Trabalho de Conclusão de Curso

Quebra e Aprimoramento do Teste de Turing da Plataforma Lattes do CNPq

Bruno Georgevich Ferreira

Orientador:

Prof. Dr. Tiago Figueiredo Vieira

Maceió, Abril de 2019

Bruno Georgevich Ferreira

Quebra e Aprimoramento do Teste de Turing da Plataforma Lattes do CNPq

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. Tiago Figueiredo Vieira

Maceió, Abril de 2019

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

Prof. Dr. Tiago Figueiredo Vieira - Orientador
Instituto de Computação
Universidade Federal de Alagoas

Prof. Dr. Marcelo Costa Oliveira - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Prof. Dr. Thales Miranda de Almeida Vieira - Examinador
Instituto de Matemática
Universidade Federal de Alagoas

Maceió, Abril de 2019

Agradecimentos

Agradeço, primeiramente, aos meus pais Joyce e Reynaldo, por me ensinarem os princípios da vida e valores que levo comigo sempre. Por me mostrarem a vida sobre uma óptica verdadeira e por serem meus amigos em todas as ocasiões.

Aos meus amados irmãos que sempre foram meus amigos, por viverem e crescerem comigo e por nunca terem desistido de mim.

A minha companheira, Lívia Enders, por ser a minha melhor amiga, por sempre acreditar em mim, por me apoiar em todas as minhas empreitadas, por ter me ajudado a realizar todos os passos da minha trajetória.

Agradeço aos meus professores, coordenadores, diretores, especialistas, mestres e doutores que agregaram conhecimento e experiências à minha trajetória acadêmica.

Agradeço aos meu amigos do laboratório Charles Babbage, por sempre acreditarem em mim, por me ajudarem e me incentivarem a ser o meu melhor.

Agradeço ao pessoal do RAS, do DIACOM e do Ramo IEEE, por me incentivarem a ser melhor e tornarem essa trajetória mais leve.

Agradeço aos meus amigos de infância, Felipe, Leonardo e Luciano, por sempre estarem ao meu lado e serem tão importantes na minha vida e trajetória.

Por fim, agradeço em especial ao meu orientador, Tiago Figueiredo Vieira, pelo apoio no Projeto de Pesquisa, na Monitoria e nos projetos desenvolvidos ao longo do curso. Por toda dedicação, paciência, sabedoria e incentivo, sendo também um dos responsáveis por esse Trabalho de Conclusão de Cursos se tornar possível.

Enfim, agradeço a todos que tornaram a realização desse sonho possível.

“A vida e o tempo são os dois maiores professores. A vida nos ensina a fazer bom uso do tempo enquanto o tempo nos ensina o valor da vida.”

– Autor Desconhecido

Resumo

A internet tornou-se um instrumento de grande importância na rotina de uma parcela considerável da população. Entretanto, houve um aumento no número de cibercrimes reportados, o que obriga alguns sistemas a intensificarem sua segurança para com o usuário. Uma das formas de filtrar usuários que são pessoas de *scripts* maliciosos é a utilização de um *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA), em português "Teste de Turing Público Completamente Automatizado para Diferenciação entre Computadores e Humanos". Entretanto, se um CAPTCHA for facilmente decodificado por uma máquina, o mesmo perde o seu propósito. Desta forma, nesse trabalho é realizada uma análise detalhada sobre as fragilidades encontradas no CAPTCHA presente na plataforma Lattes e é proposto um aprimoramento do esquema. É mostrado que o CAPTCHA do Lattes pode ser facilmente quebrado com técnicas simples, enquanto que o proposto apresenta-se mais robusto. Também foi criada uma solução robusta capaz de solucionar o CAPTCHA do Lattes. Ao final foi desenvolvido um *software* capaz de gerar os CAPTCHAs provenientes desse novo esquema.

Palavras-chave: CAPTCHA; Aprendizagem Profunda; Lattes; Processamento de Imagem

Abstract

The internet has become a very important tool in the routine of a considerable part of the population. However, there has been an increase in the number of reported cybercrimes, which forces some systems to improve their security towards the user. One of the ways to filter users who are people from malicious scripts is to use a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). However, if a CAPTCHA is easily decoded by a machine, it loses its purpose. In this work, a detailed analysis is performed on the weaknesses found in the CAPTCHA present on the Lattes platform and an improvement of the scheme is proposed. It is shown that Lattes' CAPTCHA can be easily broken with simple techniques, whereas the proposed one is more robust. A robust solution was also created to solve Lattes' CAPTCHA. At the end, a software capable of generating the CAPTCHAs from this new scheme was developed.

Keywords: CAPTCHA; Deep Learning; Lattes; New Scheme; Image Processing

Sumário

Lista de Abreviaturas e Siglas

CAPTCHA	*Completely Automated Public Turing Testto Tell Computers and Humans Apart*
CNN	*Convolutional Neural Networks*
R-CNN	*Region-based Convolutional Neural Network*
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
PDI	Processamento Digital de Imagens
HIP	*Human Interaction Proofs*
CMU	*Carnegie Mellon University*
OpenCV	*Open Source Computer Vision Library*
RGB	*Red, Green and Blue*
11 TM	T*emplate Matching*
SSD	*Sum of Squares Differences*
CCR	*Cross Correlation*
CCOE	*Cross Correlation Coefficient*
ML	*Machine Learning*
DL	*Deep Learning*
GPU	*Graphical Processor Unit*
ReLU	*Rectified Linear Unit*
Tanh	Tangente Hiperbólica
SVM	*Support Vector Machine*
BB	*Bounding Boxes*
RPN	*Region Proposal Network*
V1	Inception V1
V2	Inception V2
V3	Inception V3
V4	Inception V4
RV1	Inception ResNet V1
RV2	Inception ResNet V2

Lista de Figuras

1

Introdução

1.1 Contextualização

A internet tornou-se um instrumento importante na rotina, trabalho e lazer, de uma parcela considerável da população. De acordo com Usmani et al. (2019), houve um aumento no número de cibercrimes reportados, o que obriga alguns sistemas, que lidam com informações sigilosas, a intensificar sua segurança para com o usuário. A internet é capaz de fornecer diversos serviços e isso obteve um crescimento exponencial ao ser implantada a cultura dos aplicativos. Entretanto, muitos deles precisam realizar autenticações com usuário, seja pedindo credenciais, como login e senha, ou descobrindo se o usuário em questão trata-se de uma pessoa ou um robô de *software* malicioso. Para tentar distinguir um humano de um robô, é preciso realizar um teste conhecido como Teste de Turing, onde o sujeito é submetido a um diálogo, cujas respostas serão julgadas por um humano, o que definirá se é uma pessoa ou uma máquina.

Uma derivação do tradicional Teste de Turing é o Teste de Turing Reverso, que consiste em um robô submeter o sujeito em questão a um desafio e julgá-lo a partir do resultado, distinguindo-o entre humano ou máquina. A diferenciação entre um teste e o outro consiste que no reverso quem julga é uma máquina e não mais uma pessoa. Essa variação do Teste de Turing tem como um dos objetivos dar autonomia para *softwares* e *scripts* discernirem quais de seus usuários são robôs e, possivelmente, realizar ações com relação a isso. Segundo Gafni e Nagar (2016), um *Completely Automated Public Turing Test to Tell Computers and Humans Apart*[1] (CAPTCHA) é uma operação similar ao Teste de Turing Reverso, onde o mesmo desempenha uma autenticação do usuário conhecida como *challenge-response authentication*[2] e, caso o usuário resolva o desafio proposto corretamente, o mesmo terá

[1] Em português, "Teste de Turing Público Completamente Automatizado para Diferenciação entre Computadores e Humanos". (Tradução Livre).

[2] Em português, "Autenticação Desafio-Resposta". (Tradução Livre).

permissão de prosseguir com seu objetivo.

Para Gafni e Nagar (2016), juntamente com a criação dos CAPTCHAs em 2000, por John Langford, Nicholas J. Hooper e Luis Von Ahn, iniciaram-se os estudos para buscar suas fragilidades e solucioná-las utilizando *softwares* que imitassem a ação de humanos. Segundo Chen et al. (2017), foi na tentativa de verificar a confiabilidade e segurança dos CAPTCHAs que os estudos sobre quebrá-los – termo usado na academia para referir-se à solucionar um CAPTCHA – foram iniciados. Desta forma, na medida em que pesquisadores conseguiam resolver novos esquemas de CAPTCHAs, outros estudos surgiam e melhorias eram agregadas no que era dito como estado da arte dos CAPTCHAs. Esse ciclo evolutivo perpetuou por algum tempo, sendo aprimorado e evoluído por diversos centros de pesquisa ao redor do mundo.

Ao longo do tempo, com a intensificação dos estudos, vários tipos de CAPTCHAs foram desenvolvidos. Segundo Gafni e Nagar (2016), os tipos de CAPTCHAs mais utilizados são os textuais, de imagem, sonoros e os de pré-análise do comportamento do usuário. Para cada tipo, existem técnicas específicas de quebrá-lo, entretanto, um conjunto de técnicas que propõe possíveis soluções para todos, consiste na utilização de algoritmos que permitem que computadores possam aprender – *Machine Learning* –, mais especificamente, o *Deep Learning*, que consiste na utilização de redes neurais com mais camadas ocultas para extração de características mais complexas da base de dados. Com a popularização de *Deep Learning*, muitos problemas de alta complexidade – como por exemplo o processamento de linguagem natural, a classificação de imagens e a detecção de faces – agora, estão sendo possíveis devido ao alto poder de processamento que as *Graphical Processor Units* (GPU) estão agregando aos computadores e ao grande volume de dados presentes na internet.

No que tange CAPTCHAs textuais, Chen et al. (2017) define dois modelos de métodos para solucioná-los: métodos baseados em segmentação e em reconhecimento, onde o primeiro visa segmentar os caracteres da imagem e, o segundo, identificá-los. Para o autor, com o sucesso do modelo de reconhecimento, houve um enfoque maior dos pesquisadores na parte da segmentação. De acordo com Krizhevsky, Sutskever e Hinton (2012), técnicas de *Deep Learning* como as *Convolutional Neural Networks* (CNN) são boas em classificar imagens. Já para Girshick (2015), as *Region-based Convolutional Neural Network* (R-CNN) apresentam-se capazes de detectar os objetos de interesse, ou seja, classificá-los e indentificar as suas posições na imagem. As CNNs e as R-CNNs apresentam-se suficientemente capazes de realizar o reconhecimento dos caracteres.

O modelo proposto por Liu Rong Zhang (2017) é capaz de unir os métodos de segmentação e reconhecimento utilizando uma CNN. Porém, para isso, eles tiveram que utilizar uma espécie de janela deslizante que, a cada passo, classificava a imagem. Nessa questão, utilizar uma R-CNN faz-se mais interessante pelo fato de sua própria arquitetura de camadas segmentar diversas regiões de interesses, possíveis caracteres, e depois classificá-las.

1.2 Motivação

A plataforma Lattes é um sistema único de informação do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), que integra bases de dados de currículos, grupos de pesquisa e instituições. Na plataforma, é possível encontrar o currículo Lattes, um padrão nacional de currículos que representa a vida acadêmica, atual e pregressa, do indivíduo e hoje adotado pela maioria das instituições de fomento, universidades e institutos de pesquisa do Brasil. Alunos de curso técnico, graduação e pós-graduação são estimulados, durante toda a sua trajetória acadêmica, a alimentar e enriquecer tal currículo, manipulando-o através da própria plataforma.

No processo de busca dos currículos Lattes, indexados pela plataforma, o usuário precisa selecionar o currículo que ele deseja consultar. Para isso, o sistema exibe uma tela que apresenta um CAPTCHA textual em seu centro, de modo que para prosseguir, o usuário deve escrever corretamente o texto disposto na imagem na caixa de texto disponível para esta finalidade. O CAPTCHA apresentado tem como objetivo impedir que o sistema seja fonte de informações para outros sistemas, que utilizam de *scripts* que automatizam a coleta das informações presentes na plataforma e as exibem em seus próprios sites ou para outros fins.

