes quantidades de dados para fazer os treinamentos e, por causa disso, deve-se utilizar GPU sempre que possível.

Para a utilização do Keras com GPU, deve-se fazer download do *CUDA* e *CuDNN* (se o GPU utilizado for da NVIDIA). Uma observação importante sobre esse passo é o fato de que o Tensorflow (backend onde o Keras roda) é extremamente dependente de versão, então, versões anteriores ou superiores do *CUDA/CuDNN* podem não funcionar. Para solucionar esse problema, há duas alternativas, ou é feito o download correto segundo as recomendações atualizadas do site do Tensorflow, ou utiliza-se um gerenciador de pacotes, como o Anaconda, e "força-se" o download da versão adequada do Tensorflow e de outras dependências que funcionem em conjunto.

7.3 Organização, visualização e análise dos dados

Um dos processos mais importantes de um cientista de dados é saber observar e tirar conclusões úteis a respeito de seus dados. Sendo assim, a primeira tarefa que deve ser feita em um problema desses é decidir como os dados serão organizados. Nesse sentido, os rótulos das imagens que seriam usadas estavam em meses, conforme a figura a seguir demonstra.

Figura 36 – Formato dos dados recebidos

1	id,boneage,male
2	1377,180,False
3	1378,12,False
4	1379,94,False
5	1380,120,True
6	1381,82,False
7	1382,138,True
8	1383,150,True
9	1384,156,True
10	1385,36,True
11	1387,138,True
12	1388,126,False
13	1389,138,True

Fonte: Autor

Nesse ponto, duas decisões deveriam ser feitas. A primeira, é em relação a como o problema seria tratado. Se seria visto como um problema de regressão ou de classificação. A opção escolhida foi classificação, pois já haviam excessos de soluções prontas para esse caso que utilizavam regressão. A segunda, era em relação aos rótulos que seriam usados. Nesse sentido, escolheu-se que as imagens seriam classificadas de acordo com os valores do atlas de Greulich e Pyle (GREULICH; PYLE, 1959).

Para fazer a conversão desses valores, primeiro foi convertido o csv em xlsx, então, utilizado o MATLAB para modificar os valores segundo um algoritmo de mapeamento que foi desenvolvido e que basicamente fazia, "se um valor está dentro de x e y, ele é w". Esse mapeamento tinha dois passos, o primeiro era identificar esses intervalos, o segundo era converter os dados dentro dele em números inteiros (para facilitar visualizações futuras).

Figura 37 – Esquema de mapeamento em categorias utilizado para visualizar melhor os dados

Meses	Anos	Label
3	0,25	1
6	0,5	2
9	0,75	3
12	1	4
18	1,5	5
24	2	6
30	2,5	7
36	3	8
42	3,5	9
48	4	10
54	4,5	11
60	5	12
72	6	13
84	7	14
96	8	15
108	9	16
120	10	17
132	11	18
144	12	19
156	13	20
168	14	21
180	15	22
192	16	23
204	17	24
216	18	25
228	19	26
240	20	27

Fonte: Autor

Na Figura 38, há um trecho do resultado final do arquivo de dados modificado para validação, onde nota-se que os rótulos viraram valores inteiros, entre 1 (representando 3 meses) e 27 (representando 20 anos).

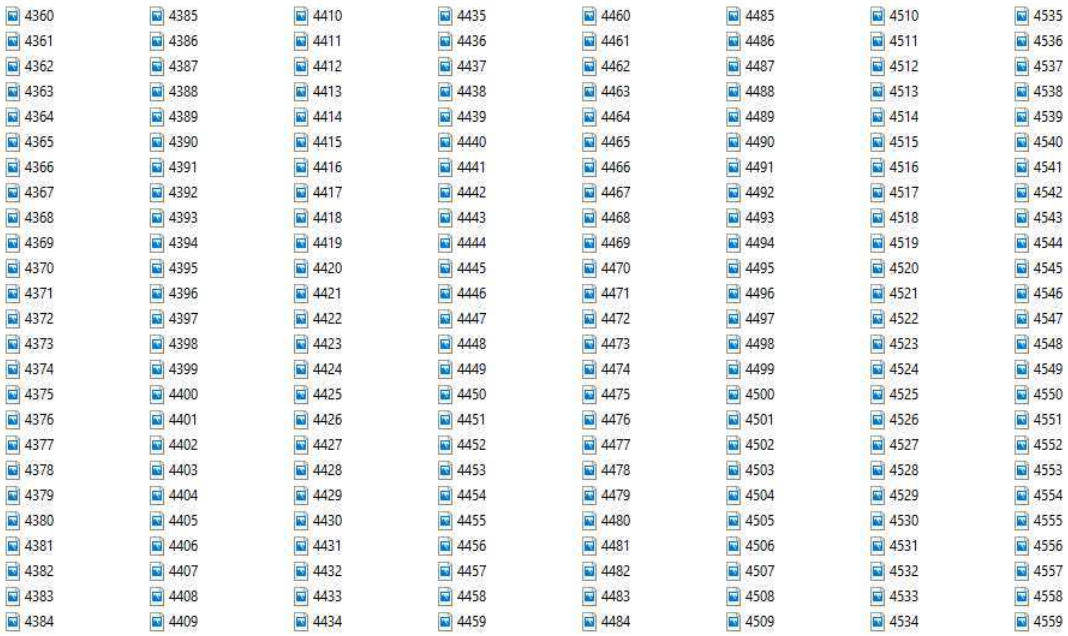
Figura 38 – Resultado do mapeamento

	filenames	male	labels
2	1386	FALSE	7
3	1392	TRUE	21
4	1397	FALSE	5
5	1401	FALSE	18
6	1410	TRUE	12
7	1413	TRUE	20

Fonte: Autor.

