

**SÉRGIO ALEXANDRE ARRUDA PINHEIRO**

**REDES NEURAIS CONVOLUCIONAIS APLICADAS AO  
RECONHECIMENTO AUTOMÁTICO DE CAPTCHAS**

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Sistemas de Informação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Bacharel em Sistemas de Informação.

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

Manaus – Dezembro – 2018

## **Ficha Catalográfica**

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).  
**Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.**

P654rr	<p>Pinheiro, Sergio Alexandre Arruda</p> <p>Redes neurais convolucionais aplicadas ao reconhecimento automático de CAPTCHAs / Sergio Alexandre Arruda Pinheiro. Manaus : [s.n], 2018.</p> <p>58 f.: color.; 33 cm.</p> <p>TCC - Graduação em Sistemas de Informação - Bacharelado - Universidade do Estado do Amazonas, Manaus, 2018.</p> <p>Inclui bibliografia</p> <p>Orientador: Elloá B. Guedes</p> <p>1. CAPTCHA. 2. Redes Neurais Artificiais Convolucionais. 3. Aprendizado de Máquina. 4. Inteligência Artificial. I. Elloá B. Guedes (Orient.). II. Universidade do Estado do Amazonas. III. Redes neurais convolucionais aplicadas ao reconhecimento automático de CAPTCHAs</p>
--------	---

SÉRGIO ALEXANDRE ARRUDA PINHEIRO

REDES NEURAIS CONVOLUCIONAIS APLICADAS AO  
RECONHECIMENTO AUTOMÁTICO DE CAPTCHAS

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Sistemas de Informação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Bacharel em Sistemas de Informação.


Aprovado em: 30/11/2018

BANCA EXAMINADORA



Profa. Dra. Elloá Barreto Guedes da Costa

UNIVERSIDADE DO ESTADO DO AMAZONAS



Prof. Raimundo Corrêa de Oliveira, D.Sc.

UNIVERSIDADE DO ESTADO DO AMAZONAS



Prof. Jucimar Maia da Silva Jr., D.Sc.

UNIVERSIDADE DO ESTADO DO AMAZONAS

# Resumo

Este trabalho aborda a utilização de Redes Neurais Convolucionais para a resolução automática de CAPTCHAs textuais da Plataforma Lattes. Para endereçar este problema foram capturados CAPTCHAs desta plataforma que então foram processados com técnicas de Visão Computacional para subtração de fundo e segmentação de caracteres individuais. Estes caracteres foram então rotulados manualmente e organizados segundo uma partição fixa do tipo *holdout* com 70% dos dados para treino, 10% para validação e 20% para teste, contendo 17.781 imagens de 28 caracteres distintos que seriam então usada para uma tarefa de classificação multiclasse em base de dados desbalanceada. Considerando a similaridade com o problema de reconhecimento de dígitos manuscritos MNIST, adotou-se a arquitetura canônica LeNet para endereçar o problema mediante aprendizado supervisionado. Apesar das semelhanças, o resultado obtido após as etapas de treino e teste foi desprezível e partiu-se então para novas abordagens de ajustes de parâmetros mediante busca em *grid*. Desta busca, organizada em dois cenários, foram treinos e testados 636 adaptações ao modelo original LeNet em que, com a utilização de regularização de pesos e normalização da entrada, foi possível identificar um modelo que endereçava a tarefa com micro F-Score de 0.9784. Os resultados obtidos foram encorajadores, sugerindo superação ao estado da arte para a resolução automática de CAPTCHAs da plataforma considerada.

**Palavras Chave:** CAPTCHA, Redes Neurais Artificiais Convolucionais, Aprendizado de Máquina, Inteligência Artificial.

# *Abstract*

*This paper discusses the use of convolutional neural networks for automatic resolution of textual CAPTCHAs of the Lattes Platform. To address this problem were caught several CAPTCHAs of this platform that were then processed with computer vision techniques for subtracting background and individual characters segmentation. These characters were then manually labeled and organized according to a fixed holdout partition with 70% of data for training, 10% for validation and 20% for testing, containing 17.781 images of 28 distinct characters that would then be used for a multiclass classification task in an unbalanced database. Considering the similarity with the MNIST handwriting recognition problem, the LeNet canonical architecture was adopted to address the problem through supervised learning. In spite of the similarities, the result obtained after the training and test was negligible so were adopted new approaches of parameter adjustments through grid search. From this search, organized in two scenarios, 636 adaptations were tested to the original LeNet model in which, with the use of weight regularization and input normalization, it was possible to identify a model that addressed the micro-F-Score task of 0.9784. The obtained results were encouraging, suggesting overcoming to the state of the art for the automatic resolution of CAPTCHAs of the considered platform.*

**Key Words:** CAPTCHA, Convolutional Artificial Neural Networks, Machine Learning, Artificial Intelligence.

# Agradecimentos

À Universidade do Estado do Amazonas, seu corpo docente, direção e administração por me proporcionarem as condições de ensino com qualidade para me tornar um profissional completo, ético e de valor à sociedade.

À minha orientadora Profa. Dra. Elloá Barreto Guedes da Costa, pela oportunidade, confiança e dedicação que se fizeram fundamentais para a concretização deste trabalho.

À minha mãe, meu pai e minha irmã, pelo amor, incentivo e apoio que construíram a pessoa que sou, não só durante o período deste trabalho, mas por toda a minha vida.

À minha esposa e amigos, por todos os momentos importantes e inesquecíveis durante essa jornada.

Agradeço também 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.

Por fim, à todos que fizeram parte, direta ou indiretamente, da minha formação, o meu muito obrigado.

## **Epígrafe**

*We can only see a short distance ahead, but we can see plenty there that needs to be done.*

Alan Turing

# Sumário

<b>Lista de Tabelas</b>	<b>ix</b>
<b>Lista de Figuras</b>	<b>xi</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Objetivos . . . . .	3
1.2 Justificativa . . . . .	3
1.3 Metodologia . . . . .	4
1.4 Cronograma . . . . .	5
1.5 Organização do Documento . . . . .	5
<b>2 Fundamentação Teórica</b>	<b>7</b>
2.1 CAPTCHA . . . . .	7
2.2 Aprendizado de Máquina . . . . .	10
2.3 Redes Neurais Artificiais . . . . .	12
2.3.1 Neurônio Artificial e Redes Perceptron . . . . .	13
2.3.2 Redes Perceptron de Múltiplas Camadas . . . . .	16
2.3.3 Redes Neurais Convolucionais . . . . .	20
2.4 Tecnologias Utilizadas . . . . .	25
<b>3 Solução Proposta</b>	<b>26</b>
3.1 Consolidação da Base de Dados . . . . .	26
3.2 Visão Geral da Base de Dados . . . . .	29



3.3	Tarefa de Classificação e Métricas de Desempenho . . . . .	30
3.4	Modelos Propostos . . . . .	31
<b>4</b>	<b>Trabalhos Relacionados</b>	<b>32</b>
<b>5</b>	<b>Resultados e Discussão</b>	<b>34</b>
5.1	Cenário 1: LeNet e Parâmetros Típicos . . . . .	34
5.2	Cenário 2: LeNet, Busca em <i>Grid</i> de Parâmetros e Regularização de Pesos . . .	36
5.3	Cenário 3: Normalizando os Dados de Entrada . . . . .	39
5.4	Cenário 4: Ampliando o Ajuste de Parâmetros da LeNet . . . . .	40
<b>6</b>	<b>Considerações Finais</b>	<b>43</b>

# Lista de Tabelas

1.1	Cronograma para execução das atividades referentes à este trabalho de conclusão de curso. . . . .	6
5.1	Métricas resultantes da etapa de testes das 288 CNNs propostas no Cenário 2. .	38
5.2	Métricas resultantes da etapa de testes das 288 CNNs propostas no Cenário 3. .	39
5.3	Métricas resultantes da etapa de testes das 384 CNNs propostas no Cenário 4. .	41

# Lista de Figuras

2.1	Exemplos de CAPTCHAs de texto. Fonte: (GOOGLE, 2018a) . . . . .	9
2.2	CAPTCHA baseado em imagem. Fonte: (GOOGLE, 2018a) . . . . .	10
2.3	Exemplos de CAPTCHAs de questão. Fonte: (GOOGLE, 2018a) . . . . .	10
2.4	Exemplo de caixa de diálogo de validação no reCAPTCHA. Fonte: (GOOGLE, 2018b) . . . . .	11
2.5	Modelo do neurônio de McCulloch e Pitts (BRAGA; CARVALHO; LUDERMIR, 2007) . . . . .	14
2.6	Exemplos de funções que podem ser utilizadas como função de ativação. . . . .	15
2.7	Modelo perceptron. Fonte: (BRAGA; CARVALHO; LUDERMIR, 2007) . . . . .	15
2.8	Objetos lineramente separáveis (FACELI et al., 2015) . . . . .	17
2.9	Exemplo de rede neural artificial com camadas ocultas. Fonte: (FACELI et al., 2015). . . . .	17
2.10	Papel desempenhado pelos neurônios nas diferentes camadas de uma rede MLP. Fonte: (FACELI et al., 2015). . . . .	18
2.11	Processo em que uma CNN inicia aprendendo características simples e a medida que aprofunda suas camadas, torna o padrão mais completo, por fim faz a classificação (KHAN et al., 2018). . . . .	20
2.12	Exemplo de convolução aplicada a uma entrada. Fonte: (BUDUMA, 2017). . . . .	21
2.13	Exemplos de arquiteturas de CNN variando em sua complexidade. O último exemplo consiste em uma arquitetura VGG. Fonte: (BUDUMA, 2017). . . . .	23

2.14	Arquitetura do modelo LeNet-5 sendo usada em um dígito do dataset MNIST (KHAN et al., 2018).	24
3.1	Script para download dos CAPTCHAs da Plataforma Lattes.	27
3.2	Exemplos de CAPTCHAs fornecidos pela Plataforma Lattes.	28
3.3	CAPTCHA antes e depois da subtração de fundo.	28
3.4	CAPTCHA antes e depois da dilatação.	29
3.5	Processo de segmentação do CAPTCHA.	29
3.6	Histograma da quantidade de dígitos classificados.	30
5.1	Métricas do cenário 1.	35
5.2	Matriz de confusão do Cenário 1.	36
5.3	Matriz de confusão do Cenário 2.	38
5.4	Matriz de confusão do Cenário 3.	40
5.5	Matriz de confusão do Cenário 4.	42

# Capítulo 1

## Introdução

O CAPTCHA, acrônimo para *Teste de Turing Público Completamente Automatizado para Diferenciação entre Computadores e Humanos*, é um recurso tecnológico que tem a finalidade de identificar se o usuário que está tentando acessar um determinado serviço é um ser humano ou uma máquina (YAN; AHMAD, 2008). Atuando como ferramenta, o CAPTCHA consiste na apresentação de um determinado padrão e na solicitação da sua identificação, para posterior permissão de acesso a um conteúdo ou serviço. A utilização de padrões e a solicitação para sua identificação baseiam-se na premissa de que esta é uma tarefa trivial para humanos, mas de difícil realização automática por algoritmos tradicionais. Assim, quando um CAPTCHA é resolvido, garante-se o provimento das informações solicitadas, pois acredita-se que um usuário humano tenha reconhecido apropriadamente o padrão apresentado (AHN; BLUM; LANGFORD, 2004). Os sistemas de CAPTCHA podem ser de diversos tipos, tais como baseados em texto, imagem, questões, etc. Nos CAPTCHAs baseados em texto, por exemplo, caracteres como letras e números devem ser identificados, mas são apresentados distorcidos e em fundo ruidoso, que dificultam o reconhecimento automático, mas em que nada impedem um fácil reconhecimento por pessoas, até mesmo crianças.

A mais frequente utilização de CAPTCHAs acontece em páginas e serviços disponíveis via internet. Cita-se, por exemplo, aqueles disponibilizados pelo Governo Brasileiro, com o intuito de proteger os dados e serviços prestados. A Receita Federal, por exemplo, visando impedir consultas indesejadas a CPFs e CNPJs, faz uso de CAPTCHAs de texto. (SANTOS, 2016).

---

Em especial, a Plataforma Lattes, base de dados de pesquisadores e de suas informações acadêmicas, também faz uso de CAPTCHAs baseados em texto para prevenir acessos em massa e prover uma camada de segurança ao disponibilizar as informações. Ao passo que esta estratégia minimiza as chances de ataques de acesso em massa, comumente do tipo DoS (*Deny of Service*), dificulta o desenvolvimento de *webcrawlers*, que poderiam explorar e analisar os dados disponíveis, propiciando o desenvolvimento de diversos trabalhos acerca do perfil das publicações e dos pesquisadores atuantes no País. Ao dificultar o acesso automatizado, estes CAPTCHAs contrariam a perspectiva de Transparência dos Dados, amplamente difundida e estimulada pelo governo.

Considerando a minimização desta problemática, este trabalho tem por objetivo endereçar a resolução automática do CAPTCHA de texto disponível atualmente na Plataforma Lattes. Para tanto, considerando uma análise preliminar dos padrões utilizados neste CAPTCHA, almeja-se investigar o potencial de aplicações de técnicas e métodos da Aprendizagem de Máquina neste contexto.

As soluções baseadas em Aprendizagem de Máquina são caracterizadas por algoritmos que se modificam e adaptam para fazer previsões ou tomadas de decisão com um índice de precisão crescente (MARSLAND, 2015). Esse modelo de computação é adequado para a resolução de problemas em que não se conhece uma solução analítica, mas que dados históricos do problema podem fornecer padrões para o aprendizado de uma solução. São exemplos de modelos de Aprendizagem de Máquina: árvores de decisão,  $k$ -NN, *Naive Bayes*, redes neurais artificiais, dentre diversos outros (FACELI et al., 2015).

Neste trabalho, dentre os diversos modelos de Aprendizado de Máquina disponíveis na literatura, serão consideradas, em particular, as Redes Neurais Artificiais (RNAs). Este modelo, inspirado nas redes neurais biológicas, é comumente usados em problemas com um grande número de entradas e com características fortemente não-lineares (PINTO, 2016), compatíveis com o contexto considerado no escopo deste trabalho, justificando a sua utilização. Em particular, as RNAs convolucionais serão consideradas, pois a literatura tem documentado um bom desempenho da mesma em problemas de aprendizado a partir de imagens.

## 1.1 Objetivos

O objetivo geral deste trabalho consiste em analisar a utilização de redes neurais convolucionais para reconhecimento automático de CAPTCHAs de texto. Para alcançar esta meta, alguns objetivos específicos precisam ser consolidados, a citar:

1. Consolidar uma base de dados representativa de CAPTCHAs de texto;
2. Realizar a fundamentação teórica acerca dos conceitos de redes neurais convolucionais;
3. Especificar parâmetros para proposição de diferentes redes neurais convolucionais;
4. Treinar e testar as redes neurais convolucionais propostas;
5. Analisar os resultados obtidos.