O CAPTCHA textual exibido na tela de consulta de um currículo Lattes apresenta diversas características que o fragilizam e, portanto, o tornam facilmente solucionável. Um dos padrões detectados no CAPTCHA do Lattes consiste na apresentação de apenas uma única fonte de letras e números, e ela não sofre deformações – rotação, deformação, translação e redimensionamento – fazendo com que uma letra seja igual em todos as imagens em que a mesma aparece. Outro padrão observado refere-se ao fato de que os caracteres são apresentados na cor branca e os fundos do CAPTCHA não são constituídos de imagens geradas aleatoriamente, ou seja, são figuras predefinidas. Um terceiro padrão percebido está nos ruídos inseridos no CAPTCHA, que são bem simples e podem ser facilmente removidos a partir de técnicas de Processamento Digital de Imagens (PDI). Por último, observou-se que os textos sempre apresentam apenas quatro caracteres e são posicionados na mesma parte da imagem, não exibem letras ou números repetidos, não contém nenhuma vogal, não tem o número zero e todas as letras são maiúsculas.

As inúmeras fragilidades do CAPTCHA do Lattes comprometem sua capacidade de fornecer autenticação adequada. Desta forma, a aplicação de técnicas mais simples de PDI ou utilização de métodos mais robustos com *Deep Learning*, conseguem obter altas taxas de sucesso ao quebrá-lo. Nesse sentido, este cenário enseja a proposta de um novo esquema de CAPTCHA, a fim de corrigir as fragilidades supracitadas, visando obter mais robustez ao ser submetido a técnicas que tentem solucioná-lo, sem comprometer a legibilidade necessária para com os humanos.

1.3 Objetivos

O presente trabalho tem como objetivo geral o desenvolvimento de um novo esquema de CAPTCHA capaz de ser mais resistente à quebra, quando comparado com o do Lattes. Entretanto, para obter tal resultado, foram definidos cinco objetivos específicos que marcaram toda a trajetória realizada por este estudo. Primeiro, desenvolver um algoritmo munido de técnicas de PDI, na tentativa de mostrar o quão frágil é o CAPTCHA do Lattes. Logo em seguida, treinar um R-CNN capaz de quebrar o modelo do Lattes, estabelecendo uma métrica de comparação de qualidade com o novo esquema de CAPTCHA que será proposto.

Após comprovar que as duas técnicas propostas conseguiram quebrar o CAPTCHA do Lattes, será desenvolvido um novo esquema de CAPTCHA mais resistente. Tendo o esquema proposto bem definido, o próximo passo será testá-lo treinando uma R-CNN, de forma a comprovar que o novo esquema se apresenta muito mais resistente à quebra. Por fim, um questionário será aplicado a um conjunto de pessoas, visando mostrar que o esquema proposto não perdeu muita legibilidade para os humanos.

A estrutura do presente trabalho inicia-se por uma breve explicação sobre todas as teorias, técnicas e tecnologias utilizadas para embasar teoricamente este estudo. Logo em seguida, será detalhada toda a metodologia utilizada nas etapas do decorrer da pesquisa. Após isso, serão explicitados todos os resultados obtidos a partir da aplicação da metodologia definida. E, por fim, serão apresentadas as conclusões obtidas a partir da análise de todos os resultados encontrados.

2

Referencial Teórico

2.1 CAPTCHA

O *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA), segundo Belk et al. (2015), é um mecanismo de defesa amplamente utilizado por provedores de serviços – em sua maioria *websites* – capaz de determinar quando a entidade que interage com seu sistema é um agente humano e não um agente malicioso. Para Gafni e Nagar (2016), o CAPTCHA é um teste que tem como objetivo distinguir entre *softwares* maliciosos automáticos e usuários reais, no que o autor chama de 'a era das ameaças cibernéticas de segurança'. O autor salienta ainda que, nesta era, é de suma importância saber se quem está utilizando o seu *website* é uma pessoa real ou um robô malicioso.

Belk et al. (2015) explicam que as *Human Interaction Proofs*[3] (HIP) são mecanismos de segurança amplamente difundidos, que provam – com alta confiabilidade – que a entidade que está interagindo com o serviço é um ser humano e não um software malicioso. Para o autor, o CAPTCHA é um teste *challenge-response* da HIP amplamente utilizado atualmente. É relatado por Chen et al. (2017), que em setembro de 2000, a equipe de pesquisa da Carnegie Mellon University (CMU) projetou os primeiros CAPTCHAs baseados em texto, da série Gimpy, para resistir a anúncios maliciosos enviados por *scripts* ilegais, nas salas de bate-papo do Yahoo. Um exemplo do Gimpy pode ser visto na Figura 1. Ainda de acordo com o autor, de 2002 a 2005, os seminários internacionais da HIP foram realizados e muitos resultados de pesquisas relacionadas foram publicados, estimulando a comunidade a estudar ainda mais sobre o assunto.

[3] Em português, "Provas de Interação Humana". (Tradução Livre)

Figura 1: Exemplo de um CAPTCHA Gimpy

Fonte: Chen et al. (2017, p. 2)

Segundo Gafni e Nagar (2016), existem diversos tipos de CAPTCHAS, entretanto, os mais conhecidos são os textuais, de imagens, sonoros e os de pré-análise do comportamento do usuário. O autor descreve que os CAPTCHAS textuais apresentam uma imagem que contém um texto, porém é composta por uma série de ruídos e distorções. Para quebrá-los, o usuário precisa decifrar a imagem e escrever o texto contido na mesma em uma caixa de texto disponibilizada, na tentativa de acertá-lo. Em relação ao CAPTCHA de imagem, há uma série de imagens que o usuário precisa selecionar as que se relacionam a um determinado tipo ou realizem alguma atividade, como ajustar à sua deformação de forma que ela se torne visível novamente. Vale ressaltar que o CAPTCHA de imagem não exige que o usuário escreva nenhum tipo de texto. Exemplos dos CAPTCHAS textual e de imagem podem ser vistos nas Figuras 2a e 2d, respectivamente.

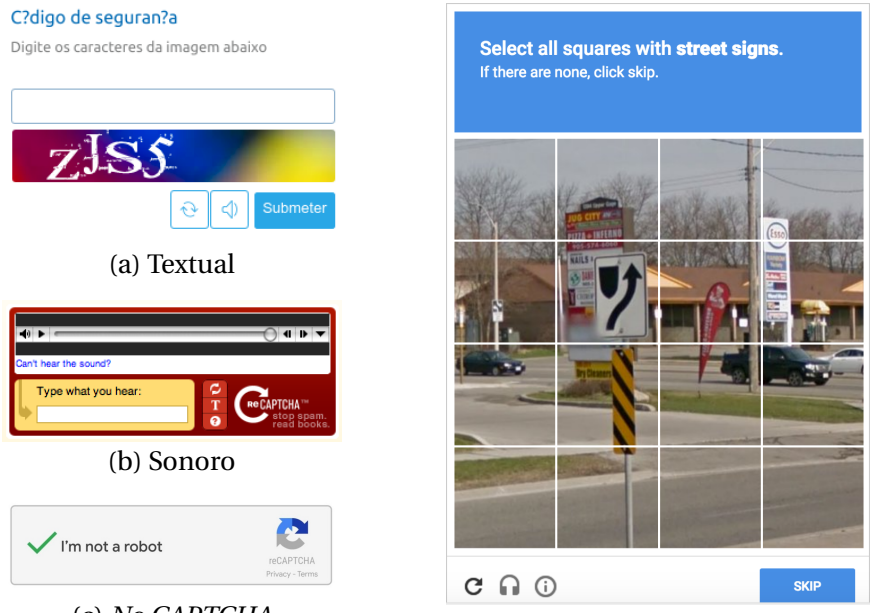
No que se trata do CAPTCHA sonoro, ainda de acordo com o autor, o mesmo foi desenvolvido no intuito de permitir que esse tipo de teste fosse realizado por pessoas que apresentassem algum impedimento em sua visão e que inviabilizasse a realização dos teste visuais. Para decifrar, o usuário precisa ouvir uma sequência de letras e/ou números e escrevê-la em uma caixa de texto. Um exemplo de CAPTCHA sonoro pode ser visto na Figura 2b. De acordo Belk et al. (2015), os CAPTCHAs visuais apresentam pouco suporte para usuários com problemas de visão.

Em 2013, o reCAPTCHA, serviço da Google, começou a implementação de um algoritmo capaz de analisar como o usuário se comportava ao manipular um *software*, no intuito de predizer se este se trata de uma pessoa ou uma máquina. Como explicado por Gafni e Nagar (2016), essa forma de análise ficou conhecida por *No CAPTCHA* ou CAPTCHA de pré-análise do comportamento do usuário, o qual consiste em uma caixa de seleção que afirma "*I'm not a robot*"("Eu não sou um robô", tradução livre). Ao marcar a caixa de seleção, o algoritmo que estava analisando o comportamento do usuário no sistema, julgará se o mesmo é um humano ou um robô. Caso seja julgado por humano, o teste CAPTCHA será finalizado e permitirá que o usuário prossiga. Caso não, o mesmo será submetido a um teste mais complexo, geralmente sendo um CAPTCHA de imagem. Um exemplo de *No CAPTCHA* pode ser visto na Figura 2c.

Para Belk et al. (2015) e Gafni e Nagar (2016), os testes CAPTCHA devem ser facilmente

desvendados por seres humanos, porém impossíveis para máquinas. De acordo com Gafni e Nagar (2016), a medida que a segurança e a robustez do CAPTCHA vai aumentando, sua usabilidade vai sendo consideravelmente depredada. Desta forma, o desafio de se desenvolver um esquema de CAPTCHA interessante deve sempre respeitar o *trade-off*[4] entre robustez e usabilidade. Para o autor, a resolução de um CAPTCHA – texto, imagem e/ou som – é uma tarefa, principalmente, de processamento cognitivo humano, reiterando que caso a capacidade de reconhecimento do CAPTCHA seja comprometida, no intuito de fazê-lo mais forte, torna-o inutilizável.

Figura 2: Exemplo de CAPTCHA Textual, Sonoro, *No CAPTCHA* e de Imagem



(a) Textual

(b) Sonoro

(c) *No CAPTCHA*

(d) Imagem

Fonte: (a) Lattes (2019) e (b,c,d) Google Imagens (2019).

2.1.1 CAPTCHAs Textuais

Para Ye et al. (2018), uma das razões para o uso difundido de CAPTCHAs de texto está no fato de que a maioria de seus ataques são bem específicos, devido ao seu esquema único, e requerem um processo demorado e trabalhoso de construção. Isso sugere que uma pequena modificação – um fundo mais ruidoso ou caracteres diferentes – no esquema do CAPTCHA pode comprometer por completo os métodos de ataque anteriores. Ainda de acordo com o autor, toda a eficácia da proposta de um ataque genérico a CAPTCHAs reside em sua competência de segmentar corretamente os caracteres do texto. Já Chen et al. (2017) sugere que caracteres com distorção, adesão e sobreposição dificultam a capacidade de segmentação dos ataques. Liu Rong Zhang (2017) explica que o processo de segmentação faz-se mais complexo e difícil que a etapa de reconhecimento, pelo fato de CAPTCHAs textuais serem

[4]Em português, "Relação de Troca". (Tradução Livre).

desenvolvidos para serem resistentes à segmentação.

Uma das modificações que mais comprometem a legibilidade – reconhecimento – do CAPTCHA textual, para Yan e Ahmad (2008), é a distorção, em principal, a deformação e redimensionamento, pois elas fazem com que o caractere perca sua forma conhecida, como pode ser visto na Figura 3. Existem outras possíveis alterações, como a utilização de outros modelos de fontes, a variação no tamanho do caractere, rotações e translações nas letras/-números, dentre outras. Essas mudanças não afetam tanto a percepção dos humanos sobre os caracteres, fazendo com que a legibilidade dos mesmos não seja comprometida. Para o autor, em CAPTCHAs textuais, a principal preocupação a se ter sobre eles deve ser a sua legibilidade. Para Ye et al. (2018), caracteres deformados e sobrepostos em conjunto com fundos complexos tornam o CAPTCHA textual mais robusto.