Além disso, os dados disponíveis eram todos concentrados em uma pasta só, conforme a imagem a seguir, para os dados de teste, demonstra.

Figura 39 – Dados disponíveis inicialmente - para teste



Fonte: Autor

Então, foi desenvolvido um script que automaticamente criava pastas de acordo com um arquivo que continha os rótulos (labels) e automaticamente colocava as imagens dentro dessas pastas. Para isso, utilizou-se Python e a biblioteca Pandas, que é excelente para manipulação de arquivos csv/xlsx. O resultado final está demonstrado a seguir.

Figura 40 – Dados após serem organizados

Fonte: Autor

7.4 Pré-processamento

Conforme observado no estudo de artigos e de outros tópicos sobre inteligência artificial, notou-se que era necessário fazer ajuste nos dados (pixels) para que a rede aprendesse de uma maneira mais eficiente.

Assim, para entender o efeito prático de algumas transformações, foram feitos alguns experimentos, primeiro utilizando a biblioteca Imagaug, que fornece diversas opções para processamento e criação de dados. Em cada um desses experimentos, calculou-se o "shape"(forma) que os dados se apresentavam, a média de seus pixels, variância, entre outras características. Alguns desses experimentos, primeiramente feito com imagens genéricas, que não eram relacionadas ao problema, serão apresentados a seguir.

Primeiro, para duas imagens, img1 e img2, foram aplicados processamentos, conforme:

Figura 41 – Teste de processamentos genéricos

Fonte: Autor

Então, um teste em relação às propriedades de canais dessas imagens foi feito, onde se nota que as imagens RGB realmente possuem 3 canais e as imagens em tons de cinza só possuem 1 canal. Esse teste está apresentado na Figura 42.

Figura 42 – Teste de canais

```
img1.shape  #CONFIRMANDO QUE RGB POSSUI 3 CANAIS
(993, 800, 3)

img3.shape #OBSERVANDO QUE GRAYSCALE NÃO POSSUI 3 CANAIS
(993, 800)
```

Fonte: Autor

Prosseguindo, algumas características das amostras de radiografia foram observadas, visando decidir que tipo de transformações/normalizações seriam feitas.

Figura 43 – Observação da média, desvio padrão e valores máximos e mínimos dos pixels em um batch de teste

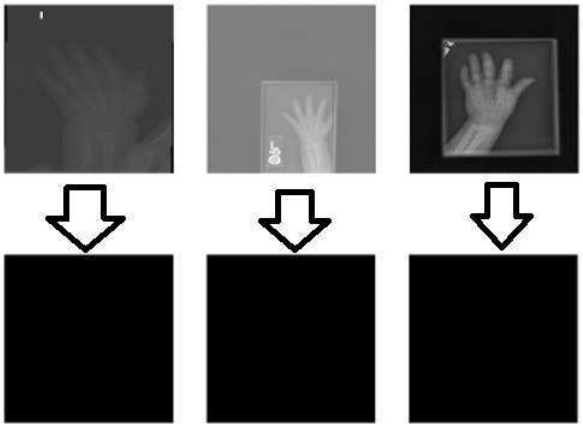
```
print('Teste= %s, Pixel min=%.3f, max=%.3f, std=%.3f, mean=%.3f' % (imgs.shape, imgs.min(), imgs.max(), imgs.std(), imgs.mean()))
```

```
Dimensões do Batch = (3, 224, 224, 3), Pixel min=0.000, max=1.000, std=0.180, mean=0.323
```

Fonte: Autor

Continuando, mas agora com um enfoque maior nos dados relevantes, algumas transformações foram feitos nas amostras, de acordo com o que foi explicado em capítulos anteriores e em recomendações de (AHMED; NORDIN, 2011) e (ALLRED, 2017). Primeiro, os valores máximos e mínimos dos pixels, que antes eram entre 0 e 255, foram ajustados para entre 0 e 1. Isso faz com que a imagem se torne completamente escura para a visão humana, o que é demonstrado a seguir com três imagens antes e três depois.

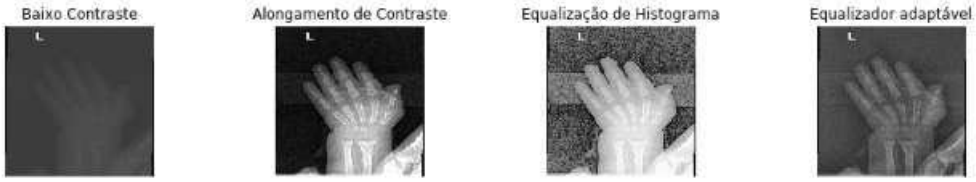
Figura 44 – Efeito de dividir os valores dos pixels por 255



Fonte: Autor

Então, foi feito o processamento das imagens usando alongamento de contraste, equalização de histograma, equalização adaptável e redução de contraste, para que se pudesse observar qual técnica seria mais adequada para ser utilizada na resolução do problema de classificação, baseado em (AHMED; NORDIN, 2011) e (ALLRED, 2017). Os resultados obtidos ao se aplicar essas técnicas em uma imagem (para exemplificar) do banco de dados são vistos na Figura 45

Figura 45 – Processamentos feitos nas imagens das radiografias



Fonte: Autor

Além dessa técnica, que visava amplificar as qualidades relevantes das imagens, também foi utilizada técnicas de geração de mais dados (data augmentation) ao girar e dar zoom nas imagens utilizando a função que já vem no Keras chamada de "Image Generator".