## 1.2 Justificativa

A utilização de CAPTCHAs para restrição de acesso a uma determinada informação ou serviço nem sempre traz somente vantagens. Quando a exposição livre desta informação pode contribuir positivamente para a sociedade, colocar uma barreira que dificulte seu acesso acaba por ser um fator limitador à livre informação. Como mencionado, a existência de um CAPTCHA na Plataforma Lattes culmina por impedir que possam ser desenvolvidas soluções com o intuito de explorar e analisar os dados dos pesquisadores e publicações encontrados nesta plataforma. Levando em conta a crescente política de transparência de acesso a dados de interesse público, o CAPTCHA vai de encontro à proposta de uma acesso rápido e fácil dos dados.

O desenvolvimento de um proposta de trabalho que promova meios para a interpretação automática desse CAPTCHA em particular, colabora para a criação e desenvolvimento de soluções em Ciência dos Dados, que venham a explorar informações de pesquisa no Brasil, como por exemplo reportando para a sociedade estatísticas e sumários, ajudando no entendimento da importância de contínuos investimentos neste setor.

Do ponto de vista do bacharel em Sistemas de Informação em formação, a presente proposta de trabalho de conclusão de curso colabora para a prática de conceitos, métodos e tecnologias de uma área de vanguarda, que é o Aprendizado de Máquina. Ademais, este trabalho alinha-se com os objetivos do Laboratório de Sistemas Inteligentes (LSI) do Núcleo de Computação (NUCOMP), motivando o desenvolvimento de uma solução inovadora que utiliza técnicas da Inteligência Artificial.

## 1.3 Metodologia

A metodologia para condução das atividades que compõem este trabalho é caracterizada pelas seguintes atividades:

1. Estudos dos conceitos a respeito de CAPTCHAs;
2. Implementação de algoritmo de raspagem para captura de CAPTCHAs do Currículo Lattes;
3. Rotulação manual dos CAPTCHAs coletados;
4. Consolidação da base de dados e descrição das características;
5. Estudo dos conceitos das redes neurais convolucionais;
6. Levantamento de técnicas e tecnologias para implementação de redes neurais convolucionais;
7. Proposição de diferentes redes neurais convolucionais para endereçamento do problema;
8. Definição de métricas de desempenho para avaliação da capacidade de generalização das redes propostas;
9. Definição de técnica de validação cruzada para análise dos resultados;
10. Realização do treinamento das redes neurais convolucionais propostas;



11. Realização dos testes e coleta das métricas de desempenho;
12. Elaboração de artefatos de visualização (gráficos, matrizes de confusão, etc.) para apresentação dos resultados das métricas;
13. Análise e comparação dos resultados obtidos.

Além destas atividades descritas, também é preciso levar em conta as atividades ligadas ao trabalho de conclusão de curso, a citar:

1. Escrita da proposta de trabalho de conclusão de curso;
2. Defesa da proposta de trabalho de conclusão de curso;
3. Escrita da monografia;
4. Defesa da monografia.

## 1.4 Cronograma

As atividades descritas na seção anterior serão executadas de acordo com o cronograma apresentado na Tabela 1.1. Os meses se referem ao ano de 2018.

## 1.5 Organização do Documento

Para apresentar os resultados deste trabalho de conclusão de curso, esta monografia está organizada como segue. O Capítulo 2 apresenta conceitos, métodos e técnicas de Aprendizagem de máquina que serão utilizadas neste trabalho. O Capítulo 3 discorre os trabalhos relacionados que servem de referência. Por fim, no Capítulo 4 encontram-se os detalhes da solução proposta para abordar o problema.

Tabela 1.1: Cronograma para execução das atividades referentes à este trabalho de conclusão de curso.

[illegible]

# Capítulo 2

## Fundamentação Teórica

Neste capítulo, serão apresentados alguns conceitos pertinentes ao entendimento da solução proposta neste trabalho. Primeiramente, os conceitos de CAPTCHA serão apresentados na Seção 2.1, bem como suas variações e limitações. Em seguida, será abordado o conceito de Aprendizado de Máquina e suas características na Seção 2.2. Os conceitos de Redes Neurais Artificiais são percorridos na Seção 2.3. Por fim, as tecnologias utilizadas no decorrer do trabalho são abordadas na Seção `refsubsec:tecnologias`.

### 2.1 CAPTCHA

CAPTCHA é um acrônimo da expressão “*Completely Automated Public Turing Test to Tell Computers and Humans Apart*” que significa a realização de um teste de Turing automatizado com o objetivo de identificar se o usuário testado é um ser humano ou uma máquina (YAN; AHMAD, 2008). Em particular, tem como finalidade evitar o acesso de maneira automática a determinados sites para fins de mineração de dados, *spam*, ataques *Deny of Service* (DoS), dentre outros (AHN; BLUM; LANGFORD, 2004).

Os testes considerados no CAPTCHA visam distinguir os humanos das máquinas, especialmente os *web bots*, por meio da solicitação da identificação de padrões não-triviais em imagens e sons, elaborados com características dificilmente capturáveis por algoritmos convencionais. Deste modo, a aprovação em um CAPTCHA aumenta as evidências de que o usuário é, de fato,

um humano (SANTOS, 2016).

No entanto, os CAPTCHAs estão sujeitos à falsos positivos e negativos. Isso se dá pois este tipo de teste pode variar em seu nível de dificuldade. Em certos tipos de CAPTCHAs baseados em imagens, por exemplo, a pouca distorção ou poluição pode facilitar a sua detecção por um algoritmo de visão computacional, culminando na aprovação indevida neste teste. Por outro lado, imagens altamente distorcidas possuem difícil interpretação para humanos, que podem vir a falhar. Em ambos os casos, o propósito desejado não é atendido. Deve-se, portanto, buscar um equilíbrio entre estas situações (SANTA-ROSA; LIBERADO, 2013).

Considerando as maneiras de testar as diferenças entre humanos e máquinas, há diversos tipos de CAPTCHAs. Os *CAPTCHAs baseados em texto*, por exemplo, consistem em imagens contendo caracteres (letras, dígitos e símbolos) que devem ser identificados pelo usuário. Porém, esses caracteres são distorcidos de diversas formas a fim de não tornar óbvio o seu entendimento. Essa distorções podem ser ondulações, diminuição da qualidade da imagem, adição de ruído, inserção de planos de fundo confusos, entre outras. A quantidade e qualidade das distorções irão influenciar diretamente a dificuldade da resolução do CAPTCHA. Este tipo de CAPTCHA é ilustrado na Figura 2.1.

Os *CAPTCHAs baseados em imagens*, por sua vez, apresentam uma ou mais imagens ao usuário e, em seguida, solicitam a identificação de um padrão ou característica presente nas mesmas. Por exemplo, no CAPTCHA ilustrado na Figura 2.2, solicita-se a identificação de todas as partes da imagem que contêm placas de trânsito.

No caso dos *CAPTCHAs baseados em áudio*, reproduz-se um som contendo caracteres falados. Para passar no teste, o usuário deve digitar corretamente quais caracteres ouviu. Este tipo de CAPTCHA é utilizado principalmente para usuários com dificuldades visuais ou como alternativa para CAPTCHAs de texto muito difíceis. Em muitos casos, vale ressaltar, o áudio apresenta ruídos de fundo e variações de voz que dificultam a sua detecção de maneira automática (MATIAS, 2011).

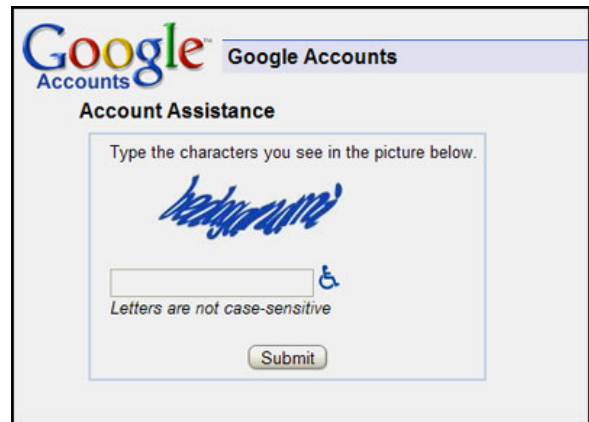
Considerando maneira não-triviais de identificar usuários humanos, os *CAPTCHAs baseados em questão* demandam a resolução de algum problema proposto. Os tipos mais comuns de

Figura 2.1: Exemplos de CAPTCHAs de texto. Fonte: (GOOGLE, 2018a)

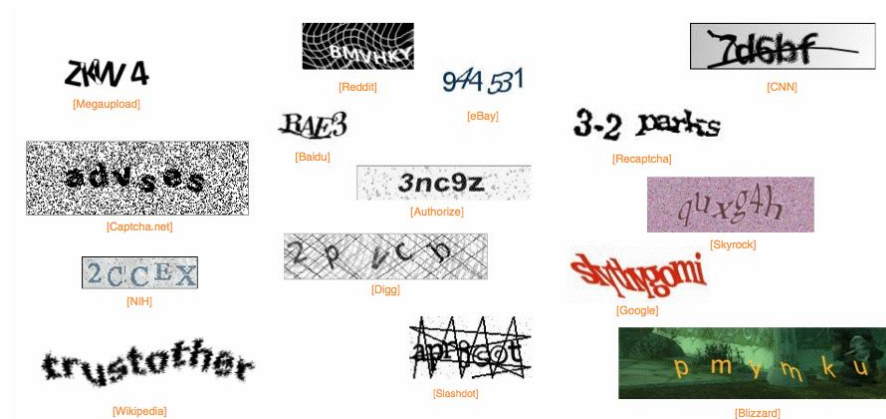
(a) CAPTCHA de texto considerado fácil.



(b) CAPTCHA de texto considerado difícil.



(c) Variações de CAPTCHAs de texto.



problemas são de natureza matemática envolvendo operações aritméticas simples, mas podem conter questões de conhecimentos gerais. Este tipo de CAPTCHA é ilustrado na Figura 2.3.

O modelo *reCAPTCHA*, desenvolvido pela empresa Google, consegue classificar o usuário em humano ou máquina a partir de padrões de navegação antes do teste ser ativado, a exemplo de tempo gasto por página, quantidade e velocidade de cliques, histórico de navegação, dentre outros. Considerando estas informações, o teste é reduzido a um único clique, conforme ilustra a Figura 2.4, que valida a classificação feita anteriormente. Esta é uma tentativa de tornar os CAPTCHAs mais agradáveis aos humanos, mas mantendo uma boa detecção de *bots* e outros algoritmos similares.

Figura 2.2: CAPTCHA baseado em imagem. Fonte: (GOOGLE, 2018a)

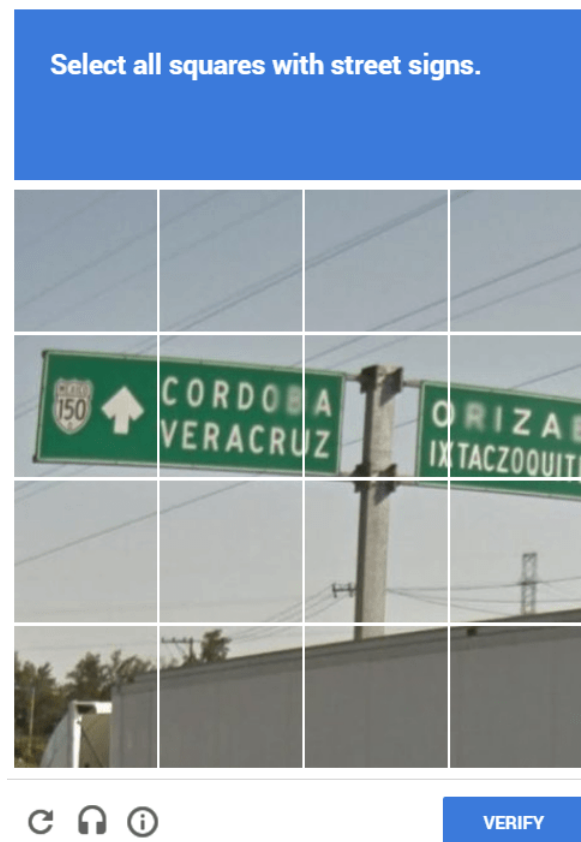


Figura 2.3: Exemplos de CAPTCHAs de questão. Fonte: (GOOGLE, 2018a)

(a) CAPTCHA de questão matemática.

A screenshot of a web form for a mathematical CAPTCHA. It includes a "User Name:" label and an input field, a "Password:" label and an input field, and a large box containing the handwritten text "What is 89 plus 1?". Below this box is a "Captcha answer:" label and an input field. At the bottom are two buttons: "Submit" and "Reset".

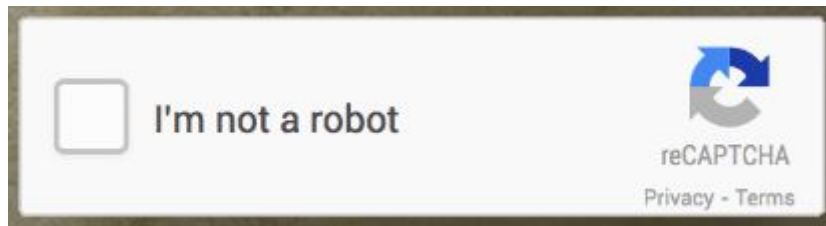
(b) CAPTCHA de questão geográfica.

A screenshot of a web form for a geographical CAPTCHA. It starts with the heading "CAPTCHA" and the text "This question is for testing whether you are a human visitor." Below this is the question "WHICH COUNTRY IS NORTH OF NEW YORK? \*". There is an empty input field for the answer. Below the input field is the text "Fill in the blank." At the bottom are two buttons: "Save" and "Preview".

## 2.2 Aprendizado de Máquina

Aprendizado de Máquina, ou *Machine Learning*, é a área da Computação responsável pelo estudo de algoritmos e sistemas que aprimoram seu conhecimento ou performance com base na experiência (FLACH, 2012). Também é possível definir como sistemas com a capacidade de

Figura 2.4: Exemplo de caixa de diálogo de validação no reCAPTCHA. Fonte: (GOOGLE, 2018b)



modificar ou adaptar suas ações para que elas constantemente melhorem sua precisão (MARSLAND, 2015). Em outras palavras, um algoritmo aprende quando modifica suas ações para se adaptar à um cenário com a finalidade de exercer uma tarefa de forma cada vez mais próxima à um estado considerado otimizado, baseando-se em situações observadas no passado.

O aprendizado pode ser sintetizado em três aspectos importantes: lembrar, adaptar e generalizar. O reconhecimento no presente de uma determinada situação já observada no passado traz o primeiro aspecto, no qual deve-se lembrar de qual atitude foi tomada para resolver aquele problema e, se essa atitude resultou em um sucesso ou um fracasso. No caso de sucesso, a atitude é repetida mas, caso contrário, é necessário tentar algo diferente, ou seja, adaptar-se. O último aspecto, a generalização, é usado para reconhecer similaridade entre situações não vistas e situações já conhecidas, permitindo a realização de atitudes bem-sucedidas nestes novos cenários (MARSLAND, 2015).

