

NICOLI PINHEIRO DE ARAUJO

ESTIMAÇÃO INTELIGENTE DE IDADE UTILIZANDO *DEEP LEARNING*

Trabalho de Conclusão de Curso apresentado
à banca avaliadora do Curso de Engenharia
de Computação, da Escola Superior de
Tecnologia, da Universidade do Estado do
Amazonas, como pré-requisito para obtenção
do título de Engenheira de Computação.

Orientador(a): Profa. Dra. Elloá Barreto Guedes da Costa

Manaus – Novembro – 2018

Universidade do Estado do Amazonas - UEA

Escola Superior de Tecnologia - EST

Reitor:

Cleinaldo de Almeida Costa

Vice-Reitor:

Cleto Cavalcante de Souza Leal

Diretor da Escola Superior de Tecnologia:

Roberto Higino Pereira Da Silva

Coordenador do Curso de Engenharia de Computação:

Salvador Ramos Bernardino Da Silva

Coordenador da Disciplina Projeto Final:

Marcia Sampaio Lima

Banca Avaliadora composta por:

Data da Defesa: 30/11/2018.

Profa. Dra. Elloá Barreto Guedes da Costa (Orientadora)

Prof. M.Sc. Mario Augusto Bessa De Figueiredo

Prof. Dr. Carlos Mauricio Serodio Figueiredo

CIP – Catalogação na Publicação

L864a

ARAUJO, Nicoli Pinheiro de

Estimação Inteligente de Idade Utilizando *Deep Learning* / Nicoli Araujo; [orientada por] Profa. Dra. Elloá Barreto Guedes da Costa – Manaus: UEA, 2018.

240 p.; il.; 30cm

Inclui Bibliografia

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação).

Universidade do Estado do Amazonas, 2018.

CDU: _____

NICOLI PINHEIRO DE ARAUJO

**ESTIMAÇÃO INTELIGENTE DE IDADE DE TELESPECTADORES PARA
APLICAÇÕES DE SUGESTÃO DE CONTEÚDO EM SMART TVs**

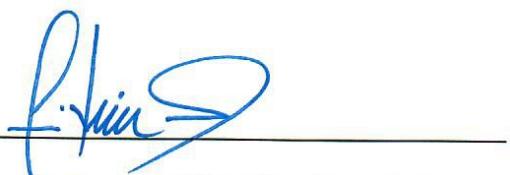
Trabalho de Conclusão de Curso apresentado
à banca avaliadora do Curso de Engenharia
de Computação, da Escola Superior de
Tecnologia, da Universidade do Estado do
Amazonas, como pré-requisito para obtenção
do título de Engenheiro de Computação.

BANCA EXAMINADORA

Aprovado em: 30/11/2018


Profa. Dra. Elloá Barreto Guedes da Costa

UNIVERSIDADE DO ESTADO DO AMAZONAS



Prof. Mario Augusto Bessa De Figueiredo, M.Sc.

UNIVERSIDADE DO ESTADO DO AMAZONAS



Prof. Carlos Mauricio Serodio Figueiredo, D.Sc.

UNIVERSIDADE DO ESTADO DO AMAZONAS

Resumo

Este trabalho apresenta uma proposta para estimação de idade de indivíduos utilizando técnicas de *deep learning*. Tal ferramenta pode ser utilizada de diversas maneiras. Em *Smart TVs*, por exemplo, pode facilitar a coleta de informações que contribuam para melhor experiência de provimento de conteúdo, criação e controle de configurações personalizadas e para a implementação de estratégias de controle parental mais eficientes. Neste trabalho, a estimação de idade de indivíduos a partir de imagens de face é tratada como tarefa de regressão. Para endereçá-la, são conduzidas nove abordagens de treino e teste de diferentes arquiteturas de redes neurais convolucionais, submetidas a variações de parâmetros e hiperparâmetros. Apesar de não ser possível constatar uma melhora progressiva, o trabalho foi conduzido conforme práticas de natureza heurística sugeridas por diversos autores da literatura. Os resultados obtidos mostraram que a melhor rede neural convolucional para esta tarefa foi LeNet cujas imagens de entrada foram submetidas a técnicas de *data augmentation* e equalização por histograma, utilizando funções de ativação *ReLU*. Este modelo obteve um MAE de 9.98 anos. Como trabalhos futuros considera-se a abordagem deste problema como uma tarefa de classificação, além de análise dos grupos de idade com maior e menor índice de erro, na tentativa de obter estimadores mais apropriados. Continuar enderezando este problema é de suma importância do ponto de vista prático para o desenvolvimento de diversas soluções de personalização automática de conteúdo.

Palavras Chave: Estimação de Idade, Redes Neurais Convolucionais, *Deep Learning*.

Abstract

This work presents a proposal for estimating the age of individuals using deep learning techniques. Such tool can be used in many ways. In smart TVs, for example, it can ease the collection of informations that can contribute to a better content delivery experience, to create and control custom settings, and to implement more efficient parental control strategies. In this work, the estimation of age of individuals using images of their faces is approached as a regression task. In order to address the referred task, nine approaches are conducted for training and testing different architectures of convolutional neural networks, which are submitted to parameters and hyperparameters variations. Although it is not possible to observe a progressive improvement, this work was conducted according to heuristic practices suggested by several authors from literature, considering models compatible with the state of the art. Obtained results showed that the best convolutional neural network for this task was LeNet which input images were submitted to data augmentation and histogram equalization techniques, using ReLU activation function. This model obtained a MAE of 9.98. As future work, it is considered to address this problem as a classification task, as well as a more detailed analysis of the age groups with the highest and lowest error rates, in an attempt to obtain more appropriate estimators considering the characteristics of each group. Further addressing this problem is of paramount importance for the development of several solutions envolving the provision of automatically customized content.

Keywords: *Age Estimation, Convolutional Neural Networks, Deep Learning.*

Agradecimentos

Agradeço primeiramente à minha mãe, Francisca Dones Nicácio Pinheiro, por ter me incentivado durante toda a vida a buscar uma formação que traga realização profissional. Agradeço pelo apoio incondicional dado à minha educação, da maneira que apenas pais dedicados conseguem fazer. Reconheço também o esforço pela conscientização e motivação pela busca por uma realidade melhor, tanto financeira quanto intelectualmente. Apesar de eu não ser uma pessoa religiosa, agradeço a minha mãe por orar por mim, gesto de carinho sempre bem vindo.

Agradeço também aos outros membros da minha família, em especial minha irmã Vitória Gabriela Pinheiro de Araujo e meu pai Valdenor Oliveria de Araujo, ambos fundamentais para a estabilidade psicológica e financeira necessária para finalizar uma graduação com turno integral.

Agradeço à Profa. Dra. Elloá B. Guedes, que em 2014 aceitou me direcionar durante a monitoria de Algoritmos e Programação I, e tem me orientado desde então. Grande mentora, agradeço por todas as lições ensinadas, desde as mais complexas, ministradas durante as aulas da antiga Fundamentos de Engenharia de Computação II até as mais triviais. Peço perdão por não ser capaz de aprender todas, e tempo para aprender mais algumas.

Agradeço aos meus colegas de curso, incontáveis, que me ajudaram em diversos momentos, sob diversas circunstâncias, a persistir em busca da graduação. Em especial, agradeço a Giovana Oliveira de Lucca e Janderson do Nascimento Lira, fundamentais defensores dos interesses dos alunos durante a troca das grades curriculares do curso de Engenharia de Computação. Agradecimentos especiais ao DNCG, a caminhada teria sido bem mais dura sem vocês, obrigada

por estarem lá em todos os momentos, pelo apoio, por me ajudarem a ser um ser humano e uma profissional melhor.

Agradeço ao Núcleo de Computação e a todos os professores e coordenadores que proporcionaram imenso aprendizado durante a graduação. Agradeço também a Universidade do Estado do Amazonas, seus servidores, prestadores de serviço, alunos, e outros envolvidos no funcionamento desta instituição de ensino, por toda a infraestrutura fornecida, desde os processos de seleção até o fim da graduação. Agradeço também ao Governo do Estado do Amazonas manter a universidade em condição útil à sociedade durante o tempo da minha formação.

Agradeço ao setor de Pesquisa e Desenvolvimento da Envision Ind. de Prod, Eletrônicos LTDA. por oferecerem um ambiente de convívio profissional em que os conceitos aprendidos em sala de aula e laboratório são efetivamente usados para endereçar problemas práticos e desenvolver soluções para produtos. Obrigada por dar o apoio profissional necessário para a minha graduação. Agradeço especialmente a Ruan Belém por incentivar a abordagem do tema central deste trabalho e por demonstrar suas aplicações práticas e relevância.

Agradeço à Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) que, por meio do Projeto PROTI Pesquisa 11/2017, colaborou para a consolidação da infraestrutura física e tecnológica do Laboratório de Sistemas Inteligentes da Escola Superior de Tecnologia da Universidade do Estado do Amazonas. Este trabalho de conclusão de curso é um dos produtos deste projeto, pois foi desenvolvido no referido laboratório, fez uso dos recursos computacionais ali disponíveis e foi melhorado graças às discussões e interações com o grupo de pesquisa nele sediado.

Ser util é ter poder.

Rupi Kaur

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

add mais
aplicações

A estimação automática de idade através de fotografias faciais tem diversas aplicações.

No caso de *Smart TVs*, é essencial que estes aparelhos sejam capazes de capturar o perfil e o interesse dos seus telespectadores a fim de oferecer uma experiência mais rica. A recomendação de conteúdo, por exemplo, pode levar em conta características individuais, tais como idade e gênero. Porém, se fornecidos de maneira habitual, via preenchimento de formulários, além de ser uma tarefa massante, pode não refletir de maneira realística o perfil individual dos vários usuários que podem estar à frente de uma *Smart TV* em um determinado momento. *Smart TVs* possuem câmeras que podem ser habilitadas para aquisição de imagens daqueles que estão à frente do televisor, respeitadas as preferências de privacidade de cada usuário. É possível usá-las como entrada para sistemas inteligentes de identificação de características, cujas previsões podem ser usadas, por exemplo, para recomendação de conteúdo. No caso da idade, em particular, é possível usar estas informações para realizar um controle parental mais eficiente, protegendo crianças e adolescentes de conteúdos inadequados à sua faixa etária.(??).

Diante do que foi exposto, este trabalho de conclusão de curso considera o desenvolvimento de estratégias inteligentes, baseadas na utilização de técnicas de *Deep Learning*, para estimação da idade de telespectadores a partir de fotografias faciais. Embora a estimação de outras características também pudesse ser realizada mediante a análise de fotografias faciais, desde gênero até a presença de doenças, optou-se pela idade por ser um atributo comum a todos

os indivíduos, pelo potencial de aplicações, pela existência de bases de dados adequadamente rotuladas com este atributo e pelo menor potencial de infringência das searas privadas dos usuários.

1.1 Objetivos

O objetivo geral deste trabalho consistiu em aplicar técnicas de *Deep Learning* para estimação de idade de telespectadores de *Smart TVs* a partir de suas respectivas fotografias faciais. Para alcançar esta meta, alguns objetivos específicos precisaram ser contemplados, a citar:

1. Formular um referencial teórico sobre redes neurais convolucionais, contemplando seu arcabouço matemático, suas características, principais arquiteturas, métodos de treinamento e teste;
2. Consolidar uma base de dados com exemplos realísticos para treinamento dos modelos, tendo em vista a captura de padrões representativos ao domínio do problema;
3. Identificar tecnologias adequadas para implementação dos estimadores;
4. Propor, treinar e testar diferentes estimadores de idade baseados em redes neurais convolucionais para a tarefa em questão;
5. Avaliar comparativamente os estimadores propostos.

1.2 Justificativa

A realização de um trabalho de conclusão de curso desta natureza é justificada por várias razões. No contexto da interação entre telespectador e *Smart TV*, um estimador de idade pode ser utilizado para facilitar a coleta de informações que contribuam para melhor experiência de provimento de conteúdo e de configurações personalizadas. Em particular, a estimação de idade dos telespectadores pode ser especialmente empregada na implementação de um controle

parental mais eficiente, protegendo crianças e adolescentes de conteúdos inadequados à sua faixa etária.

Um outro aspecto que ressalta a importância da realização de um trabalho desta natureza é a prática e a proposição de soluções envolvendo *Machine Learning*. Esta é uma área de vanguarda na Computação e seu potencial para resolução de problemas práticos está em franco desenvolvimento. Ao considerar a elaboração do estimador proposto, será necessário dominar conhecimentos de ferramental tecnológico atual, o que pode colaborar na minimização da distância entre o profissional em formação e os anseios do mercado de trabalho da área.

Por fim, há que se mencionar a relação entre a área de pesquisa considerada neste trabalho de conclusão de curso e o Laboratório de Sistemas Inteligentes (LSI). Este trabalho alinhou-se com os objetivos desta iniciativa do Núcleo de Computação (NUCOMP), motivando o desenvolvimento de uma solução inovadora que utiliza técnicas da Inteligência Artificial.

1.3 Metodologia

A metodologia para o desenvolvimento deste trabalho consistiu na realização da *fundamentação teórica sobre Machine Learning*, em especial contemplando os conceitos relativos às redes neurais convolucionais. Para tanto, considerou-se a literatura desta área para que haja o entendimento das bases matemáticas deste modelo computacional, como funcionam, quais as características e as arquiteturas mais importantes. Neste estudo, além dos aspectos teóricos, foram considerados os ambientes de desenvolvimento, bibliotecas e outras tecnologias para implementação dos conceitos contemplados.

Os demais passos que compõem a metodologia deste trabalho baseiam-se no *fluxo de atividades de machine learning* (??). Inicialmente, houve a aquisição e o pré-processamento de imagens para *consolidar uma base de dados* para esta tarefa de aprendizado. Nesta etapa, foi considerada a literatura e uma base de dados já disponível e apropriadamente anotada, com licença livre de utilização.

A seguir, houve a *proposição de diferentes modelos de redes neurais convolucionais* para a tarefa de aprendizado considerada. Nesta etapa, foram elencados diferentes parâmetros e hiper-

add mais
aplicações
aq

parâmetros de configuração, bem como arquiteturas. Estes procedimentos visaram consolidar um espaço de busca de modelos que possam endereçar a tarefa de maneira mais eficiente.

O próximo estágio consistiu no *treinamento das redes neurais convolucionais* para o problema em questão. Durante este processo, uma parte da base de dados foi apresentada aos modelos para que houvesse o ajuste de pesos, compreendendo o aprendizado das características relevantes. O treinamento das redes ocorreu utilizando computação em nuvem e computadores disponíveis no Laboratório de Sistemas Inteligentes (LSI), tendo em vista a infra-estrutura de hardware necessária para realizar este procedimento.

Seguiu-se então o *teste das redes*, respeitando uma abordagem de validação cruzada e utilizando métricas de desempenho apropriadas. O objetivo desta fase consistiu em aferir os modelos propostos e treinados quanto à sua capacidade de generalização.

Por fim, para identificação de um modelo mais adequado à esta tarefa, as *métricas de desempenho foram comparadas* e os melhores modelos elencados a partir destes valores, apontando assim um estimador apropriado para o problema inicialmente considerado.

Alem destas atividades, há que se considerar a escrita da proposta e do projeto final do trabalho de conclusão de curso, bem como as defesas parcial e final.

1.4 Cronograma

O cronograma de realização das atividades pode ser visto na Tabela ???. As atividades listadas possuem relação com a metodologia detalhada na seção anterior, compreendendo os requisitos elementares para a realização deste trabalho.

1.5 Organização do Documento

Para a apresentação desta proposta de trabalho de conclusão de curso, o presente documento está organizado como segue. Inicialmente, uma fundamendação teórica pode ser vista na Seção ???. Uma análise dos trabalhos relacionados encontra-se na Seção ???. Na Seção ???. detalha-

Tabela 1.1: Cronograma de atividades levando em consideração os dez meses (de 02/2018 a 12/2018) para a realização do TCC.

	2018										
	02	03	04	05	06	07	08	09	10	11	12
Escrita da Proposta	X	X	X	X	X						
Fundamentação Teórica sobre Machine Learning	X	X	X	X							
Consolidação da Base de Dados			X	X							
Proposição de Modelos de Redes Neurais Convolucionais					X	X	X	X	X		
Defesa da Proposta						X					
Escrita do Trabalho Final							X	X	X	X	X
Treinamento das Redes Neurais Convolucionais						X	X	X	X	X	
Teste das Redes Neurais Convolucionais							X	X	X	X	X
Comparação de Métricas de Desempenho							X	X	X	X	X
Defesa do Trabalho Final											X

se uma solução proposta para a tarefa endereçada. Na Seção ?? estão os resultados obtidos. Finalmente, as considerações finais podem ser encontradas na Seção ??.

Capítulo 2

Fundamentação Teórica

A fundamentação teórica para a realização deste trabalho compreende conceitos ligados ao *Machine Learning*. Assim, a Seção ?? comprehende os conceitos essenciais de *Machine Learning*, em que as redes neurais são particularmente detalhadas na Seção ???. Os conceitos mais emergentes desta área, envolvendo *Deep Learning*, são descritos na Seção ???. As tecnologias utilizadas para o desenvolvimento deste trabalho estão na ??.

2.1 *Machine Learning*

Machine Learning (ML), também chamado de Aprendizado de Máquina, é uma subárea da Inteligência Artificial que trata do estudo sistemático de algoritmos e sistemas que são capazes de melhorar seu desempenho com a experiência. Um algoritmo neste domínio é capaz de aprender a partir de dados, capturando padrões e efetuando inferências. Estes algoritmos podem ser entendidos em uma analogia com humanos e outros animais que, ao se depararem com determinada situação, costumam procurar lembranças de situações similares, de como agiram, e se o comportamento adotado foi vantajoso, e deve ser repetido, ou prejudicial, devendo ser evitado (??????).

Para consolidar o aprendizado, os algoritmos de ML precisam passar por um processo de aquisição da experiência, comumente chamado de treinamento. De acordo Mitchell (??), um algoritmo que aprende a partir da experiência E quanto a um conjunto de tarefas T e medida de

performance P , se sua performance nas tarefas em T , medida por P , melhora com a experiência E .

Ao preparar um algoritmo de ML para desenvolver determinada tarefa, busca-se um modelo, ou seja, uma função, que mapeie as instâncias do espaço de entrada para o de saída (??). Estes modelos podem ser agrupados em diferentes categorias ao se considerar o tipo de aprendizado e a saída desejada para o algoritmo. A Figura ?? apresenta uma visão geral do estado da arte acerca dos modelos de ML e suas subdivisões.

Quanto ao tipo de aprendizado, as tarefas de ML podem ser agrupadas em três tipos diferentes, a depender da presença e do tipo de resposta dada ao algoritmo quanto ao desempenho de suas saídas. No *aprendizado supervisionado*, o algoritmo deve aprender a inferir valores a partir de atributos preditores e do respectivo atributo alvo fornecido como exemplo, ou seja, de cenários em que os valores de saída são conhecidos. Os modelos mais frequentemente utilizados neste tipo de aprendizado são as máquinas de vetores de suporte, redes neurais artificiais *feedforward*, regressão linear e logística, etc. Já no *aprendizado não-supervisionado*, o algoritmo deve inferir padrões e estruturas a partir de dados não rotulados, buscando alguma estrutura interna que os caracterize. Exemplos de modelos aplicáveis a este cenário são os algoritmos *k-means* e *k-medoids*. Por fim, no *aprendizado por reforço* o algoritmo não recebe dados nem tampouco rótulos, e deve aprender a partir das recompensas positivas ou negativas dadas por ações que modifiquem o ambiente de maneira satisfatória ou não (??).