7.5 Escolha do projeto e de seus hiperparâmetros

Como já foi explicado em seções anteriores, os parâmetros (peso e viés) possuem seus valores ajustados durante o treino, no entanto, outros parâmetros devem ser definidos para que se tenha um resultado satisfatório. O processo de escolhas e implementação realizado está descrito a seguir.

7.5.1 Teste inicial

Para se ter noção se a tarefa seria difícil ou não, foi feita uma rede inicial com apenas uma camada convolucional, uma *flatten* e uma densa. Então, colocou-se essa rede para ser treinada com parte dos dados (cerca de mil imagens de treino). A estrutura da rede está mostrada na Figura 46.

Figura 46 – Camadas do teste inicial

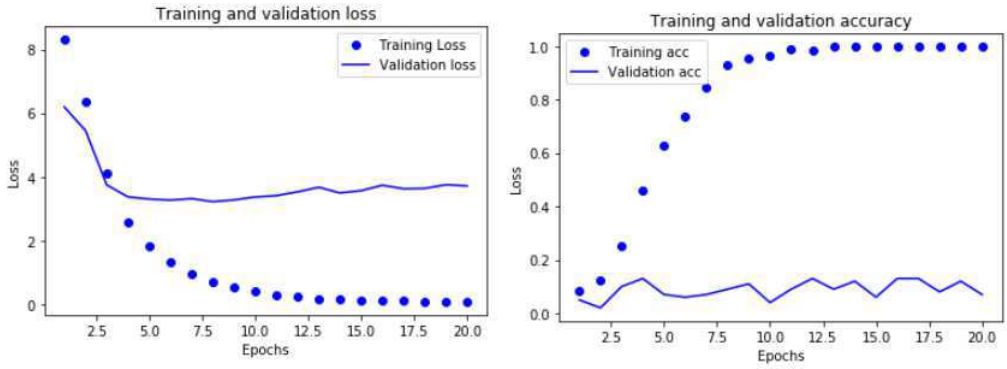
```
model.summary()

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 32)      896
_________________________________________________________________
flatten_1 (Flatten)          (None, 1605632)           0
_________________________________________________________________
dense_1 (Dense)              (None, 27)                43352091
=================================================================
Total params: 43,352,987
Trainable params: 43,352,987
Non-trainable params: 0
_________________________________________________________________
```

Fonte: Autor

Os resultados obtidos não foram satisfatórios, a rede tinha poucos dados e não estava mapeando (generalizando) corretamente, não sendo capaz de classificar corretamente qual seria o rótulo de uma nova entrada. Esse comportamento observado se caracterizava como *overfitting* e isso foi um sinal de que seria melhor utilizar todos os dados (para dificultar a tarefa da rede de "decorar") e que seria interessante usar outra arquitetura. Os resultados que geraram essas conclusões estão apresentados na Figura 47

Figura 47 – Resultados do primeiro teste



Fonte: Autor

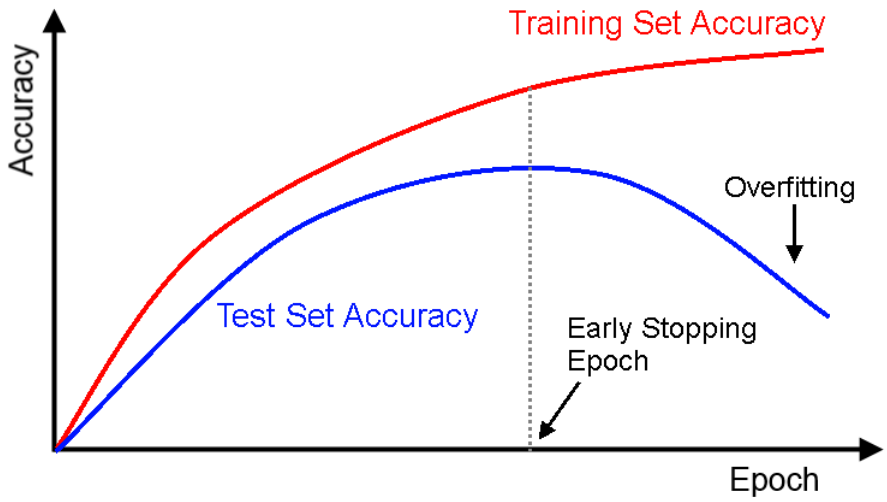
7.5.2 VGG16

Prosseguindo, foram feitos testes ao se modificar a arquitetura VGG16 (*fine-tuning*). Esses testes foram feitos inicialmente com parte dos dados do conjunto de treinamento e depois com todos os dados.

Os resultados obtidos ainda foram insatisfatórios. Então, tentou-se mudar outros hiperparâmetros, por exemplo, aumentando ou diminuindo o tamanho do *batch*, usando mais ou menos camadas densa no final, usando ou não *dropout*. Mesmo assim, nada do que foi feito teve resultados satisfatórios.

Early stopping é um algoritmo que visa evitar que a rede continue a sintonizar se não estiver tendo melhora suficiente no conjunto de validação e se a rede está mostrando um comportamento repetitivo tendendo a overfitting. A imagem a seguir ilustra onde esse ponto de parada seria.