Figura 3: Exemplos de caracteres distorcidos que foram descaracterizados

(a) A distorcido parece 4 (b) T distorcido parece 7 (c) Z distorcido parece N

Fonte: Elaborada pelo autor.

De acordo com Yan e Ahmad (2008), translação é o movimento dos caracteres para cima ou para baixo, e para a direita ou a esquerda, por uma determinada quantidade de espaço. A rotação consiste em girar os caracteres no sentido horário ou anti-horário por um determinado ângulo. O redimensionamento é esticar ou comprimir o caractere na direção x e/ou y. Por fim, a deformação é o redimensionamento do caractere em diferentes proporções e direções. Desta forma, ao tentar distorcer o caractere preservando a capacidade de legibilidade do mesmo, deve-se definir um máximo e mínimo de distorção possível, de forma a não torná-lo irreconhecível. A pior das distorções é a deformação, pois ela rapidamente faz com que o caractere se torne muito diferente do conhecido. Exemplos das distorções supracitadas podem ser vistos na Figura 4.

Figura 4: Exemplos de rotação, redimensionamento e deformação

(a) Rotação (b) Redimensionamento (c) Deformação

Fonte: Elaborada pelo autor.

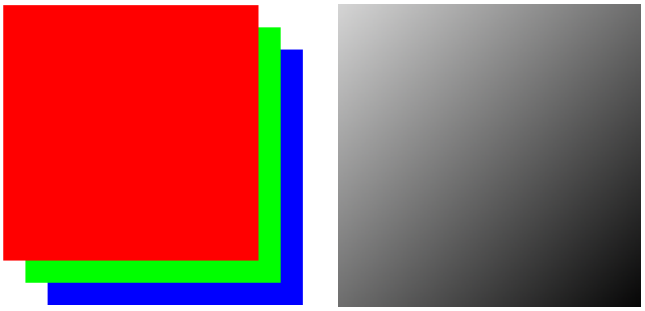
2.2 Técnicas de Processamento Digital de Imagens

Para Gonzalez, Woods et al. (2002), uma imagem pode ser definida como uma função bidimensional, $f(x, y)$, onde x e y são coordenadas espaciais e a intensidade, nível de cinza, da imagem no par (x, y) é a amplitude de f nesse ponto. Para o autor, quando uma imagem apresenta uma quantidade finita de valores para x, y e f, então a imagem é definida como imagem digital. Desta forma, o autor define o PDI como o processamento de imagens digitais em um computador digital. Uma das bibliotecas mais conhecidas no processamento de imagem e campo da visão computacional, que fornece um arcabouço de ferramentas para processar as imagens de forma digital, é a *Open Source Computer Vision Library*[5] (OpenCV).

Uma imagem é constituída de pixels e canais. Nesse sentido, Gonzalez, Woods et al. (2002) explica que pixel é o termo mais utilizado para denotar os elementos constituintes de uma imagem digital. O pixel, originado da aglutinação de ***Pic**ture **El**ement*[6], é o elemento mais básico de uma imagem digital e é composto por um par de coordenadas (x, y) e uma intensidade, que por convenção define-se de 0 a 255.

Uma imagem também é formada por canais, que dependendo da representação, pode descrever a intensidade de cada cor – RGB – em um determinado pixel, ou a matiz, saturação e luminância. Uma imagem em escala de cinza apresenta apenas um canal e imagens coloridas (RGB) apresentam três, uma para as core vermelho, verde e azul. Do ponto de vista matemático, interpreta-se uma imagem como uma matriz multidimensional de dimensão $[l, h, n]$, onde l é a largura, h é a altura e n o número de canais da imagem. Na Figura 5 é possível ver uma representação de imagens coloridas e em escala de cinza, respectivamente.

Figura 5: Canais de uma imagem RGB e em escala de cinza



(a) RGB (b) Escala de Cinza

Fonte: Elaborada pelo autor.

Na tentativa de construir um método de solucionar o CAPTCHA do Lattes de forma simples – o qual terá uma explicação mais detalhada no Capítulo 3 –, fez-se necessária a utilização de algumas técnicas de PDI, as quais são normalização, limiarização, morfologia e *Template Matching*[7].

[5] Em português, "Biblioteca *open source* de Visão Computacional". (Tradução Livre).
[6] Em português, "Elemento da Figura". (Tradução Livre).
[7] Em português, "Correspondência de modelos". (Tradução Livre).

2.2.1 Normalização

Para Abu-Mostafa e Psaltis (1985), normalização é o processo de transformar a função da imagem de $f(x, y)$ para $g(x, y)$, de modo que ela retenha toda a informação relevante da imagem original. Geralmente utiliza-se da normalização quando se quer transformar as intensidades dos pixels das imagens de 0 a 255 para 0 a 1. Isso se faz interessante, pois o número 0 é o elemento nulo e o número 1 é o elemento neutro da multiplicação. Portanto, a multiplicação, pixel a pixel, de uma máscara binária de 0 a 1 e uma imagem de interesse mantém apenas a parte que é branca na máscara, desprezando o valor da intensidade dos outros pixels. Todo o processo detalhado anteriormente pode ser visualizado na Figura 6.

Figura 6: Exemplo de utilização de uma máscara binária

(a) Máscara Binária (b) Imagem (c) Resultado da Multiplicação.

Fonte: Elaborada pelo autor.

Existe um tipo de normalização chamada de normalização Min Max, a qual recebe como parâmetros o menor valor – definido como o mínimo – e o maior valor – definido como o máximo – para qual os pixels da imagem serão mapeados. Entretanto, se a imagem tem como a intensidade do menor pixel 40 e a do maior 220, e é realizada a normalização Min Max de 0 a 255, a imagem final apresentará a menor e a maior intensidade dos seus pixels como 0 e 255 e as cores da imagem vão sofrer uma distorção. O exemplo dado só contemplou imagens com um canal. Caso essa imagem tivesse três canais – colorida – essa distorção poderia ser bem mais agravante. A distorção descrita anteriormente pode ser vista na Figura 7.

Figura 7: Exemplo da distorção provocada pela normalização Min Max

(a) Escala de Cinza (b) Normalizada

Fonte: Elaborada pelo autor.

2.2.2 Limiarização

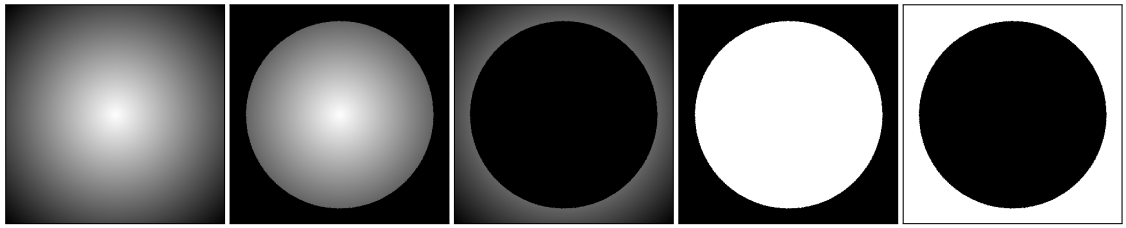
Para Haralick e Shapiro (2001), a limiarização escolhe alguns dos pixels da imagem e os definem como *foreground*[8] – que compõem os objetos de interesse – e o resto como *background*[9]. No caso mais simples de limiarização, define-se um valor para o limiar e escolhe se é uma limiarização acima ou abaixo. Se for realizada uma limiarização acima, todos os pixels que tiverem suas intensidades maiores ou iguais ao limiar, podem se manter intactos ou tornam-se brancos, portanto, são definidos como *foreground* e o resto vira *background*, como pode ser observado na Equação 1, onde $p(i,l)$ é a função que retornará a nova intensidade do pixel, i é a intensidade atual do pixel e l é valor limiar. A limiarização abaixo realiza o contrário da acima, logo, para todos os pixels que tiverem suas intensidades menores que o limiar tornam-se *foreground* e o resto *background*, como descrito na Equação 2.

Uma variação da limiarização é a limiarização binária ou binarização. Caso seja uma binarização acima, todos os pixels que compuserem o *foreground* terão suas intensidades definidas para um valor máximo, geralmente 255 ou 1, como mostrado na Equação 3. A binarização abaixo se comporta de forma similar a limirização abaixo e está descrita na Equação 4. A binarização se faz útil quando é necessário gerar uma máscara binária de um fragmento da imagem. É importante salientar que a limiarização é uma operação que atua individualmente em cada canal da imagem. Caso a limiarização apresente mais de um limiar – um superior e um inferior – definem-se dois possíveis tipos de limiarização: interior e exterior. Dessa forma, em uma limiarização interior, todos os pixels que estiverem entre os limiares inferior e superior tornam-se *foreground* e o resto *background*. De forma similar funciona a limiarização exterior, onde os pixels que estiverem contidos entre os limiares inferior e superior serão definidos como *background* e o resto *foreground*. Exemplos de limiarização podem ser vistos na Figura 8.

$$p(i,l) = \begin{cases} 0, & se\ i < l \\ i, & se\ i \geq l \end{cases} \quad (1) \qquad p(i,l) = \begin{cases} i, & se\ i < l \\ 0, & se\ i \geq l \end{cases} \quad (2)$$

$$p(i,l) = \begin{cases} 0, & se\ i < l \\ 1, & se\ i \geq l \end{cases} \quad (3) \qquad p(i,l) = \begin{cases} 1, & se\ i < l \\ 0, & se\ i \geq l \end{cases} \quad (4)$$

Figura 8: Exemplos de limiarizações



(a) Original (b) Limiarização Acima (c) Limiarização Abaixo (d) Binarização Acima (e) Binarização Abaixo

Fonte: Elaborada pelo autor.

[8]Em português, "Primeiro Plano". (Tradução Livre).
[9]Em português, "Plano de Fundo". (Tradução Livre).

2.2.3 Morfologia

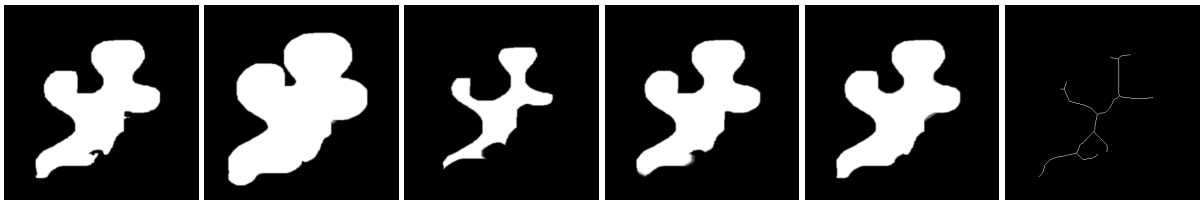
Para Pratt (2007), o processamento morfológico de imagens é um tipo de processamento no qual a forma espacial ou estrutura de objetos dentro de uma imagem é modificada. Dentre as operações morfólogicas, existem cinco que são muito utilizadas, são elas: dilatação, erosão, abertura, fechamento e esqueletonização. Em resumo, com a dilatação um objeto cresce uniformemente em extensão espacial, enquanto que na erosão o objeto encolhe uniformemente. A dilatação é representada com o operador matemático $\oplus$ e a erosão com o $\ominus$. As Equações 5 e 6 apresentam as funções que descrevem a dilatação e erosão, respectivamente, onde A é a imagem, a qual será aplicada a morfologia, B é o elemento estruturante (filtro) que será utilizado pela operação morfológica. A_b descreve a translação de A por b.