Quanto ao tipo de saída desejada, os problemas que podem ser endereçados segundo ML são de classificação, regressão, transcrição, tradução automática, detecção de anomalia, síntese e amostragem. No caso do aprendizado supervisionado, em particular, as principais tarefas realizadas são de classificação e regressão (??).

Um algoritmo proposto a uma tarefa de classificação deve especificar cada entrada x como pertencente a uma dentre k categorias pré-determinadas, produzindo uma saída $y = f(x)$ tal que a função f é definida como $f : \mathbb{R}^n \rightarrow \{1, \dots, k\}$, ou seja, f mapeia sequências de números reais x de dimensão n para um valor inteiro y dentre k possibilidades (??).

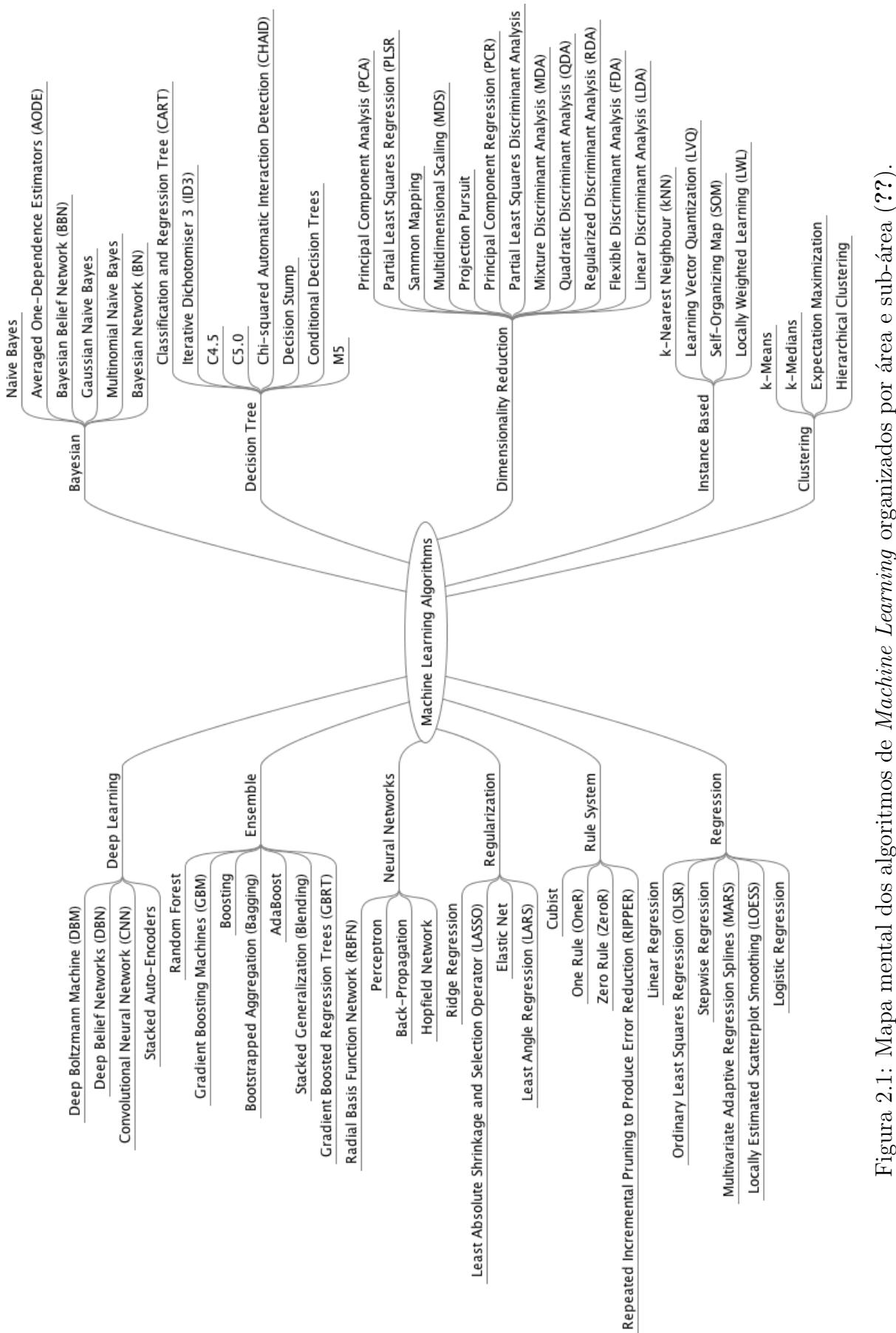


Figura 2.1: Mapa mental dos algoritmos de *Machine Learning* organizados por área e sub-área (??).

Dentre as tarefas de classificação estão, por exemplo, o reconhecimento de objetos em uma imagem, determinar se um indivíduo será ou não vítima de determinada doença, se sobreviverá ou não a determinado acidente, etc.

Numa tarefa de regressão, por sua vez, objetiva-se aprender uma função de valor real a partir de uma entrada (??). Assim, a saída $y = f(x)$ é dada pela função $f : \mathbb{R}^n \rightarrow \mathbb{R}$, ou seja, f mapeia uma entrada multidimensional x para um valor y real (??). Algumas tarefas de regressão envolvem a previsão de preços de um mercado de ações, a determinação do risco do seguro para um carro, do volume diário de precipitação em determinada cidade, entre outros.

Os modelos de ML são organizados em dois grandes grupos, dos tipos paramétricos ou não paramétricos. Segundo Russel e Norvig, um modelo de aprendizado que resume dados utilizando um conjunto de parâmetros de tamanho definidos independente do número de exemplos de treinamento é chamado de *modelo paramétrico*. Dentre os modelos paramétricos está a regressão logística. Já um *modelo não-paramétrico* é aquele que não pode ser caracterizado por um conjunto limitado de parâmetros. Alguns exemplos de modelos não-paramétricos são máquinas de vetores de suporte, redes neurais artificiais, k vizinhos mais próximos e árvores de decisão CART e C4.5 (??).

Dentre os modelos não paramétricos, as redes neurais artificiais têm demonstrado resultados satisfatórios em tarefas de classificação e regressão quando aplicadas em diversas áreas. Em especial, aplicações de *Deep Learning* no reconhecimento de objetos e no processamento de linguagem natural, por exemplo, têm trazido ainda mais atenção a este modelo. Diante desta importância e da utilização no contexto deste trabalho, estes conceitos serão abordados com mais profundidade nas seções a seguir.

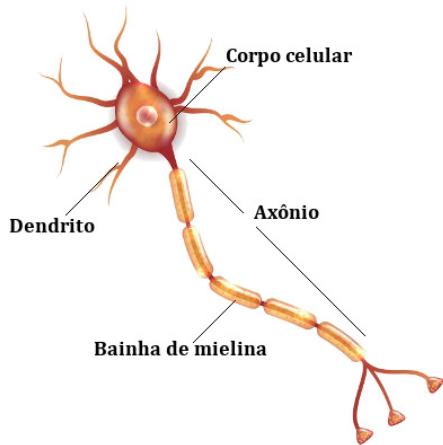
2.2 Redes Neurais Artificiais

As *Redes Neurais Artificiais* (RNAs) são um modelo de computação caracterizado por sistemas que, em algum nível, lembram a estrutura do cérebro humano. São sistemas paralelos e distribuídos, compostos por unidades de processamento simples, os *neurônios artificiais*, que calculam funções matemáticas, normalmente não-lineares. Estes neurônios são dispostos em

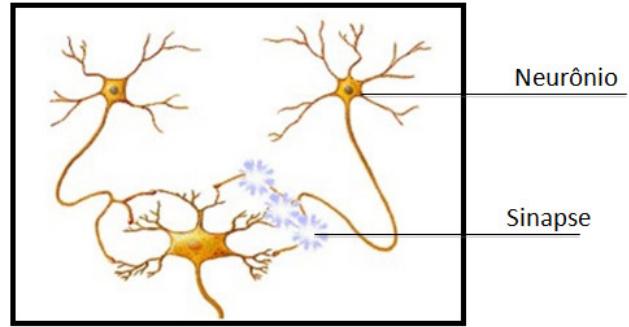
uma ou mais camadas e interligados por um grande número de conexões normalmente unidireccionais e comumente associadas a pesos, que armazenam o conhecimento representado no modelo e ponderam a entrada recebida por cada neurônio da rede. Os principais atrativos das RNAs envolvem a capacidade de capturar tendências a partir de um conjunto de exemplos e dar respostas coerentes para dados não-conhecidos, ou seja, de generalizar a informação aprendida (??).

Figura 2.2: Redes neurais biológicas.

(a) Neurônio biológico e seus componentes. Fonte:
(??)



(b) Sinapse entre neurônios. Fonte: (??)

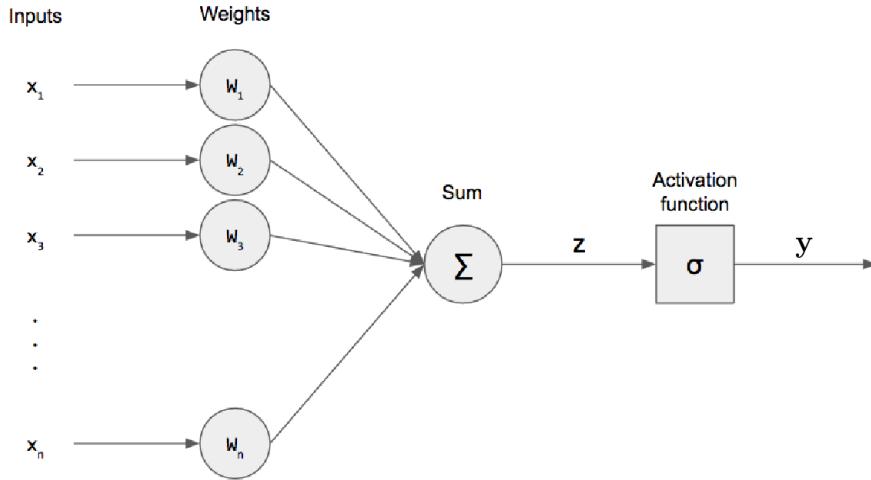


A motivação para a criação deste modelo vem do funcionamento do cérebro biológico, que é formado por neurônios interligados e que se comunicam entre si de modo contínuo e paralelo através de impulsos nervosos. Esta complexa rede neural biológica é capaz de reconhecer padrões e relacioná-los, produzir emoções, pensamentos, percepção e cognição. Cada neurônio biológico é composto de um corpo, dendritos e um axônio, como ilustrado na Figura ???. Os dendritos são responsáveis pela recepção de impulsos nervosos vindos de outros neurônios; o corpo combina os sinais recebidos pelos dendritos e caso o resultado ultrapasse determinado limiar de excitação do neurônio, são gerados novos impulsos nervosos, que são transmitidos pelo axônio até os dendritos dos neurônios seguintes. Esta conexão unilateral entre neurônios biológicos, denominada sinapse, encontra-se ilustrada na Figura ??.

Com base no modelo biológico, McCulloch e Pitts propuseram em (??) um neurônio artificial. Como mostrado na Figura ??, o modelo de McCulloch e Pitts de neurônio artificial

contém n terminais de entrada, denotados por $x = x_1, \dots, x_n$, e um terminal de saída y . Esta organização faz uma alusão aos dendritos, corpo celular e axônio de um neurônio biológico.

Figura 2.3: Representação de um neurônio artificial. Fonte: (??)



A saída y é o resultado do mapeamento da função de ativação σ , conforme expresso na Equação ??, aplicada à soma ponderada do vetor de entrada x pelo conjunto de pesos $w = w_1, \dots, w_n$. No caso de um neurônio mais simples, como o de McCulloch e Pitts, a ativação do neurônio é obtida através da aplicação de uma função degrau deslocada, a depender da comparação da entrada com um limiar de ativação θ , conforme Equação ?? (??).

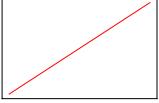
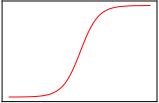
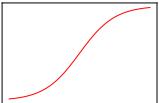
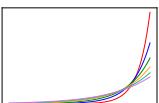
$$y = \sigma(z), \quad (2.1)$$

$$z = \sum_{i=1}^n x_i w_i \quad (2.2)$$

$$\sigma(z) = \begin{cases} 1, & \text{se } z \geq \theta \\ 0, & \text{caso contrário.} \end{cases} \quad (2.3)$$

Considerando desenvolvimentos posteriores, sabe-se atualmente que a escolha da função de ativação σ das camadas ocultas e da camada de saída deve considerar funções contínuas e deriváveis (??), em que comumente são optadas pelas funções apresentadas na Tabela ??.

Tabela 2.1: Exemplos de funções de ativação (??)

Nome	Gráfico	Equação	Intervalo
Identidade ou Linear		$\sigma(z) = z$	$(-\infty, +\infty)$
Tangente Hiperbólica		$\sigma(z) = \tanh(z) = \frac{(e^z - e^{-z})}{(e^z + e^{-z})}$	$(-1, 1)$
Sigmoide ou Logística		$\sigma(z) = \frac{1}{1 + e^{-x}}$	$(0, 1)$
Unidade Linear Retificada		$\sigma(z) = \max(0, z)$	$[0, \infty)$
Softmax		$g(z_j) = \frac{e^{z_j}}{\sum_{j=1}^K e^{z_k}} \quad j = 1, \dots, K$	$(-\infty, \infty)$

Em 1958, Frank Rosenblatt desenvolveu o neurônio *Perceptron* (??), que mais tarde seria empregado como a unidade de processamento das RNA e de outros modelos de ML, a exemplo das máquinas de vetores de suporte. O Perceptron de Rosenblatt agregou ao neurônio de McCulloch e Pitts conceitos cruciais para a caracterização das RNAs como são conhecidas hoje, como a não obrigatoriedade de igualdade dos pesos e limiares de ativação, a possibilidade de os pesos serem positivos ou negativos, a diversidade de funções de ativação, entre outros. Além desta caracterização, uma contribuição relevante deste trabalho contempla a proposição de um algoritmo de aprendizado que permite a adaptação dos pesos de uma RNA através da otimização do desempenho da rede. Isto atribuiu ao modelo Perceptron a capacidade de aprender tarefas que contenham dados linearmente separáveis (??).

Na ocasião de sua proposição, o modelo Perceptron apresentava algumas limitações, atribuídas principalmente à sua linearidade e simplicidade, características que possibilitam resolver apenas problemas linearmente separáveis (??). Com o objetivo de tornar as RNAs aplicáveis em uma gama maior de problemas, os esforços de pesquisadores da área foram concentrados

em aumentar e diversificar as técnicas aplicáveis a este modelo. Atualmente, as RNAs podem apresentar diversos tipos de arquitetura, ao variar-se parâmetros como o número de camadas, quantidade de neurônios em cada camada, os tipos de conexões entre neurônios e topologia de rede. Alguns exemplos de arquiteturas podem ser encontrados na Figura ??.

Figura 2.4: Arquiteturas populares de RNAs. Fonte: (??)

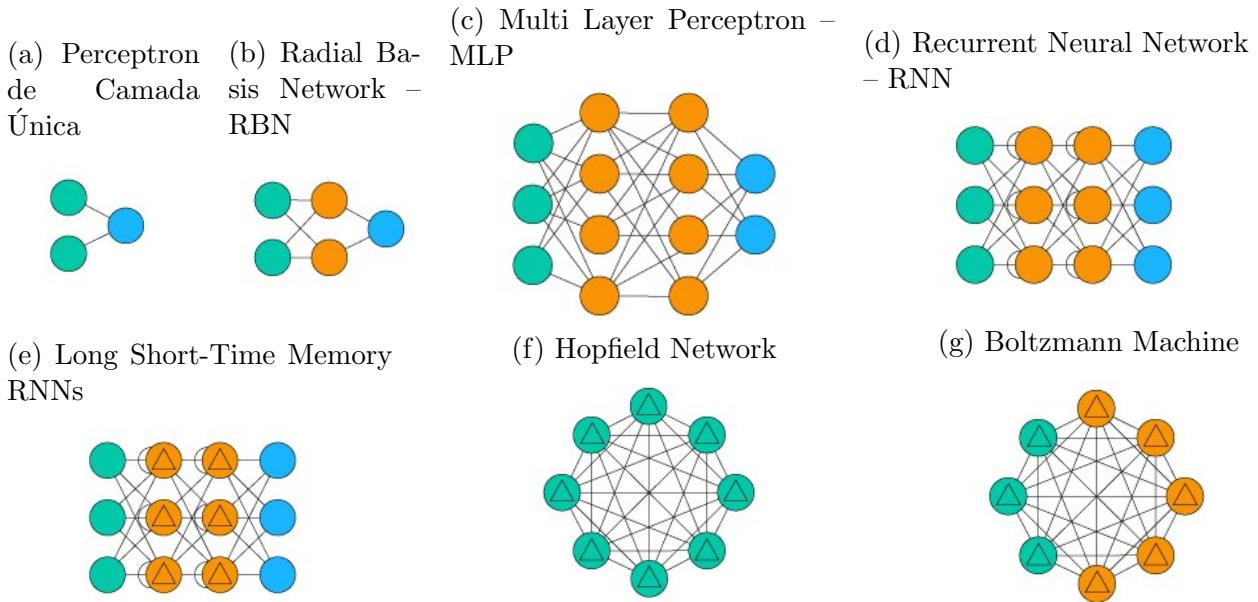


Tabela 2.2: Legenda das unidades presentes nas redes neurais da Figura ??.

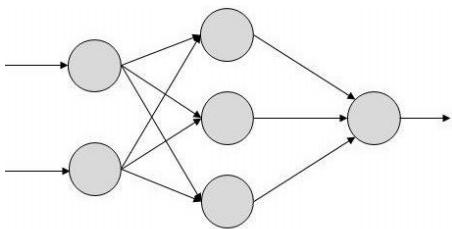
Imagen	Unidade
●	Entrada
○	Saída
■	Oculta
○△	Feedback com memória
○△	Entrada retroalimentada
■△	Oculta probabilística

Quanto aos tipos de conexão possíveis entre os neurônios, tem-se que as RNAs podem ser do tipo *feedforward* ou recorrente. As RNAs *feedforward*, como a exemplificada na Figura ??, são comumente associadas a um grafo acíclico em que as saídas de uma camada servem de entrada à camada seguinte, e assim sucessivamente, até que seja produzida uma saída da rede. As RNAs

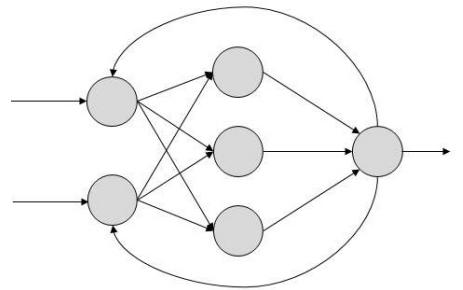
recorrentes, tais como a rede exemplificada na Figura ??, contém conexões entre neurônios de modo a formar um grafo direcionado cíclico, o que permite que o modelo capture sequências de comportamentos organizados em séries temporais.