Figura 48 – Early stopping



Fonte:(ACADEMY, 2018)

Sendo assim, ao observar a imagem abaixo, que mostra que ocorreu early stopping, nota-se que a rede não era eficiente para essa tarefa. Então, a partir desse ponto, foi decidido mudar de arquitetura.

Figura 49 – Observação dos resultados durante as épocas

```
394/394 [==============================] - 341s 867ms/step - loss: 2.6538 - acc: 0.1494 - mean_absolute_error: 0.0676 - val_l
oss: 2.7019 - val_acc: 0.1628 - val_mean_absolute_error: 0.0677
Epoch 13/50
394/394 [==============================] - 340s 862ms/step - loss: 2.6420 - acc: 0.1536 - mean_absolute_error: 0.0675 - val_l
oss: 2.6908 - val_acc: 0.1766 - val_mean_absolute_error: 0.0675
Epoch 14/50
394/394 [==============================] - 342s 868ms/step - loss: 2.6414 - acc: 0.1532 - mean_absolute_error: 0.0675 - val_l
oss: 2.7148 - val_acc: 0.1508 - val_mean_absolute_error: 0.0675
Epoch 15/50
394/394 [==============================] - 341s 866ms/step - loss: 2.6587 - acc: 0.1488 - mean_absolute_error: 0.0676 - val_l
oss: 2.6696 - val_acc: 0.1685 - val_mean_absolute_error: 0.0677
Epoch 16/50
394/394 [==============================] - 338s 858ms/step - loss: 2.6495 - acc: 0.1490 - mean_absolute_error: 0.0676 - val_l
oss: 2.7176 - val_acc: 0.1685 - val_mean_absolute_error: 0.0677
Epoch 17/50
394/394 [==============================] - 341s 865ms/step - loss: 2.6495 - acc: 0.1534 - mean_absolute_error: 0.0676 - val_l
oss: 2.6844 - val_acc: 0.1766 - val_mean_absolute_error: 0.0674
Epoch 00017: early stopping
```

Fonte: Autor

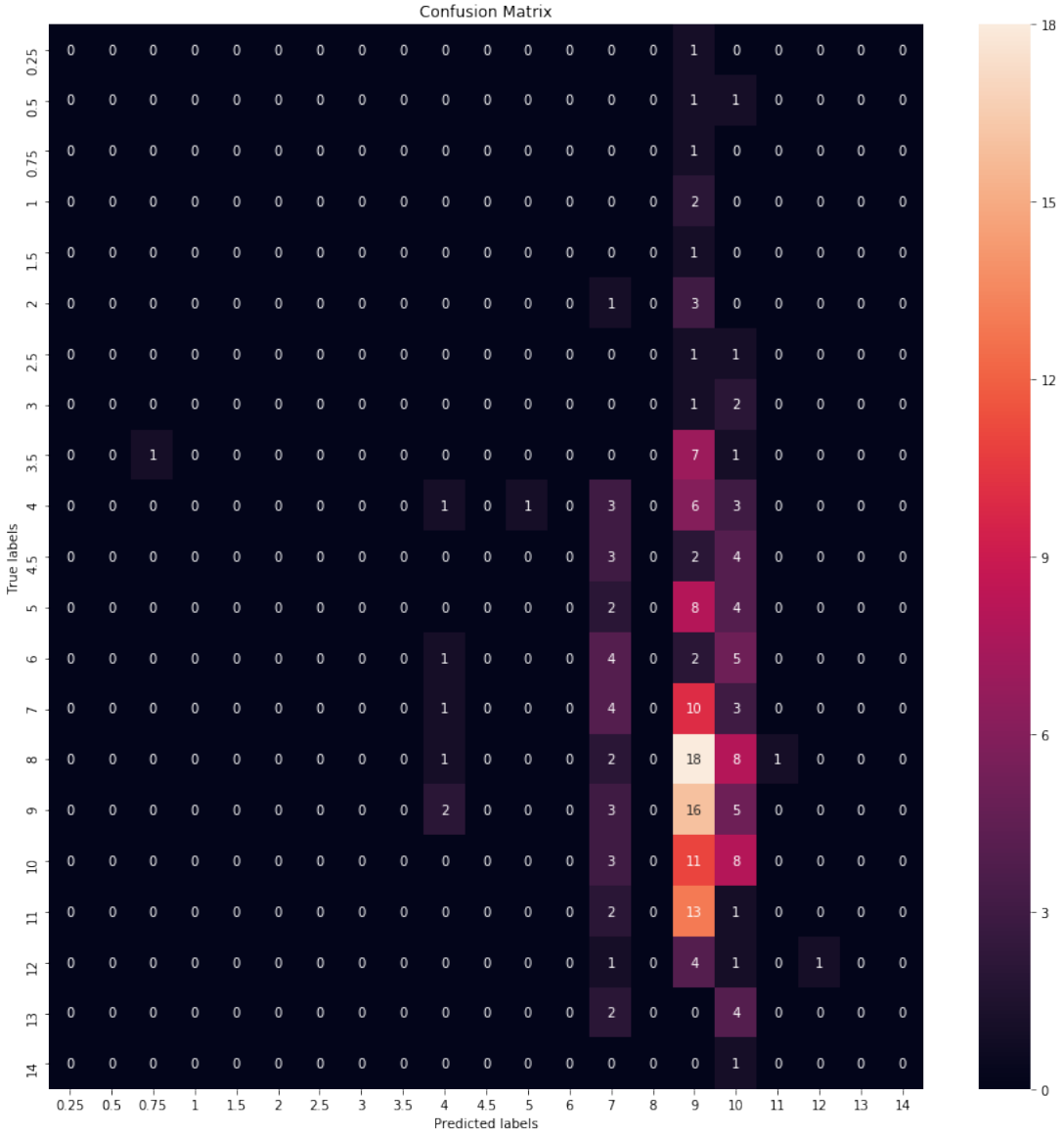
7.5.3 Inception V3

Foi decidido que seria utilizada a rede Inception V3 como ponto de partida dessa nova fase de testes, de maneira semelhante ao que o vencedor da competição (CICERO; BILBILY, 2017), de onde os dados foram tirados, tinha realizado. No entanto, essa rede tem dois grandes problemas. O primeiro é que ela tem concatenação, o que torna o seu "desmembramento"menos intuitivo e mais complicado de implementar em código.