Segundo Pratt (2007), a abertura e o fechamento de uma imagem tende a suavizar os contornos dos objetos, eliminar pequenos furos, preencher lacunas pequenas entre os objetos ou quebrar objetos estritos. A abertura, diferentemente da erosão, consegue erodir pequenos objetos sem deteriorar o corpo principal do objeto de interesse, sendo de suma importância também na remoção de ruídos. Já o fechamento, consegue diminuir pequenas lacunas no *foreground*, com a dilatação total do objeto de interesse. A abertura consiste em uma erosão seguida de uma dilatação e o fechamento é uma dilatação seguida de uma erosão, como pode ser visto nas Equações 7 e 8, respectivamente. Por fim, a esqueletonização resulta em uma representação de um esqueleto do objeto. A Equação 9 apresenta a função que descreve a esqueletonização, onde N é o último passo possível antes da imagem A ser erodida ao conjunto vazio. Exemplos das operações morfológicas supracitadas podem ser vistos na Figura 9.

$$A \oplus B = \bigcup_{b \in B} A_b \quad (5) \qquad A \ominus B = \bigcap_{b \in B} A_{-b} \quad (6) \qquad A \circ B = (A \ominus B) \oplus B \quad (7) \qquad A \bullet B = (A \oplus B) \ominus B \quad (8)$$

$$S(A) = \bigcup_{n=0}^{N} [(A \ominus nB) - (A \ominus nB) \circ B], \text{ onde } N = max\{k | (A \ominus nB) \neq \emptyset)\} \quad (9)$$

Figura 9: Exemplo de dilatação, erosão e esqueletonização



(a) Original (b) Dilatação (c) Erosão (d) Abertura (e) Fechamento (f) Esqueletonização

Fonte: Elaborada pelo autor.

2.2.4 *Template Matching*

Para Pratt (2007), o *Template Matching* (TM) é uma das formas mais fundamentais de detecção de objetos dentro de uma imagem, a qual um modelo – uma réplica – do objeto de interesse é comparada a todos os objetos, os quais são desconhecidos ou não, da imagem. Caso algum dele forneça uma correspondência ao modelo – modelo e objeto apresentam evidentes similaridades – este objeto desconhecido será rotulado a partir do *template* que foi atribuído a ele. Ao realizar a comparação – pixel a pixel – deve-se utilizar algum método capaz de avaliar o quão próximo o pixel do modelo e o seu correspondente na imagem estão um do outro, tentando quantificar a similaridade entre eles. Dentre os possíveis métodos a ser utilizados, segundo Gonzalez, Woods et al. (2002), os mais conhecidos são *Sum of Squares Differences* (SSD), a SSD normalizada, *Cross Correlation* (CCR), CCR normalizada, *Cross Correlation Coefficient* (CCOE) e CCOE normalizado. Os métodos, anteriormente mencionados, são descritos nas Equações 10 a 15, respectivamente.

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \quad (10)$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \quad (11)$$

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))^2 \quad (12)$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \quad (13)$$

$$R(x,y) = _{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))^2 \quad (14)$$

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \quad (15)$$

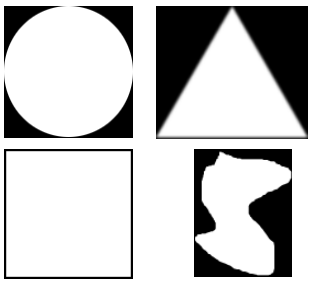
Nas Equações 10 a 15, T é a matriz correspondente ao *template* e I a imagem a qual será aplicada o *Template Matching*. Tendo em vista que o *template* deve ter seu tamanho menor ou igual que o da imagem, define-se o número possível de iterações no eixo das abscissas e no das ordenadas de forma a contemplar todas as possíveis posições que o *template* pode assumir na imagem. Obtém-se o número possível de iterações ao subtrair o tamanho da imagem ao tamanho do *template*, caso eles tenham o mesmo tamanho, só haverá uma iteração. As variáveis x e y correspondem ao ponto na imagem que define onde será iniciada a comparação do *template* com o pedaço correspondente da imagem. Esse ponto, por convenção, localiza-se na parte superior esquerda do *template* e da mesma forma no trecho correspondente daquela iteração na imagem. Esse ponto irá guardar, no fim, o valor de similaridade do *template* com o fragmento da imagem. Para fazer a comparação pixel a pixel do *template* com a fração da imagem em questão, manipula-se o ponto constituído das variáveis x' e y', que irão iterar sobre a superfície do *template* e da imagem ao mesmo tempo e, além disso, aplicarão os métodos supracitados para atribuir um valor de semelhança entre os dois.

As variáveis x e y correspondem ao ponto de aplicação do *template*, onde será calculada a equivalência naquela iteração, enquanto que o x' e o y' descrevem o ponto que está iterando na intersecção entre a imagem e o *template*, naquele instante de tempo. O x' e o y' representam uma abstração da comparação elemento a elemento da equação. As matrizes T' e I', propostas no método CCOE, são descritas nas Equações 16 e 17. O T' é o T subtraído do valor médio dos pixels do *template* e o I' é o I subtraído do valor médio dos pixels da imagem na parcela correspondente ao *template*, naquela iteração. Um exemplo de utilização do TM pode ser visto nas Figuras 10 e 11, onde na primeira é ilustrado os *templates* e a imagem que serão usados e, na segunda, a aplicação de cada *template*, utilizando do método CCOE, na imagem definida anteriormente. As regiões mais avermelhadas implicam em uma maior chance do *template* estar posicionado naquele trecho da imagem, em contrapartida as azuis denotam o contrário. É importante salientar que, no TM, tanto o *foreground* quanto o *background* são levados em consideração.

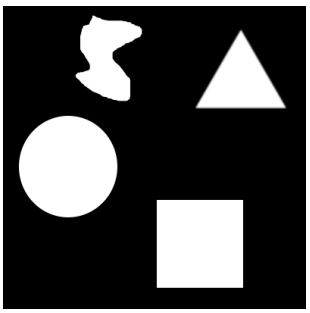
$$T'(x',y') = T(x',y') - \frac{\sum_{x'',y''} T(x'',y'')}{w \cdot h} \quad (16)$$

$$I'(x+x',y+y') = I(x+x',y+y') - \frac{\sum_{x'',y''} I(x+x'',y+y'')}{w \cdot h} \quad (17)$$

Figura 10: Modelos de *Templates* e Imagem para aplicação do TM



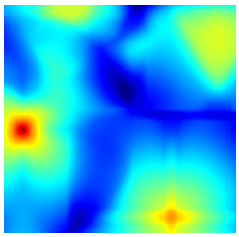
(a) Modelos de *Templates*



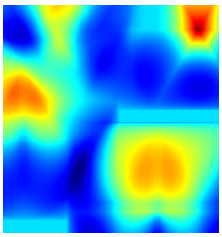
(b) Imagem

Fonte: Elaborada pelo autor.

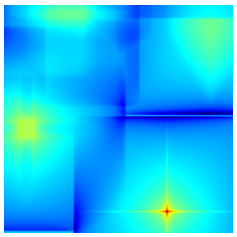
Figura 11: Aplicação do CCOE com os *Templates* da Figura 10a na imagem da Figura 10b.



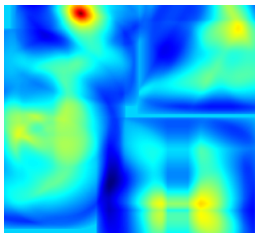
(a) Círculo



(b) Triângulo



(c) Quadrado



(d) Modelo Arbitrário.

Fonte: Elaborada pelo autor.

2.3 *Machine Learning e Deep Learning*

Para Mitchell (1997), um programa de computador aprende com a experiência (E) com relação a um conjunto de tarefas (T) e com um desempenho (P), de forma a melhorar P de T com E, ou seja, a experiência do computador ao executar um conjunto de tarefas deve aumentar o seu desempenho. Segundo Goodfellow, Bengio e Courville (2016), *Machine Learning* (ML) é a capacidade de um sistema artificialmente inteligente adquirir seus próprios conhecimentos a partir da extração de padrões de dados brutos. De acordo com Chollet (2017), *Deep Learning* (DL) é um subcampo de *Machine Learning* que desmembra a representação do aprendizado em múltiplas camadas – utilizando geralmente redes neurais–, de forma que as camadas iniciais apresentam o conhecimento descrito em estruturas simples e, as mais profundas, em estruturas mais complexas.

ML e DL tiveram um crescimento exponencial nos últimos tempos devido à evolução do poder computacional, com a popularização das GPUs, e ao grande volume de dados dispostos na internet. Os dois critérios mencionados anteriormente – base de dados e poder computacional – são fundamentais para o desenvolvimento de soluções que utilizam ML e DL. Ao longo dos anos, a internet vem evoluindo e, com isso, sua forma de utilização vem sendo transformada ao passo dessa evolução. No início, a internet funcionava apenas como uma plataforma de consulta e comunicação, entretanto, nos dias atuais, a mesma fornece diversos serviços, como comércio, redes sociais, plataformas de ensino, serviços multimídia, entre outros.

Essa nova forma de utilização da internet fez com que os usuários depositassem os mais diversos tipos de dados, de todos os formatos, sobre suas vidas, rotinas e interesses, enriquecendo os dados já existentes e ensejando à utilização dos mesmos em busca de padrões que os explicassem. Desta forma, empresas e indústrias, que já apresentavam algumas rotinas automatizadas, poderiam propor produtos customizados aos seus clientes e automatizar esse processo de personalização. Em outras palavras, empresas e indústrias utilizariam de soluções com ML, construídas a partir dos dados de seus próprios clientes, porém, que estão dispostos na internet. Este exemplo de utilização de ML, mais especificamente de DL, mostra o quanto sistemas inteligentes são capazes de modificar o mundo. As CNNs e R-CNNs revolucionaram a visão computacional, permitindo que projetos como carros autônomos, drones de reconhecimento, validação biométrica e diversos outros, pudessem ser realizados.

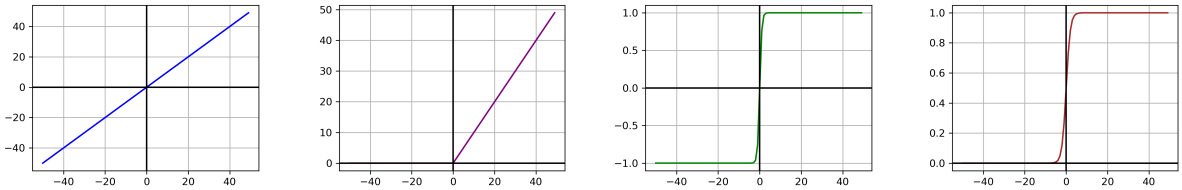
2.3.1 *Convolutional Neural Networks* (CNNs)

As *Convolutional Neural Networks* (CNN) ou *Convolutional Networks* são um tipo especializado de rede neural utilizado para processamento de dados, como imagens. Para Goodfellow, Bengio e Courville (2016), as redes convolucionais são redes neurais que usam a convolução no lugar da multiplicação geral da matriz em pelo menos uma de suas camadas. De acordo com Chollet (2017), uma das primeiras utilizações práticas das CNNs foi em 1989, no Bell Labs, onde Yann LeCun combinou a ideia de CNN com *backpropagation*[10] para classificar dígitos escritos à mão. Segundo Goodfellow, Bengio e Courville (2016), ao processar uma imagem, a imagem de entrada pode ter milhares ou milhões de pixels, mas pode-se detectar objetos pequenos e significativos ao aplicar a convolução na imagem de entrada com *kernels* que ocupam apenas dezenas ou centenas de pixels. Em relação à imagem de saída, a mesma será uma versão filtrada da entrada, a qual terá as suas características mais determinantes explicitadas. As CNNs são amplamente utilizadas para classificação de imagens.