Figura 2.5: Exemplos de RNA com diferentes tipos de conexões entre neurônios (??).

(a) Exemplo de RNA *feedforward*.



(b) Exemplo de RNA recorrente.



Um dos parâmetros relacionados à arquitetura de uma RNA é a quantidade de camadas ocultas. Pode-se ter redes de camada única, compostas por um neurônio que conecta todos os parâmetros de entrada às saídas do modelo, a exemplo das redes Perceptron. Há também as redes de múltiplas camadas, que consistem de mais de um neurônio entre entrada e saída da rede, como retratado na Figura ???. Redes com múltiplas camadas, as chamadas Redes Neurais *Feedforward Multilayer Perceptron* (MLP), são capazes de aproximar diversas funções. Quando a Equação ?? é aplicada a uma camada oculta c , de modo que a entrada da função nesta camada seja a saída da camada anterior $c - 1$, o resultado a^c é construído como mostrado na Equação ?? e chamado de mapa de características ou *feature map*. Para uma RNA com profundidade C , a saída prevista \hat{y} é dada pelo mapa de características a^C (????).

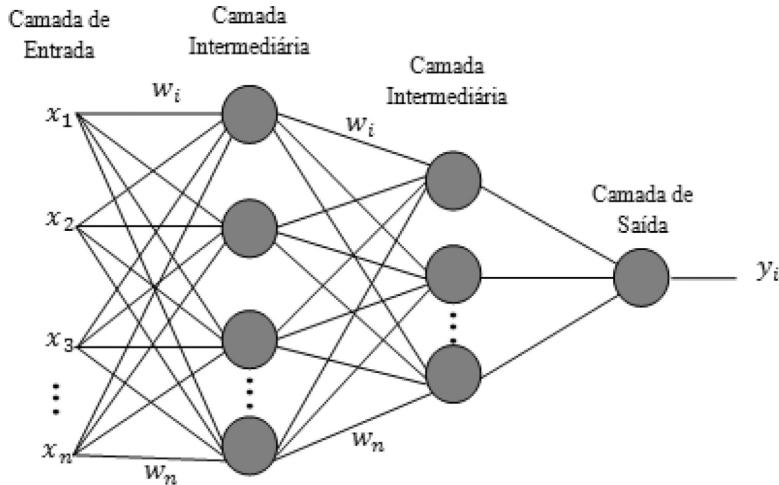
$$z^c = \sum_{i=1}^n a_i^{c-1} w_i^c + b_i^c \quad (2.4)$$

$$a^c = \sigma(z^c) \quad (2.5)$$

Segundo o Teorema da Aproximação Universal definido por Hornik em 1991, se a ativação de uma rede neural MLP for uma função limitada e não-constante, então dada uma entrada x , a rede é capaz de aproximar qualquer função contínua, provida uma quantidade adequada

de camadas ocultas. Esta característica atribui às redes neurais artificiais o potencial de se tornarem máquinas de aprendizado universal (??).

Figura 2.6: Rede Neural MLP com duas camadas ocultas.



O objetivo das RNAs é aproximar funções que mapeiem entradas x às suas respectivas saídas y . Para atingir este objetivo, é necessário minimizar a disparidade entre as saídas previstas \hat{y} e as saídas desejadas y através da atualização dos pesos w e do *bias* b dos neurônios das camadas. A função que calcula tal disparidade é chamada *função custo*, dada por J e tida como a soma das funções de perda, $L(\hat{y}_i, y_i)$, entre cada saída esperada y_i e obtida pela RNA \hat{y}_i , que é acumulada à medida que o modelo é apresentado a m exemplos do evento que se deseja aprender. A função custo está representada na Equação ??, e a perda pode ser calculada de diversas maneiras, a depender do tipo de tarefa de aprendizado, função de ativação e algoritmo de otimização escolhidos.

$$J = \frac{1}{m} \sum_{i=1}^m L(\hat{y}_i, y_i). \quad (2.6)$$

No contexto de maximizar as previsões corretas de uma RNA, deve-se atualizar gradualmente os pesos e *bias* do modelo de maneira a encontrar uma função que melhor represente o comportamento do fenômeno apresentado através dos dados, de maneira a minimizar a função custo para que haja a otimização do desempenho da RNA. Este procedimento é descrito pelo algoritmo de *back-propagation*, que consiste em duas fases que se alternam: a fase *forward* e a

fase *backwards* (??).

A fase *forward*, também chamada *forward propagation*, consiste na inferência das saídas da rede perante um conjunto de m entradas, fornecidas à rede uma de cada vez. Neste processo, dada uma entrada x_i , a informação flui para frente pela rede, isto é, as informações iniciais que são propagadas até as camadas ocultas, e de lá até que a saída \hat{y}_i seja produzida. Ao final de uma fase *forward*, a função custo J é calculada. Modificações presentes na literatura também consideram algumas execuções da fase *forward* para obtenção do custo médio nestas iterações. A representação da sequência de passos realizados nesta fase *forward* está detalhada no Algoritmo ?? (????).

Algoritmo 1: Fase *forward*

Entrada: i -ésimo par de exemplos de entrada x_i e saída y_i do conjunto de treinamento

Saída: Perda $L(\hat{y}_i, y_i)$

início

A entrada x_i é apresentada à primeira camada da rede, como o primeiro vetor de características a^0

para cada camada $c=1, \dots, C$ **faça**

| Calcular a saída da camada $a^c = \sigma^c(z^c)$, $z^c = w^c \times a^{c-1}$

fim

Tomar como valor de saída do modelo a saída da última camada $\hat{y} = h^C$

Calcular a perda $L(\hat{y}_i, y_i)$.

fim

A fase *backwards* é responsável por permitir que a informação referente à diferença entre os valores de saída obtidos \hat{y} e os esperados y calculada através da função custo na fase *forward* flua para trás. Isto ocorre por meio da atualização dos pesos dos neurônios, a começar por aqueles localizados na camada de saída, passando pelas camadas ocultas, até atingir a camada de entrada. Ao percorrer este caminho contrário, para cada camada é calculado o gradiente δ da perda L em função dos pesos e *bias*, dado pela fórmula mostrada na Equação ??.

$$\delta_w = \nabla_w J = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \dots, \frac{\partial J}{\partial w_n} \right]^T \quad (2.7)$$

$$\delta_b = \nabla_b J = \left[\frac{\partial J}{\partial b_1}, \frac{\partial J}{\partial b_2}, \dots, \frac{\partial J}{\partial b_n} \right]^T \quad (2.8)$$

No contexto do cálculo, o gradiente indica o sentido e a direção para as quais se devem mover os valores dos pesos e *bias* das camadas de maneira a se obter o maior incremento possível de perda. Considerando que o objetivo consiste na minimização *gradual* dos parâmetros da rede na iteração $t + 1$, o ajuste dos pesos e *bias* dos neurônios na iteração t é comumente realizado por meio do método gradiente descendente indicado na Equação ??, no qual o valor do gradiente δ é multiplicado por uma taxa de aprendizado η e então subtraído dos valores de w e b (????).

$$w^c(t + 1) = w(t) - \eta \nabla_{w^c} \quad (2.9)$$

$$b^c(t + 1) = b(t) - \eta \nabla_{b^c} \quad (2.10)$$

A fase *backwards*, denotada no Algoritmo ??, lança mão de sequências de operações de regras da cadeia para calcular os gradientes. Inicialmente, o gradiente δ da função de ativação σ da camada l é obtido ao realizar o produto interno do gradiente calculado sob a função de custo da RNA pela derivada da função de saída σ^c . Esta combinação de derivações encadeadas é altamente eficiente, o que faz com que o custo operacional da computação do gradiente seja $O(m)$, sendo m o número de exemplos no conjunto de treinamento. Assim, o custo computacional cresce de maneira proporcional e linear à quantidade de exemplos presentes no conjunto de treinamento (????).

Algoritmo 2: Fase *backwards*.

Entrada: Custo J

Saída: Gradientes dos parâmetros da RNA MLP atualizados

início

Calcular gradiente do custo, ou seja, da perda da camada de saída

$$\delta = \nabla_y J = \nabla_y L(\hat{y}_i, y_i)$$

para cada camada $c=C, \dots, 1$ **faça**

Calcular o gradiente da camada c , dado por $\delta^c = \delta^{c-1} \cdot \sigma^c(z^c)$

Atualizar variação em w , dada por $\nabla_{w^c} = \delta^c a^{(c-1)}$

Atualizar variação em b , dada por $\nabla_{b^c} = \delta^c$

fim

fim

Além dos parâmetros w e b , as RNAs MLP têm hiperparâmetros, responsáveis por contro-

lar as mudança ocorridas nos parâmetros. Bengio define hiperparâmetros como variáveis cujos valores devem ser definidos antes que o algoritmo de treinamento se inicie. Alguns hiperparâmetros já discutidos neste texto são a taxa de aprendizado η , número de camadas ocultas, quantidade de neurônios e tipos de funções de ativação escolhidos para cada camada. O número de vezes que o conjunto de dados é apresentado ao modelo no treinamento, conhecido como número de épocas, também é um hiperparâmetro, assim como a quantidade de exemplos de treinamento apresentados à rede de uma só vez na fase *forward*, chamado de *batch size* (??).

Sumarizando os conceitos apresentados, o algoritmo para treinamento supervisionado de uma RNA MLP se dá como mostrado no Algoritmo ??.

Algoritmo 3: Algoritmo de treinamento de uma RNA (??).

Entrada: Conjuntos de exemplos e respectivos rótulos (X, Y), rede neural a ser treinada, número de épocas e , taxa de aprendizado η e *batch size* b .

Saída: Rede neural treinada.

início

Inicialização dos vetores de pesos w e *bias* b

para cada $batch = 1, \dots, b$ do conjunto de dados **faça**

Fase *forward*: Calcular previsões \hat{y} e custos J .

Fase *backwards*: Calcular gradientes dos pesos ∇_{w^c} e *bias* ∇_{b^c}

Atualizar valores dos pesos e *bias* a partir do gradiente descendente.

fim

fim

Ao lidar com técnicas que visam acelerar ou potencializar o processo de aprendizado, o número de hiperparâmetros aumenta. Alguns exemplos destas técnicas incluem os algoritmos que realizam a otimização do gradiente descendente através da regularização dos pesos, como a Estimação Adaptativa do Momento, ou *Adam* e o algoritmo de otimização da família dos métodos quase Newtonianos *L-BFGS*. Outros hábitos comuns incluem a adoção de métodos de inicialização dos pesos que evitem o gradiente de convergir para pontos de sela, e de uma taxa de aprendizado que diminui conforme o número de épocas executadas aumenta para que o gradiente converja para um ponto mínimo mais rapidamente (??).

As RNAs *feedforward* MLP são amplamente utilizadas em aplicações de diversos domínios. Inicialmente, destacaram-se as aplicações voltadas para o mercado financeiro visando, por exemplo, otimizar estratégias de marketing. Aplicações posteriores consideraram a alocação de

assentos em aviões, aprovação de empréstimo, controle de qualidade em processos industriais, dentre outros (??). O escopo de aplicações deste modelo continua a crescer nos dias atuais, especialmente diante do desenvolvimento de variantes, a exemplo das redes neurais convolucionais, com grande capacidade de detecção de padrões e pouco esforço de pré-processamento. Reconhecimento de caracteres e dígitos (??), processamento de imagens médicas para reconhecimento de características associadas à doenças cardíacas (??), pulmonares (??) e mamárias (??) são alguns exemplos de aplicações de vanguarda destes modelos compreendidos dentro da sub-área de *Deep Learning*, que será caracterizada na seção a seguir.

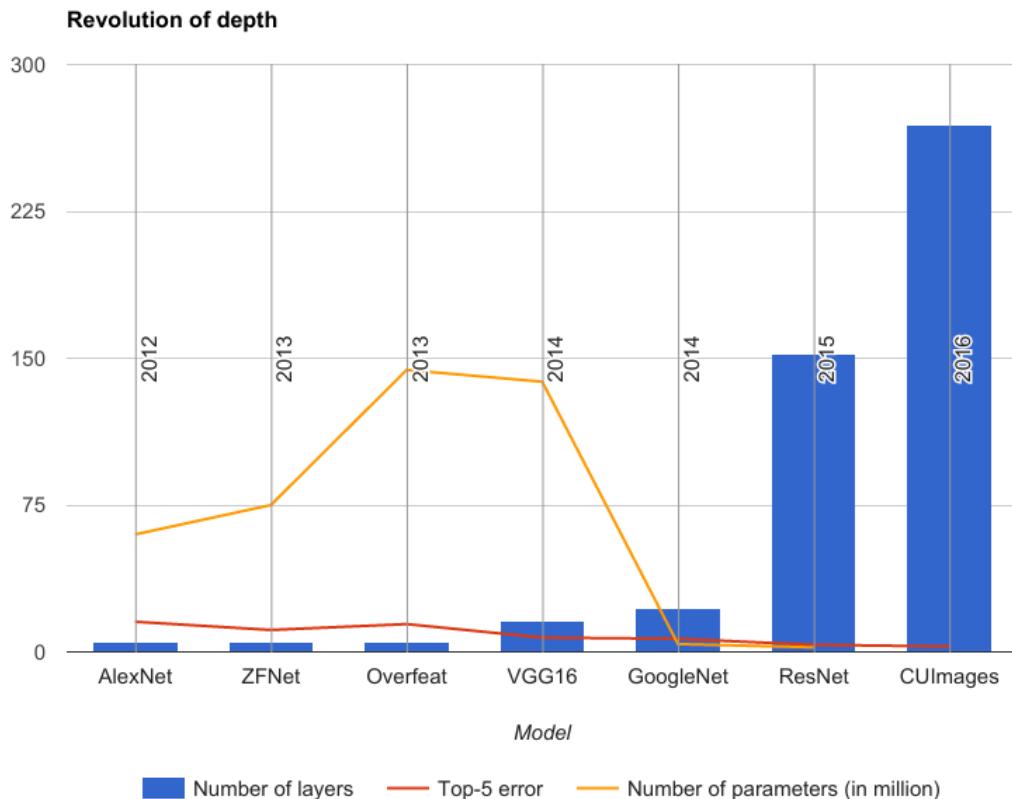
2.3 *Deep Learning*

Deep Learning (DL), também conhecido como Aprendizado Profundo, compreende um conjunto de técnicas de ML que podem ser aplicadas em problemas de aprendizado supervisionado e não-supervisionado. A principal característica dos modelos neste domínio é a capacidade de representar e reconhecer características sucessivamente complexas, por meio da adição de níveis ou camadas de operações não-lineares em sua arquiteturas, a exemplo das redes neurais profundas, máquinas de Boltzmann profundas e fórmulas proposicionais. Modelos deste tipo ganharam popularidade ao se mostraram capazes de resolver problemas complexos com um desempenho cada vez maior (??).

A melhoria do desempenho de modelos de DL é decorrente do aumento recente da quantidade de dados disponíveis sobre temas complexos, aliado ao aumento da disponibilidade de recursos computacionais para executar modelos mais robustos (????). Alguns dados fornecidos pela IBM reforçam esta afirmação: em 2017 foram gerados 2,5 quintilhões de bytes de dados por dia, e 90% do volume total de dados gerados até 2017 no mundo foi criado somente nos últimos dois anos (??). Estes fatores possibilitaram a implementação de modelos que apresentaram uma melhoria significativa na eficiência de generalização frente a modelos existentes até então, especialmente em virtude da capacidade de organizar a computação como uma composição de várias operações não-lineares (funções de ativação) e uma hierarquia de características reutilizadas (adição de camadas) (??).

Para exemplificar o efeito da adição de camadas aos modelos de DL, a Figura ?? mostra uma visão geral do aumento da profundidade das camadas de redes neurais e o desempenho destas em problemas de detecção de objetos em imagens. Nota-se que, à medida que a profundidade aumenta, há uma diminuição no erro. Mais recentemente, isto também tem implicado na redução do número de parâmetros treináveis, por meio da implementação de técnicas de subamostragem (??). Este panorama reforça a hipótese de que o aumento da profundidade das redes neurais impacta positivamente na captura de características e que estes avanços têm tornado as tarefas mais factíveis, com uma diminuição do esforço computacional associado, em comparação com modelos mais rasos (??).

Figura 2.7: Evolução de profundidade, taxa de erro e número de parâmetros das redes neurais profundas com o passar dos anos. Fonte: (??).



2.3.1 Breve Histórico

O termo *Deep Learning* (DL) não é recente, foi utilizado pela primeira vez por Dechter, no contexto da descoberta de todas as configurações de conflitos mínimas a fim de resolver um

problema de satisfação de restrições (??). Porém, ganhou força a partir de pesquisas sobre RNAs *feedforward* com muitas camadas ocultas, também conhecidas por redes neurais profundas (??).

Considera-se que o desenvolvimento de DL pode ser dividido em três momentos. No primeiro momento, houve a proposição de modelos lineares simples, compostos apenas por um neurônio, a exemplo dos neurônios de McCulloch e Pitts (??) e *Perceptron* de Rosenblatt (??). No segundo instante, iniciado nos anos 1980, teve-se como eixo central a interconexão entre vários neurônios e a proposição do algoritmo *back-propagation* para ajuste de pesos no treinamento das RNAs (????). Com estas contribuições, houve aplicação das RNAs em diversos domínios. Ainda no final deste segundo momento, duas contribuições relevantes foram feitas: os modelos *Long Short-Term Memory* (LSTM) e LeNet. Esta última utiliza o algoritmo de *backpropagation* para treinar uma rede neural convolucional profunda para reconhecer dígitos escritos à mão (??).