Além disso, a entrada padrão dela é 299x299x3, o que fez com que o GPU onde esses testes estavam sendo realizados (*Geforce 1080*) não aguentasse alguns dos experimentos. Diversos outros problemas foram percebidos. Ao fim, a solução foi diminuir o tamanho do batch e usar o mínimo possível de modificações na arquitetura, para que se pudesse "congelar"o que já tinha de parâmetros sintonizados (da Inception V3) e só sintonizar mais alguns outros de camadas adicionadas.

Mesmo usando uma arquitetura potente como a Inception V3, fazendo testes usando ou não dropout, com mais ou menos camadas densas, com *batch* maior ou menor, com ou sem regularização L2, com ou sem preprocessamentos distintos (como o que o autor (CICERO; BILBILY, 2017) utilizou e o que foi mencionado anteriormente) o modelo ainda não obteve resultados satisfatório, o que é observado, por exemplo, na Figura 50, que foi gerada usando a biblioteca *Seaborn* e que mostra as predições feitas em relação aos valores testados usando *cross entropy*, *batch size* de 32, Adam (variação de gradiente descendente) e taxa de aprendizado de 10^{-3}.

Figura 50 – Mapa de calor das predições



Fonte: Autor

7.5.4 MobileNET

Por fim, em termos de implementação, ao observar os resultados anteriores, percebeu-se que talvez a melhor escolha para esse problema não tenha sido aplicada. E o baixo valor de MAE (média absoluta do erro) nos modelos desenvolvidos é um indicativo de que esse problema poderia ter sido melhor abordado usando regressão. Considerando isso, a diferença entre regressão e classificação é só o fato de que a última camada tem um nó (ao invés de vários) e este tem ativação linear, ou seja, ela gera um valor que é a predição, ao invés de selecionar entre escolhas de categorias (rótulos) possíveis.

Infelizmente, a ideia do trabalho era usar redes neurais convolucionais para clas-

sificação, mas, como não foi possível obter um resultado adequado com as estruturas utilizadas de rótulos e modelos escolhidos, foi decidido que valeria a pena pelo menos testar a implementação com regressão. Assim, procurou-se arquiteturas e implementações relacionadas e descobriu que há uma rede otimizada para aplicações com baixo poder computacional, como os computadores antigos e alguns dispositivos embarcados, e essa arquitetura se chama MobileNET. Então, foram testados scripts disponíveis no Kaggle (MADER, 2018b) e (MADER, 2018a). Os resultados iniciais usando regressão com a arquitetura MobileNET possuíam média absoluta do erro próxima de 12 meses, o que não foi considerado um desempenho aceitável.

Então, modificou-se (MADER, 2018a) usando os conceitos aplicados anteriormente, fazendo testes com mais camadas, até que se obtivesse *overfitting* e, finalmente, fazendo regularização utilizando *dropout* para que se tivesse, finalmente, um resultado satisfatório. A arquitetura final que foi utilizada está apresentada na Figura 51.

Figura 51 – Arquitetura modificada final

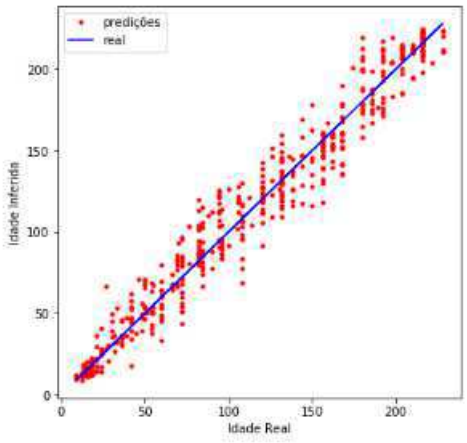
Layer (type)	Output Shape	Param #
batch_normalization_1 (Batch	(None, 512, 512, 3)	12
mobilenet_1.00_512 (Model)	(None, 16, 16, 1024)	3228864
batch_normalization_2 (Batch	(None, 16, 16, 1024)	4096
global_average_pooling2d_1 (	(None, 1024)	0
dense_1 (Dense)	(None, 1000)	1025000
dropout_1 (Dropout)	(None, 1000)	0
dense_2 (Dense)	(None, 1000)	1001000
dropout_2 (Dropout)	(None, 1000)	0
dense_3 (Dense)	(None, 1)	1001

Total params: 5,259,973
Trainable params: 5,236,031
Non-trainable params: 23,942

Fonte: Autor

Prosseguindo, observou-se que o conjunto de validação estava com média absoluta do erro menor que 9 meses e com custo próximo a 0,02. Nesse ponto, foi feito um teste com o conjunto de dados reais em comparação ao previsto (gerado) com regressão, para se observar a dispersão que havia entre um modelo ideal (reta azul) e o modelo obtido (pontos vermelhos), conforme a Figura 52 apresenta. Analisando os resultados, observou-se que a variação do valor real e do previsto estava dentro dos limites apresentados na Figura 3.