Uma CNN apresenta alguns tipos de camadas que realizam operações diferentes na imagem de entrada. Tipicamente, as camadas ocultas de uma CNN consistem em camadas convolucionais, camada de ativação, camadas de *pooling*[11], camadas densas ou totalmente conectadas, camadas de normalização e camada Softmax. As camadas convolucionais realizam uma convolução entre a imagem de entrada e um *kernel* ou filtro, no intuito de realçar as *features*[12] de interesse. O filtro utilizado na operação de convolução é definido anteriormente e para cada camada convolucional pode ter tamanhos e valores distintos. As camadas de ativação submetem a imagem de entrada – pixel a pixel – a uma função, conhecida como função de ativação. As mais conhecidas funções de ativação são a *Rectified Linear Unit*[13] (ReLU), Tangente Hiperbólica (Tanh) e Sigmoide. A ReLU define todos os valores da distribuição de entrada abaixo de 0 como iguais a 0. A Tanh normaliza os valores da entrada de -1 a 1. E a Sigmoide, de uma forma bem semelhante a Tanh, normaliza os valores da entrada de 0 a 1. O comportamento das funções de ativação podem ser vistos na Figura 12.

[10] Em português, "Retropropagação". (Tradução Livre).
[11] Em português, "Agrupamento". (Tradução Livre).
[12] Em português, "Características". (Tradução Livre).
[13] Em português, "Unidade Linear Retificada". (Tradução Livre).

Figura 12: Aplicações de funções de ativação em uma distribuição arbitrária

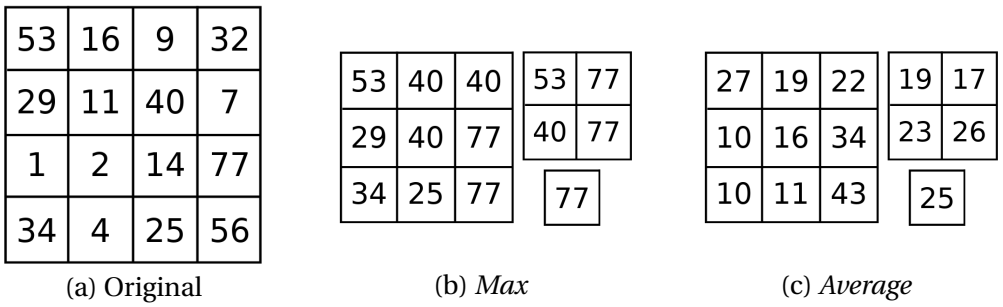


(a) Distribuição (b) ReLU (c) Tanh (d) Sigmoide.

Fonte: Elaborada pelo autor.

As camadas de *pooling*, de forma semelhante as camadas de convolução, transladam um *kernel* por cima da imagem. Ao passo que o *kernel* vai avançando sobre a entrada, ele realiza uma combinação entre os pixels em questão, de modo a retornar um valor sobre eles. O *max pooling*[14] avalia qual é o pixel de maior valor do conjunto em observação e retorna o seu valor. O *average pooling*[15] retorna a média dos pixels atuais. Além do *max pooling* e do *average pooling* existem o *global max pooling*[16] e o *global average pooling*[17], que se comportam de forma semelhante às suas versões locais, tendo como diferença os métodos globais que realizam a operação de *pooling* sobre toda a imagem de entrada e não mais sobre um *kernel*, de forma que a imagem inteira é sumarizada a 1 número. Se a imagem tiver mais de uma camada, as operações de *pooling* serão realizadas para cada camada individualmente. As operações de *pooling* são utilizadas para reduzir a dimensão da entrada e também para aguçar as *features* mais relevantes. Exemplos de *max* e *average pooling* podem ser vistos nas Figuras 13b e 13c utilizando *kernels* de dimensões 2x2, 3x3 e Global.

Figura 13: *Max* e *Average Pooling* com *kernels* de 2x2, 3x3 e Global em uma matriz arbitrária



(a) Original (b) *Max* (c) *Average*

Fonte: Elaborada pelo autor.

[14]Em português, "Agrupamento Máximo", (Tradução Livre).
[15]Em português, "Agrupamento Médio", (Tradução Livre).
[16]Em português, "Agrupamento Máximo Global", (Tradução Livre).
[17]Em português, "Agrupamento Médio Global", (Tradução Livre).

As camadas densas ou totalmente conectadas realizam a conexão da camada anterior com a próxima, na forma de um grafo completo, onde todos os neurônios de uma camada se conectam com todos os neurônios da outra. As camadas de normalização mapeiam a saída da camada de ativação, de forma que ela apresente sua média próxima a 0 e o desvio padrão próximo a 1. Essa camada se torna fundamental ao utilizar a função de ativação ReLU, pois a mesma satura os elementos menores que 0 para 0, porém não apresenta nenhum limite superior, tendendo a infinito. A camada Softmax transforma os valores brutos – reais – de sua entrada em valores percentuais. A transformação realizada pela Softmax é descrita pela Equação 18, onde x_j é o valor de uma das entradas da Softmax.

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}} \tag{18}$$

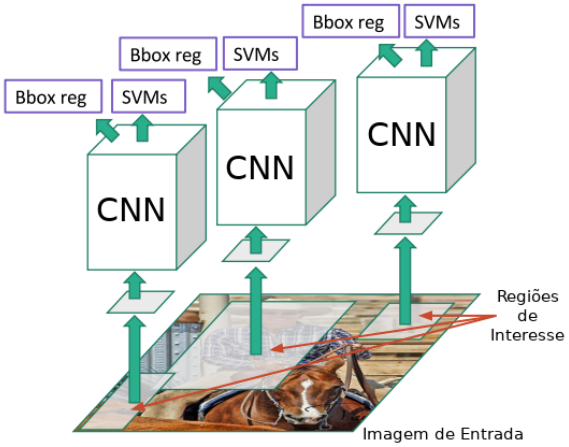
2.3.2 *Region-based Convolutional Neural Networks* (R-CNNs)

Segundo Girshick et al. (2014), a R-CNN consiste na união de um algoritmo capaz de propor regiões de interesse da imagem, com uma CNN – que irá extrair as *features* de tais regiões – e do algoritmo *Support Vector Machine*[18] (SVM), que irá classificá-las. Desta forma, é comum definir que uma R-CNN é capaz de detectar objetos em uma imagem, ou seja, não somente definir uma classe para a imagem como um todo, mas atribuir classes a possíveis objetos que estão inclusos na imagem. De acordo com Li, Johnson e Yeung (2017), o componente responsável por detectar as regiões de interesse consegue descobrir mais de 1000 regiões em alguns segundos. Em geral, além das classes a serem detectadas por uma R-CNN, também é inserida uma classe nula, a qual representará o fundo da imagem.

A arquitetura de uma R-CNN pode ser vista na Figura 15. O processo de detecção de uma R-CNN se dá, primeiramente, pela coleta das regiões de interesse, as quais são redimensionadas para que possam ser inseridas na CNN. A CNN irá fazer a extração das *features* de interesse da imagem, buscando filtrar o que não é interessante, além de salientar as características mais relevantes. A saída da CNN é diretamente ligada à entrada do SVM, que irá analisar as *features* extraídas da imagem de entrada e classificá-la em uma de suas classes, podendo ser a nula. Também sai da CNN o *Bounding Boxes*[19] (BB) que irá englobar o objeto detectado na imagem. De acordo com Li, Johnson e Yeung (2017), a R-CNN apresenta um treinamento demorado e apresenta um longo tempo de inferência. A R-CNN possui duas versões melhoradas, a *Fast* R-CNN e a *Faster* R-CNN. Ambas foram propostas após a R-CNN e agregam melhorias a ela. A *Faster* R-CNN é uma melhoria da *Fast* R-CNN.

[18] Em português, "Máquinas de Vetores de Suporte". (Tradução Livre).
[19] Em português, "Caixas Delimitadoras". (Tradução Livre).

Figura 14: Arquitetura de uma R-CNN

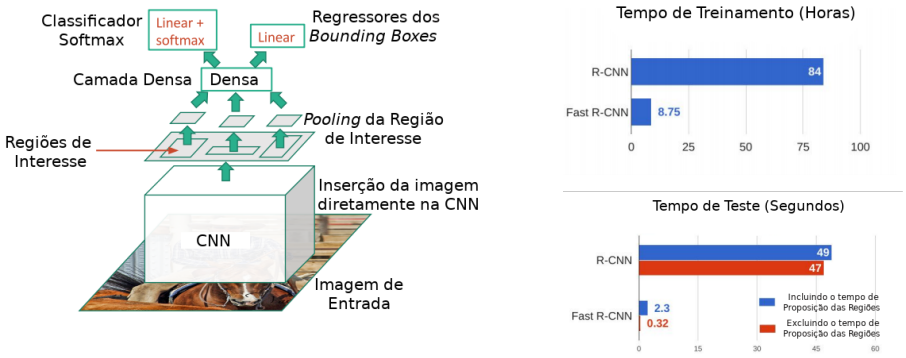


Fonte: Adaptada de Li, Johnson e Yeung (2017, p. 68).

De acordo com Girshick (2015), as mudanças propostas na *Fast* R-CNN consistem na substituição do classificador SVM por uma camada densa seguida de uma camada Softmax, as quais realizarão a classificação das imagens, uma vez que a segmentação das regiões de interesse agora são feitas após a extração das *features* pela CNN. Entre o segmentador de regiões e a camada densa há uma camada de *pooling* que irá redimensionar a região proposta. Os *Bounding Boxes* são calculados por um regressor linear. A arquitetura da *Fast* R-CNN pode ser vista na Figura 15a.

Os autores Li, Johnson e Yeung (2017) realizaram uma comparação do tempo de treinamento, com o *Dataset* ImageNet, e obtiveram que a *Fast* R-CNN levou 8,75 horas para treinar, enquanto que a R-CNN precisou de 84 horas para obter o mesmo resultado. Além do tempo de treinamento, realizou-se uma comparação no tempo de teste das duas abordagens e a *Fast* R-CNN realizou a inferência em uma imagem em dois/três segundos – sendo dois segundos o tempo necessário para a infraestrutura de proposição de regiões fornecer as regiões de interesse – e a R-CNN levou 49 segundos, demandando dos mesmos dois segundos de proposição de regiões. Esses resultados podem ser vistos na Figura 15b.

Figura 15: Arquitetura de uma *Fast* R-CNN.



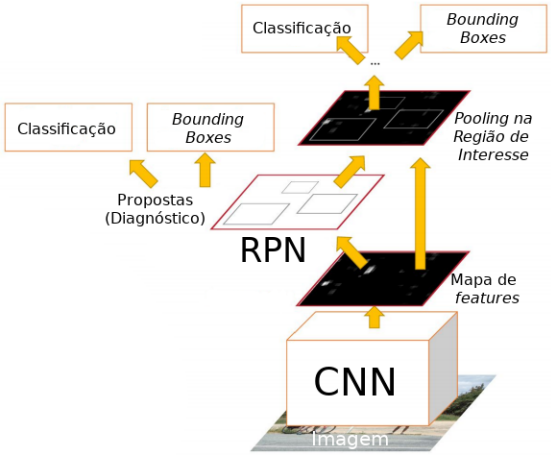
(a) Arquitetura da *Fast* R-CNN

(b) Comparação da R-CNN com a *Fast* R-CNN

Fonte: Adaptada de Li, Johnson e Yeung (2017, p. 75,79).

Ren et al. (2015) propõem uma arquitetura melhorada, a partir da *Fast* R-CNN chamada de *Faster* R-CNN. Essa nova arquitetura traz algumas modificações. Uma das modificações mais relevantes é que haverá uma CNN responsável por propor as regiões de interesse, não necessitando mais de uma infraestrutura externa que realizasse isso. Essa CNN é chamada de *Region Proposal Network*[20] (RPN). A RPN fará uma pré-análise de algumas *Anchors*[21] – Possíveis Regiões de Interesse – e entregará um diagnóstico para a CNN classificadora, que irá decidir se as regiões propostas tratam-se de objetos ou *backgrounds*. A RPN também entregará um conjunto de possíveis *Bounding Boxes* para o regressor de BB, que irá refiná-los. Vale salientar que a RPN e a CNN de classificação são treinadas separadamente. A arquitetura da *Faster* R-CNN pode ser vista na Figura 16.