Diante destes conceitos, é notória a experiência como crucial para o aprendizado, e no caso do Aprendizado de Máquina, essa experiência encontra-se no conjunto de dados disponíveis sobre o problema, comumente chamado de *dataset*. Assim, é de suma importância que o *dataset* seja representativo do contexto do problema, isto é, que os dados que o compõem tenham sido coletados e tratados de forma a representar fielmente o cenário de futura utilização da solução desenvolvida.

Os algoritmos de Aprendizado de Máquina podem ser organizados segundo o paradigma de aprendizado, a citar:

- **Aprendizado Supervisionado.** Neste paradigma, o aprendizado é guiado pelas respostas corretas associadas à cada exemplo, os chamados atributos alvos. Assim, este

paradigma é preferível em cenários em que há uma quantidade substancial de exemplos com suas respectivas resoluções ideais;

- **Aprendizado Não Supervisionado.** Neste caso, não se objetiva prever ou descobrir uma relação do tipo entrada e saída, mas sim efetuar uma descrição dos dados a partir de suas características intrínsecas, a exemplo de similaridades;
- **Aprendizado por Reforço.** Está no limiar entre o aprendizado supervisionado e o não supervisionado. Neste paradigma, é informado quando a resposta está errada, mas não quando está correta. Assim, tenta-se uma abordagem diferente até que uma resposta aceitável seja produzida;
- **Aprendizado Evolucionário.** É baseado nas ideias de evolução biológica, na qual é feita a seleção de um ou mais configurações, que obtiveram os resultados mais próximos do ideal. Em seguida, uma adaptação é efetuada, pois continuamente são inseridos fatores de aleatoriedade e soluções com desempenho ruim são eliminadas (FLACH, 2012).

No caso do aprendizado supervisionado, as tarefas de classificação, em particular, são amplamente endereçadas. Este tipo de tarefa consiste em receber um conjunto de valores de entrada e decidir à qual classe este exemplo corresponde, dentre um conjunto finito e discreto de possibilidades (MARSLAND, 2015). Assim, após a etapa de treinamento para aprendizado de características, o algoritmo generaliza perante um exemplo ainda não visto com vistas a atribuir o rótulo da classe mais conveniente. É importante salientar que esse tipo de tarefa de aprendizado de máquina considera problemas em que cada exemplo possui um único rótulo e que este contempla uma das classes que foram apresentadas durante o treinamento do modelo.

## 2.3 Redes Neurais Artificiais

As *Rede Neurais Artificiais* (RNAs), inspiradas no cérebro humano, são sistemas paralelos distribuídos compostos por unidades de processamento simples, denominados *neurônios artificiais*,



que calculam determinadas funções matemáticas (normalmente não lineares) (BRAGA; CARVALHO; LUDERMIR, 2007). As capacidades de aprender e generalizar são fatores que tornam as RNAs atrativas para a resolução de problemas, com aplicações em diversos domínios.

Em uma RNA, os neurônios artificiais ficam dispostos em camadas interligadas por um grande número de conexões, denominadas *pesos*. O ajuste desses pesos está diretamente ligado ao aprendizado das RNAs. A quantidade de neurônios, a forma que são dispostos e conectados determinam a *topologia* da RNA, que pode ser completa ou parcialmente conectada, retroalimentada, *feedforward*, etc. (FACELI et al., 2015).

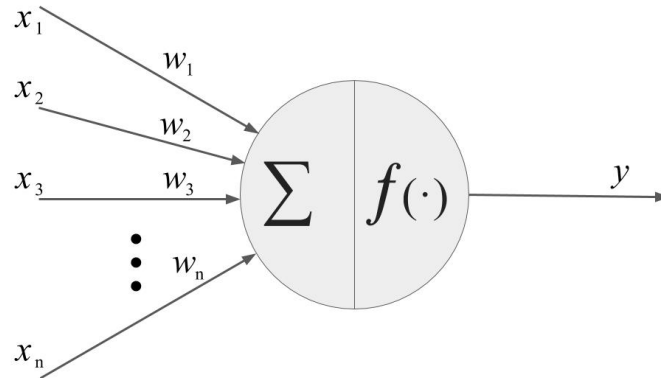
As seções a seguir detalham os conceitos relativos às RNAs utilizados, elementares para o desenvolvimento da solução proposta neste trabalho. Os conceitos relativos aos neurônios artificiais e a combinação destes em redes estruturalmente simples são apresentados na Seção 2.3.1. As RNAs Perceptron de Múltiplas Camadas, que combinam os neurônios artificiais em camadas sequenciais e são utilizadas na resolução de diversos problemas envolvendo Aprendizado de Máquina, são mostradas na Seção 2.3.2. Por fim, as RNAs convolucionais, mais voltadas para aprendizado de padrões em imagens, encontram-se apresentadas na Seção 2.3.3.

### 2.3.1 Neurônio Artificial e Redes Perceptron

Em 1943, McCulloch e Pitts desenvolveram um modelo de simplificação do neurônio baseado no que se sabia até então a respeito dessa estrutura biológica. Esta descrição matemática resultou em um modelo, denominado *neurônio MCP* e ilustrado na figura 2.5, com  $n$  terminais de entrada, que representavam os dendritos de um neurônio biológico, recebendo os valores  $\mathbf{X} = (x_1, x_2, \dots, x_n)$ , e produzindo apenas um valor de saída  $y$ , análogo ao axônio. Com o intuito de representar o comportamento das sinapses, foram atribuídos valores  $\mathbf{W} = (w_1, w_2, \dots, w_n)$  como pesos associados às entradas, podendo ser positivos ou negativos. O efeito de uma sinapse particular em um neurônio pós-sináptico é dada por  $\sum_i x_i \cdot w_i$ .

A função dos pesos em um neurônio artificial é de mensurar o quanto a entrada associada deve ser considerada para a ativação daquele neurônio. Dessa maneira, considerando as várias entradas, o neurônio processa essa informação por meio de uma soma ponderada conforme

Figura 2.5: Modelo do neurônio de McCulloch e Pitts (BRAGA; CARVALHO; LUDERMIR, 2007)



demonstrada na Equação 2.1.

$$u = \sum_{i=1}^n x_i \cdot w_i. \quad (2.1)$$

O neurônio irá disparar, emitindo a saída  $y = 1$ , quando o resultado da soma  $u$  ultrapassar o limiar de excitação  $\theta$ . Caso esse valor não seja ultrapassado, a saída então será  $y = 0$ . Esse limiar é determinado através de uma *função de ativação*.

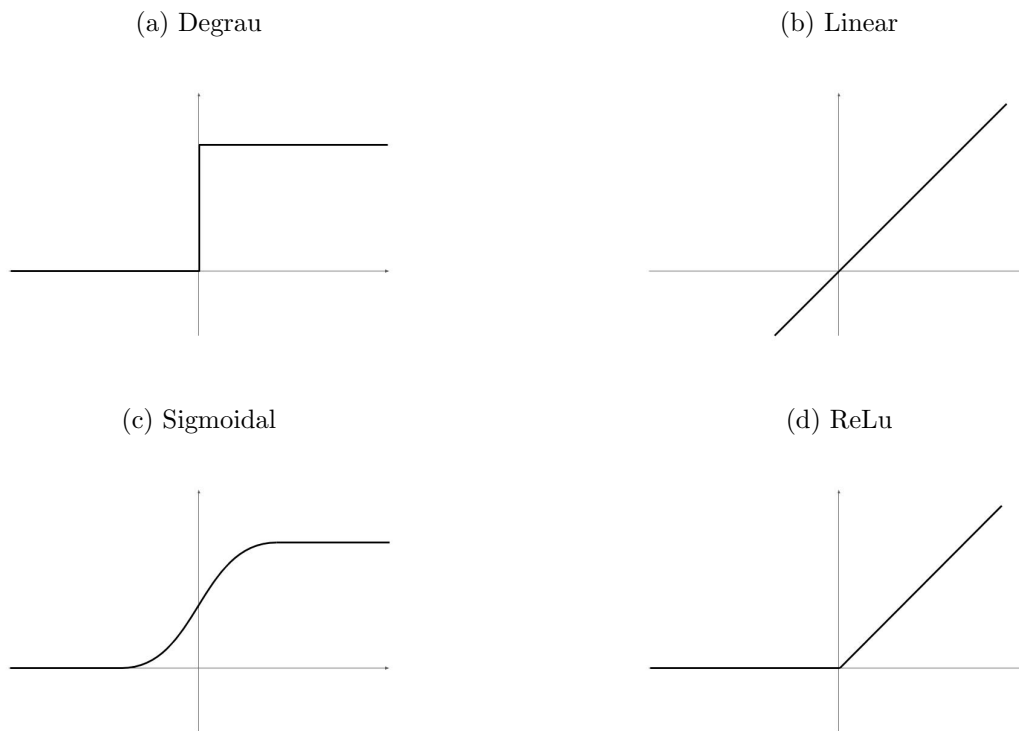
A função de ativação é responsável por determinar a saída  $y$  do neurônio a partir dos valores de entrada  $\mathbf{X}$  e seus determinados pesos  $\mathbf{W}$ . No caso do MCP, a função de ativação é do tipo degrau deslocada do limiar de ativação  $\theta$  em relação à origem, como mostrado na Equação 2.2.

$$f(u) = \begin{cases} 1, & \text{se } u \geq \theta, \\ 0, & \text{caso contrário.} \end{cases} \quad (2.2)$$

Além da função degrau deslocada, outras funções podem ser utilizadas na formulação de neurônios artificiais, desde que sejam diferenciáveis. As funções linear, sigmoidal e ReLu (retificada linear), juntamente com a função degrau, são ilustradas na Figura 2.6.

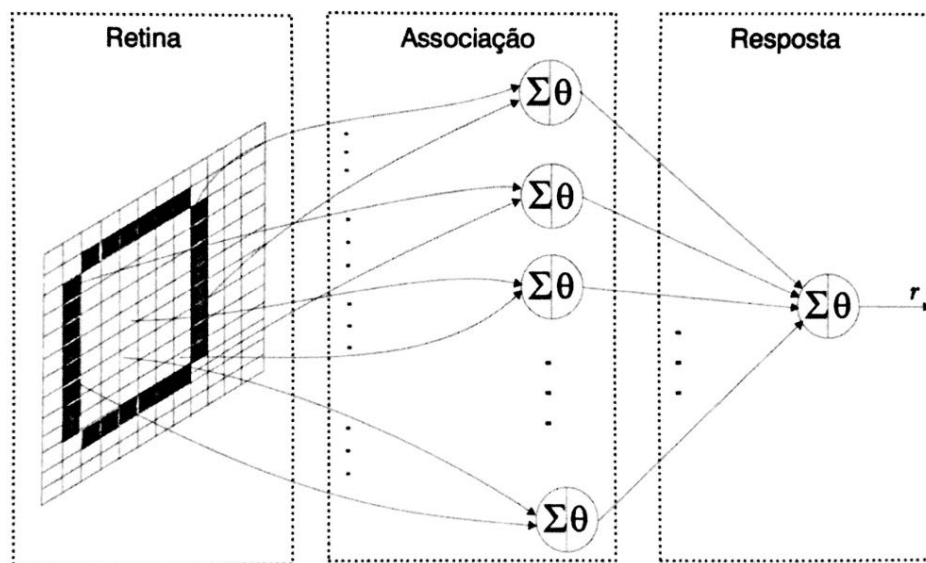
Fazendo uso do modelo de neurônios MCP, em 1958 foi publicado o modelo *perceptron*, composto de três camadas (retina, associação e resposta), dispostas conforme ilustrado na Figura 2.7. A camada retina era composta de unidades sensoras, aptas a receberem a informação do mundo externo; a camada de associação, por sua vez, era formada por neurônios MCP

Figura 2.6: Exemplos de funções que podem ser utilizadas como função de ativação.



possuindo pesos fixos definidos previamente; e a camada de resposta, por sua vez, possuindo um neurônio MCP responsável por disponibilizar o processamento produzido pela rede para o mundo externo (ROSENBLATT, 1958).

Figura 2.7: Modelo perceptron. Fonte: (BRAGA; CARVALHO; LUDERMIR, 2007)



No modelo perceptron apresentado, apenas o neurônio da camada de resposta poderia ter seus pesos ajustados mediante um processo de treinamento. Assim, o processo de treinamento deste modelo, responsável pela apresentação de exemplos visando promover o ajuste de pesos para reduzir os erros cometidos, tem como objetivo encontrar um incremento  $\Delta \mathbf{W}(t)$  a ser aplicado no vetor de pesos  $\mathbf{W}(t)$  original, em que  $t$  é o instante em que se encontra o treinamento. Assim, o aprendizado acontece mediante a atualização no vetor de pesos conforme a Equação 2.3.

$$\mathbf{W}(t + 1) = \mathbf{W}(t) + \Delta \mathbf{W}(t). \quad (2.3)$$

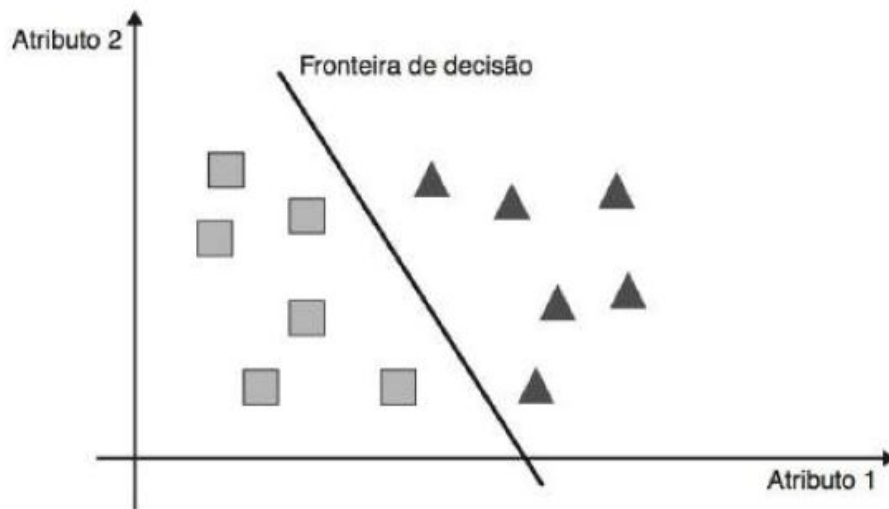
A diferença  $\Delta \mathbf{W}(t)$  é proporcional a um valor  $\eta$ , denominado *taxa de aprendizado*. Esta taxa está intimamente ligada ao tempo necessário para a convergência da rede. Caso esse valor seja muito pequeno, será necessária uma grande quantidade de ciclos para obter uma solução aceitável. Por outro lado, um valor muito grande pode provocar oscilações que dificultam a convergência. Uma possível medida para amenizar esse problema é a introdução do termo *momentum*  $\alpha$  que tem a função de quantificar o grau de importância da variação de peso do ciclo anterior em relação ao ciclo atual. Esses parâmetros são importantes no ajuste do processo de aprendizado do modelo em questão.