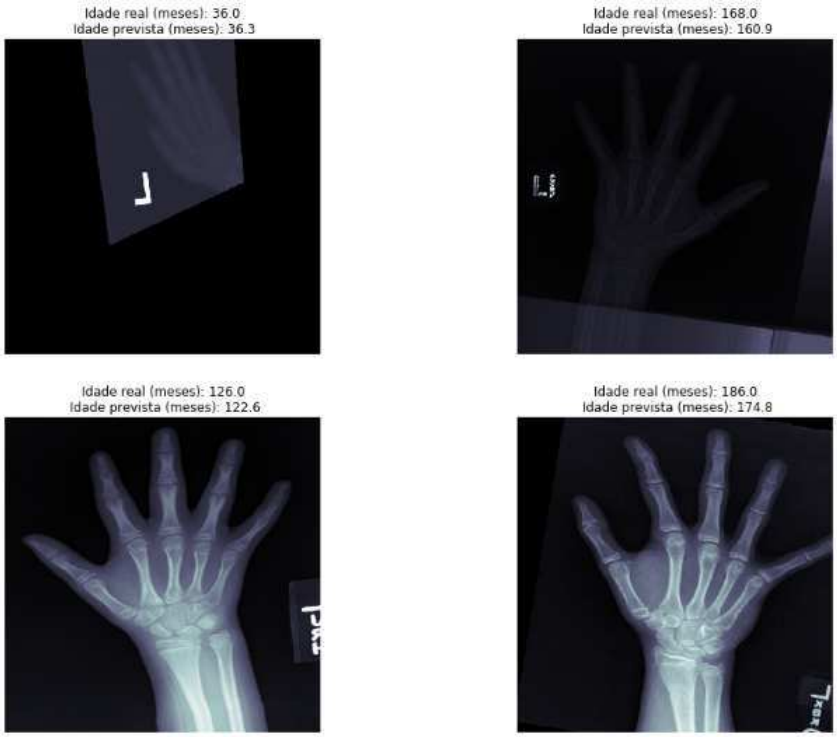
Figura 52 – Comparação ideal com resultado obtido



Fonte: Autor

Por fim, alguns testes observando as imagens reais do banco de dados e suas idades ósseas, junto das previsões obtidas pela rede, visando validar o procedimento adotado de maneira visual, são apresentadas na Figura 53.

Figura 53 – Resultados finais



Fonte: Autor

8 Considerações Finais

Apesar de no final a solução encontrada (regressão) tenha sido diferente da proposta inicialmente (classificação), esse trabalho foi relevante em termos pessoais, fornecendo uma base no assunto teórico e em diversos ferramentas (Pandas, Matplotlib, Seaborn, Keras) práticas ao aluno, e também em termos gerais, levando em conta que aqui está descrito o procedimento geral de escolha de hiperparâmetros (regularização), algumas metodologias de pré-processamento (usando *Image Generator* e *Imgaug*) e questões gerais relacionadas à abordagem de um problema de aprendizado de máquina.

Além disso, esse trabalho abre espaço para algumas sugestões e questionamentos, que são: quando um problema de regressão pode ser resolvido adequadamente com classificação? Quando não é possível? Como transitar entre esses mundos? Como solucionar problema de aprendizado de máquina com máquinas limitadas? Entre outras.

Outras possibilidades de desenvolvimento para continuação do trabalho são em termos de aplicação de conceitos mencionados aqui, como *autoencoders* e redes GAN para o pré-processamento de dados, o que já é proposto inclusive em alguns artigos (FRIDADAR et al., 2018), (SHEN; WU; SUK, 2017). Ou, ainda, pode-se substituir todo esse trabalho manual por algum algoritmo genético que otimize esses hiperparâmetros automaticamente (SUGANUMA; SHIRAKAWA; NAGAO, 2017). Outra ideia é adicionar mais rótulos (*labels*), simulando uma regressão, mais camadas, aumentar as dimensões das imagens e tentar fazer o processamento em computadores mais potentes.

Ainda, um dos motivos da classificação ter falhado em solucionar a tarefa adequadamente foi o desbalanceamento do banco de dados usado para treinamento. Isso pode ser resolvido utilizando *image augmentation* (geração de imagens) para que se forme um banco de dados com quantidades de imagens semelhantes em cada categoria de classificação definida.

Por fim, como sugestão de trabalho futuro, há a possibilidade de fazer um aplicativo usando o modelo final de regressão adotado, semelhante ao feito pelos vencedores da competição da RSNA (CICERO; BILBILY, 2017). E também se pode utilizar o conhecimento desenvolvido neste trabalho para resolver outros problemas de inteligência artificial da área médica, como geração automática de anotações (rótulos) e organização de bancos de dados antigos que possuam radiografias de mão e punho.

Referências

ACADEMY, D. S. *Usando Early Stopping Para Definir o Número de Épocas de Treinamento*. 2018. Disponível em: <http://deeplearningbook.com.br/usando-early-stopping-para-definir-o-numero-de-epocas-de-treinamento/>. Acesso em: 25 jun. 2019. Citado na página 53.

ACKLEY, D. H.; HINTON, G. E.; SEJNOWSKI, T. J. A learning algorithm for boltzmann machines. *Cognitive science*, Elsevier, v. 9, n. 1, p. 147–169, 1985. Citado na página 18.