Figura 16: Arquitetura de uma *Faster* R-CNN.



Fonte: Adaptada de Li, Johnson e Yeung (2017, p. 81).

[20] Em português, "Rede de Proposta de Região". (Tradução Livre).
[21] Em português, "Âncoras". (Tradução Livre).

2.3.3 Inception

A Inception é uma topologia de rede neural proposta por Szegedy et al. (2015), onde sua principal motivação é permitir a classificação de imagens cujos conteúdos apresentam-se em tamanhos e posições distintos. Por exemplo, na Figura 17 podem ser vistas quatro fotos de gatos, que ocupam parcelas distintas da imagem, de forma que seria bastante complexo escolher qual o melhor tamanho de *kernel* para convolução. Para as Figuras 17a e 17d, que apresentam informações globais na imagem, *kernels* maiores seriam mais interessantes. Já para as Figuras 17b e 17c, cujas informações são dispostas localmente na imagem, *kernels* menores seriam preferidos.

Figura 17: Exemplos de imagens com conteúdos de diversas dimensões

(a) (b) (c) (d)

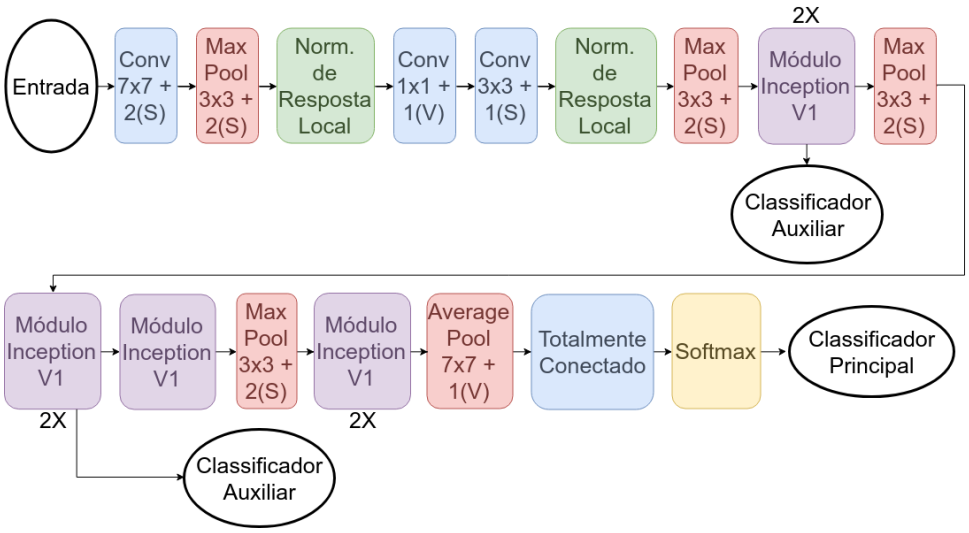
Fonte: Google Imagens (2019).

Desta forma, a Inception V1 propõe uma maneira de utilizar diversos *kernels* para atender diversos conteúdos da imagem. Ao invés de tornar a rede mais profunda – mais custosa computacionalmente –, a Inception V1 tornou-se mais larga, utilizando paralelamente os diferentes tamanhos de *kernels*, como pode ser visto na Figura 18. Além da questão do custo computacional, redes mais profundas apresentam uma tendência maior de apresentarem *overfitting*[22], devido ao fato de ser difícil propagar as atualizações do gradiente sobre toda a rede.

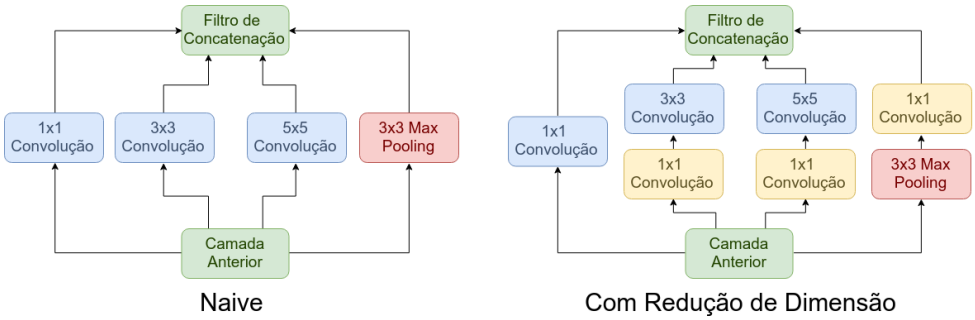
Para Szegedy et al. (2015), a solução encontrada para o problema dos diferentes tamanhos de conteúdo seria a utilização de diversos tamanhos de *kernel* em um mesmo nível, como pode ser visto na Figura 18. Deste modo, criou-se o módulo Inception, que foi proposto de duas formas: a *naive*[23] e a com redução de dimensão. A forma *naive* realiza a convolução com três dimensões diferentes de filtros – 1x1, 3x3 e 5x5 – e um *max pooling* 3x3. A saída das três convoluções e do nó *pooling* são concatenadas e inseridas na próxima camada da rede. O módulo com redução de dimensão foi proposto devido ao fato de que a forma *naive* apresenta-se computacionalmente custosa, pois não há limites sobre o número de canais de entrada.

[22] Em português, "Sobreajuste". (Tradução Livre).
[23] Em português, "Ingênua". (Tradução Livre).

Figura 18: Topologia e Módulos da Inception V1



(a) Topologia da Inception V1.



(b) Propostas de Módulos da Inception V1.

Fonte: (a) Adaptada de Szegedy et al. (2015, p. 7) e (b) Adaptada de Szegedy et al. (2015, p. 5).

Nesse sentido, os autores precederam as convoluções 3x3 e 5x5 com convoluções 1x1, as quais não alteram a largura e altura da imagem de entrada, apenas o número de canais. Isso permitiu que o módulo Inception fosse menos custoso, pois convoluções 3x3 e 5x5 com muitos canais são mais custosas que com canais limitados. O mesmo aconteceu com o *max pooling*, sendo que a redução de dimensionalidade ocorreu depois de sua execução. A Inception V1 é conhecida também como GoogLeNet.

A GoogLeNet tem 27 camadas – incluindo as camadas de *pooling* –, o que a faz ser um classificador com uma grande profundidade, levando-a a ter problemas como *vanishing gradient*[24]. Tendo em vista essa possibilidade de problema, os autores que idealizaram a Inception V1 introduziram dois classificadores auxiliares no meio da rede. Nesse contexto, possibilitaria computar suas perdas auxiliares. Desse modo, a perda total é calculada a partir de uma média ponderada entre as perdas auxiliares e a perda real, como mostrado na Equação 19. Vale ressaltar que as perdas auxiliares são utilizadas apenas no treinamento da rede. A constante evolução da Inception levou a criação de diversas versões da rede, de forma que

[24]Em português, "Dissipação do Gradiente". (Tradução Livre).

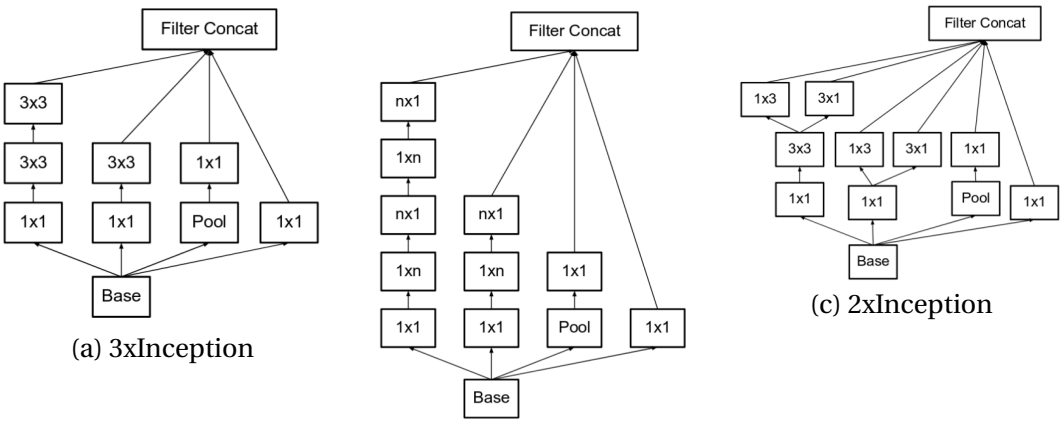
cada versão apresenta uma melhoria com relação à sua predecessora. Dentre as versões da Inception as mais populares são a V1 (GoogLeNet), V2, V3, V4, ResNet V1 e ResNet V2.

$$P_{total} = P_{real} + 0.3 * P_{auxiliar\ 1} + 0.3 * P_{auxiliar\ 2} \tag{19}$$

A Inception V2 e V3 foram propostas no mesmo artigo, onde os autores visaram, principalmente, a redução da complexidade computacional e um aumento da acurácia do modelo. Um dos problemas averiguados, pelo autores, na Inception V1, foi que a drástica diminuição da dimensão da entrada da rede causava uma excessiva perda de informações. Esse problema é chamado de *representational bottleneck*. Foi constatado que utilizar métodos de fatorização inteligentes fariam com que as convoluções se tornassem mais eficientes em termos computacionais. Assim, a melhoria proposta na Inception V2 foi a substituição das convoluções 5x5 por duas 3x3 seguidas, como pode ser visto na Figura 19a. Os autores perceberam que uma convolução 5x5 apresenta-se 2,78 vezes mais custosa que a 3x3. Desta forma, empilhar duas convoluções 3x3 aumentou o desempenho da rede.

Outra fatorização proposta por Szegedy et al. (2016), foi a substituição de convoluções nxn para 1xn e nx1, como pode ser visto na Figura 19b. Eles descobriram que a convolução 3x3 é equivalente a realizar a 1x3 seguida da 3x1. Percebeu-se que esse método mostrou-se 33% menos custoso que uma única convolução 3x3. Entretanto, esse tipo de operação torna a rede mais profunda. Desta forma, os autores propuseram uma terceira fatorização onde os bancos de filtros foram expandidos tornando-os mais largos e menos profundos – no intuito de remover a *representational bottleneck* –, como pode ser visto na Figura 19c. Devido à segunda fatorização tornar o modelo mais profundo, a entrada era submetida a mais convoluções sucessivas e, por consequência, mais informações eram perdidas. As três fatorizações propostas pelos autores deram origem à três novos módulos Inception chamados no artigo de 3xInception, 5xInception e 2xInception.

Figura 19: Fatorizações propostas para a Inception V2



(a) 3xInception

(b) 5xInception

(c) 2xInception

Fonte: Obtidas em Szegedy et al. (2016, p. 5).

Na Inception V3, foi removido um dos classificadores auxiliares, mantendo-se apenas um. Além da remoção de um deles, o que restou teve seu propósito modificado. Antes ele era utilizado para apresentar uma classificação auxiliar e burlar o *vanishing gradient*, porém, na V3, ele será utilizado como um regularizador. Além disso, a proposta era utilizar do otimizador RMSProp e da fatorização das convoluções 7x7. Por fim, percebeu-se que alguns *logits* tornavam-se muito maiores que os outros, de modo que fosse proposta uma regularização chamada de *Label Smoothing*[25], a qual buscava diminuir essa discrepância entre as classes e prevenir que a rede se tornasse muito confiante sobre uma classe – *overfitting*. Ao comparar a Inception V1 com a V2 e V3, percebeu-se que a V2 obteve um erro Top-1 de 3,8 pontos percentuais a menos que V1 e um erro Top-5 de 1,4 menor que GoogLeNet. Já a V3, obteve um erro Top-1 7,8 pontos percentuais menor que a V1 e 4 que a V2, ao analisar o Top-5 alcançou-se um erro 3,6 menor que GoogLeNet e 2,2 que a V2. Esses resultados podem ser vistos na Figura 20.