Apesar do modelo perceptron ser dividido em três camadas, ele é conhecido como *modelo perceptron de camada única*, justamente por apresentar propriedades adaptativas somente na camada de saída. Isso restringe sua capacidade de solução à um único tipo de problema, aqueles linearmente separáveis (FACELI et al., 2015). Neste tipo de problema, uma reta é capaz de distinguir entre objetos de duas classes distintas, conforme ilustrado na Figura 2.8.

### 2.3.2 Redes Perceptron de Múltiplas Camadas

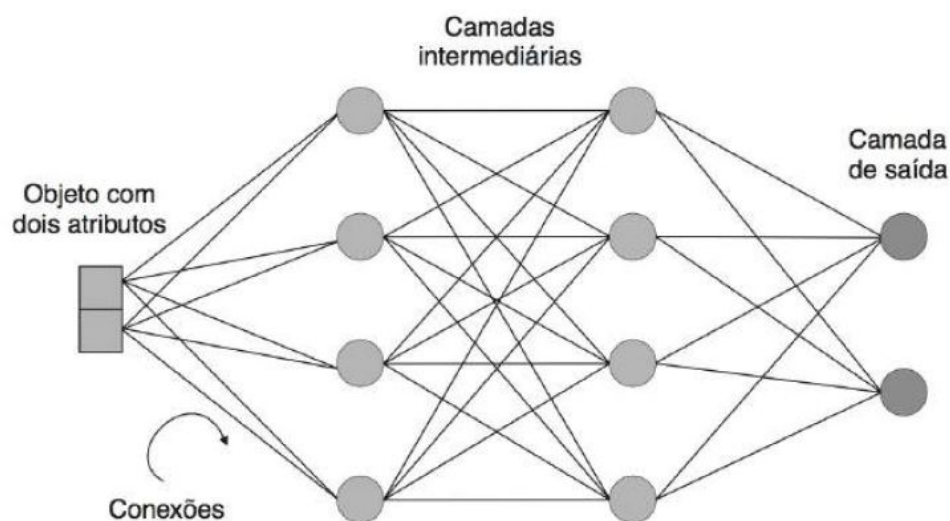
O escopo dos problemas linearmente separáveis é pequeno e, para o tratamento de problemas com um perfil mais complexo, mais passíveis de ocorrência em cenários reais, faz-se necessária a adição de mais camadas de neurônios às redes neurais artificiais (FACELI et al., 2015). Essas camadas, denominadas *camadas ocultas* ou intermediárias, localizam-se entre as camadas de

Figura 2.8: Objetos linearmente separáveis (FACELI et al., 2015)



entrada e de saída, conforme mostrado na Figura 2.9.

Figura 2.9: Exemplo de rede neural artificial com camadas ocultas. Fonte: (FACELI et al., 2015).



As camadas de uma RNA são conectadas de forma que os valores gerados na saída de um neurônio sejam aplicados como valor de entrada de um neurônio da próxima camada. Com isso, existem três padrões de conexões para uma RNA de múltiplas camadas: a *completamente conectada*, a *parcialmente conectada* e a *localmente conectada*.

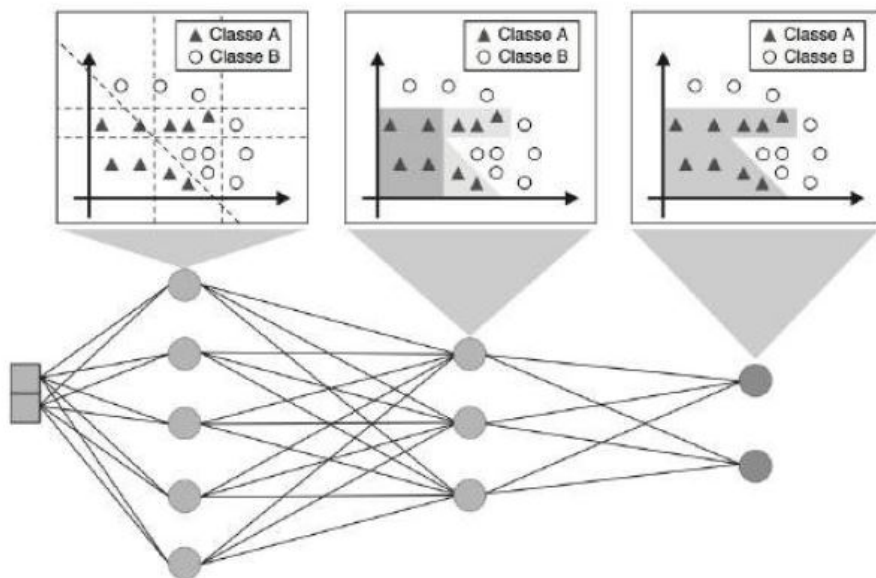
- Uma rede **completamente conectada** tem como característica o fato de que as saídas de todos neurônios de uma camada  $\ell$  estão conectados às entradas de todos os neurônios

da próxima camada  $\ell + 1$ ;

- Em uma rede **parcialmente conectada** tem-se que apenas alguns neurônios de uma camada  $\ell$  estão conectados à apenas alguns da próxima camada  $\ell + 1$ ;
- Uma rede **localmente conectada** é uma rede parcialmente conectada, porém suas conexões entre camadas são feitas em áreas bem definidas.

Da-se o nome de *Rede Neural Perceptron de Múltiplas Camadas* (MLP, do inglês *Multilayer Perceptron*) à rede neural artificial completamente conectada e composta de pelo menos uma camada oculta contendo neurônios com funções de ativação sigmoideal (BRAGA; CARVALHO; LUDERMIR, 2007). Graças à existência das camadas ocultas, este tipo de RNA é capaz de resolver problemas não-linearmente separáveis, da seguinte forma: na primeira camada, cada neurônio é responsável pelo aprendizado de uma função linearmente separável, definindo hiperplanos; os neurônios da camada posterior combinam os resultados recebidos como entrada e passam a ser capazes de reconhecer regiões convexas; a partir daí, as camadas seguintes são capazes de reconhecer regiões de formato arbitrário (FACELI et al., 2015), conforme apresentado na Figura 2.10.

Figura 2.10: Papel desempenhado pelos neurônios nas diferentes camadas de uma rede MLP. Fonte: (FACELI et al., 2015).



No processo de aprendizado supervisionado de uma rede neural é necessário efetuar o ajuste de pesos em decorrência do erro, isto é, da diferença entre a saída produzida e a saída desejada. No caso de uma RNA de camada única, este erro é obtido de maneira trivial. Com este valor do erro, pode-se então fazer ajustes no vetor de pesos  $\mathbf{W}$ , conforme descrito anteriormente na Equação 2.3. Contudo, em uma RNA MLP, esse processo só pode ser aplicado à última camada, visto que não existem saídas desejadas definidas para as camadas ocultas (BRAGA; CARVALHO; LUDERMIR, 2007).

Para solucionar o problema do ajuste de pesos nas camadas ocultas das RNAs MLPs, utiliza-se o algoritmo de retropropagação de erros, ou *back-propagation*. Esse algoritmo faz uso da técnica do gradiente descendente para estimar o erro das camadas internas por meio do efeito que estas causam no erro da camada de saída. Para tanto, é calculado o erro na camada de saída, que por sua vez é repassado para as camadas intermediárias de forma retroativa. Assim, é possível ajustar os pesos proporcionalmente aos valores das conexões entre as camadas internas (BRAGA; CARVALHO; LUDERMIR, 2007). De maneira simples, o algoritmo *back-propagation* é composto de duas fases:

1. **Fase *forward*.** Nesta fase, o fluxo da informação na RNA dá-se da primeira camada até a camada de saída. A primeira camada recebe a entrada, que por sua vez passa o resultado gerado para a camada seguinte e, assim sucessivamente, até a camada de saída, onde é produzida a resposta da rede para a informação fornecida como entrada;
2. **Fase *backwards*.** Esta fase é responsável pelo ajuste de pesos na rede mediante erro. Assim, o fluxo da informação ocorre no sentido oposto, onde na camada de saída calcula-se o erro e ajustam-se os pesos da camada de saída, depois da camada anterior à esta e assim sucessivamente, até a camada de entrada (HAYKIN, 2009).

É importante salientar que por conta da utilização do gradiente descendente no algoritmo *back-propagation*, restringem-se as funções de ativação às funções contínuas e diferenciáveis (BRAGA; CARVALHO; LUDERMIR, 2007).

Considerando a capacidade de endereçar problemas não-linearmente separáveis, as RNAs MLP são amplamente utilizadas em diversos domínios. Previsão do valor de ações no mercado

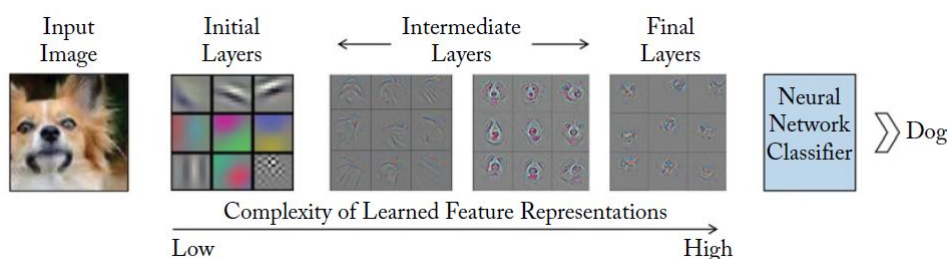
financeiro, reconhecimento de fonemas em sinais de voz e identificação de assinaturas manuscritas são apenas alguns exemplos dentre um vasto corpo de soluções utilizando este modelo de Aprendizado de Máquina (BRAGA; CARVALHO; LUDERMIR, 2007).

### 2.3.3 Redes Neurais Convolucionais

Apesar de redes MLPs possuírem a capacidade de abordar grande variedade de problemas, a busca por padrões em imagens é uma situação que tem como característica a necessidade de relacionar a vizinhança de *pixels* ou mesmo as noções dos componentes de cores, relevantes para o entendimento da informação ali contida. As *Redes Neurais Convolucionais* (CNNs, do inglês *Convolutional Neural Networks* ou ainda *ConvNets*) são modelos de RNAs inspirados na organização do córtex visual de animais, muito populares para abordar esses problemas relacionados a visão computacional (VASCONCELOS; CLUA, 2017).

A diferença entre uma CNN e uma RNA convencional é que cada unidade em uma determinada camada é um filtro de duas ou mais dimensões. Esses filtros incorporam um contexto espacial, mas menor, da mídia de entrada e usam compartilhamento de parâmetros para reduzir significativamente o número de possíveis variáveis a serem processadas (KHAN et al., 2018). Em outras palavras, cada neurônio responde à pequenos campos receptivos da imagem relacionados a sua posição espacial e com sobreposição dos campos receptivos de seus vizinhos. A partir desses filtros, diferentes mapas são produzidos, variando de acordo com o posição e níveis de abstração que acontecem com o aprofundamento dos níveis de neurônios, conforme ilustrado na Figura 2.11.

Figura 2.11: Processo em que uma CNN inicia aprendendo características simples e a medida que aprofunda suas camadas, torna o padrão mais completo, por fim faz a classificação (KHAN et al., 2018).



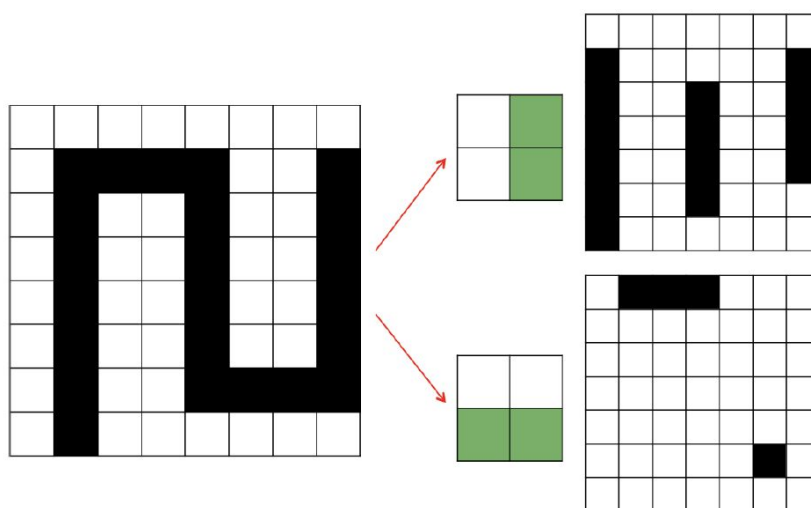


Uma CNN é basicamente composta por tipos específicos de camadas, são elas: camadas convolucionais, camadas de *pooling*, camadas de ativação, camadas completamente conectadas e, por fim, uma camada de saída.

As camadas convolucionais são de maior importância em uma CNN. Nelas são definidos os filtros, também chamados de *kernels*, que fazem o processo de varrer a imagem para gerar como saída um *mapa de características*, processo este denominado de *convolução*. O filtro é um conjunto de valores organizados sob a forma de um volume, isto é, organizados de maneira tridimensional com altura, largura e profundidade, dependendo da aplicação.

Na Figura 2.12 a realização da operação de convolução é ilustrada em detalhes. A imagem à esquerda é fornecida como entrada, a qual será sujeita à convolução por dois filtros distintos. Os filtros possuem dimensão  $2 \times 2$  e contêm apenas valores binários, em que os pixels na cor verde correspondem ao valor 1. O resultado da convolução da entrada com cada um dos filtros é mostrada mais à direita, onde percebe-se o mapa de características que foi produzido como resposta. É importante salientar que os mapas produzidos possuem dimensões menores que da imagem original (BUDUMA, 2017).

Figura 2.12: Exemplo de convolução aplicada a uma entrada. Fonte: (BUDUMA, 2017).



Ter um número grande de filtros pequenos é uma boa forma de conseguir uma representação fiel das características da imagem com uma quantidade pequena de parâmetros. Também é sugerido que os filtros se movam apenas um pixel por vez (parâmetro *stride* igual a 1) sem usar

nenhum tipo de preenchimento (parâmetro *padding* igual a 0), para que o mapa de saída mantenha a proporção da imagem original, mas com um número de parâmetros significativamente menor (BUDUMA, 2017).