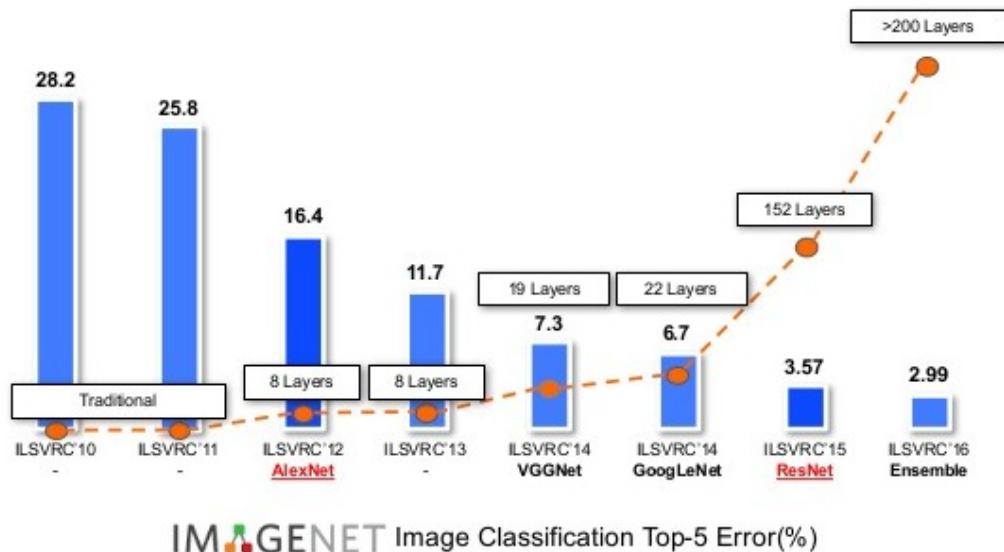
A terceira fase tem um marco inicial definido: compreende o ano de 2006, quando Hinton utilizou o termo *deep belief network* para designar um tipo de modelo de RNA MLP cujo treinamento adota uma estratégia gulosa e orientada a camadas. Cada camada é pré-treinada individualmente como uma máquina de Boltzmann restrita, e o modelo inteiro é então ajustado utilizando técnicas de treinamento supervisionado, incluindo o algoritmo de *backpropagation*. A partir deste marco, outros pesquisadores passaram a investigar a técnica de Hinton e, com o tempo, o termo *deep learning* passou a designar modelos compostos de várias camadas sucessivas de operações não lineares utilizados para o aprendizado de determinada tarefa (????????).

Na conjectura atual, modelos de DL têm superado significativamente o estado da arte de modelos inteligentes em diversas competições em todo o mundo. A *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) (??) é uma competição em que equipes de pesquisa avaliam seus algoritmos em um conjunto de dados fornecido, e competem para chegar à melhor acurácia em várias tarefas de reconhecimento visual automático. O conjunto de dados utilizado, denominado ImageNet (??), consiste de um conjunto de aproximadamente 14 milhões de imagens de 21 mil categorias organizadas hierarquicamente, em que cada uma contém algumas centenas de imagens de exemplo. A performance dos modelos submetidos para a competição é chamada taxa de erro top-5, e representa o erro computado quando a classe alvo não se encontra entre

as 5 apontadas pelo modelo como as que têm maior probabilidade de estarem na imagem. Em 2011, os melhores resultados de classificação no ILSVRC tinham por volta de 25% de erro top-5 nas tarefas propostas. Em 2012, o modelo AlexNet, uma rede neural convolucional proposta segundo as ideias de DL, atingiu apenas 16,4% de erro, propondo um ganho até então nunca visto entre duas edições sucessivas da competição (??).

O gráfico da Figura ?? sintetiza o histórico da competição ILSVRC, em que a partir do ano de 2012 houve a introdução de modelos baseados em DL. O histograma mostra a diminuição do erro na tarefa de aprendizado proposta e a linha tracejada enfatiza o número de camadas ocultas utilizadas nos modelos vencedores.

Figura 2.8: Evolução do erro dos modelos vencedores da competição ILSVRC pela profundidade das redes neurais (????)



Apesar do foco inicial de DL ter sido concentrado no desenvolvimento de técnicas de aprendizado não-supervisionado e na habilidade de modelos profundos de boa generalização a partir de conjuntos de dados pequenos, o cenário atual das pesquisas nesta área consideram o uso de técnicas de aprendizado supervisionado visando o endereçamento de conjuntos de dados massivos e categorizados, e também redes neurais profundas híbridas, que misturem técnicas e conceitos de diferentes origens (????).

Dentre estes resultados de vanguarda no ILSVRC destacam-se os modelos de DL construídos

com redes neurais convolucionais, que têm impulsionado os mais recentes avanços na área de Visão Computacional desde a proposição na literatura da rede LeNet (??). Em particular, desde a proposição e vitória do modelo AlexNet (??) no ano de 2012, as redes neurais convolucionais têm dominado o ranking dos anos seguintes da competição, conforme ilustrado na Figura ?? (??). Dada a importância deste tipo de rede neural, os conceitos sobre as mesmas e os modelos canônicos serão apresentados a seguir.

2.3.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNNs, do inglês *Convolutional Neural Networks*) são uma classe de redes neurais *feedforward* com topologia bem definida e estrutura em grade, com o uso de operações de convolução em pelo menos uma de suas camadas (??). Aplicadas em tarefas de classificação, regressão, localização, detecção e outras, este tipo de modelo se destaca no reconhecimento de padrões em dados de alta dimensionalidade, a exemplo de séries temporais, imagens e vídeos (??).

A operação de convolução possui um papel central nas CNNs. Esta operação descreve a média ponderada de uma determinada função $x_1(t)$ sob um intervalo fixo de uma variável, enquanto os pesos da média ponderada considerada pertencem à função $x_2(t)$ amostrados em intervalos a (??). Assim, a convolução $s(t)$ de duas funções $x_1(t)$ e $x_2(t)$ é uma função $s : \mathbb{Z} \rightarrow \mathbb{R}$, denotada $s(t) = x_1(t) * x_2(t)$, e definida conforme Equação ?? (??):

$$s(t) = x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(a)x_2(t-a)da. \quad (2.11)$$

No contexto de ML, a função $x_1(t)$ é chamada de *input*, a função $x_2(t)$ é o *kernel*, e a saída $s(t)$ consiste no *feature map*, ou mapa de características. No contexto prático, o *input* normalmente é um vetor multidimensional de dados e o *kernel* é um vetor multidimensional de pesos que devem ser ajustados para aprendizado das CNNs. Considerando, por exemplo, uma imagem I de dimensões (m, n) como *input* e a aplicação de um *kernel* K , a versão discreta da convolução, passível de implementação computacional e equivalente à Equação ??, é mostrada

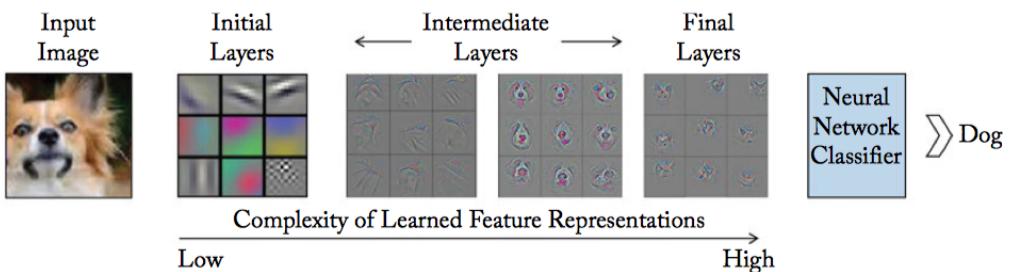
na Equação ??:

$$S(i, j) = I(i, j) * K(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n), \quad (2.12)$$

em que S é o *feature map* resultante e (i, j) é a posição correspondente nesse mapa. Para otimizar os aspectos de implementação, os valores resultantes da operação de convolução são armazenados apenas nas posições (i, j) explicitamente declaradas ??.

Os *feature maps*, resultantes das operações de convolução, compreendem a noção de filtros, responsáveis por capturarem características relativas à entrada, tais como contornos, linhas, texturas, etc. Quando combinados de maneira sequencial, como proposto pelas CNNs, as características capturadas pelas camadas convolucionais vão se tornando mais complexas à medida que se aumenta a profundidade da rede. Assim, um primeiro *feature map* de uma camada convolucional captura um simples contorno, enquanto um *feature map* em uma camada mais profunda da rede pode capturar uma forma, um rosto ou até um objeto inteiro ??). Esta noção é ilustrada na Figura ??.

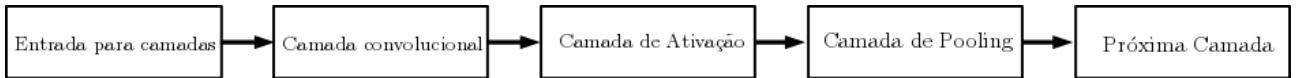
Figura 2.9: Papel das camadas convolucionais e *feature maps* nas CNNs. Fonte: ??).



As camadas convolucionais, que capturam os *feature maps* e contém os pesos da rede, normalmente são seguidas por funções de ativação como as exemplificadas na Seção ??, mais especificamente na Tabela ???. Via de regra, a toda camada convolucional em uma CNN, segue-se uma função de ativação, finalizando em uma operação de *pooling*, como mostra a Figura ??.

Uma função de *pooling*, por sua vez, substitui a saída da rede em determinada localização por uma síntese estatística das saídas vizinhas. Por exemplo, a operação *max pooling* retorna o valor

Figura 2.10: Componentes de uma camada de uma rede neural convolucional. Adaptado de (??).



máximo em uma área retangular, enquanto a *average pooling* retorna a média das saídas de um retângulo. O objetivo desta operação é fazer com que o *feature map* seja invariante a pequenas mudanças na entrada. Esta invariância a pequenas mudanças locais é uma propriedade útil quando o mais importante for a existência da característica e não exatamente a sua posição, o que aumenta a eficiência geral da CNN ao reduzir drasticamente o número de valores a serem passados entre duas camadas quaisquer (??).

Outros parâmetros, como *padding* e *strides*, são importantes para a captura de características. O parâmetro *padding* consiste em adicionar um número de linhas e colunas em cada lado da entrada de maneira a controlar o tamanho do *feature map* resultante da operação de convolução. Já a distância entre duas janelas da convolução, ou da operação de *pooling*, é medida através do número de *strides*. Ambos parâmetros manipulam as dimensões das saídas das camadas de uma CNN (??).

Embora se tenha uma noção clara das camadas individuais e de suas respectivas funções, a combinação das mesmas em uma rede neural convolucional não é uma tarefa trivial, podendo resultar em um número arbitrariamente grande de redes com milhares de parâmetros ajustáveis, cujo desempenho acerca de um problema ainda precisará ser aferido. Considerando os esforços computacionais para isto, a maioria das soluções atuais baseadas em DL fazem uso de CNNs canônicas já propostas na literatura, as quais são apresentadas a seguir.

2.3.3 Modelos Canônicos de Redes Neurais Convolucionais

Os modelos canônicos de CNNs são arquiteturas que trouxeram contribuições importantes, pioneiras na aplicação de técnicas que são comuns ainda hoje no cenário de DL, comumente utilizadas em diversas tarefas de aprendizado (??).

A LeNet é o primeiro modelo de rede neural a aplicar convoluções ao invés das camadas

totalmente conectadas convencionais. Foi proposta por LeCun em 1998 e é composta de 7 camadas, sendo uma de entrada, duas camadas convolucionais, duas camadas de *pooling*, uma camada totalmente conectada e a camada de saída. A tarefa de aprendizado endereçada por esta rede na ocasião de sua proposição foi o reconhecimento de dígitos manuscritos. Para o treinamento e teste desta rede foi utilizado o conjunto de dados *Modified National Institute of Standards and Technology* (MNIST), composto de 60000 imagens de treinamento e 10000 de teste dos dígitos de 0 a 9 escritos à mão (??). A LeNet foi amplamente utilizada por bancos para o reconhecimento de números escritos à mão em cheques digitalizado em imagens em escala de cinza de tamanho 32×32 (??). Apesar de pesquisas nesta área continuarem no decorrer dos anos, a quantidade insuficiente de bases de imagens catalogadas e o baixo poder computacional da época fizeram com que as CNNs permanecessem sem grandes destaque até o ano de 2012 (??).

AlexNet foi a primeira CNN ganhadora do desafio ILSVRC, em 2012, ao atingir um erro top-5 igual a 15.4%. O segundo melhor modelo daquele ano atingiu um erro de 26.2%. Esta rede, treinada para uma tarefa de classificação utilizando imagens de 1000 categorias da ImageNet, é formada por 5 camadas convolucionais com *kernels* de tamanho 11×11 , intercaladas com camadas de *max-pooling* e *dropout* e 3 camadas totalmente conectadas. Utilizava a função de ativação *ReLU* ao invés da tradicional tangente hiperbólica. Na ocasião, para obter uma quantidade de exemplos razoável mediante o número de parâmetros ajustáveis, foi realizado um aumento artificial nas imagens de entrada, modificando-as segundo translações, reflexões horizontais e cortes. A AlexNet foi então treinada utilizando duas GPU GTX 580 por 5 a 6 dias, com o algoritmo de *backpropagation* utilizando gradiente descendente estocástico para *batch* e técnicas como *momentum* e *weight decay*. Estas técnicas garantiram à rede um desempenho significativamente melhor que o dos modelos tradicionais que estavam sendo aplicados para a ILSVRC daquele ano (??).

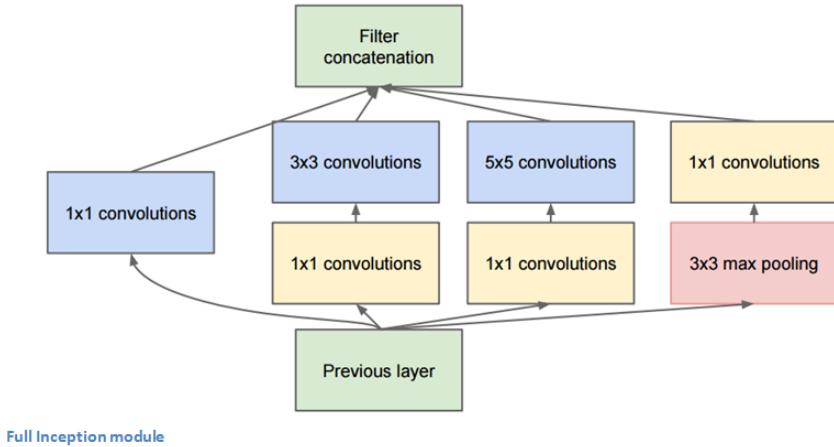
Uma CNNs com um amplo destaque pela simplicidade e profundidade é a VGG. Apesar de não ter ganhado a ILSVRC 2014, alcançou um erro top-5 de 7.3%. Foi concebida na Universidade de Oxford, a VGG-19 contendo 19 camadas que utiliza estritamente *kernels* de 3×3

com *stride* e *pad* de 1, juntamente com *max-pooling* de tamanho 2×2 e *stride* 2. O número de filtros dobra após cada camada de *max-pooling*, o que reforça a idéia de diminuir dimensões espaciais de largura e altura e aumentar a profundidade. A VGG-19 foi treinada em parte da base ImageNet utilizando 4 GPUs NVIDIA Titan Black por duas a três semanas (??).

Uma das primeiras arquiteturas de CNNs que se desviou do caminho normal de simplesmente empilhar camadas convolucionais e de pooling em uma estrutura sequencial foi a GoogLeNet, também chamada de Inception. Dotada de 22 camadas convolucionais, a rede ganhou o ILSVRC 2014 com um erro top-5 de 6.7%. Esta performance se deve aos chamados de módulos Inception, compostos de camadas da rede que ocorrem em paralelo. Ao invés de realizar uma operação de convolução ou *max-pooling* de cada vez, várias operações diferentes são realizadas e os mapas de características obtidos são condensados e conectados ao próximo bloco Inception. A Figura ?? mostra que há 4 conjuntos de operações a serem realizadas, todas contendo convoluções com *kernels* 1×1 , podendo haver também a convolução com *kernels* de tamanho 3×3 , 5×5 , ou uma operação de *max-pooling* de 3×3 . A convolução com filtro 1×1 é aplicada para reduzir a dimensionalidade do problema reduzindo a profundidade do mapa de características, além de retirar da entrada informações bem detalhadas em volume. As convoluções que utilizam *kernels* de 3×3 e 5×5 dão ao modelo a capacidade de extrair informações relevantes em larga escala. A camada de *max-pooling* é aplicada a fim de reduzir a largura e altura do mapa de características e combater *overfitting*. Na GoogLeNet, as camadas totalmente conectadas são substituídas por *average pooling*, o que economiza um grande número de parâmetros. A rede, que foi treinada em algumas GPUs de alta performance por uma semana, utiliza *ReLU* como função de ativação e dispõe de 9 módulos Inception com mais de 100 camadas no total. Todas estas técnicas utilizadas para reduzir a dimensionalidade do problema surtiram efeito pois, como mostrado na Figura ??, nota-se que esta rede tem 12 vezes menos parâmetros que a sua predecessora AlexNet (??).

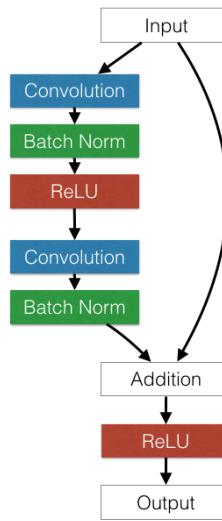
A Microsoft ResNet foi a CNN vencedora do ILSVRC 2015, com uma taxa de erro top-5 de 3.6%. Composta de um total de 152 camadas, esta rede neural deve o sucesso de sua profundidade ao bloco residual, representado na Figura ?? que, de maneira resumida, soma à

Figura 2.11: Bloco Inception da CNN GoogLeNet. Fonte: (??).



saída de um certo bloco de convoluções a saída de um bloco anterior, para que ambos *feature maps* sejam alimentados à função de ativação *ReLU*.

Figura 2.12: Bloco Residual da CNN ResNet. Fonte: (??).



Em CNNs tradicionais, a partir de uma imagem de entrada, obtém-se o *feature map* resultante de uma operação de convolução e então aplica-se o *max-pooling*. Este resultado produz uma nova representação, que possui pouca relação com a entrada original. No caso da ResNet, o bloco residual faz com que a entrada original persista em camadas mais profundas da rede, o que garante que as características determinantes para a tarefa de aprendizado sejam disseminadas para camadas mais profundas, ao mesmo tempo em que mantém a dimensionalidade reduzida. Outra razão para o bloco residual ser efetivo é que durante a fase *backwards* o gradi-

ente vai fluir mais facilmente pela rede, pois os blocos residuais estão diretamente conectados às camadas mais rasas na arquitetura, o que facilita a distribuição do gradiente. Na ocasião da sua proposição, a ResNet foi treinada em 8 GPUs por duas a três semanas (??).

Observada a importância das CNNs apresentadas nesta seção, vale notar o esforço computacional para o treino das mesmas, que demora dias mesmo com um hardware específico para este fim. Todo o resultado do treinamento destas redes com o conjunto de dados ImageNet encontram-se disponíveis em *frameworks* para implementação de CNNs, como o *Keras* (??) e *Tensorflow* (??). Disponibilizar estas versões de tais CNNs favorece o aproveitamento das mesmas em contextos análogos, como mostrado a seguir.

2.3.4 Transfer Learning

As várias redes convolucionais treinadas com milhares de exemplos da base de dados ImageNet e validadas na competição ILSVRC, a exemplo das que foram mencionadas na seção anterior, registram alto desempenho nas tarefas de aprendizado para as quais foram originalmente projetadas. Considerando a arquitetura das mesmas, percebe-se como resultado do treinamento efetuado que uma grande quantidade de parâmetros foi ajustada mediante os milhares de exemplos apresentados, processo este que demorou vários dias mesmo mediante uso de hardware altamente especializado.