Figura 20: Comparação entre a Inception V1, V2 e V3

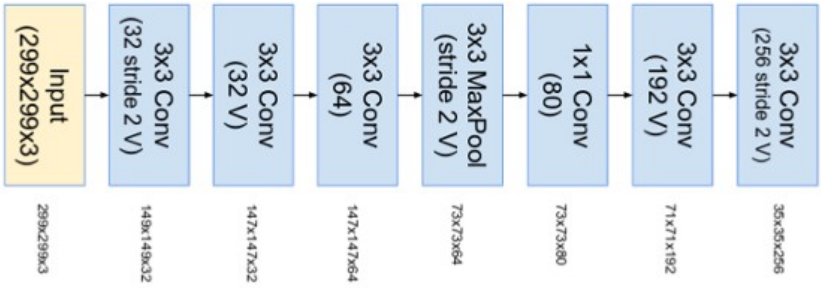
	Network	Top-1 Error	Top-5 Error	Cost Bn Ops
Inception-v1	GoogLeNet [20]	29%	9.2%	1.5
	BN-GoogLeNet	26.8%	-	**1.5**
Inception-v2	BN-Inception [7]	25.2%	7.8	2.0
Inception-v3	Inception-v3-basic	23.4%	-	3.8
Inception-v3 + RMSProp	Inception-v3-rmsprop RMSProp	23.1%	6.3	3.8
Inception-v3 + RMSProp + Label Smoothing	Inception-v3-smooth Label Smoothing	22.8%	6.1	3.8
Inception-v3 + RMSProp + Label Smoothing + 7×7 Factorization	Inception-v3-fact Factorized 7 × 7	21.6%	5.8	4.8
Inception-v3 + RMSProp + Label Smoothing + 7×7 Factorization + Auxiliary Classifier	Inception-v3 BN-auxiliary	**21.2%**	**5.6%**	4.8

Fonte: Google Imagens (2019).

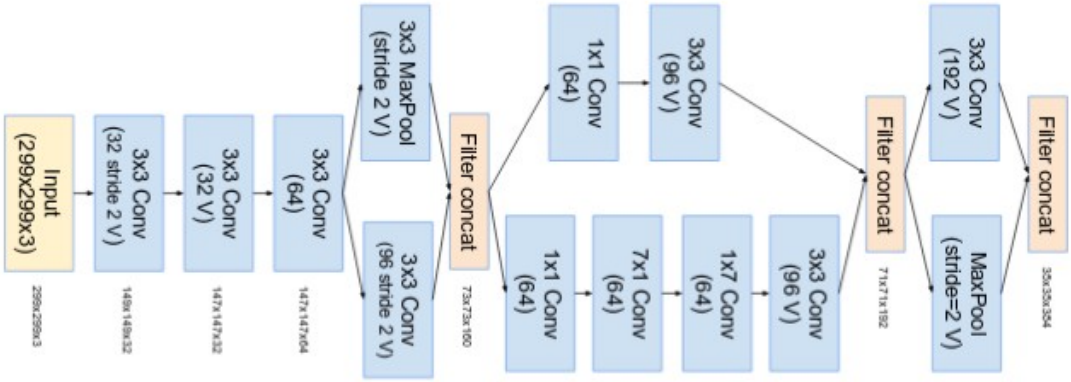
A Inception V4 e a ResNet – V1 e V2 – foram propostas em Szegedy et al. (2016). A premissa da proposição da Inception V4 pauta-se na observação dos autores de que os módulos precisavam ser mais uniformes, pois eles apresentavam-se mais complexos que o necessário. Dessa forma, buscavam também impulsionar o desempenho à medida que fossem ajustando tais módulos. Portanto, os autores propuseram a solução para tal premissa, modificando o *stem*[26] da Inception V4, ou seja, começou nos blocos introdutórios da topologia, como mostrado na Figura 21. Além da modificação do *stem*, propôs a substituição dos blocos introduzidos na V2 por três novos blocos, os quais foram nomeados como Módulo A, B e C. Os novos módulos propostos podem ser vistos na Figura 23. Outra modificação foi a criação de blocos especializados na redução dimensional – chamados de *Reduction Blocks*[27] –, os quais não existiam na rede, entretanto, sua funcionalidade já estava implementada em outros blocos. É possível visualizar os novos *Reduction Blocks* na Figura 22.

[25] Em português, "Suavização de Rótulos". (Tradução Livre).
[26] Em português, "Haste". (Tradução Livre).
[27] Em português, "Blocos de Redução". (Tradução Livre).

Figura 21: Comparação do *Stem* da ResNet V1 com o proposto na V4 e ResNet V2



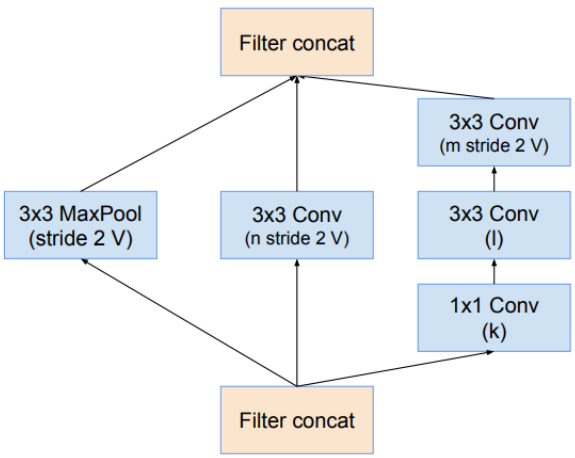
(a) *Stem* da Inception ResNet V1.



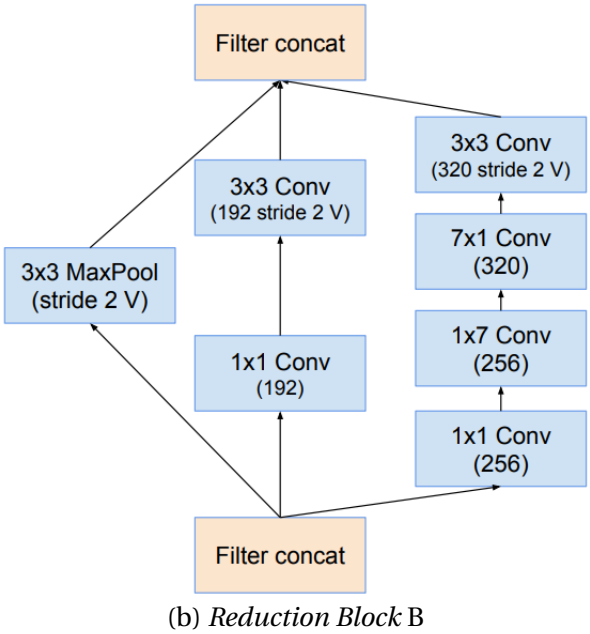
(b) *Stem* da V4 e da ResNet V2.

Fonte: Obtidas em Szegedy et al. (2016, p. 3(b), 6(a)).

Figura 22: *Reduction Blocks* propostos para a Inception V4



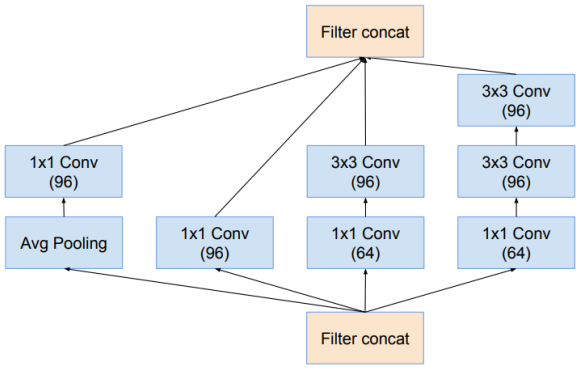
(a) *Reduction Block* A



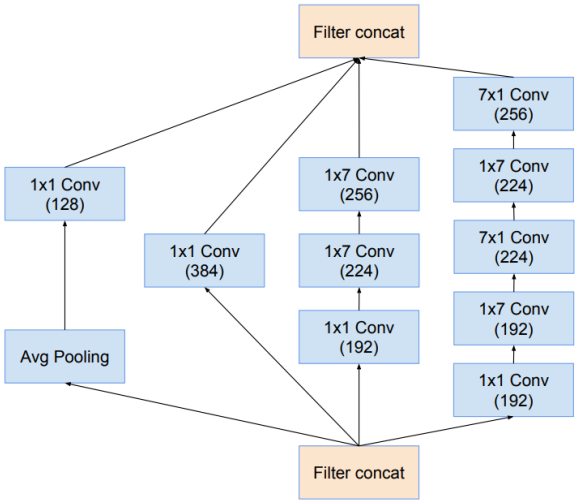
(b) *Reduction Block* B

Fonte: Obtidas em Szegedy et al. (2016, p. 4).

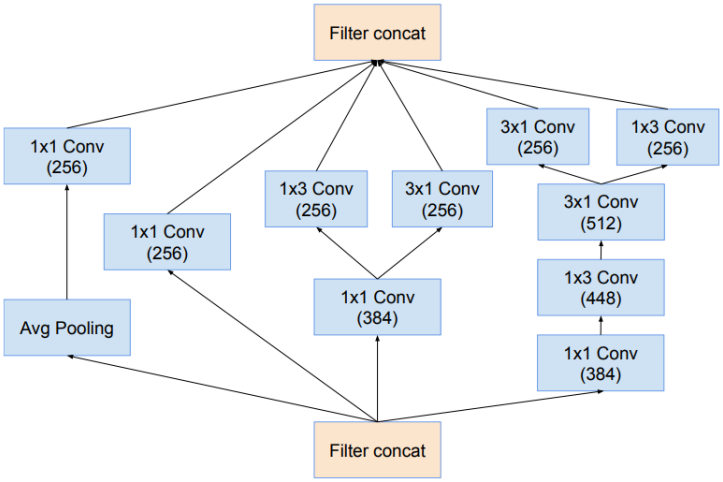
Figura 23: Módulos propostos para a Inception V4



(a) Módulo Inception A



(b) Módulo Inception B

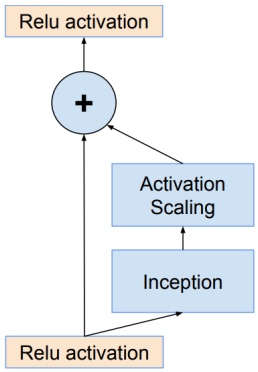


(c) Módulo Inception C

Fonte: Obtidas em Szegedy et al. (2016, p. 4).

A Inception ResNet é uma variação da Inception que teve inspiração no desempenho da ResNet, proposta por He et al. (2016). Devido a um módulo híbrido, a Inception ResNet apresenta duas variações, a V1 (RV1) e a V2 (RV2). Dentre as diferenças que a RV1 tem da RV2, pode-se afirmar que o custo computacional da RV1 assemelha-se ao da V3 e o da RV2 ao da V4. A RV1 apresenta o *stem* mostrado na Figura 21a e a RV2 o proposto na V4. Com relação aos módulos e a estrutura, a RV1 e RV2 são iguais apresentando diferenças nos hiperparâmetros. Tendo em vista as menores diferenças da topologia Inception ResNet entre suas subversões e com as anteriores, estabelece-se que a premissa da Inception ResNet é de introduzir conexões residuais, de modo que seja adicionada à saída da operação de convolução do módulo Inception em relação à sua entrada. Este novo modelo de módulo pode ser visto na Figura 24.

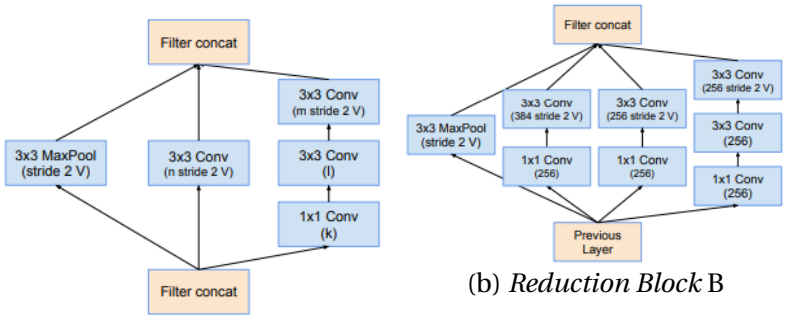
Figura 24: Modelo de módulo proposto para a Inception ResNet



Fonte: Obtidas em Szegedy et al. (2016, p. 9).