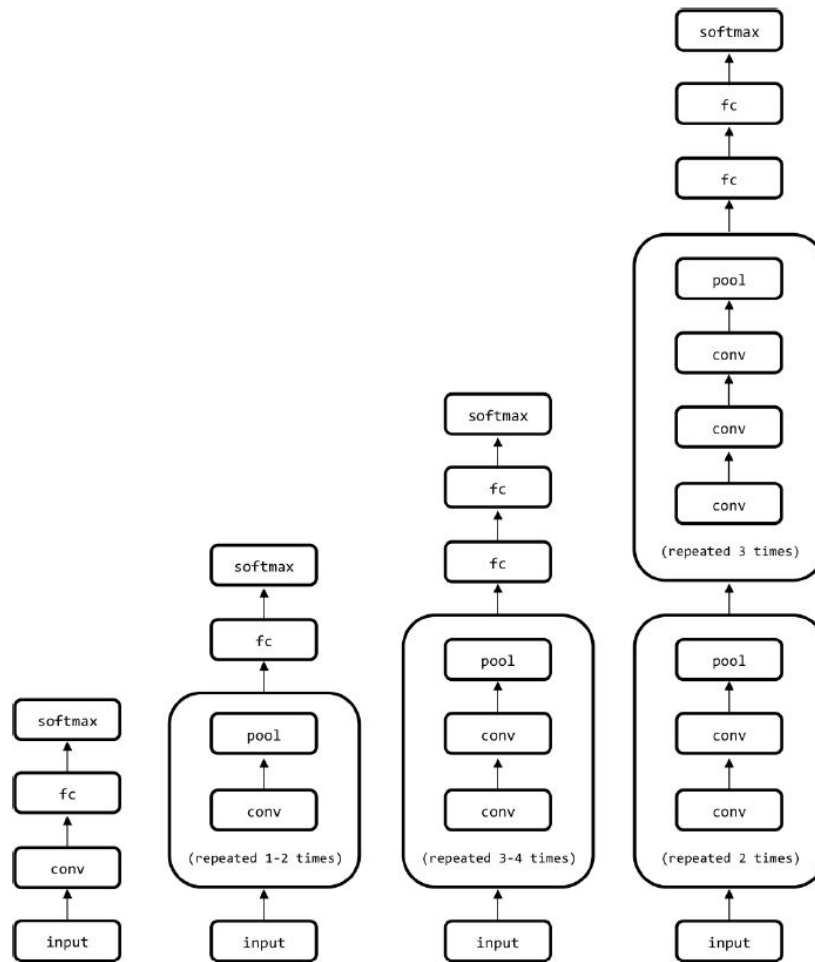
A *Camada de Pooling* recebe como entrada os mapas de características tipicamente gerados pelas camadas de convolução que a antecedem. Nela, são aplicadas funções com o objetivo de reduzir drasticamente as dimensões dos mapas de características recebidos como entrada, a fim de torná-los compactos e invariantes perante pequenas mudanças da imagem de entrada. Essas funções variam de acordo com a aplicação, mas a função *Max Pooling*, por exemplo, gera como saída um mapa com os valores mais altos encontrados em cada iteração da varredura. Já a *Average Pooling*, por sua vez, faz o mesmo que a *Max Pooling*, mas com a média dos valores (KHAN et al., 2018). A *Camada de Ativação* recebe um valor de entrada e transforma em uma saída com uma extensão pré definida, a exemplo do intervalo  $[0, 1]$ . O papel dessa camada é atuar como um mecanismo que decide quando e o quanto um neurônio deve ser disparado, dada determinada entrada. Assim como nas MLPs, as funções dessa camada devem ser contínuas e diferenciáveis, como a sigmoideal ou a ReLu, mostradas anteriormente na Figura 2.6.

A *Camada Completamente Conectada* contempla um conjunto de neurônios totalmente interconectados e corresponde em termos práticos à uma camada convolucional com filtros de dimensão  $1 \times 1$ , sendo geralmente encontrada ao final de uma CNN. Esta camada tem a capacidade de separar as diferentes variações de classificação que serão geradas na saída, sintetizando os resultados dos múltiplos mapas de características produzidos. (GULLI; PAL, 2017)

A *Camada de Saída*, por fim, é responsável por receber o resultado de todo o processamento feito pelas camadas anteriores e ativar os neurônios de acordo com a possível resposta encontrada, produzindo uma saída inteligível em relação ao problema considerado. Se o problema for de classificação, por exemplo, esta camada produz um vetor de probabilidades em relação às classes do problema que a entrada está associada.

Tendo em vista as camadas apresentadas, pode-se organizá-las de diferentes maneiras com vistas a compor a arquitetura da CNN desejada, desde mais simples até mais complexa. Algumas arquiteturas de CNNs são ilustradas na Figura 2.13.

Figura 2.13: Exemplos de arquiteturas de CNN variando em sua complexidade. O último exemplo consiste em uma arquitetura VGG. Fonte: (BUDUMA, 2017).

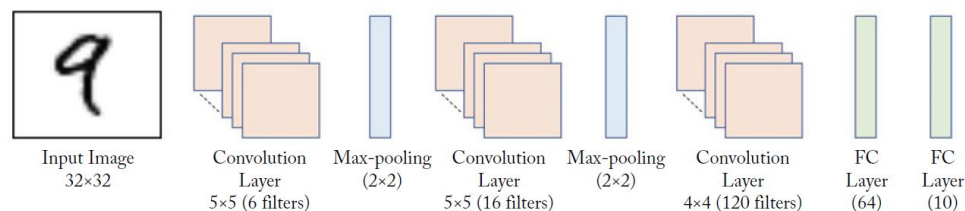


É importante salientar que, quanto mais camadas são introduzidas (redes mais profundas), o número de camadas de *pooling* torna-se reduzido em relação às camadas convolucionais. Isso ocorre por que as operações de *pooling* causam perda de informações e adicionar mais camadas convolucionais antes permite alcançar representações mais precisas (BUDUMA, 2017).

A arquitetura *LeNet* é uma das primeiras e mais básicas arquiteturas de CNNs aplicada na identificação de imagens contendo dígitos escritos à mão. Uma versão dessa arquitetura que apresenta bons resultados neste problema é a *LeNet-5* que é composta por dois blocos, cada um contendo uma camada convolucional seguida de uma camada de *Max Pooling*. Ao final destes dois blocos, segue-se uma camada convolucional e duas camadas completamente conectadas que agem como o classificador das características. Na Figura 2.14, é mostrada uma CNN *LeNet-5*

sendo usada para reconhecimento de um dígito manuscrito.

Figura 2.14: Arquitetura do modelo LeNet-5 sendo usada em um dígito do dataset MNIST (KHAN et al., 2018).



O processo de treinamento de uma CNN começa com a atribuição de pesos aleatórios aos neurônios que a compõem. Uma restrição importante nesta atribuição inicial é que os pesos não sejam iniciados todos iguais à zero, pois isto dificultará o ajuste de parâmetros necessário ao aprendizado. No mais, a apresentação de exemplos, estimativa de erro e ajuste de parâmetros é feito de maneira análoga às MLPs, por meio do algoritmo *back-propagation*, apresentado anteriormente. Diversas técnicas de otimização de busca no gradiente descendente podem ser usadas neste algoritmo com vistas a acelerar o treinamento, como: diferenciação numérica, diferenciação analítica, entre outras.

Devido ao grande número de parâmetros que uma CNN possui, é comum que ela tenha uma tendência ao *overfitting*, processo em que o treinamento de uma rede neural provoca um superajustamento aos exemplos do conjunto de treinamento, podendo aumentar a taxa de erro em situações de generalização (FACELI et al., 2015). Para contornar essa tendência, técnicas de regularização costumam ser aplicadas, nas quais é forçada uma generalização para o conjunto de dados empregado. O aumento artificial da base de dados por um processo de *data augmentation*, por exemplo, pela rotação, inversão e cortes das imagens na base de dados original, é uma forma de efetuar esta regularização. (KHAN et al., 2018)

Uma aplicação bastante consolidada e que retrata a importância prática das CNNs diz respeito ao problema do reconhecimento de dígitos manuscritos. Neste problema, são dadas como entrada imagens monocromáticas de dimensão  $28 \times 28$  *pixels*, contendo representações de dígitos escritos à mão. O objetivo do problema é classificar as representações encontradas na imagem, identificando o número ou letra correspondente ao que está apresentado. A base

de dados mais conhecida para esta tarefa é o MNIST, contendo 60.000 exemplos de imagens para treinamento e 10.000 para os testes. O estado da arte desse problema, segundo Benenson (BENENSON, 2018), é alcançado por CNNs, com taxa de erro de 0,21%. Este resultado, juntamente com diversos outros disponíveis na literatura, reforçam a importância deste modelo de Aprendizado de Máquina e o seu potencial de uso em aplicações práticas.

## 2.4 Tecnologias Utilizadas

As tecnologias utilizadas para o desenvolvimento deste trabalho compreenderam a linguagem de programação Python e algumas bibliotecas disponíveis nesta linguagem. Python é uma linguagem de propósito geral, multi-paradigma, com sintaxe concisa e de fácil entendimento, que vem ganhando progressivamente espaço em diversos âmbitos do desenvolvimento de software comercial e científico (GRIES; CAMPBELL; MONTOJO, 2013). Em particular, o desenvolvimento foi efetuado com a utilização do *Jupyter Notebooks*<sup>1</sup>, uma aplicação cliente-servidor que combina código executável com elementos textuais complexos (figuras, parágrafos, gráficos, equações, etc.), propiciando fácil aprendizagem e compreensão.

Os *frameworks* *sci-kit learn*<sup>2</sup> e *keras*<sup>3</sup>, da linguagem Python, foram elencados como mais adequados dentre os disponíveis para a implementação das tarefas de Aprendizado de Máquina, pois possuem uma grande quantidade de modelos, permitem uma vasta configuração de parâmetros e hiperparâmetros, possibilitam uma fácil obtenção de métricas de desempenho e abstraem detalhes de implementação e configuração.

Além do que foi mencionado, a API *OpenCV*<sup>4</sup>, multiplataforma e disponível na linguagem Python, também será considerada para as tarefas de pré-processamento dos CAPTCHAs. Esta biblioteca possui um vasto conjunto de funções que implementam tarefas de Visão Computacional, as quais consideram aspectos de eficiência e são geralmente voltadas para aplicações de tempo-real.

---

<sup>1</sup><<http://jupyter.org/>>

<sup>2</sup><<http://scikit-learn.org>>

<sup>3</sup><<http://keras.io>>

<sup>4</sup><<https://opencv.org/>>

# Capítulo 3

## Solução Proposta

Para apresentar a solução proposta para o problema do reconhecimento automático de CAPTCHAs considerando o reconhecimento de padrões por modelos de Aprendizado de Máquina, faz-se necessário obedecer os passos canônicos das tarefas neste domínio (MARSLAND, 2015, vide Seção 1.5). Para tanto, primeiramente foi necessário consolidar uma base de dados, processo este já concluído e detalhado na Seção 3.1, que também contempla a limpeza e preparação dessa base. Em seguida, uma visão geral desta base e das classes disponíveis na mesma é apresentado na Seção 3.2. A tarefa de previsão e as métricas de desempenho são mostradas na Seção 3.3. Por último, a proposição de modelos e escolha de parâmetros é apresentada na Seção 3.4.

### 3.1 Consolidação da Base de Dados

Para fornecer experiência a respeito do problema considerado, se fez necessário obter uma quantidade razoável de amostras do tipo de CAPTCHA que estava sendo endereçado, isto é, do CAPTCHA de texto utilizado pela Plataforma Lattes.

Primeiramente, ao acessar a URL <<http://buscatextual.cnpq.br/buscatextual/visualizacv.do>> foi possível identificar um *web service* provedor das imagens de CAPTCHA. Por meio da linguagem de programação Python, foi construído um script para consumo deste *web service* de maneira automática, o qual era responsável por fazer requisições sequenciais de conexão com o endereço provedor do serviço para realizar o download das imagens. O código-fonte deste script

encontra-se na Figura 3.1.

Figura 3.1: Script para download dos CAPTCHAs da Plataforma Lattes.

```
import os
import datetime
import time
import urllib.request

img_file = 'captchas' # path of captchas file
url = "http://buscatextual.cnpq.br/buscatextual/" +
      "servlet/captcha?metodo=getImagemCaptcha" # url that will be crawled
tries = 5 # number of page reloads
sleep_time = 1.5 # seconds sleeping before a new try

def saveImg(img, name): # image save function
    try:
        urllib.request.urlretrieve(url, os.path.join(img_file, name) + '.jpg')
        saveLog('image saved: ' + name)
    except:
        saveLog('cant save image: ' + name)

def saveLog(msg): # log register function
    with open('log.txt', "a") as fh:
        fh.write(str(datetime.datetime.now()) + ' - ' + msg + '\n')
    fh.close()

keep_working = True
count = 0

while(keep_working):
    count += 1
    saveImg(url, 'img' + str(count))
    if (count < tries):
        time.sleep(sleep_time)
    else:
        keep_working = False
```

Para possibilitar esta coleta, um cuidado particular se fez necessário na implementação do script: aumentar o espaçamento temporal entre duas requisições sequenciais, o que culminaria em evitar uma grande quantidade de acessos rápidos, com vistas a não prejudicar o funcionamento do serviço e nem tampouco causar o bloqueio das requisições, que poderiam ser confundidas com um ataque do tipo *Deny of Service*. Ao ser executado, o script produzido também gerava um arquivo de *log*, registrando o histórico de solicitações e seus resultados,

possibilitando análises detalhadas de tempo e recuperação em caso de eventuais problemas.

Como resultado da execução deste script durante cerca de 10 horas, foram coletados 6.000 CAPTCHAs de texto gerados pela Plataforma Lattes, tais como os ilustrados na Figura 3.2.

Figura 3.2: Exemplos de CAPTCHAs fornecidos pela Plataforma Lattes.



A partir de uma inspeção visual nas imagens coletadas, foi possível perceber uma característica em comum: toda a informação textual encontrava-se majoritariamente na cor branca, enquanto o fundo da imagem encontrava-se com padrões de cores variados. Isso ensejou a realização da subtração de fundo, tarefa de processamento digital de imagens com vista a eliminar informações irrelevantes. O processo de subtração de fundo, exemplificado na Figura 3.3, foi aplicado em todas as imagens previamente coletadas.

Figura 3.3: CAPTCHA antes e depois da subtração de fundo.



Após a etapa de subtração de fundo, foi verificado que alguns caracteres possuíam partes com linhas muito estreitas, que poderiam dificultar a identificação dos caracteres correspondentes. Para amenizar esse problema, foi utilizado um algoritmo de dilatação por iteração, disponível na biblioteca *OpenCV*. Esse algoritmo torna os elementos visuais de uma imagem mais espessos. De maneira análoga, todos os exemplos da base de dados foram submetidos à esse processamento, exemplificado na Figura 3.4.