AHMED, H. S.; NORDIN, M. J. Improving diagnostic viewing of medical images using enhancement algorithms. *Journal of Computer Science*, v. 7, n. 12, p. 1831, 2011. Citado na página 50.

ALLRED, R. *Image Augmentation for Deep Learning using Keras and Histogram Equalization*. 2017. Disponível em: <https://towardsdatascience.com/image-augmentation-for-deep-learning-using-keras-and-histogram-equalization-9329f6ae5085>. Acesso em: 25 jun. 2019. Citado na página 50.

AMABIS, J. M.; RODRIGUES, G. M. *Fundamentos da biologia moderna*. [S.l.]: Moderna, 2016. Citado 2 vezes nas páginas 2 e 20.

BROWNLEE, J. *A Tour of The Most Popular Machine Learning Algorithms*. 2013. Disponível em: <https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/>. Acesso em: 22 jun. 2019. Citado na página 16.

BULL, R. et al. Bone age assessment: a large scale comparison of the greulich and pyle, and tanner and whitehouse (tw2) methods. *Archives of disease in childhood*, BMJ Publishing Group Ltd, v. 81, n. 2, p. 172–173, 1999. Citado na página 1.

CAI, E. *Machine Learning Lesson of the Day – Overfitting and Underfitting*. 2014. Disponível em: <https://chemicalstatistician.wordpress.com/2014/03/19/machine-learning-lesson-of-the-day-overfitting-and-underfitting/>. Acesso em: 20 jun. 2019. Citado na página 29.

CHOLLET, F. *Deep Learning with Python*. [S.l.]: Manning Publications Co., 2018. Citado 4 vezes nas páginas 24, 25, 29 e 35.

CICERO, M.; BILBILY, A. *Machine Learning and the Future of Radiology: How we won the 2017 RSNA ML Challenge*. 2017. Disponível em: <https://www.16bit.ai/blog/ml-and-future-of-radiology>. Acesso em: 25 jun. 2019. Citado 4 vezes nas páginas 41, 53, 54 e 58.

DECHTER, R. *Learning while searching in constraint-satisfaction problems*. [S.l.]: University of California, Computer Science Department, Cognitive Systems ..., 1986. Citado na página 17.

FRID-ADAR, M. et al. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, Elsevier, v. 321, p. 321–331, 2018. Citado na página 58.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, Springer, v. 36, n. 4, p. 193–202, 1980. Citado na página 17.

GILSANZ, V.; RATIB, O. *Hand bone age: a digital atlas of skeletal maturity*. [S.l.]: Springer Science & Business Media, 2005. Citado na página 1.

GOODFELLOW, I. et al. Generative adversarial nets. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 2672–2680. Citado na página 18.

GREULICH, W.; PYLE, S. Radiographic atlas of skeletal development of the hand and wrist. *The American Journal of the Medical Sciences*, v. 238, n. 3, 1959. Citado 4 vezes nas páginas 1, 4, 5 e 46.

HINTON, G. E. et al. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. Citado 2 vezes nas páginas 18 e 32.

HSU, F.-h. Ibm's deep blue chess grandmaster chips. *IEEE Micro*, IEEE, v. 19, n. 2, p. 70–81, 1999. Citado na página 12.

IGLOVIKOV, V. I. et al. Paediatric bone age assessment using deep convolutional neural networks. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. [S.l.]: Springer, 2018. p. 300–308. Citado 2 vezes nas páginas 42 e 43.

KESKAR, N. S. et al. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. Citado na página 28.

KING, P. *How do artificial neural networks work?* 2016. Disponível em: <https://www.quora.com/How-do-artificial-neural-networks-work>. Acesso em: 20 jun. 2019. Citado na página 21.

KUPERMAN RAPHAEL LIBERATORE, A. S.-M. H. *Recomendações, atualizações em condultas de pediatria 38 - Idade óssea e distúrbios do crescimento*. 2007. Citado 2 vezes nas páginas 3 e 8.

LIGHTHILL, J. *Artificial Intelligence: A General Survey*. Report. 1973. Citado na página 11.

LIN, J. *Categorical Crossentropy VS Sparse Categorical Crossentropy*. 2018. Disponível em: <https://jovianlin.io/cat-crossentropy-vs-sparse-cat-crossentropy/>. Acesso em: 25 jun. 2019. Citado na página 31.

LONGUI, C. A. *Temas de Pediatria 61 - Determinação da Idade Óssea na avaliação de crescimento*. 1996. Citado 7 vezes nas páginas 2, 3, 5, 6, 7, 8 e 9.

MACHINERY, C. Computing machinery and intelligence-am turing. *Mind*, v. 59, n. 236, p. 433, 1950. Citado na página 11.

MADER, K. *MobileNet for Bone Age*. 2018. Disponível em: <https://www.kaggle.com/kmader/mobilenet-for-bone-age>. Acesso em: 25 jun. 2019. Citado na página 56.

MADER, K. *Pretrained-InceptionV3 for Bone Age*. 2018. Disponível em: <https://www.kaggle.com/kmader/pretrained-inceptionv3-for-bone-age>. Acesso em: 25 jun. 2019. Citado na página 56.

MAHAPATRA, S. *Why Deep Learning over Traditional Machine Learning?* 2018. Disponível em: <https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional//-machine-learning-1b6a99177063>. Acesso em: 22 jun. 2019. Citado na página 17.