Segundo Oquab, representações de imagens aprendidas por CNNs a partir de conjuntos de dados com grande número de exemplos podem ser transferidas eficientemente para outras tarefas de reconhecimento visual que tenham uma quantidade limitada de dados de treinamento (??). Para tanto, faz-se uso de uma técnica denominada *transfer learning*, que consiste em transferir os conhecimentos entre domínios relacionados. No contexto das CNNs aplicadas em reconhecimento de objetos em imagens, as camadas internas podem agir como detectores de características em médio nível. Estas podem ser pré-treinados na tarefa fonte, na qual utiliza-se um conjunto de dados vasto como o ImageNet, e então re-utilizados na tarefa alvo, que pode conter um conjunto de dados mais restrito.

Após o treinamento para a tarefa original, a rede aprende a identificar características mais

elementares, como linhas, contornos e objetos, que podem ser redirecionadas na tarefa alvo, que possui um objetivo mais específico. O trabalho de Zeiler e Rob, por exemplo, ilustra uma aplicação de *transfer learning* em uma tarefa de classificação, em que os autores transferem o conhecimento de uma CNN AlexNet treinada originalmente para o conjunto de dados ImageNet adaptando-a para o conjunto de dados Caltech-256, cuja base de dados possui apenas cerca de 30 mil imagens (??).

Na prática, o *transfer learning* é feito ao transferir os parâmetros de peso w e de bias b de um modelo já consolidado na literatura e pré-treinado com um conjunto de dados mais robusto, para um modelo similar ainda não treinado. Com o objetivo aproveitar os parâmetros pré-treinados, remove-se a última camada, ou seja, a camada de saída, e em seguida adiciona-se uma ou mais camadas novas, sem treinamento. Neste ponto, outros hiperparâmetros são considerados (número de camadas removidas, quantidade e tipo de camadas adicionadas, por exemplo) e o modelo passa por um processo de ajuste, chamado *fine tuning*, para ajustar os novos parâmetros adicionados em conformidade com os pré-existentes, permitindo a extração das características do conjunto de dados da tarefa alvo de maneira satisfatória (??).

2.4 Tecnologias Utilizadas

Para a realização deste trabalho são utilizadas tecnologias comumente relacionadas às práticas de ML. A linguagem de programação adotada é o Python 3, em conjunto com bibliotecas como a *Python Data Analysis Library* (Pandas) (??) utilizada para análise e consolidação de conjuntos de dados, as bibliotecas de visualização de dados Seaborn (??) e Matplotlib (??), as bibliotecas de ML com suporte a DL Keras (??) e TensorFlow (??).

A fim de executar o treinamento e teste dos modelos propostos, utilizou-se uma instância de máquina virtual disponibilizada através da *Google Compute Engine* (GCE) (??), dotada de 4 núcleos de processamento com 4 GHz cada e 16 GB de memória RAM. A GCE é parte da *Google Cloud Platform*, uma suíte de computação em nuvem oferecida pelo *Google*. Utilizou-se também um desktop físico pertencente à infraestrutura do LSI – UEA, com processador Intel Core I7 8a geração, 16 GB de RAM, 480 GB de memória SSD, 2 TB de memória HD e duas

placas de fídeo NVIDIA GeForce 1080 com 11 GB de memória cada.

Capítulo 3

Trabalhos Relacionados

A proposta apresentada está relacionada com inúmeros trabalhos envolvendo a aplicação de redes neurais convolucionais e outros modelos de ML para a estimativa de idade de indivíduos a partir de fotos.

Segundo Fu et al., a idade pode ser inferida a partir de padrões distintos que emergem através da aparência da face (??). Técnicas comuns para a estimativa da idade envolvem a dedução de modelos matemáticos a partir do estudo do crescimento de medidas da face e do crânio (??), da textura do rosto (??), da captura de tendências de envelhecimento a partir de várias imagens de indivíduos de mesma idade (??) e da extração de características específicas relacionadas à idade (????). Modelos de ML também são utilizados para a tarefa, em especial as redes neurais artificiais, K-vizinhos mais próximos e máquinas de vetores de suporte.

Recentemente, a aplicação de redes neurais convolucionais em problemas de classificação e detecção de objetos em imagens têm obtido resultados significativamente positivos. Em diversos trabalhos da literatura são descritas arquiteturas robustas capazes de detectar dezenas de objetos em várias situações (?????????). Treinadas com conjuntos de dados visuais que contam com milhares de exemplos como a ImageNet (??), Pascal VOC (??) e COCO (??), estas redes são conhecidas por seu bom desempenho. Algumas destas redes foram ajustadas utilizando conjuntos de dados menores e especializados para a tarefa de estimativa de idade.

O trabalho de Rothe em (??) relata um método para estimativa de idade aparente em imagens de faces imóveis utilizando DL. O método proposto consiste em detectar uma face em

uma imagem, para, em seguida estimar, sua idade. Para esta última tarefa, propôs um conjunto de 20 redes neurais convolucionais classificadoras com arquiteturas VGG-16 pré-treinadas com a base de dados ImageNet, e ajustadas utilizando imagens disponibilizadas pelos sites do IMDb, da Wikipedia, e o conjunto de dados *Looking At People* para anotação de idade aparente. Cada modelo tem como saída um número discreto entre 0 e 100, representando a idade prevista. A saída final do modelo consiste na localização do rosto e na média entre as idades previstas pelas 20 redes para o rosto detectado. A solução atingiu um MAE (*Mean Average Error*) de 3.221 na fase de testes.

Em Liu et al. cria-se um estimador de idade composto pela fusão de um modelo regressor e outro classificador (??). Realiza-se um pré-processamento das imagens de entrada, que envolve a detecção das faces presentes em cada imagem, seguida pela etapa de localização de pontos de referência, como olhos, nariz e boca, e por fim há a normalização das faces. Dois métodos de normalização de face são testados, a normalização exterior e interior. Após este pré-processamento, as imagens resultantes são alimentadas a modelos de redes neurais convolucionais profundas inspiradas na *GoogLeNet* (??). O modelo sofreu modificações em sua arquitetura, como adição de normalização do *batch*, remoção de camadas de *dropout* e perda. Foram treinados e testados diversos modelos com variações no tipo de normalização da face, tamanho do corte dos rostos, tipo de tarefa preditiva, etc. Os modelos resultantes destas variações foram unidos em um conjunto, que conseguiu prever idades com MAE de 3.3345 (??).

Ademais, é possível encontrar resultados satisfatórios para a tarefa de aprendizado proposta utilizando modelos menos complexos. Com o objetivo de consolidar um método de classificação de idade e gênero, o trabalho de Levi e Hassner propõe uma rede neural convolucional de natureza mais simples, se comparada com as citadas acima (??). A arquitetura proposta consiste em três camadas convolucionais com *dropout* e funções de ativação *ReLU*, seguidas por três camadas totalmente conectadas. A camada de saída tem como função de ativação a *Softmax*. A escolha por um design de rede menor é motivado pelo desejo de reduzir o risco de *overfitting* e pela natureza do problema, que contém somente 8 classes de idade. O modelo é treinado utilizando apenas o conjunto de referência *Adience*, composto por imagens não

filtradas para classificação de idade e gênero. Considerando uma margem de erro de uma classe de idade vizinha, a melhor rede obteve acurácia de $84.7\% \pm 2.2$ ao empregar a técnica de sobre-amostragem.

Capítulo 4

Solução Proposta

A solução proposta para a realização deste trabalho compreende a caracterização da tarefa de aprendizado, exposta na Seção ???. A descrição do conjunto de dados estão na Seção ???. A Seção ?? compreende a limpeza, o pré-processamento e a consolidação do conjunto de dados. Por fim, na Seção ?? estão os modelos e hiperparâmetros de CNNs considerados.

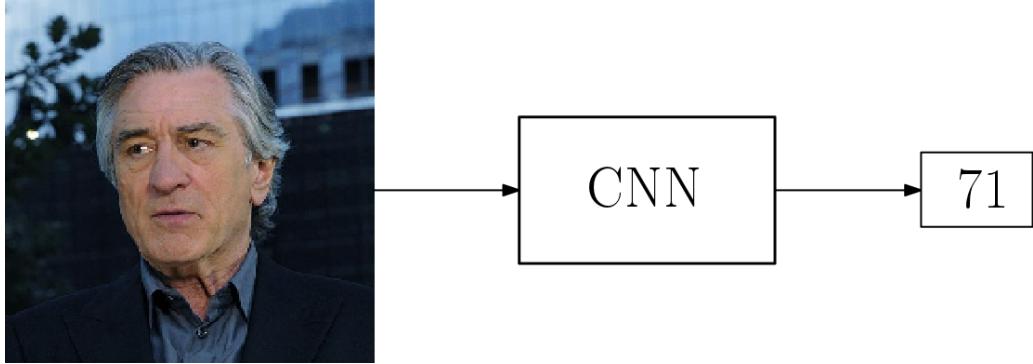
4.1 Tarefa de Aprendizado

A tarefa de aprendizado considerada para a estimativa de idade de telespectadores é a regressão. Neste contexto, uma imagem em cores RGB de dimensões 224×224 pixels contendo uma face humana centralizada será fornecida como entrada. A saída desejada é a estimativa de idade, em anos, da pessoa correspondente, conforme exemplificado na Figura ???. Esta tarefa será abordada segundo o paradigma de aprendizado supervisionado.

Os dados disponíveis para este contexto serão particionados em três conjuntos disjuntos, sendo 70% reservados para o treino, 10% para validação e 20% para teste. Esta partição obedece à tecnica *Holdout* de validação cruzada (??).

Os modelos propostos para esta tarefa terão seu desempenho aferido perante os dados do conjunto de testes de acordo com duas métricas de desempenho, *Root Mean Squared Error* (RMSE) e *Mean Average Error* (MAE). Estas métricas são análogas na medida em que consideram a diferença entre cada um dos valores previstos \hat{y} e os reais y , e posteriormente quantificam uma

Figura 4.1: Tarefa de aprendizado



média imune à variação positiva ou negativa desta diferença. A Equação ?? denota o cálculo do RMSE, enquanto a Equação ?? define o cálculo do MAE (??).

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y})^2}. \quad (4.1)$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}| \quad (4.2)$$

O RMSE será considerado para o cálculo da perda e por conseguinte da atualização dos pesos dos modelos, enquanto o MAE será utilizado para fins de comparação do desempenho dos modelos propostos neste trabalho com os modelos consolidados na bibliografia e detalhados nos trabalhos relacionados na Seção ??.

4.2 Conjunto de Dados

Para a tarefa de aprendizado apresentada, dispôs-se da base de dados experimentais IMDb, composta de 452.132 exemplares contendo imagens e outras informações de 20.284 dos atores mais populares listados no site IMDb. O conjunto de dados foi construído utilizando técnicas de *web crawling* aplicadas aos perfis de atores do site, em que foram coletadas todas as imagens relacionadas à celebridade, além de informações como data de nascimento, nome e gênero (??).

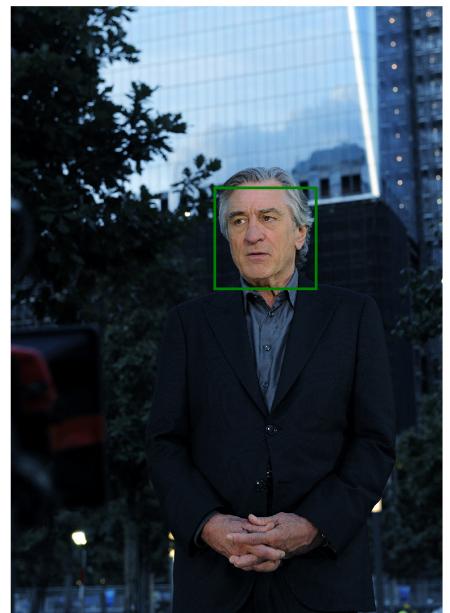
A base de imagens oriunda do site IMDb foi organizada por Rothe et al. considerando uma tarefa de aprendizado análoga à deste trabalho, conforme mencionado na Seção ?? (??). Nesta

base, há também as coordenadas da localização de um rosto detectado na imagem, além de uma pontuação atribuída ao rosto produzida pelo detector de face, quantificando o grau de certeza na detecção do rosto. Partindo da possibilidade de haver mais de um rosto por imagem, uma segunda pontuação é atribuída pelo detector, referente ao grau de certeza de que há outro rosto na mesma imagem.

Neste contexto, cada exemplo deste conjunto de dados é referente a uma imagem, cujos metadados estão descritos em seus atributos, que compreendem o nome, gênero, data de nascimento e um número de identificação da celebridade cujo perfil estava atrelado à imagem, o endereço da foto em disco, a suposta localização da face da celebridade, e pontuações referentes a duas possíveis faces encontradas. Assim, há exemplos de imagens em que há apenas um rosto, como mostrado na Figura ???. Já na Figura ?? está o exemplo de uma imagem onde há mais de um rosto, porém a localização do rosto está correta. Por fim, na Figura ?? há uma imagem com mais de um rosto, porém o rosto identificado neste item não é o da celebridade cujos dados estão referenciados.

Figura 4.2: Exemplo de imagem do conjunto de dados contendo apenas um rosto.

Meta-dado	Valor
ID Celebridade	16349
Nome	Robert De Niro
Endereço da imagem	imdb/34/nm0000134_rm3340090368_1943-8-17_2011.jpg
Pontuação da Face	5.21396
Pontuação da Segunda Face	NaN
Localização da Face	(663.65, 992.475, 590.134, 918.959)
Data de Nascimento	1943 – 08 – 17
Ano da Foto	2011
Gênero	Masculino



A versão original das imagens do conjunto de dados IMDb ocupava 267 GB em disco. Porém, uma versão pré-processada dessas imagens está disponível, contendo as faces recortadas com

Figura 4.3: Exemplo de imagem do conjunto de dados contendo mais de um rosto com a classificação correta.

Meta-dado	Valor
ID Celebridade	16349
Nome	Robert De Niro
Endereço da imagem	imdb/34/nm0000134_rm17663 60064_1943-8-17_2010.jpg
Pontuação da Face	5.12527
Pontuação da Segunda Face	5.08887
Localização da Face	(914.886, 1426.31, 287.31, 798.734)
Data de Nascimento	1943 – 08 – 17
Ano da Foto	2010
Gênero	Masculino

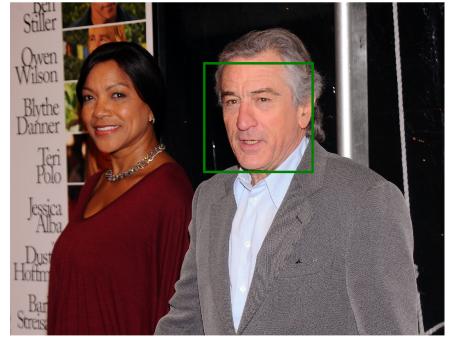


Figura 4.4: Exemplo de imagem do conjunto de dados contendo mais de um rosto com a classificação errônea.

Meta-dado	Valor
ID Celebridade	16349
Nome	Robert De Niro
Endereço da imagem	imdb/34/nm0000134_rm14800 44288_1943-8-17_2012.jpg
Pontuação da Face	5.51656
Pontuação da Segunda Face	4.55379
Localização da Face	(1392.72, 1614.18, 225.55, 447.003)
Data de Nascimento	1943 – 08 – 17
Ano da Foto	2012
Gênero	Masculino



40% da largura e altura da imagem original, totalizando 7,1 GB de dados. Esta versão foi considerada neste trabalho.

4.3 Limpeza e Pré-processamento dos dados

A fim de adequar melhor o conjunto de dados para os modelos de CNNs utilizados, realizou-se uma limpeza e pré-processamento dos meta-dados e das imagens da base IMDb, que se iniciou

com o cálculo do atributo alvo, a idade, a partir dos atributos originais fornecidos. A idade foi aferida através da data de nascimento da celebridade e do ano em que a fotografia em questão foi capturada.

Uma análise do conjunto de dados revelou a presença de itens com idade e gênero apresentando valores nulos, inválidos ou negativos, que foram descartados. Observou-se também a presença de múltiplos exemplos referentes à mesma pessoa com a mesma idade. Houve a remoção de tais exemplos, a fim de evitar que a apresentação de um mesmo rosto com a mesma idade provocasse *overfitting* nos modelos. Exemplos atípicos, possivelmente resultados de rotulação incorreta, como idade maior que 100 anos ou não compatível com os dados da celebridade referida nos meta-dados também foram descartados. Os atributos de pontuação de rostos foram úteis para identificar e remover exemplos em que não havia nenhum rosto identificado, ou em que havia mais de uma face na imagem. Este descarte foi realizado com o objetivo de eliminar rotulações errôneas, como a mostrada na Tabela ??.

Foram realizados também procedimentos de pré-processamento nas imagens. Considerando a literatura, padronizou-se o modo RGB e o tamanho das imagens para 224×224 pixels. Também foi efetuada a normalização das imagens, com o objetivo de escalar os valores dos pixels para o intervalo $[0, 1]$, realizada por meio da divisão dos valores de entrada por 255. Conforme sugerido pela literatura (??, vide Seção 5.2.4), esta prática propicia a convergência das redes neurais convolucionais, as quais tem um melhor ajuste de pesos quando as entradas são números pequenos. Em alguns cenários, conforme será apresentado posteriormente, considerou-se também a realização a equalização do histograma das imagens, a fim de aumentar o contraste global e evidenciar características das faces, como ilustrado na Figura ?? (??).

Após o pré-processamento das imagens de entrada, o cálculo do atributo alvo idade, a adequação do caminho para as imagens em disco e a remoção de exemplos impróprios, seguiu-se o descarte dos outros meta-dados irrelevantes para a tarefa de estimativa de idade de um indivíduo a partir de imagem. A data em que a foto foi tirada, nome, número de identificação, gênero, data de nascimento, localização do rosto da celebridade e pontuações de rostos nas imagens foram removidos.

Figura 4.5: Exemplo de imagem do conjunto de dados antes e depois do processo de equalização por histograma.

- (a) Imagem sem normalização por equalização de histograma. (b) Imagem após normalização por equalização de histograma.

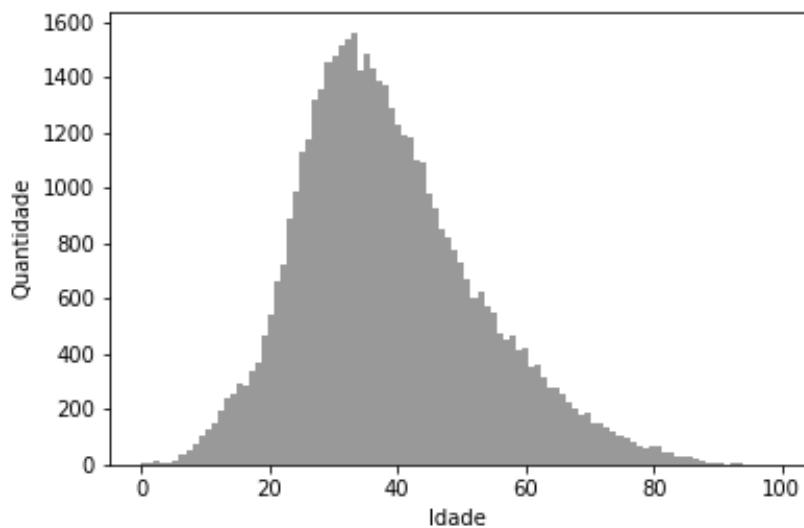


Figura 4.6: Histograma de frequência da idade do conjunto de dados utilizado.