Também a partir da inspeção visual foi verificado que todos os CAPTCHAs consistiam na identificação de um número fixo de quatro caracteres. A partir disto foi consolidado o próximo

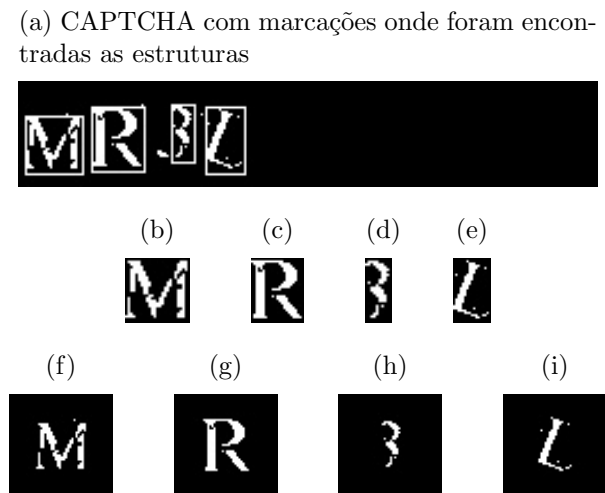


Figura 3.4: CAPTCHA antes e depois da dilatação.



passo, que tratou da separação dos caracteres individuais que compunham o CAPTCHA textual. Esse processo, exemplificado na Figura 3.5, também foi realizado com o auxílio da biblioteca *OpenCV*, por meio de um algoritmo de segmentação de estruturas. Por fim, foi acrescentado às imagens dos caracteres, bordas para torna-las com dimensões fixas.

Figura 3.5: Processo de segmentação do CAPTCHA.



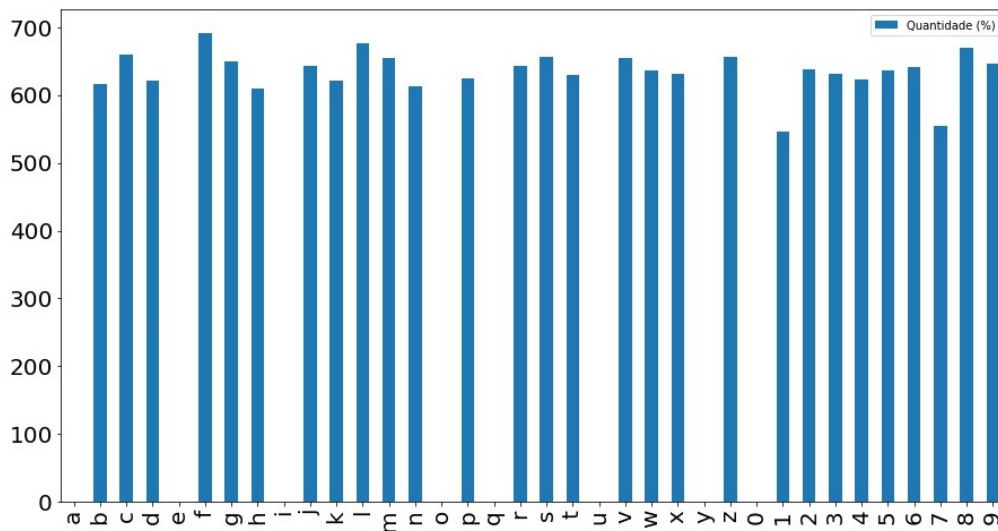
Durante a etapa de pré-processamento, houve a escolha experimental de alguns parâmetros para ajustar o processo de forma a aumentar sua eficácia. Por exemplo, na fase de segmentação, foram escolhidos valores com o intuito de eliminar estruturas muito pequenas, geralmente encontradas devido à algum ruído que não foi totalmente removido na fase de subtração do fundo.

## 3.2 Visão Geral da Base de Dados

Após a etapa anterior, foi obtida uma base de dados contendo 24 mil exemplos, dos quais 17.781 foram categorizados manualmente em pastas rotuladas de *A* a *Z* e de 0 a 9. Dentre o total de

exemplos rotulados, observou-se 70,68% correspondiam a caracteres alfabéticos e que o restante correspondia a caracteres numéricos, distribuídos conforme o histograma da Figura 3.6.

Figura 3.6: Histograma da quantidade de dígitos classificados.



A partir desta análise da base de dados, foi possível perceber que determinados caracteres, tais como o 0 e as vogais, não ocorreram no total de amostras coletadas. Dentre os demais caracteres, observou-se uma ocorrência heterogênea, de modo que a base de dados pode ser vista como não balanceada.

### 3.3 Tarefa de Classificação e Métricas de Desempenho

A tarefa de previsão considerada no escopo deste trabalho será do tipo classificação multi-rótulo em que, dada uma imagem oriunda do CAPTCHA de texto da Plataforma Lattes pré-processada conforme descrito, tem-se por objetivo identificar a qual caractere esta corresponde, dentre um conjunto finito de possibilidades (letras ou dígitos).

Os 24 mil exemplos da base de dados serão particionados de maneira aleatória em dois conjuntos: conjunto de treinamento, com 70% das amostras; e conjunto de teste, com as demais amostras. O conjunto de treinamento, contendo as imagens e seus respectivos rótulos, serão utilizadas para ajuste de parâmetros e aprendizado de padrões pelos modelos, enquanto a outra fração dos dados será utilizada para aferir o poder de generalização. Estes procedimentos

obedecem à técnica de *holdout* para validação cruzada (BRINK; RICHARDS; FETHEROLF, 2017).

Para aferir os modelos será utilizada a métrica de desempenho micro  $F$ -Score, que contempla a média harmônica entre precisão e revocação num domínio multi-rótulos em que a frequência das classes é desigual. Esta métrica pondera a capacidade do modelo em rotular corretamente os exemplos das diferentes classes, mas penalizando os erros proporcionalmente à quantidade de amostras na respectiva classe associada (KUBAT, 2015).

### 3.4 Modelos Propostos

Os modelos de Aprendizado de Máquina a serem considerados neste trabalho são as redes neurais convolucionais, onde serão contempladas diferentes arquiteturas e parâmetros de treinamento, para fins de comparação dos resultados obtidos e identificação de um modelo mais apto para a tarefa.

Dentre as redes neurais convolucionais a serem consideradas para este cenário, enfatiza-se a importância de treinar e testar o modelo LeNet para esta tarefa. Desenvolvida por Yann le Cun e colaboradores (LECUN et al., 1998), esta arquitetura foi proposta inicialmente para o reconhecimento de dígitos manuscritos da base de dados MNIST (CUN et al., 1990; KHAN et al., 2018), a qual possui muitas similaridades com a base de dados considerada neste trabalho.

## Capítulo 4

# Trabalhos Relacionados

Ao consultar a literatura, foram identificados alguns trabalhos que também consideram o reconhecimento automático de CAPTCHAs textuais similares ao abordado neste trabalho.

A monografia de Arins Pinto considerou o reconhecimento de CAPTCHAs disponibilizados para controle de acesso a um serviço online do Estado de Santa Catarina (PINTO, 2016). Este CAPTCHA é de natureza textual sendo composto por cinco caracteres, incluindo letras e dígitos. De maneira análoga à solução proposta neste trabalho, o autor em questão utilizou Redes Neurais Convolucionais como modelo de Aprendizado de Máquina para reconhecimento do CAPTCHA. Porém, nenhum pré-processamento foi efetuado e as redes deveriam prever todos os caracteres de uma única vez.

Para resolver a tarefa em questão, o autor propôs uma CNN com quatro camadas convolucionais, tendo uma camada de ativação (ReLU) e uma camada de *pooling* entre cada camada convolucional. Posteriormente, existe uma camada de *dropout* e duas camadas completamente conectadas. A última camada produz 5 saídas, em que cada uma é um vetor de 36 posições correspondendo aos caracteres possíveis no CAPTCHA ( $[0, 9]$  e  $[a, z]$ ). Ao testar este modelo utilizando a acurácia como métrica de desempenho, obteve-se um resultado igual a 92,87%, um resultado satisfatório para o cenário considerado. Com este resultado, o autor argumentou acerca da eficácia de controles de acesso que utilizam CAPTCHAs de texto e ainda complementou a discussão em termos da limitação da transparência no acesso à informações públicas (PINTO, 2016).

---

O trabalho de Santos (SANTOS, 2016), por sua vez, abordou pontualmente o reconhecimento de caracteres do CAPTCHA de texto da Plataforma Lattes, mesmo problema considerado no escopo deste trabalho. Para o reconhecimento automático o autor comparou a utilização de redes neurais artificiais rápidas (FANNs, do inglês *Fast Artificial Neural Networks*) e duas abordagens de reconhecimento óptico (OCR). De maneira análoga ao que está sendo considerado neste trabalho, o autor também propôs uma estratégia de pré-processamento dos CAPTCHAs. Os resultados obtidos por Santos evidenciaram as FANNs como melhor modelo para esta tarefa, com uma acurácia de 87,5%, enquanto os demais modelos tiveram um desempenho baixo, de 28,7% e 0,8%.

Embora considere a princípio o mesmo CAPTCHA do trabalho de Santos (SANTOS, 2016), esta proposta de conclusão de curso se propõe a, de maneira análoga ao trabalho de Arins Pinto (PINTO, 2016), considerar o modelo de redes neurais convolucionais. Este modelo de Aprendizado de Máquina tem consolidado o estado da arte da maioria das competições envolvendo classificação de imagens (RUSSAKOVSKY et al., 2015). Assim, almeja-se investigar se haverá desempenho superior ao que está posto na literatura para o reconhecimento automático do CAPTCHA da Plataforma Lattes. Além do modelo a ser utilizado, também serão considerados passos de pré-processamento, que venham a diminuir a quantidade de informações irrelevantes antes das etapas de treinamento e teste.

# Capítulo 5

## Resultados e Discussão

Esta capítulo compreende a apresentação e discussão dos resultados obtidos decorrentes do treinamento e teste das CNNs propostas para endereçar a tarefa de aprendizado de máquina considerada. Os cenários descritos tiveram sua execução conduzida em um computador do tipo *desktop* com processador Intel i5 7500, 8 GB de memória RAM e placa de vídeo nVidia GTX 1050 Ti com 4 GB de memória. A GPU disponível na placa de vídeo foi utilizada para promover um ganho de desempenho e acelerar a realização do cálculo dos parâmetros livres das CNNs.

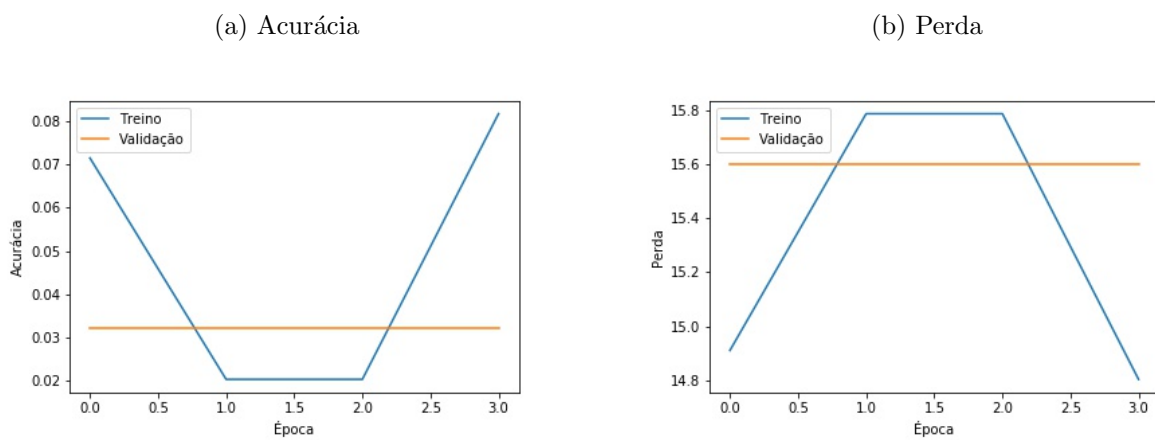
### 5.1 Cenário 1: LeNet e Parâmetros Típicos

Neste primeiro cenário, foi considerada a arquitetura LeNet e os valores de parâmetros típicos utilizados nesta rede para endereçar o problema de classificação de dígitos manuscritos MNIST (CUN et al., 1990; GULLI; PAL, 2017). Esta configuração foi adotada em virtude da semelhança entre as bases de dados de ambos os problemas e pelo bom desempenho desta configuração no problema MNIST, visando transpor a boa performance neste cenário análogo. Assim, nessa abordagem foram utilizados 20 *kernels* de dimensão  $5 \times 5$  pixels no primeiro bloco e 50 *kernels* com a mesma dimensão no segundo bloco. As camadas de *Maxpooling* utilizaram stride  $2 \times 2$ . Em todas as camadas de ativação internas foi utilizada a função de ativação ReLu. Considerando o problema de classificação multiclasse, para função de ativação da camada de saída foi utilizada a função *Softmax*. A camada densa do último bloco possui 500 neurônios.

Como *optimizer* para o algoritmo de *backpropagation* utilizou-se o *solver* Adam, fornecido pela biblioteca Keras. A taxa de aprendizado considerou valor *default* igual a 0,001 e o número de épocas adotado foi de 20 com tamanho de *batch* de 128.

A etapa de treinamento da rede foi concluída em 26s, sendo interrompida pela função de *early stop* com vistas a prevenir *overfitting*. Durante o treinamento foram coletadas métricas de desempenho, cujos gráficos encontram-se ilustrados na Figura 5.1.

Figura 5.1: Métricas do cenário 1.