MCCARTHY, J. et al. *Proposal for the 1956 Dartmouth Summer Research Project on Artificial Intelligence, Dartmouth College, Hanover, NH, USA*. 1955. Citado na página 10.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 17.

NIELSEN, M. *Improving the way neural networks learn*. 2018. Disponível em: <http://neuralnetworksanddeeplearning.com/chap3.html>. Acesso em: 25 jun. 2019. Citado na página 31.

PRABHU. *Understanding of Convolutional Neural Network (CNN) - Deep Learning*. 2018. Disponível em: <https://medium.com/@RaghavPrabhu/understanding-of-/convolutional-neural-network-cnn-deep-learning-99760835f148>. Acesso em: 20 jun. 2019. Citado 4 vezes nas páginas 29, 34, 36 e 37.

RADHAKRISHNAN, P. *What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?* 2017. Disponível em: <https://towardsdatascience.com/what-are-hyperparameters-and-/how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a>. Acesso em: 20 jun. 2019. Citado na página 21.

RAJ, B. *A Simple Guide to the Versions of the Inception Network*. 2018. Disponível em: <https://towardsdatascience.com/a-simple-guide-to-the-versions-of-/the-inception-network-7fc52b863202>. Acesso em: 25 jun. 2019. Citado na página 39.

REYNOSO, R. *A Complete History of Artificial Intelligence*. 2019. Disponível em: <https://learn.g2.com/history-of-artificial-intelligence#ai-1>. Acesso em: 23 jun. 2019. Citado na página 11.

ROSENBLATT, F. *The perceptron, a perceiving and recognizing automaton Project Para*. [S.l.]: Cornell Aeronautical Laboratory, 1957. Citado na página 17.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. *Learning Internal Representations by Error Propagation. David E. Rumelhart, James L. McClelland, and the PDP research group.(editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations*. [S.l.]: MIT Press, 1986. Citado na página 18.

SAMUEL, A. L. Programming computers to play games. In: *Advances in Computers*. [S.l.]: Elsevier, 1960. v. 1, p. 165–192. Citado na página 11.

SANDHU, S. *What is an inception layer?* 2017. Disponível em: <https://datascience.stackexchange.com/questions/14984/what-is-an-inception-layer>. Acesso em: 25 jun. 2019. Citado na página 38.

SARLE, W. *Improving the way neural networks learn*. 2002. Disponível em: <http://www.faqs.org/faqs/ai-faq/neural-nets/part2/>. Acesso em: 25 jun. 2019. Citado na página 32.

SEIF, G. *The 5 Clustering Algorithms Data Scientists Need to Know*. 2018. Disponível em: <https://towardsdatascience.com/the-5-clustering-algorithms/ /-data-scientists-need-to-know-a36d136ef68>. Acesso em: 20 jun. 2019. Citado na página 24.

SHANNON, C. E. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Taylor & Francis, v. 41, n. 314, p. 256–275, 1950. Citado na página 10.

SHEN, D.; WU, G.; SUK, H.-I. Deep learning in medical image analysis. *Annual review of biomedical engineering*, Annual Reviews, v. 19, p. 221–248, 2017. Citado na página 58.

SHIMAO. *When to "add" layers and when to "concatenate" in neural networks?* 2018. Disponível em: <https://stats.stackexchange.com/questions/361018/when-to-add-layers-/ /and-when-to-concatenate-in-neural-networks>. Acesso em: 25 jun. 2019. Citado na página 37.

SUGANUMA, M.; SHIRAKAWA, S.; NAGAO, T. A genetic programming approach to designing convolutional neural network architectures. In: ACM. *Proceedings of the Genetic and Evolutionary Computation Conference*. [S.l.], 2017. p. 497–504. Citado na página 58.

SYNCED, S. *A Review of AI History Beyond Deep Learning*. 2018. Disponível em: <https://medium.com/syncedreview/ intro-a-review-of-ai-history-beyond-deep-learning-f166ff8b4b2f>. Acesso em: 24 jun. 2019. Citado na página 13.

TANNER, J. Assessement of skeletal maturity and predicting of adult height (tw2 method). *Prediction of adult height*, Academic Press, p. 22–37, 1983. Citado na página 5.

TOKUNAGA, A. P. P. *Métodos de Greulich e Pyle e Eklof e Ringertz na estimativa da idade óssea*. 2013. Citado na página 2.

VANIAN, J. *Here's How Much Companies Are Spending on Artificial Intelligence: Eye on A.I.* 2019. Disponível em: <https://fortune.com/2019/06/18/ business-leaders-artificial-intelligence/>. Acesso em: 24 jun. 2019. Citado na página 12.

VáZQUEZ, F. *A "weird" introduction to Deep Learning*. 2018. Disponível em: <https: //towardsdatascience.com/a-weird-introduction-to-deep-learning-7828803693b0>. Acesso em: 25 jun. 2019. Citado na página 19.

WERBOS, P. Beyond regression: New tools for prediction and analysis in the behavioral sciences//phd thesis, dept. of applied mathematics. harvard university, cambridge, ma., 1974. 1974. Citado na página 17.

WERTHEIMER, M. Gestalt theory. Kegan Paul, Trench, Trubner & Company, 1938. Citado na página 18.

YUAN, R. *How does model.fit () calculate loss and acc ? Documentation will be helpful.* 2018. Disponível em: <https://github.com/keras-team/keras/issues/10426>. Acesso em: 20 jun. 2019. Citado na página 31.

ZHU, J.-Y. et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. [S.l.: s.n.], 2017. Citado na página 18.