Por fim, o conjunto de dados consolidado consiste de 47.950 exemplos contendo imagens e idades de 14.607 celebridades distintas, ocupando 1,2GB. O histograma de frequência da distribuição de idades, compreendendo o intervalo de 0 a 100 anos, presente nos exemplos da base de dados pode ser visualizado na Figura ???. Este total foi então dividido como proposto: conjunto de treinamento, contendo 70% dos exemplos, ou seja, 33.565 amostras; conjunto de validação, referente a 10% dos dados, ou seja, 4.795 itens; e, por fim, conjunto de testes, contendo os 20% restantes, ou seja, 9.590 exemplos. Esta divisão obedece ao que preconiza a

validação cruzada segundo o método *Holdout*.

4.4 Modelos de CNN Considerados

Levando em conta a adoção de CNNs como o modelo de ML a ser usado neste trabalho, considerou-se inicialmente a utilização das arquiteturas LeNet e AlexNet. A implementação da AlexNet, em particular, seguiu a prática de utilizar apenas uma GPU em seu treinamento, segundo a qual as camadas ora paralelas no trabalho original foram unificadas de forma sequencial (??). Adotou-se um *batch size* igual a 64 para o treinamento, e o método de otimização do gradiente descendente foi o *Adam*. O número de épocas foi obtido de maneira experimental e a taxa de aprendizado é adaptativa, sendo iniciada com 10^{-3} e com decaimento suave de 10^{10} , observando a perda obtida ao final de cada época.

Além da LeNet e AlexNet, duas redes mais profundas foram aplicadas para a tarefa: a VGG-16 e a SqueezeNet. A arquitetura profunda VGG-16 foi utilizada em uma tarefa de estimação de idade previamente descrita na literatura obtendo resultados satisfatórios, conforme Seção ?? e (??). A SqueezeNet, por sua vez, é uma CNN proposta recentemente na literatura, datada de 2018, especialmente voltada para os mesmos cenários da AlexNet, prometendo o mesmo desempenho, mas com 50 vezes menos parâmetros (??). No contexto de uma aplicação em que se necessita prover ao usuário respostas rápidas obtidas em *hardware* limitado, tal como ocorre neste cenário em que se consideram as *Smart TVs*, o potencial das características positivas destas redes torna propícia a investigação de sua adequação. O número de épocas de treinamento e a taxa de aprendizado inicial seguiram as mesmas práticas e valores daqueles adotados na LeNet e AlexNet.

É importante ressaltar que as quatro arquiteturas de CNNs consideradas, LeNet, AlexNet, VGG-16 e SqueezeNet, foram originalmente propostas para problemas de classificação, entretanto, no contexto deste trabalho, considera-se um problema de regressão. A fim de promover esta adaptação, a camada de saída, que anteriormente era constituída de múltiplos neurônios sujeitos à função de ativação *softmax*, foi substituída por um único neurônio. Considerou-se utilizar duas funções de ativação distintas na saída deste neurônio: a função *ReLU*, previamente

apresentada na Seção ??, e também a função *Leaky ReLU*, expressa na Figura ???. Esta última visa contornar um problema de estagnação típico durante o treinamento de CNNs com função de ativação *ReLU*, o chamado *dying ReLU problem*, tendo como características ser diferenciável e monotônica, colaborando também para a introdução de não-linearidade.

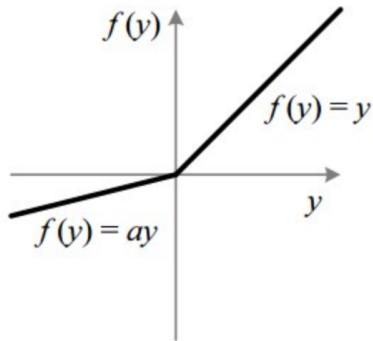


Figura 4.7: Função de Ativação *Leaky ReLU*

Definidos os parâmetros relativos às arquiteturas das CNNs e de suas funções de ativação, bem como os hiperparâmetros relativos ao treinamento no que diz respeito ao número de épocas, taxa de aprendizado e otimizador, partiu-se então para, enfim, o treino e teste dos modelos propostos, cujos resultados são discutidos no capítulo a seguir.

Capítulo 5

Resultados e Discussão

Considerando a estratégia descrita na solução proposta, os resultados da execução das CNNs aplicadas ao problema de estimativa de idade a partir de uma imagem de face são apresentados a seguir. Estes resultados estão organizados segundo abordagens sequenciais, que contemplam desde as técnicas mais elementares, e que vão aumentando o grau de complexidade conforme uso de estratégias específicas da prática de DL visando obter melhores resultados.

5.1 Abordagem 1: LeNet e AlexNet com Imagens Normalizadas

A primeira abordagem de treinamento considerou o uso dos modelos LeNet e AlexNet de maneira canônica, isto é, tais como são definidos na literatura. Adotou-se as funções de ativação não-lineares *ReLU* e *Leaky ReLU* por serem simples de calcular e por satisfazerem os critérios de continuidade e diferenciação, requeridos pelo algoritmo de *backpropagation*, conforme discutido anteriormente na Seção ??.

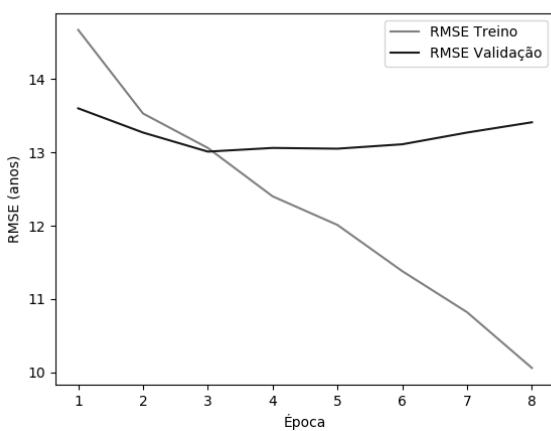
As imagens da base de dados foram normalizadas antes de serem apresentadas às redes, conforme a estratégia de pré-processamento previamente definida, isto é, todos os valores dos *pixels* componentes das imagens foram escalonados para o intervalo [0, 1] por meio de uma divisão por 255. Conforme mencionado, a prévia normalização das imagens antes da apresentação

às CNNs colabora para uma melhor execução do gradiente descendente e diminui a variância nos pesos.

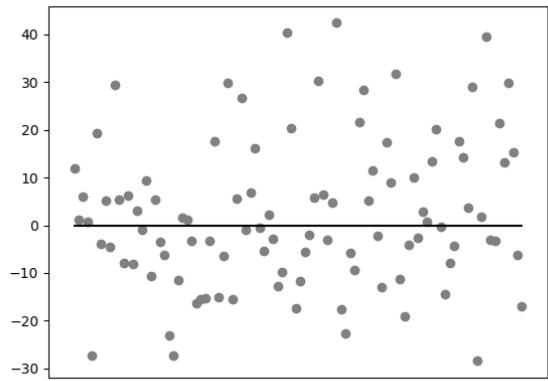
Os treinamentos destas duas arquiteturas duraram aproximadamente 16 e 12 horas respectivamente, em uma instância do Google Compute Engine com 4 CPUs virtuais e 15 GB de RAM. Os gráficos de treinamento e as retas zero obtidas a partir da apresentação do conjunto de teste aos modelos treinados podem ser vistos na Figura ??.

Figura 5.1: Resultados do treinamento e teste da CNN LeNet de acordo com a Abordagem 1.

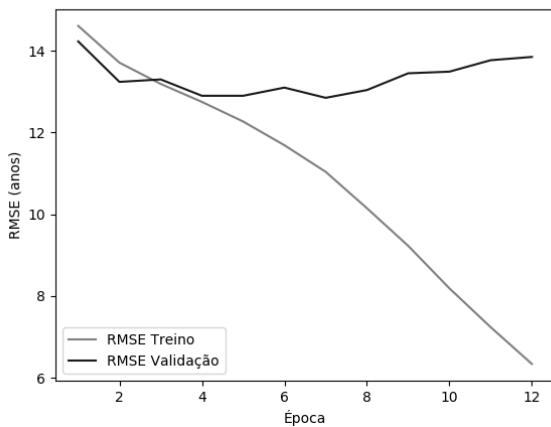
(a) RMSE de treinamento da arquitetura LeNet utilizando funções de ativação *ReLU*.



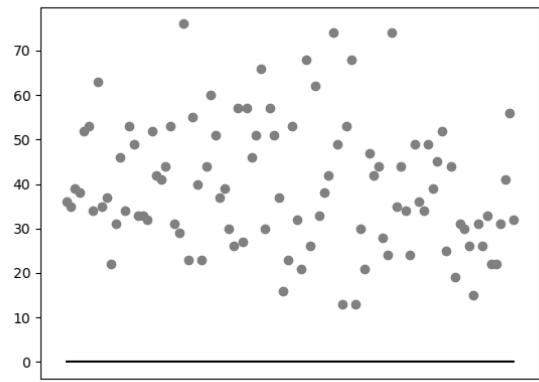
(b) Reta-0 LeNet *ReLU*.



(c) RMSE de treinamento da arquitetura LeNet utilizando funções de ativação *Leaky ReLU*.



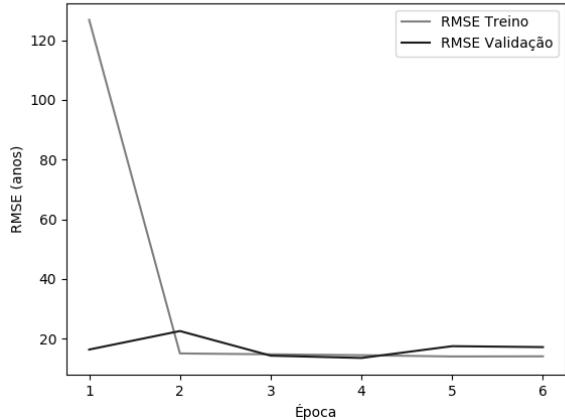
(d) Reta-0 LeNet *Leaky ReLU*.



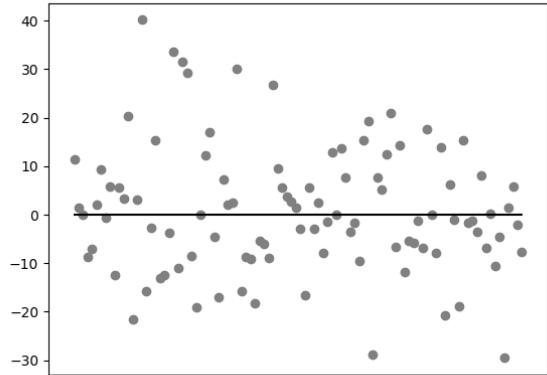
Obedecendo ao método de validação cruzada *holdout* previamente mencionado, os resultados desta abordagem encontram-se sintetizados na Tabela ???. De maneira geral, ambas as

Figura 5.2: Resultados do treinamento e teste da CNN AlexNet de acordo com a Abordagem 1.

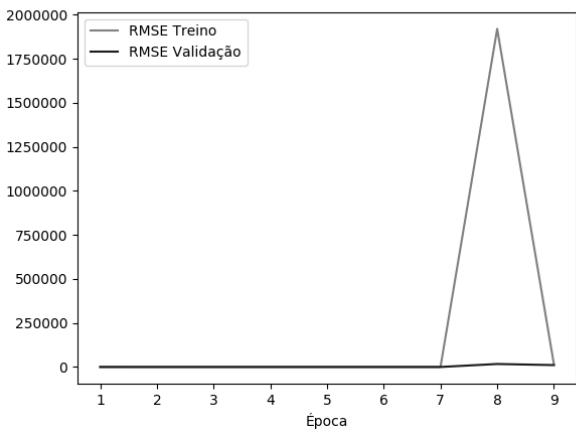
(a) Treinamento AlexNet *ReLU*.



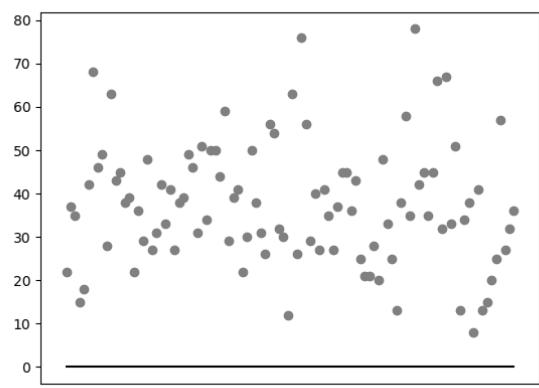
(b) Reta-0 AlexNet *ReLU*.



(c) Treinamento AlexNet *Leaky ReLU*.



(d) Reta-0 AlexNet *Leaky ReLU*.



redes obtiveram melhor desempenho com o uso da função de ativação *ReLU* em que, em particular, a arquitetura LeNet apresentou resultados ligeiramente superiores. Apesar disso, nota-se que o treinamento não seguiu de maneira estável, com grande variação entre as métricas de treinamento e validação, ocasionando *early stopping* após um número relativamente curto de épocas. Tal cenário é sugestivo de *overfitting*, especialmente mais evidentes nos casos em que se utilizou a função de ativação *Leaky ReLU*.

Tabela 5.1: Resultados do treino e teste dos modelos propostos na Abordagem 1.

Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
LeNet	<i>ReLU</i>	4	10.53	13.55
LeNet	<i>Leaky ReLU</i>	8	38.33	40.82
AlexNet	<i>ReLU</i>	5	11.03	13.76
AlexNet	<i>Leaky ReLU</i>	5	39.27	41.97

5.2 Abordagem 2: Introduzindo *Data Augmentation*

A abordagem anterior consistiu essencialmente da utilização dos modelos tal como foram definidos e com uma simples operação de adequação dos dados de entrada por meio de normalização. Porém, em problemas de Visão Computacional, é comum aplicar técnicas de *data augmentation* com vistas a aumentar artificialmente o conjunto de dados, fazendo com que o modelo, em sua fase de treinamento, não seja exposto à mesma entrada em mais de uma ocasião. Esta estratégia de regularização colabora na mitigação do *overfitting* e costuma promover uma melhor generalização (??).

As técnicas de *data augmentation* consideradas foram a rotação entre 0 e 20 graus no sentido horário ou anti-horário, zoom de 0.8 a 1.2 vezes, inversão horizontal com probabilidade de ocorrência de 0.5 ou translação com probabilidade igual a 0.2.

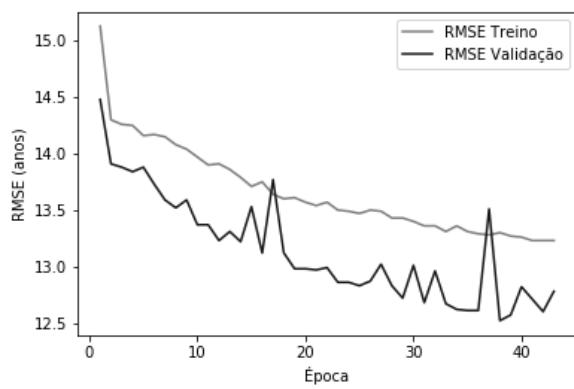
Desta feita, após o aumento artificial do conjunto de dados, partiu-se então para o treinamento e teste dos modelos conforme a metodologia adotada, considerando as arquiteturas LeNet e AlexNet e as funções de ativação *ReLU* e *Leaky ReLU*. Os gráficos das métricas de desempenho coletadas durante o treinamento e a reta-0 obtida a partir dos dados de teste em cada uma destas quatro configurações são ilustrados nas Figuras ?? e ??.

De maneira análoga, as métricas de desempenho coletadas encontram-se detalhadas na Tabela ???. Nota-se que o número de épocas no treinamento foi maior que a abordagem anterior, indicando que houve um cenário mais favorável para o aprendizado dos padrões nos dados. De maneira geral, as métricas obtidas não fornecem uma evidência forte de que esta segunda abordagem produz resultados mais significativos que a primeira mas, no caso da CNN AlexNet

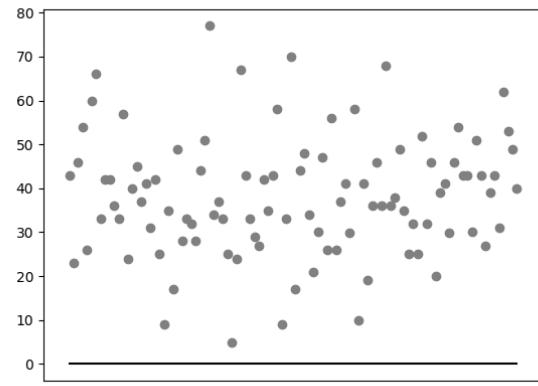
com *ReLU*, os resultados foram comparáveis. O efeito positivo esperado pelo *data augmentation* não se mostrou tão evidente quanto se esperava inicialmente. Porém, isto pode acontecer em razão dos valores dos hiperparâmetros e da necessidade de melhor pré-processamento das imagens antes da apresentação às CNNs, o que motivou a realização da abordagem a seguir.

Figura 5.3: Resultados do treinamento e teste da CNN LeNet de acordo com a Abordagem 2.

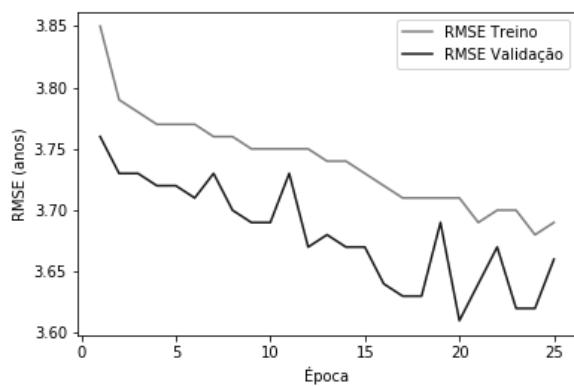
(a) RMSE de treinamento da arquitetura LeNet utilizando funções de ativação *ReLU*.



(b) Reta-0 LeNet *ReLU*.



(c) RMSE de treinamento da arquitetura LeNet utilizando funções de ativação *Leaky ReLU*.



(d) Reta-0 LeNet *Leaky ReLU*.