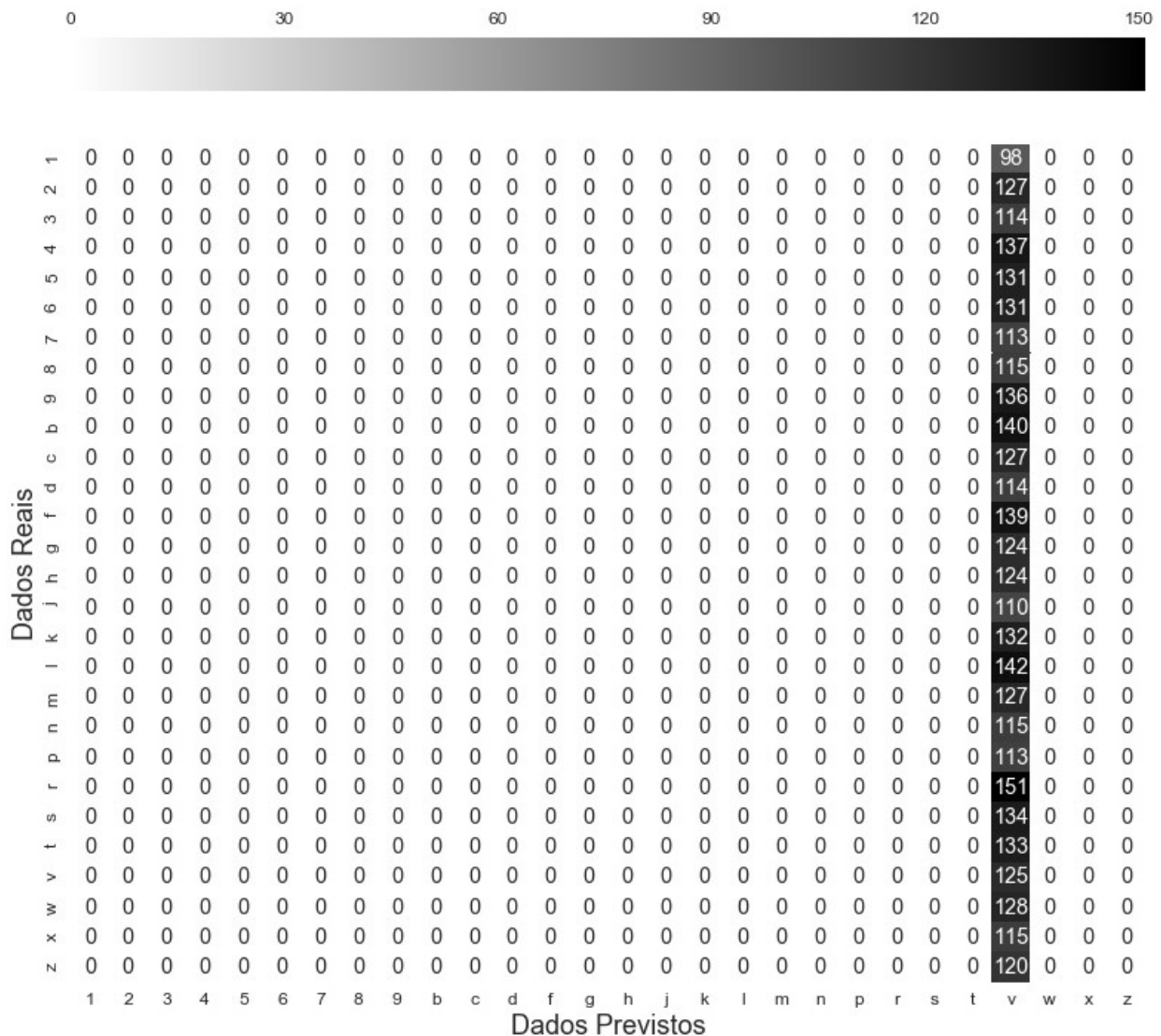
Após o treinamento, foi então realizada a fase de testes com os dados reservados para este fim. A métrica de desempenho obtida foi o  $F$ -Score com valor de 0.0356, acurácia igual a 0.0322 e matriz de confusão ilustrada na Figura 5.2.

Uma análise mais detalhada dos resultados desse cenário revela que o treinamento da rede não foi efetivo para promover o aprendizado e que esta minimizou a métrica de erro com uma estratégia contraproducente: previu todos os exemplos do conjunto de teste como sendo de uma única classe. Pode-se então concluir que uma simples transposição de uma CNN e de seus parâmetros a um problema análogo não foi uma estratégia efetiva para endereçar o novo cenário considerado. Em decorrência, novos cenários foram considerados visando contornar os problemas encontrados.

## 5.2 Cenário 2: LeNet, Busca em *Grid* de Parâmetros e Regularização de Pesos

Uma das justificativas para o mau desempenho no cenário anterior pode ser decorrente da má escolha de parâmetros para a rede em questão. Assim, o Cenário 2 contemplou a utilização de uma busca em *grid* com vistas a ajustar os parâmetros que otimizam o modelo, pois esta busca atua permutando uma variedade finita e pré-definida de valores de parâmetros (BUDUMA, 2017). Além disso, considerou-se também a regularização de pesos, a qual impõe restrições

Figura 5.2: Matriz de confusão do Cenário 1.





aos valores que estes podem assumir com vistas a mitigar *overfitting* e diminuir a entropia do modelo resultante (CHOLLET, 2017).

Neste Cenário 2 manteve-se a utilização da arquitetura LeNet, considerou-se a regularização dos tipos L1 e L2, as quais se diferenciam no tocante ao cálculo dos coeficientes dos pesos, e variou-se valores de parâmetros escolhidos, a citar:

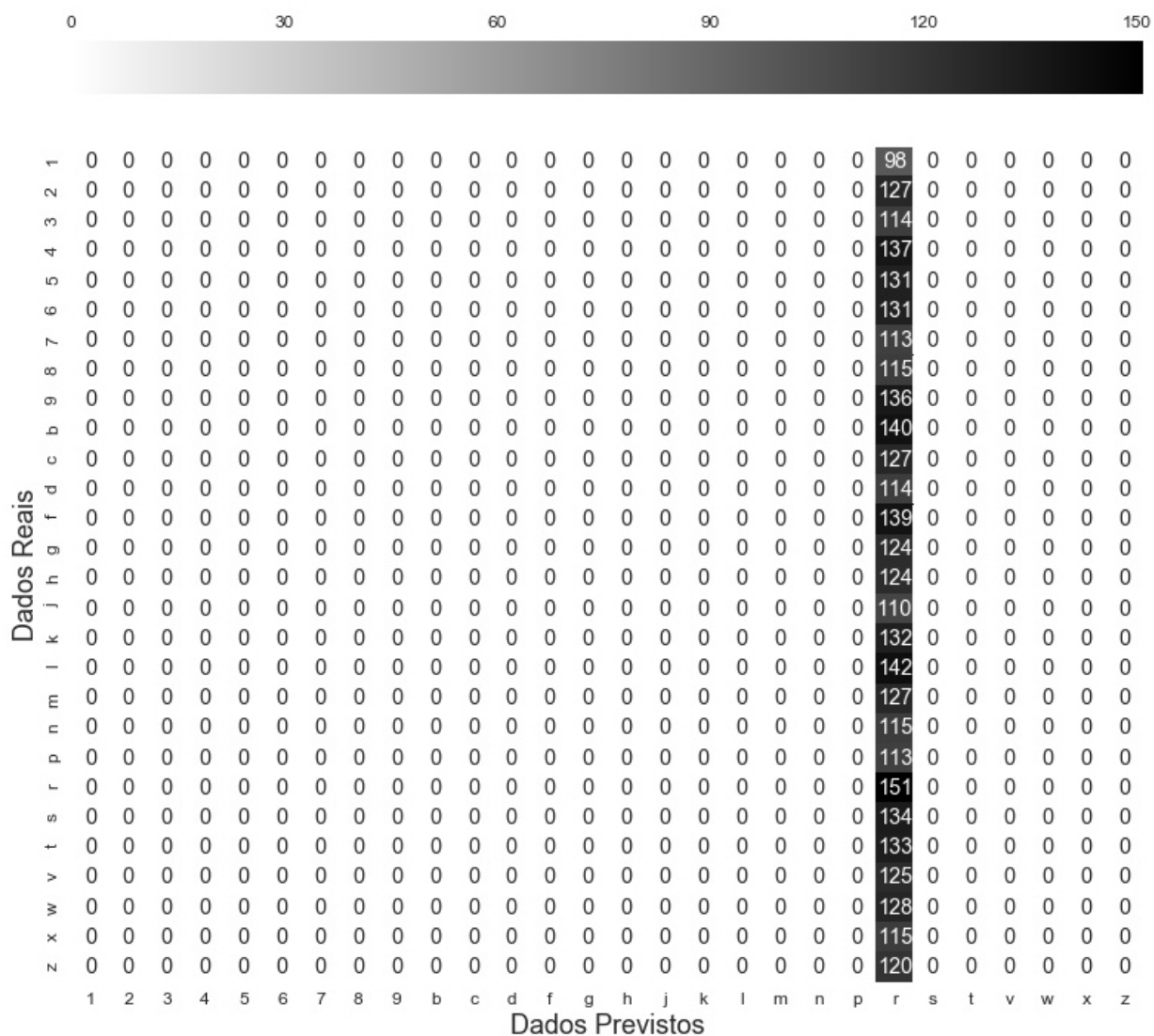
- Tamanho do *kernel*:  $3 \times 3$  ou  $5 \times 5$ ;
- Número de neurônios na primeira camada convolucional: 10, 20 ou 30;
- Número de neurônios na segunda camada convolucional: 50 ou 100;
- Taxa de aprendizado: 0.01 ou 0.005;
- Funções de ativação das camadas internas: ReLu ou Leaky ReLu;
- *Solvers*: SGD ou Adam.

A combinação desses parâmetros resultou na proposição de 288 modelos distintos para o problema que, após treinados e testados, obtiveram as métricas de F-Score mostradas na Tabela 5.1. Tomando a CNN com melhor desempenho neste cenário, a qual utiliza *kernels* de tamanho  $3 \times 3$ , 10 neurônios da primeira camada convolucional, 50 neurônios da segunda camada convolucional, taxa de aprendizado de 0.01, *Leaky ReLu* como função de ativação, Adam como *solver* e regularização L2 e que obteve F-Score igual a 0.0430, tem-se a matriz de confusão ilustrada na Figura 5.3. A partir destes resultados, embora tenha havido uma leve melhora na métrica de desempenho, de 0.0356 no Cenário 1 para 0.0430 no Cenário 2, concluiu-se que a busca em *grid* não foi efetiva em encontrar melhores parâmetros que promovessem uma melhoria de desempenho significativa, pois o valor da métrica continua muito distante de valores considerados razoáveis. Novas estratégias então foram exploradas nos cenários seguintes com vistas a mitigar estas dificuldades.

Tabela 5.1: Métricas resultantes da etapa de testes das 288 CNNs propostas no Cenário 2.

Resultado	Valor Obtido
Média do F-Score	0.0350
Mediana do F-Score	0.03528
Desvio Padrão do F-Score	0.0034
Maior F-Score Obtido	0.0430
Acurácia do Modelo com Maior F-Score	0.0310
Menor F-Score Obtido	0.0278

Figura 5.3: Matriz de confusão do Cenário 2.



### 5.3 Cenário 3: Normalizando os Dados de Entrada

Com o Cenário 2 não obteve resultados satisfatórios, e observando todos os seus modelos treinados repetindo a abordagem contraproducente encontrada no Cenário 1, cogitou-se mudar a forma em que os exemplos do *Dataset* eram entregues à CNN. Assim, seguindo sugestões da literatura (CHOLLET, 2017, vide Seção 5.2.4), os valores do canal da escala de cinza de cada imagem foram normalizados, passando de uma escala de  $[0, 255]$  para  $[0, 1]$ , ainda respeitando as mesmas proporções, onde o valor 0 indica a cor totalmente preta e o valor 1 indica a cor totalmente branca, passando por todo o espectro de tonalidades de cinza nos valores intermediários.

Após a normalização efetuada, repetiu-se a mesma busca em *grid* da rede neural LeNet realizada no Cenário 2 para verificar se a nova forma de entrada de dados iria trazer métricas superiores, e assim, após 6 horas e 5 minutos de processamento, foram obtidos os valores mostrados na Tabela 5.2.

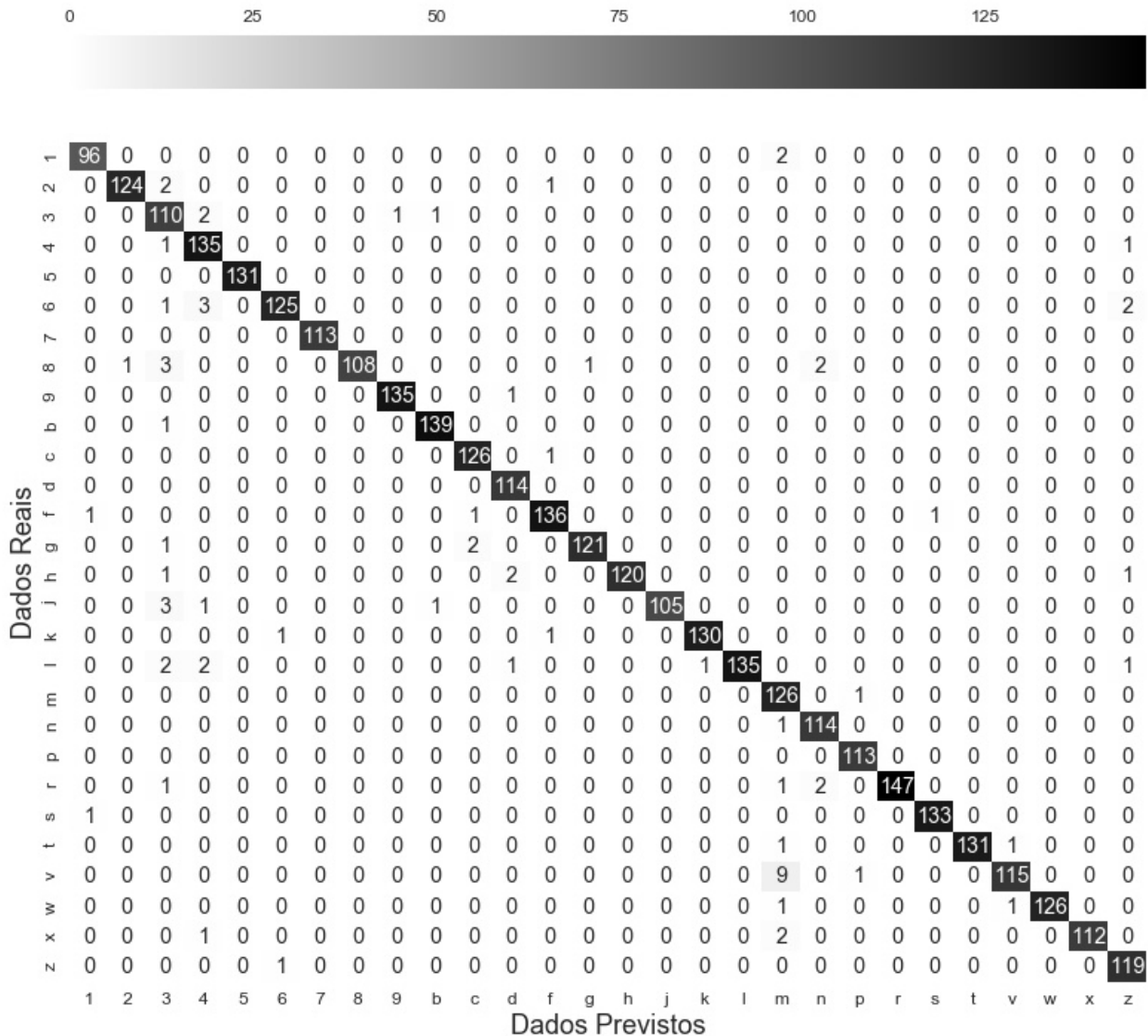
Tabela 5.2: Métricas resultantes da etapa de testes das 288 CNNs propostas no Cenário 3.

Resultado	Valor Obtido
Média do F-Score	0.5430
Mediana do F-Score	0.9172
Desvio Padrão do F-Score	0.4528
Maior F-Score Obtido	0.9784
Acurácia do Modelo com Maior F-Score	0.9737
Menor F-Score Obtido	0.0279

A partir da análise dos resultados obtidos, foi possível verificar que a normalização de pesos provou, de fato, uma melhoria significativa no desempenho, garantindo um aprendizado efetivo que refletiu positivamente na etapa de testes. Ao considerar a CNN com melhor desempenho e valor de F-Score igual a 0.9784, foi gerada então a matriz de confusão mostrada na Figura 5.4. Assim, a partir desta matriz, pode-se olhar a prevalecência de valores na diagonal principal, refletindo um alto número de acertos na previsão pela rede. Os erros ainda restantes, encontrados em valores fora da diagonal principal, passam a ser típicos de cenários de Aprendizado

de Máquina, podendo ser justificados em função do ruído introduzido pelo CAPTCHA, que dificultou a real identificação do caractere associado.