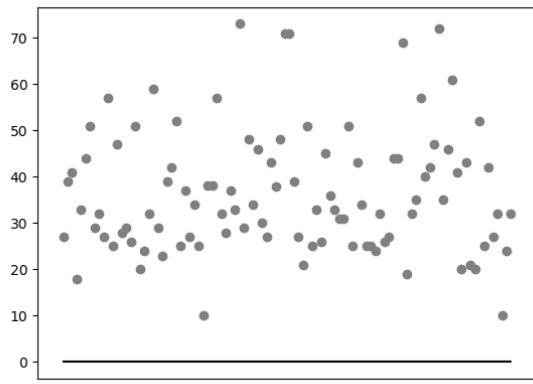
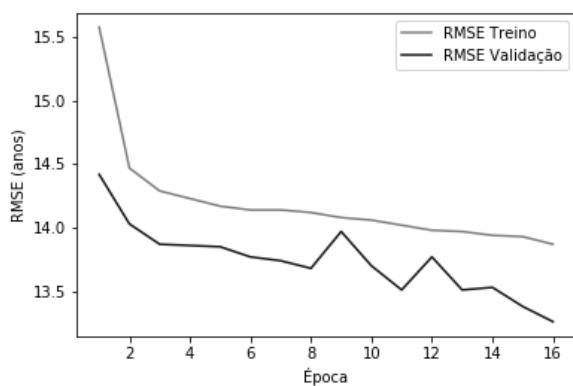
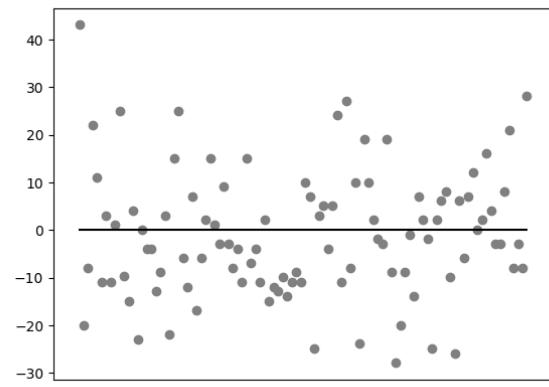


Figura 5.4: Resultados do treinamento e teste da CNN AlexNet de acordo com a Abordagem 2.

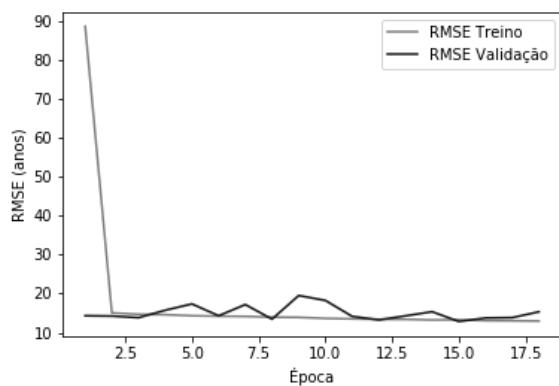
(a) Treinamento AlexNet *ReLU*.



(b) Reta-0 AlexNet *ReLU*.



(c) Treinamento AlexNet *Leaky ReLU*.



(d) Reta-0 AlexNet *Leaky ReLU*.

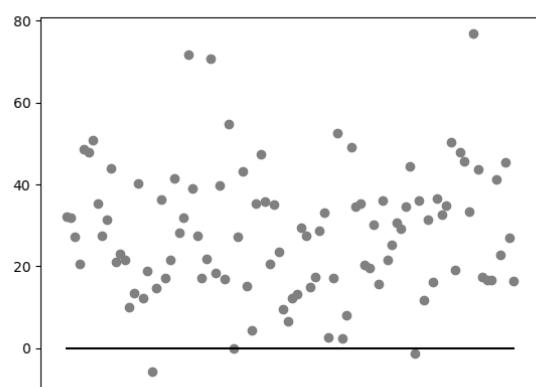


Tabela 5.2: Resultados do treino e teste dos modelos propostos na Abordagem 2.

Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
LeNet	<i>ReLU</i>	39	37.85	40.27
LeNet	<i>Leaky ReLU</i>	21	38.50	41.06
AlexNet	<i>ReLU</i>	16	11.59	14.59
AlexNet	<i>Leaky ReLU</i>	16	28.06	31.81

5.3 Abordagem 3: Introduzindo Equalização de Histograma

A terceira abordagem utilizou as imagens da base de dados normalizadas e técnicas de *data augmentation* previamente mencionadas. Além disto, visando melhorar as métricas de desempenho, introduziu-se o processo de equalização das imagens por histograma, que ajusta o contraste da imagem utilizando o histograma de cores. Conforme mencionado na Seção ??, este método aumenta o contraste global de imagens, especialmente quando os dados úteis da imagem são representados por cores próximas. No contexto da detecção de idade por meio da imagem da face de determinado indivíduo, a equalização por histograma reforça marcas de expressões e outras imperfeições ??.

Considerado uma nova equalização do conjunto de dados segundo o histograma, somado à normalização anterior e à aplicação de *data augmentation*, consolidou-se então o treinamento e teste das redes LeNet e AlexNet com função de ativação *ReLU* e *Leaky ReLU* segundo a terceira abordagem. Os gráficos de treinamento e a reta-0 obtidos encontram-se nas Figuras ?? e ??, respectivamente. A partir destes gráficos, observa-se um treinamento mais consistente, com queda progressiva da perda à medida que se avançam as épocas. Apesar disso, nota-se também que houve retrocessos na métrica de perda da validação ao longo do treinamento, o que promoveu *early stopping*.

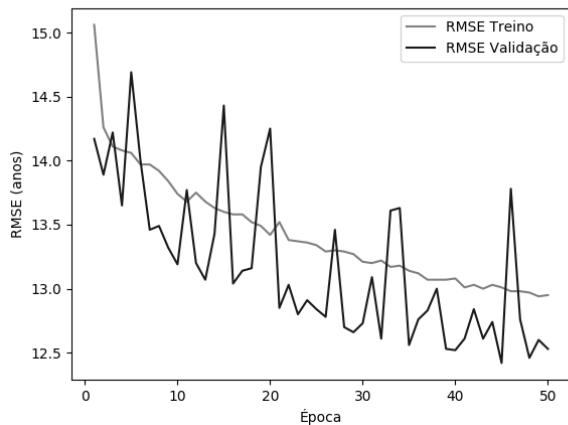
Obedecendo ao método de validação cruzada *holdout* previamente mencionado, os resultados desta abordagem encontram-se sintetizados na Tabela ?? . É interessante notar que o aumento nas épocas do treinamento não foi efetivo em obter melhores métricas de testes. Tem-se que, embora as CNNs aprendam por mais épocas que nos cenários anteriores, este aprendizado é menor para fins de generalização. Apesar de estar havendo um esforço de pré-processamento por meio de *data augmentation* e de equalização por histograma, seguindo práticas tipicamente adotadas em trabalhos desta área ??, as melhorias promovidas por estas estratégias são de cunho heurístico, podendo ser boas em determinados cenários, mas não garantidamente em todos, tal como ocorreu no contexto em questão.

Tabela 5.3: Resultados do treino e teste dos modelos propostos na Abordagem 3.

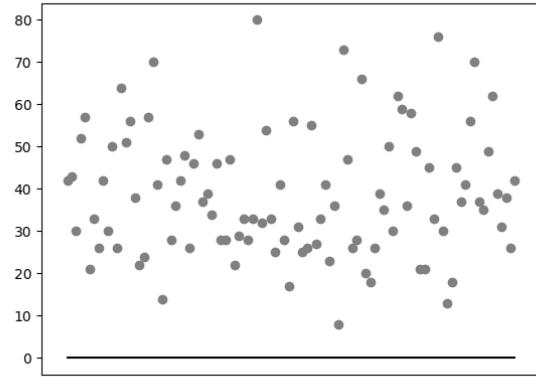
Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
LeNet	<i>ReLU</i>	46	38.66	41.20
LeNet	<i>Leaky ReLU</i>	38	38.26	40.85
AlexNet	<i>ReLU</i>	7	13.10	15.88
AlexNet	<i>Leaky ReLU</i>	18	35.25	38.04

Figura 5.5: Resultados do treinamento e teste da CNN LeNet de acordo com a Abordagem 3.

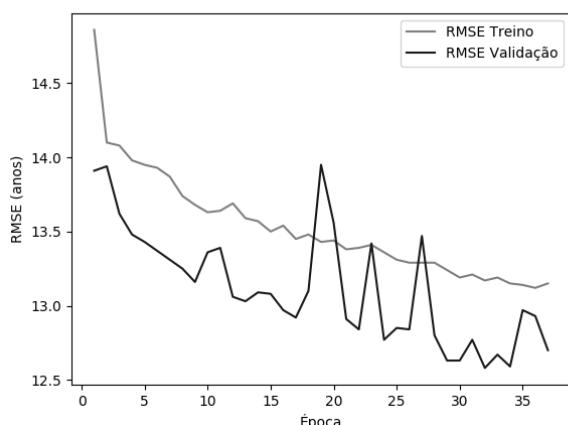
(a) RMSE de treinamento da arquitetura LeNet utilizando funções de ativação *ReLU*.



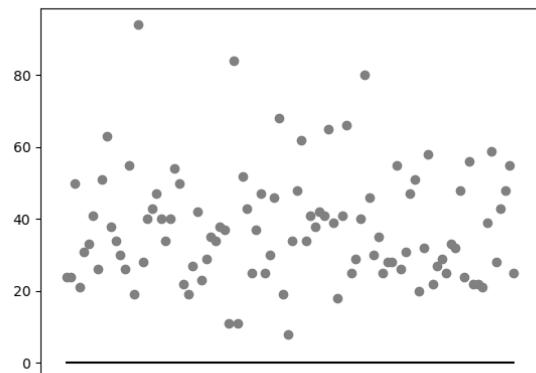
(b) Reta-0 LeNet *ReLU*.



(c) RMSE de treinamento da arquitetura LeNet utilizando funções de ativação *Leaky ReLU*.



(d) Reta-0 LeNet *Leaky ReLU*.

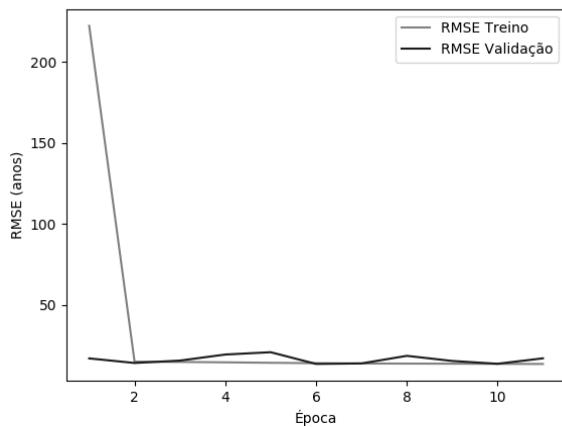


5.4 Abordagem 4: Utilizando MAE para o Cálculo da Perda

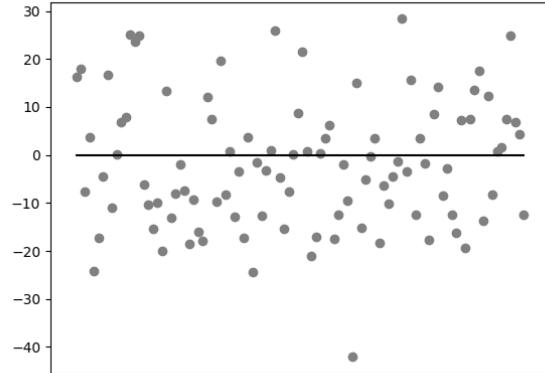
A análise dos gráficos de treinamento das redes anteriores levou à suposição de que a métrica utilizada para a atualização dos pesos RMSE estivesse trazendo instabilidade para o treinamento. Desta maneira, esta abordagem considera a rede com melhor desempenho verificado até então, a LeNet com função de ativação *ReLU*, imagens normalizadas mas sem *data augmentation*.

Figura 5.6: Resultados do treinamento e teste da CNN AlexNet de acordo com a Abordagem 3.

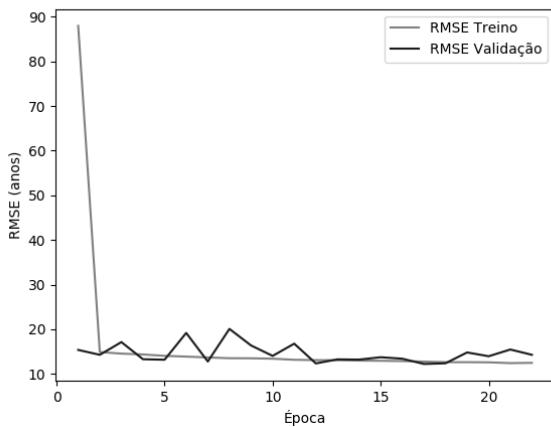
(a) RMSE de treinamento da arquitetura AlexNet utilizando funções de ativação *ReLU*.



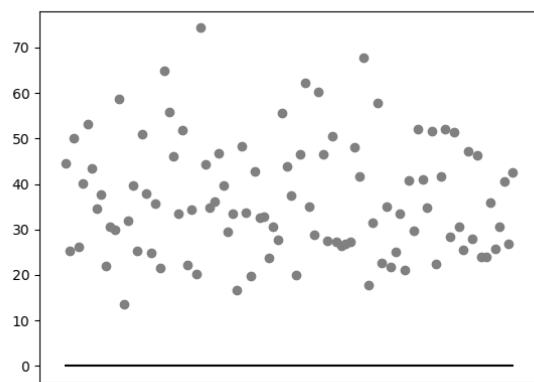
(b) Reta-0 AlexNet *ReLU*.



(c) RMSE de treinamento da arquitetura AlexNet utilizando funções de ativação RMSE de treinamento da arquitetura LeNet utilizando funções de ativação *Leaky ReLU*.



(d) Reta-0 AlexNet *Leaky ReLU*.

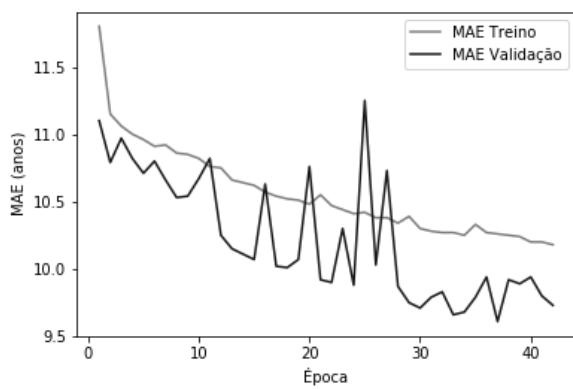


mentation ou equalização por histograma descrita ainda na Abordagem 1. Porém, decidiu-se sujeitar as entradas desta rede às técnicas de *data augmentation* e equalização por histograma, mas com a utilização do MAE para cálculo da perda.

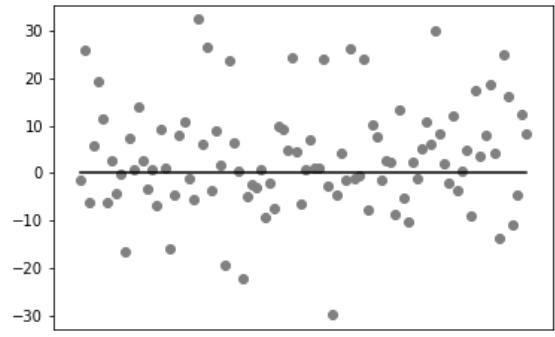
Esta abordagem foi considerada em virtude do treinamento com o otimizador *Adam* construir superfícies estocásticas de erro a partir da métrica de perda. Assim, após treinamento e teste, obteve-se os gráficos ilustrados nas Figuras ??.

Figura 5.7: Resultados do treinamento e teste da CNN LeNet de acordo com a Abordagem 4.

(a) MAE de treinamento da arquitetura LeNet utilizando funções de ativação *ReLU*.



(b) Reta-0 LeNet *ReLU*.



Obedecendo ao método de validação cruzada *holdout* previamente mencionado, os resultados desta abordagem encontram-se sintetizados na Tabela ??.

Tabela 5.4: Resultados do treino e teste do modelo proposto na Abordagem 4.

Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
LeNet	<i>Leaky ReLU</i>	38	9.98	12.91

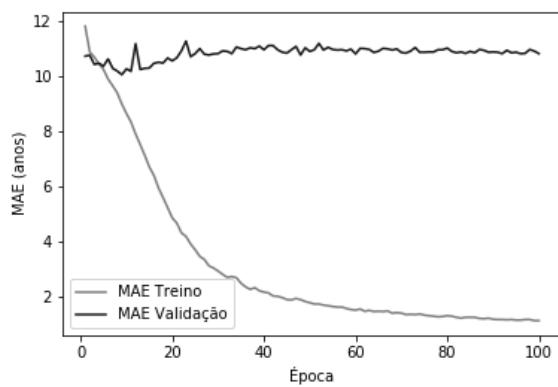
Os resultados obtidos evidenciam melhores métricas de desempenho até então, oferecendo diretrizes positivas a cerca das estratégias adotadas. Apesar da identificação de uma rede satisfatória até então, fica o questionamento se as melhorias obtidas em si foram introduzidas pelas heurísticas de pré-processamento ou se são intrínsecas do modelo adotado. Como a equalização por histograma e os processos de *data augmentation* introduzem ônus de pré-processamento aos dados, desejou-se verificar se o desempenho positivo desta rede seria mantido mesmo sem tais práticas, o que motivou a realização da abordagem seguinte.

5.5 Abordagem 5: LeNet Apenas com Normalização da Entrada

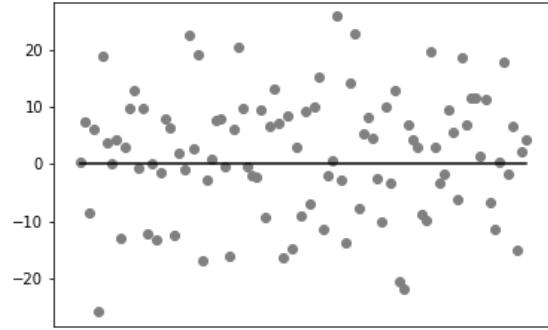
A quinta abordagem também adotou a rede com melhor desempenho obtido até o momento, a LeNet descrita na Abordagme 1, com função de ativação *ReLU*, treinada apenas com imagens da base de dados normalizadas, mas sem equalização de histograma nem tampouco técnicas de *data augmentation*. Seguiu-se utilizando a métrica MAE para o cálculo da perda e da atualização dos pesos. Buscando garantir maior estabilidade nas métricas de desempenho durante o treinamento, aumentou-se o tamanho do batch para 128, com vistas a estimar melhor a superfície de erro à medida que o algoritmo de otimização percorre o gradiente descendente.

Figura 5.8: Resultados do treinamento e teste da CNN LeNet de acordo com a Abordagem 5.

(a) MAE de treinamento da arquitetura LeNet utilizando funções de ativação *ReLU*.



(b) Reta-0 LeNet *ReLU*.



Obedecendo ao método de validação cruzada *holdout*, os resultados desta abordagem encontram-se sintetizados na Tabela ???. É interessante notar que não houve melhorias na métrica de desempenho, pelo contrário, o erro aumentou em relação ao cenário anterior, sugerindo que as práticas de pré-processamento colaboraram positivamente no melhor aprendizado das características da tarefa considerada. Uma vez que foram exploradas diferentes práticas de treinamento das redes LeNet e AlexNet, em que as melhorias alcançadas estagnaram após a quarta abordagem, partiu-se então para a avaliação de arquiteturas mais profundas aplicadas ao problema considerado, o que ensejou as abordagens a seguir.

Tabela 5.5: Resultados do treino e teste dos modelos propostos na Abordagem 5.

Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
LeNet	<i>ReLU</i>	9	10.09	13.04

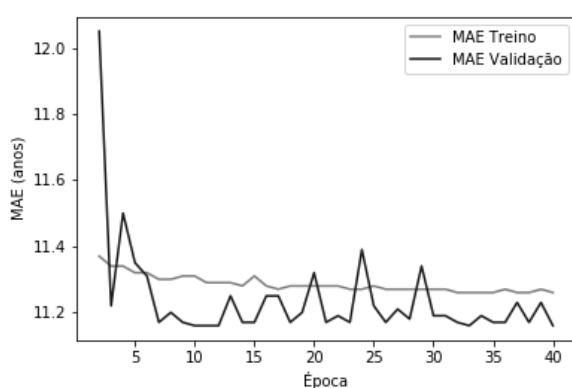
5.6 Abordagem 6: VGG-16 e Dados Normalizados

A sexta abordagem considerou a utilização da arquitetura VGG-16, previamente apresentada na Seção ???. Para utilização neste contexto, de maneira análoga aos cenários anteriores, removeu-se a camada de saída, tipicamente utilizada para fins de classificação, e adicionou-se uma camada densa com função de ativação *ReLU*. Embora tipicamente esta rede seja utilizada com *transfer learning* do conjunto de dados ImageNet, optou-se por seguir a prática adotada nas demais redes, com inicialização aleatória dos pesos e treinamento integral com o conjunto de dados apenas sujeito à normalização.

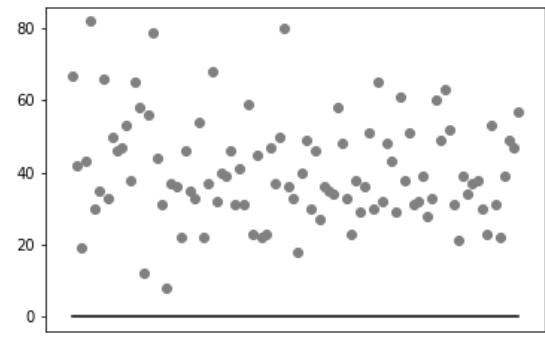
Em decorrência do treinamento e teste nos mesmos moldes dos cenários anteriores, obteve-se os gráficos ilustrados na Figura ???. As métricas de desempenho obtidas são elencadas na Tabela ???.

Figura 5.9: Resultados do treinamento e teste da CNN VGG-16 de acordo com a Abordagem 6.

(a) MAE de treinamento da arquitetura VGG-16 utilizando funções de ativação *ReLU*.



(b) Reta-0 VGG-16 *ReLU*.



É interessante notar que a rede VGG-16 é mais profunda em termos de camadas que as redes LeNet e AlexNet, o que impacta em uma maior quantidade de parâmetros livres a serem calculados e que acaba por exigir mais recursos computacionais para seu treino. Segundo

Tabela 5.6: Resultados do treino e teste dos modelos propostos na Abordagem 6.

Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
VGG-16	<i>ReLU</i>	11	40.90	38.35

as ideias de DL, imagina-se que uma rede com um número maior de camadas hierárquicas seja capaz de produzir melhores representações, impactando positivamente no aprendizado da tarefa. Apesar disso, teve desempenho aquém. Uma suposição para este resultado é que a rede sofreu *dying ReLU problem*, pois apenas produziu saídas iguais a 37.05 para todas as imagens dadas como entrada, maximizando o erro aferido no teste. Para contornar este problema, esta arquitetura foi novamente testada, mas introduzindo também *data augmentation* e normalização por equalização de histograma.

5.7 Abordagem 7: VGG-16 com *Data Augmentation* e Equalização de Histograma

A rede VGG-16 utilizada nesta abordagem foi instanciada e treinada com os mesmos parâmetros descritos na Abordagem 6, porém, passou-se a utilizar *data augmentation* e equalização de histograma, obedecendo às mesmas práticas de configuração utilizada nas Abordagens 2 a 4 para as redes LeNet e AlexNet e dos resultados positivos observados.

Os resultados obtidos do treinamento e teste encontram-se ilustrados nos gráficos do treinamento e reta-0 da Figura ?? e detalhados na Tabela ???. É interessante notar que as técnicas para mitigar *overfitting* não foram efetivas para contornar o *dying ReLU problem* previamente verificado, impactando em métricas análogas de erro.

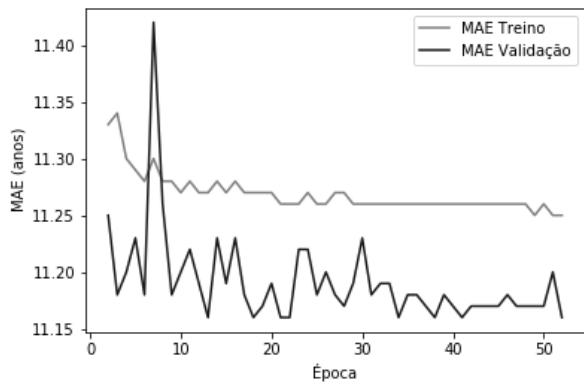
Tabela 5.7: Resultados do treino e teste dos modelos propostos na Abordagem 7.

Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
VGG-16	<i>ReLU</i>	23	40.99	38.39

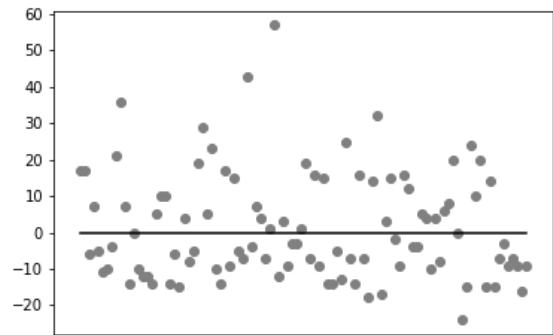
A partir da análise dos resultados obtidos, observa-se a VGG-16 também sofreu dificuldades durante o treinamento. Esta rede possivelmente foi vítima de *dying ReLU problem*, mesmo

Figura 5.10: Resultados do treinamento e teste da CNN VGG-16 de acordo com a Abordagem 7.

(a) MAE de treinamento da arquitetura VGG-16 utilizando funções de ativação *ReLU*.



(b) Reta-0 VGG-16 *ReLU*.



havendo mudanças nas entradas apresentadas, haja vista que o modelo obtido testado após o treino seguindo a abordagem descrita previa apenas o valor 37.02 para todas as imagens de entrada apresentadas. Portanto, optou-se por realizar ajustes diretamente nos parâmetros da rede, conforme é descrito na próxima solução para treinamento considerada.

5.8 Abordagem 8: VGG-16 com *data augmentation*, *Transfer Learning* e *Leaky ReLU*

A abordagem agora considerada tenta endereçar de maneira mais objetiva a superação do *dying ReLU problem*, pois considera a adoção da função de ativação *Leaky ReLU* que não está sujeita à este fenômeno, segundo a literatura (??). Além disto, tentou-se utilizar uma taxa de aprendizado inicial maior, igual a 0.003, tentando favorecer uma caminhada mais objetiva no gradiente descendente em função do método de otimização utilizado. É importante salientar que, embora possa implicar em mais avanços que impliquem na eventual convergência, esta taxa de aprendizado torna o treinamento consideravelmente mais instável. *Data augmentation* e equalização de histograma foram empregados seguindo os mesmos objetivo e configuração descritos em estratégias anteriores. *Transfer Learning* de pesos obtidos a partir do treinamento

da VGG-16 para classificação de objetos por meio da base de dados ImageNet foi empregado com o objetivo de inicializar os pesos de maneira a facilitar a convergência do modelo.

O comportamento do MAE durante o treinamento e a reta-0 obtida durante o teste da VGG-16 obtida utilizando estas configurações podem ser visualizados na Figura ???. Na Tabela ?? estão as métricas de desempenho RMSE e MAE.

Figura 5.11: Resultados do treinamento e teste da CNN VGG-16 de acordo com a Abordagem 8.

(a) MAE de treinamento da arquitetura VGG-16 utilizando funções de ativação *Leaky ReLU*.

(b) Reta-0 VGG-16 *Leaky ReLU*.

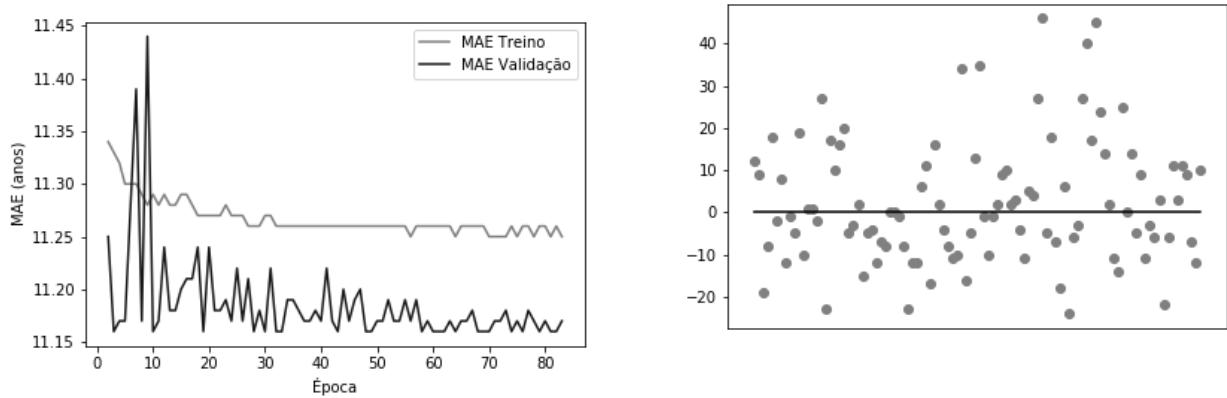


Tabela 5.8: Resultados do treino e teste dos modelos propostos na Abordagem 8.

Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
VGG-16	<i>Leaky ReLU</i>	83	11.02	14.24

Assim como as VGG-16 treinadas anteriormente, esta CNN prevê um valor fixo para qualquer entrada apresentada, neste caso 36.99. Este é um indicador de que o modelo continua sofrendo *dying ReLU problem*, apesar da configuração descrita nesta abordagem. Salienta-se que existem diversas estratégias, descritas na literatura, capazes de contornar este problema. Porém, dada a característica empírica e o tempo de treinamento das CNNs, em especial modelos extensos como a VGG-16, optou-se por buscar um modelo de CNN tido como mais simples.

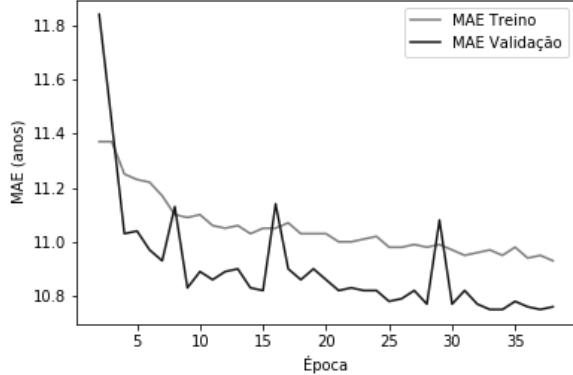
5.9 Abordagem 9: SqueezeNet

A última abordagem considerada neste trabalho, buscando sempre a perspectiva de melhores métricas e uma boa adequação ao contexto final de utilização do modelo, tratou do treinamento e teste da rede SqueezeNet. Dada a proposição recente deste modelo na literatura, ainda há poucos guias práticos sobre sugestões de uso e inicialização. Assim, esta arquitetura foi mantida tal como inicialmente proposta, mas considerou-se o uso de *data augmentation* e equalização de histograma, objetivando os mesmos ganhos verificados nas abordagens anteriores melhor sucedidas até então.

Os resultados obtidos pelo treinamento e teste encontram-se ilustrados nos gráficos do treinamento e reta-0 da Figura ?? e detalhados na Tabela ??.

Figura 5.12: Resultados do treinamento e teste da CNN SqueezeNet de acordo com a Abordagem 9.

(a) MAE de treinamento da arquitetura SqueezeNet utilizando função de ativação *ReLU* e *data augmentation*.



(b) Reta-0 SqueezeNet.

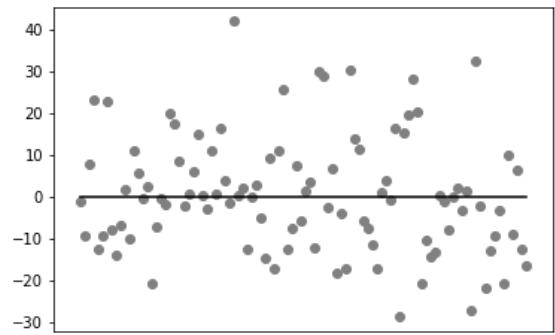


Tabela 5.9: Resultado do teste da SqueezeNet proposta na Abordagem 9.

Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
SqueezeNet	<i>ReLU</i>	38	10.72	13.84

A SqueezeNet obteve métricas de desempenho satisfatórias considerando os valores obtidos por outras redes treinadas utilizando diferentes abordagens. Esta CNN foi capaz de gerar valores diferenciados para cada imagem de entrada, ou seja, não caiu em *dying ReLU problem*

mesmo empregando *ReLU* como função de ativação para suas camadas. Estes resultados são expressivos ao se considerar os recursos necessários para executar um sistema de tempo real utilizando uma CNN, já que a SqueezeNet ocupa menos de 1 MB de memória (??) e obteve, para este problema, resultados similares aos de redes que necessitam de mais espaço e poder de processamento, a citar LeNet e AlexNet.

5.10 Sumarizando os Resultados

Considerando todas as abordagens conduzidas, a Tabela ?? sintetiza todos os treinamentos e testes realizados e os resultados obtidos da métrica de desempenho MAE adotada para esta tarefa. Pode-se identificar então que, para a tarefa elencada, a rede com melhor desempenho observado foi a LeNet descrita na Abordagem 4, que se utiliza de funções de ativação *ReLU*, com imagens de entrada sujeitas às técnicas de *data augmentation* e equalização por histograma. Esta abordagem também trouxe a mudança de métrica utilizada para o cálculo da perda e da atualização dos pesos durante o treinamento de RMSE para MAE, o que acabou trazendo mais estabilidade para o treinamento das redes.

É interessante notar que o ajuste de parâmetros e hiperparâmetros para as redes neurais convolucionais segue as mesmas dificuldades das redes neurais *multilayer perceptron*, em que a superfície de erro é hiperdimensional e complexa, e que, quanto maior o número de pesos na rede, maior é o espaço de busca pelos parâmetros ideais, dificultando, por conseguinte, a tarefa de encontrá-los (??). No caso das CNNs, em especial, soma-se isto ao fato do aprendizado destas redes ser naturalmente mais oneroso em virtude do número de operações necessárias para realização das convoluções e do algoritmo de *backpropagation*.

Tabela 5.10: Sumário dos resultados obtidos de todas as abordagens conduzidas.

Abordagem	Rede	Função de ativação	Épocas	MAE Teste	RMSE Teste
1	LeNet	<i>ReLU</i>	4	10.53	13.55
1	LeNet	<i>Leaky ReLU</i>	8	38.33	40.82
1	AlexNet	<i>ReLU</i>	5	11.03	13.76
1	AlexNet	<i>Leaky ReLU</i>	5	39.27	41.97
2	LeNet	<i>ReLU</i>	39	37.85	40.27
2	LeNet	<i>Leaky ReLU</i>	21	38.50	41.06
2	AlexNet	<i>ReLU</i>	16	11.59	14.59
2	AlexNet	<i>Leaky ReLU</i>	16	28.06	31.81
3	LeNet	<i>ReLU</i>	46	38.66	41.20
3	LeNet	<i>Leaky ReLU</i>	38	38.26	40.85
3	AlexNet	<i>ReLU</i>	7	13.10	15.88
3	AlexNet	<i>Leaky ReLU</i>	18	35.25	38.04
4	LeNet	<i>Leaky ReLU</i>	38	9.98	12.91
5	LeNet	<i>ReLU</i>	9	10.09	13.04
6	VGG-16	<i>ReLU</i>	11	40.90	38.35
7	VGG-16	<i>ReLU</i>	23	40.99	38.39
8	VGG-16	<i>ReLU</i>	83	11.02	14.24
9	SqueezeNet	<i>ReLU</i>	38	10.72	13.84

Capítulo 6

Considerações Finais

Este trabalho teve por objetivo elaborar estratégias inteligentes para estimação de idade de telespectadores de *Smart TVs* a partir de suas respectivas fotografias faciais. Para endereçar este problema, consolidou-se um base de dados contendo 47.950 exemplos de faces centralizadas e com as respectivas idades dos sujeitos retratados. Estes dados foram então utilizados para treinar e testar redes neurais convolucionais para a tarefa de Aprendizado de Máquina considerada segundo uma abordagem de validação cruzada do tipo *Holdout*.

Considerando o modelo adotado, foram conduzidas várias abordagens de treino e teste utilizando arquiteturas canônicas de CNNs, a citar: LeNet, AlexNet, VGG-16 e SqueezeNet, esta última, em particular, oriunda de um trabalho recente na literatura. Houve uma adaptação na camada de saída destas redes para uma tarefa de regressão e variou-se o parâmetro de função de ativação para os valores *ReLU* e *Leaky ReLU*. Considerou-se também técnicas de normalização da entrada, equalização de histograma e ainda *data augmentation*. Estas redes, parâmetros, hiperparâmetros e técnicas foram explorados de maneira sistemática em nove cenários distintos.

De maneira geral, não foi possível constatar uma progressiva melhora ao passo que as abordagens foram conduzidas, fato decorrente dos desafios na busca de soluções ideias em redes neurais quando considerado o aumento no espaço de busca. Apesar disso, o trabalho foi conduzido conforme práticas de natureza heurística sugeridas por diversos autores da literatura, considerando modelos compatíveis com o estado da arte. Os resultados obtidos mostraram que a melhor CNN para esta tarefa foi LeNet sujeita à abordagem 4, a qual considerou a entrada de

dados com *data augmentation*, equalização por histograma, função de ativação *ReLU* e MAE para o cálculo da perda, obtendo-se um MAE de 9.98 anos.

Embora os resultados aqui apresentados não tenham superado o estado da arte na literatura para este problema, pretensão não esperada para o escopo de um trabalho de conclusão de curso em face à comparação com trabalhos de níveis mais avançados, os resultados obtidos ressaltam a condução assertiva da metodologia, ilustram a prática de métodos e tecnologias desta área de pesquisa emergente e propõem um patamar para comparação com outros autores que não utilizam técnicas de extração de características nem tampouco *ensembles* de CNNs, em contraste como os melhores resultados observados atualmente.

Como trabalhos futuros considera-se a abordagem deste problema como uma tarefa de classificação e a avaliação do impacto da utilização de técnicas de *transfer learning*. Há ainda a possibilidade de uma análise mais detalhada dos grupos de idade com maior e menor índice de erro, na tentativa de obter estimadores mais apropriados considerando as características de cada grupo. Continuar endereçando este problema é de suma importância do ponto de vista prático para o desenvolvimento de diversas soluções de recomendação de conteúdo e controle parental em *Smart TVs*, auxiliando no desenvolvimento destas soluções tecnológicas.