Figura 5.4: Matriz de confusão do Cenário 3.



## 5.4 Cenário 4: Ampliando o Ajuste de Parâmetros da LeNet

Ainda perseguindo uma melhoria nos valores de métricas encontradas no Cenário 3, uma nova busca em *grid* foi feita, mantendo a normalização, mas considerando uma variação maior de

parâmetros que não foram alterados até então. Estes parâmetros considerados foram:

- Tamanho do *kernel*:  $3 \times 3$  ou  $5 \times 5$ ;
- Número de neurônios na primeira camada convolucional: 5, 7 ou 10;
- Número de neurônios na segunda camada convolucional: 3 ou 5;
- Taxa de aprendizado: 0.01 ou 0.005;
- Funções de ativação das camadas internas: SeLu ou Leaky ReLu;
- *Solvers*: Adadelta ou Adam.
- *Strides* (variando independentemente nas camadas): 1 ou 2;

Assim, foram geradas 384 novas variações da LeNet que, após 17 horas e 38 minutos de processamento, tiveram seus resultados obtidos, os quais encontram-se sintetizados na Tabela 5.3.

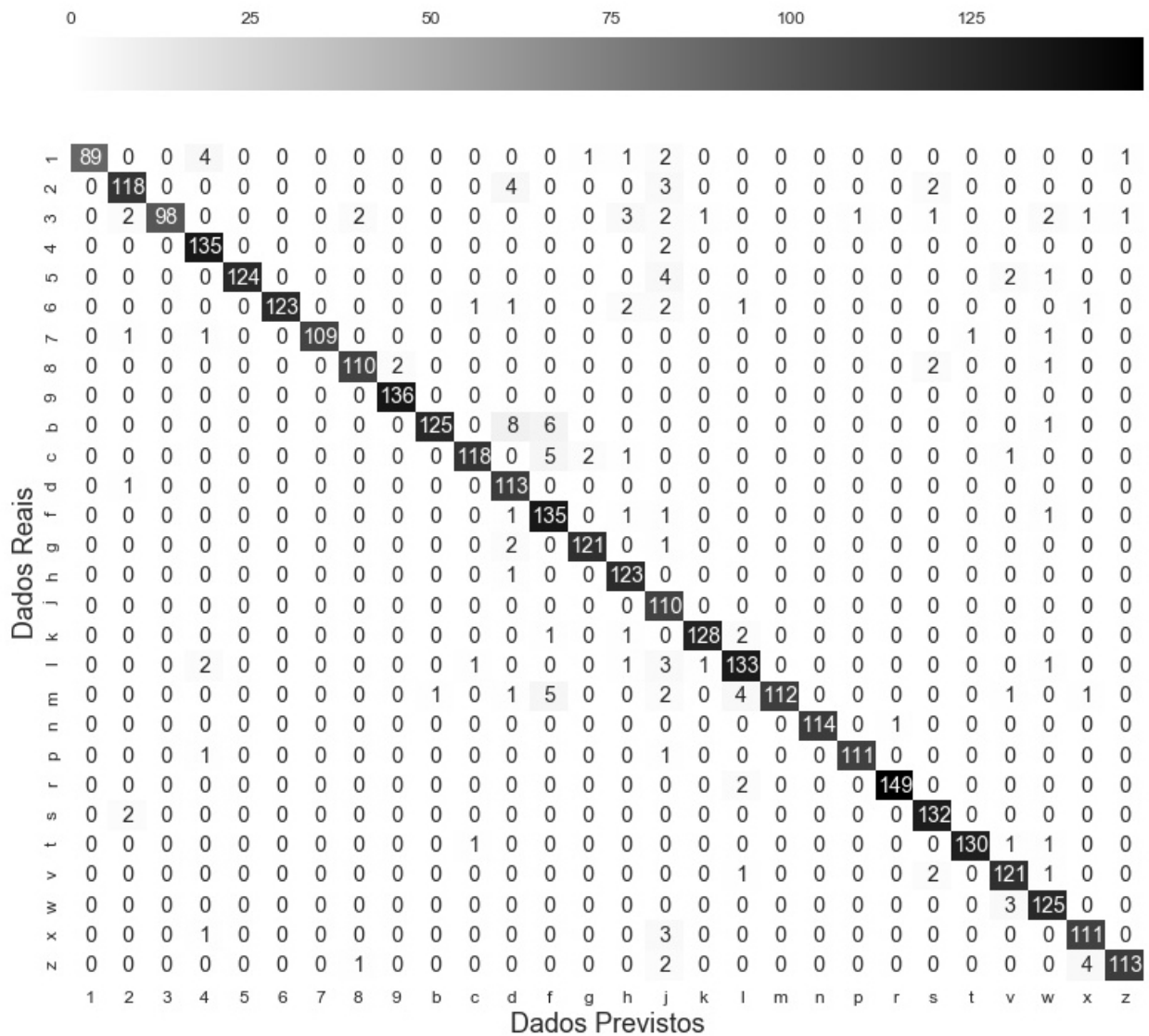
Tabela 5.3: Métricas resultantes da etapa de testes das 384 CNNs propostas no Cenário 4.

Resultado	Valor Obtido
Média do F-Score	0.5532
Mediana do F-Score	0.8135
Desvio Padrão do F-Score	0.3959
Maior F-Score Obtido	0.9576
Acurácia do Modelo com Maior F-Score	0.9561
Menor F-Score Obtido	0.0036

Apesar de resultados próximos aos encontrados no Cenário 3, o melhor modelo apresentou F-Score de 0.9576 e não conseguiu superar o valor encontrado anteriormente, que havia sido de 0.9784. A matriz de confusão deste modelo pode ser vista na Figura 5.5. Apesar de não ter havido melhoras no desempenho nesta nova busca de parâmetros, esta é uma estratégia típica de Aprendizado de Máquina conduzida quando se obtém um bom modelo para a tarefa, na tentativa de otimizar os parâmetros, melhorando-os. Ela foi conduzida por repetir a *práxis*

sugerida normalmente neste domínio, mas não é garantida de produzir melhores resultados, tais como o que ocorreu neste Cenário 4.

Figura 5.5: Matriz de confusão do Cenário 4.



## Capítulo 6

# Considerações Finais

Neste trabalho foi ilustrada uma aplicação de redes neurais convolucionais para endereçar o problema de identificação de caracteres em CAPTCHAs de texto. Como prova de conceito utilizou-se o CAPTCHA adotado pela Plataforma Lattes. Uma das vantagens de endereçar a resolução automática deste CAPTCHA consiste na possibilidade de trabalhos posteriores endereçarem técnicas de mineração de dados nesta plataforma, corroborando para a crescente política de transparência de acesso a dados de interesse público.

Para tanto, foi necessário criar um *dataset*, para o qual desenvolveu-se um *script* para fazer o *download* automático das imagens dos CAPTCHAs. Em seguida, as imagens foram tratadas com para que se subtraísse o *background* e extraísse os caracteres individualmente com uma menor quantidade de ruídos, utilizando, para tanto, técnicas de segmentação oriundas da Visão Computacional. Assim, 17.781 imagens obtidas e classificadas manualmente em 28 tipos de letras e números, contabilizados de forma desbalanceada. A partir desse ponto, o *dataset* foi então dividido em 3 partes: treinamento com 70% das imagens, validação com 10% e teste com 20%. Este particionamento fixo foi então utilizado como método *Holdout* de validação cruzada das CNNs que seriam propostas, treinadas e testadas.

Devido a natureza do problema se assemelhar à outros já documentados pela literatura (LECUN et al., 1998; CUN et al., 1990), a arquitetura canônica LeNet foi utilizada como referência para as redes neurais convolucionais que seriam propostas. Inicialmente, esta rede foi utilizada com sua configuração original e os resultados do treino e teste revelaram que não

se mostrava adequada para uso, obtendo um desempenho risível. Partiu-se então para uma busca em *grid* por melhores parâmetros, complementada por técnicas de regularização de pesos e normalização. Esta busca em *grid* resultou na proposição de 636 variações da LeNet original, as quais foram então treinadas e testadas segundo dois cenários. Ao fim, o resultado obtido permitiu a identificação de uma variação da LeNet com desempenho aferido por micro F-Score igual a 0.9784. Seguiu-se ainda com uma busca em *grid* para melhorar este resultado, mas que não resultou em valores encorajadores.

Desta feita, os resultados obtidos decorrentes deste trabalho de conclusão de curso permitiram a identificação de uma rede LeNet capaz de endereçar a tarefa de reconhecimento automático do CAPTCHA da Plataforma Lattes, com índice de acerto superior a 97,37% por caractere. Assim, como proposto, a resolução do CAPTCHA da Plataforma Lattes é feita inicialmente com sua captura, seguido do processamento e separação dos caracteres individuais e classificação com o uso da CNN LeNet previamente treinada, de maneira que a probabilidade de acerto de um CAPTCHA contendo 4 caracteres é igual a, aproximadamente, 89,89%. Em contraste com os trabalhos relacionados identificados na literatura também para a Plataforma Lattes, conforme apresentado no Capítulo 4, o estado da arte para este problema correspondia à taxa de acerto de 87,5%. Pode-se afirmar, portanto, que a solução proposta neste trabalho corroborou para uma melhoria na tarefa considerada, propondo novo *baseline* para esta tarefa.

Embora houvesse uma analogia com o problema do reconhecimento de dígitos do MNIST, é interessante ressaltar que a solução previamente existente para este domínio não pôde ser diretamente aproveitada no cenário do CAPTCHA da Plataforma Lattes aqui considerado. Para contornar, diferentes outras técnicas de busca e ajuste de parâmetros e de pré-processamento foram consideradas. Isto ressalta a não-trivialidade dos problemas de Aprendizado de Máquina, a importância do conhecimento técnico deste domínio e a consulta constante à literatura na área para uso de técnicas consolidadas e de novas práticas que venham a ser propostas.

Trabalhos futuros podem dar continuidade ao estudo aqui realizado, sendo desenvolvidos com o intuito de alcançar melhores métricas através da utilização de arquiteturas diferentes, novas técnicas e outras ferramentas disponíveis no mercado.



# Referências Bibliográficas

AHN, L. von; BLUM, M.; LANGFORD, J. Telling humans and computer apart (automatically) or how lazy cryptographers do ai. Communications of the ACM, 2004.

BENENSON, R. *What is the class of this image?* 2018. Disponível em <[http://rodrigob.github.io/are\\_we\\_there\\_yet/build/classification\\_datasets\\_results.html](http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html)>. Acessado em 5 de dezembro de 2018.

BRAGA, A. de P.; CARVALHO, A. P. de Leon F. de; LUDERMIR, T. B. *Redes Neurais Artificiais – Teoria e Aplicações*. 2. ed. Brasil: LTC, 2007.

BRINK, H.; RICHARDS, J. W.; FETHEROLF, M. *Real-World Machine Learning*. Estados Unidos: Manning Publications, 2017.

BUDUMA, N. *Fundamentals of Deep Learning*. Estados Unidos: O'Reilly Media, 2017.

CHOLLET, F. *Deep Learning with Python*. 1. ed. Shelter Island, New York: Manning Publications, 2017.

CUN, Y. L. et al. Handwritten zip code recognition with multilayer networks. In: IEEE. *Proceedings of the 10th International Conference on Pattern Recognition*. New Jersey, Estados Unidos, 1990. v. 2, p. 35–40.

FACELI, K. et al. *Inteligência Artificial – Uma abordagem de aprendizado de máquina*. Rio de Janeiro: Editora LTC, 2015.

FLACH, P. *MACHINE LEARNING - The Art and Science of Algorithms that Make Sense of Data*. 1. ed. Nova York, Estados Unidos: United States of America by Cambridge University Press, 2012.

GOOGLE. *Google Imagens*. 2018. Disponível em <<https://images.google.com/>>. Acessado em 5 de dezembro de 2018.

GOOGLE. *What is reCAPTCHA?* 2018. Disponível em <<https://developers.google.com/recaptcha/>>. Acessado em 5 de dezembro de 2018.

GRIES, P.; CAMPBELL, J.; MONTOJO, J. *Practical Programming: An Introduction to Computer Science Using Python*. 2. ed. Estados Unidos: Pragmatic Bookshelf, 2013.

GULLI, A.; PAL, S. *Deep Learning with Keras*. 1. ed. Reino Unido: Packt Publishing, 2017.

HAYKIN, S. *Neural Networks and Learning Machines*. 3. ed. Nova Jersey: Pearson, 2009.

- KHAN, S. et al. *A Guide to Convolutional Neural Networks for Computer Vision*. [S.l.]: Morgan and Claypool, 2018.
- KUBAT, M. *An Introduction to Machine Learning*. Estados Unidos: Springer, 2015.
- LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998.
- MARSLAND, S. *Machine Learning - An algorithmic perspective*. 2. ed. Estados Unidos: CRC Press, 2015.
- MATIAS, P. *Resolução automática de CAPTCHAs de voz*. 2011. Disponível em <<https://hal9k.ifsc.usp.br/~matias/disciplinas/posgrad/2011-Processamento-Digital-de-Audio-e-Voz/paper/paper.pdf>>. Acessado em 5 de dezembro de 2018.
- PINTO, V. A. *Redes Neurais Convolucionais de Profundidade para Reconhecimento de Textos em Imagens CAPTCHA*. Florianópolis, Brasil: Universidade Federal de Santa Catarina, 2016. Trabalho de Conclusão de Curso, Bacharelado em Sistemas de Informação.
- ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, p. 386–408, 1958.
- RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.
- SANTA-ROSA, J. G.; LIBERADO, J. R. G. V. Recurso de interface CAPTCHA: Segurança e usabilidade na distinção entre humano e máquina. *Ergodesign & HCI*, v. 1, p. 34–43, 2013.
- SANTOS, V. D. dos. *Análise de tecnologias de reconhecimento para quebra de CAPTCHAs*. Marília, São Paulo: Centro Universitário Eurípedes de Marília, 2016. Trabalho de Conclusão de Curso, Bacharelado em Ciência da Computação.
- VASCONCELOS, C. N.; CLUA, E. W. G. Deep learning – teoria e prática. In: *Anais do Congresso da Sociedade Brasileira de Computação, Jornadas de Atualização em informática*. São Paulo: Sociedade Brasileira de Computação, 2017.
- YAN, J.; AHMAD, A. S. E. Usability of captchas or usability issues in captcha design. In: *Proceedings of the 4th symposium on Usable privacy and security*. Reino Unido: ACM, 2008. p. 44–52.