De acordo com Szegedy et al. (2016), para agregar as conexões residuais ao trabalho, a entrada e a saída após a convolução possuem as mesmas dimensões. Desta forma, utilizou-se de convoluções 1x1 após as originais, a fim de limitar a profundidade de origem, tendo em vista que a mesma aumenta após a convolução. A operação de *pooling*, dentro dos principais módulos, foi substituída para melhor estruturar as conexões residuais. No entanto, ainda é possível encontrar essas operações nos *Reduction Blocks*. O *Reduction Block* A é o mesmo que o do Inception v4. O formato dos *Reduction Blocks* na arquitetura Inception ResNet encontra-se descrito na Figura 25. As topologias finais da Inception V4 e da Resnet podem ser vistas na Figura 26. Por fim, ao analisar os resultados propostos na Figura 27, observa-se que a RV1 superou a V3 no erro Top-5 mas foi pior no erro Top-1. Já a V4 foi melhor que a RV1 em ambos os erros. Quando compara-se os resultados da RV2 com a V4, conclui-se que a RV2 obteve um desempenho levemente superior ao da V4.

Figura 25: *Reduction Blocks* propostos para a Inception V4

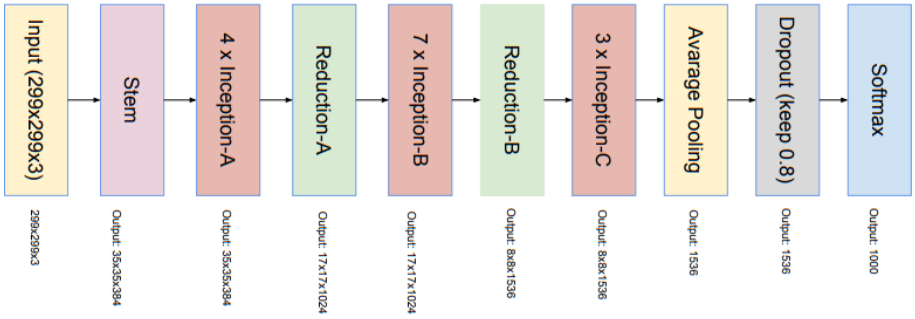


(a) *Reduction Block* A

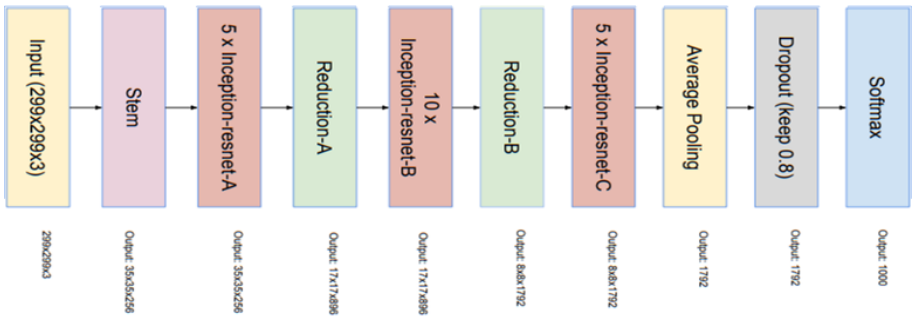
(b) *Reduction Block* B

Fonte: Obtidas em Szegedy et al. (2016, p. 4).

Figura 26: Topologia da Inception V4 e ResNet



(a) Inception V4



(b) Inception ResNet

Fonte: Obtidas em Szegedy et al. (2016, p. 4).

Figura 27: Comparação entre a Inception V3, V4, RV1 e RV2

Network	**Top-1 Error**	**Top-5 Error**
BN-Inception [6]	25.2%	7.8%
Inception-v3 [15]	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

Fonte: Obtida em Szegedy et al. (2016, p. 10).

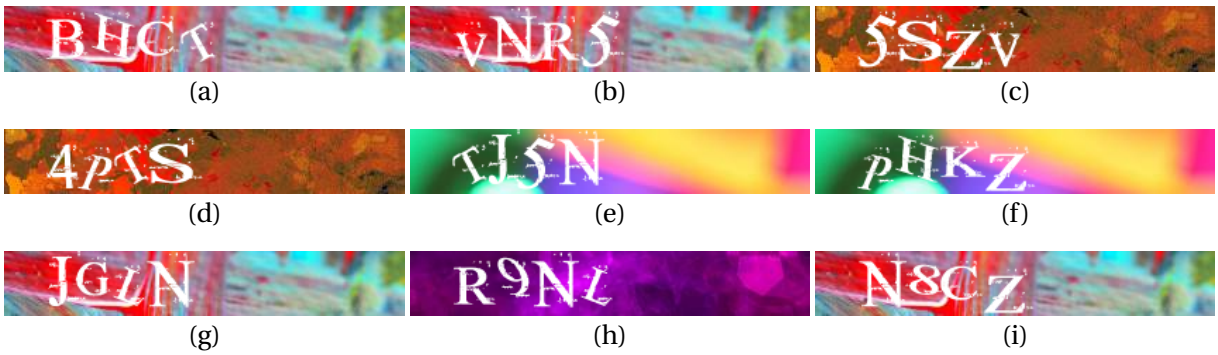
3

Metodologia

3.1 Análise do CAPTCHA do Lattes

Tendo em vista que o CAPTCHA do Lattes era um possível objeto de estudo, buscou-se avaliar as suas fragilidades e analisar a possibilidade de desenvolver um trabalho sobre elas. Primeiramente, fez-se necessário a realização do *download* de algumas amostras provenientes do próprio site do Lattes, de forma que alguns testes fossem realizados. À medida que as amostras eram baixadas e a análise sobre cada uma era realizada, elucidou-se uma das fragilidades mais óbvias desse CAPTCHA, o mesmo sempre apresenta quatro caracteres. Ao passo que a análise foi se tornando mais criteriosa, percebeu-se que o mesmo apresenta diversas outras falhas, as quais são comprometedoras para a segurança do CAPTCHA. A primeira a ser observada é que todos os CAPTCHAs gerados pelo Lattes apresentam as mesmas fontes de letras, ou seja, um N sempre será o mesmo N em todas as imagens que o contiver, como pode ser visto na Figura 28b, 28e, 28g, 28h e 28i.

Figura 28: Exemplos de CAPTCHAs do Lattes



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fonte: Lattes (2019).

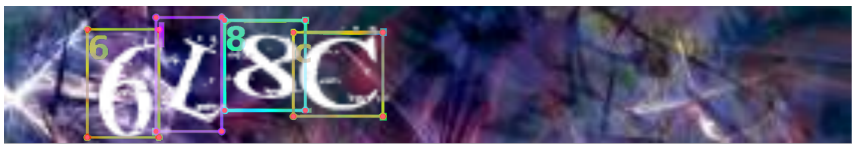
Mais crítico que apresentar a mesma fonte em todos os CAPTCHAs, é não apresentar nenhum tipo de distorção na mesma, fazendo com que o caractere seja sempre igual em todos os CAPTCHAs que o contiver. Essa falha no CAPTCHA do Lattes encejou à construção da primeira solução proposta nesse trabalho. Outra fragilidade detectada, consiste no fato dos caracteres sempre se apresentarem na cor branca e o fundo não ser gerado aleatoriamente, ou seja, repete-se em muitas imagens. O ruído inserido nas imagens é muito brando e facilmente removido com aplicação de morfologias e outras técnicas de PDI.

Foi percebido que os CAPTCHAs do Lattes também não apresentam vogais e algumas outras letras – Q e Y – além de serem todas maiúsculas. Outra observação importante consiste no CAPTCHA nunca apresentar caracteres repetidos na mesma imagem. Por fim, os caracteres sempre são alocados no mesmo espaço da imagem, ou seja, eles não transladam sobre todo o possível espaço dela. Isso também se configura uma fragilidade, pois pode-se descartar um espaço considerável da imagem pelo fato do mesmo nunca apresentar uma letra ou número contido nele. Todas as fragilidades supracitadas podem ser vistas na Figura 28.

3.2 Construção da Base de Dados

Tendo em vista que uma primeira análise foi feita sobre os CAPTCHAs do Lattes, iniciou-se a construção de uma base de dados, que será utilizada em todas as soluções propostas nesse trabalho. Para construir uma base de dados deve-se, primeiramente, obter as imagens e em seguida anotar os caracteres presentes nela. Entende-se por anotar, desenhar um quadrado em volta do caractere e escrever qual é a letra ou número dele, como pode ser visto na Figura 29. Esse processo se torna bastante moroso e cansativo, na medida em que a base aumenta. A base de dados é constituída por três conjuntos de imagens, sendo o primeiro um conjunto de treinamento com 3000 imagens, o segundo de teste com 1000 e o terceiro de validação, também com 1000. Os conjuntos de treinamento e teste foram usados para treinar as soluções robustas deste trabalho, enquanto que o conjunto de validação foi utilizado para medir a qualidade de todas as soluções. Para a primeira solução, utilizou-se apenas o conjunto de validação.

Figura 29: CAPTCHAs do Lattes anotado



Fonte: Adaptada de Lattes (2019).

3.3 Solução Simples para o CAPTCHA do Lattes

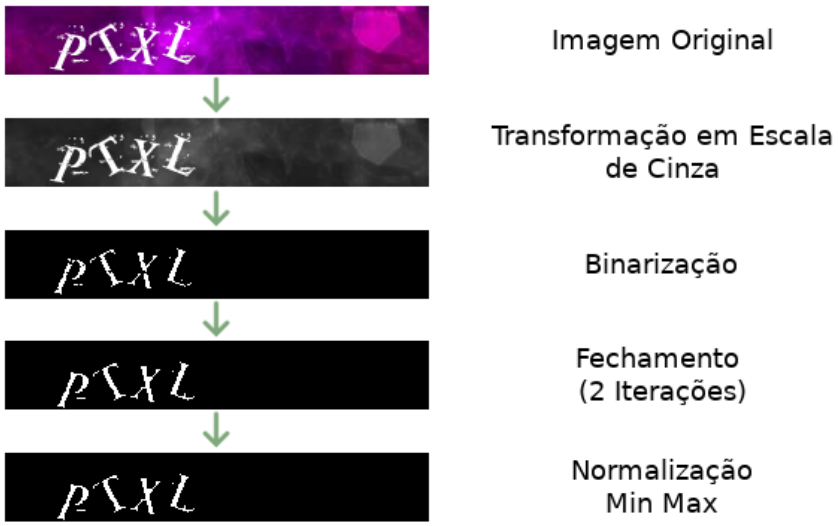
Após a criação e anotação da base de dados, partiu-se para desenvolver uma solução capaz de quebrar o CAPTCHA do Lattes de uma forma simplificada, focando nas fragilidades anteriormente constatadas. Ao observar que os caracteres não mudam, ou seja, a fonte é a mesma e eles não sofrem nenhuma distorção, a primeira solução mais interessante foi a utilização de *Template Matching*, era necessário somente coletar os devidos *templates* e aplicá-los nas imagens de interesse. Os caracteres que tiverem maiores similaridades, provavelmente, serão os certos. A primeira etapa no desenvolvimento dessa solução foi a seleção dos *templates* que seriam utilizados no TM, conforme pode ser visto na Figura 30. Os *templates* selecionados passaram por uma etapa de pré-processamento, onde eles foram binarizados e tiveram os seus ruídos removidos. A primeira solução utilizou das bibliotecas OpenCV e NumPy, ambas na linguagem Python.

Figura 30: Exemplos de *Templates* do CAPTCHA do Lattes

Fonte: Elaborada pelo autor.

A intenção de utilizar uma solução simples é tentar evidenciar a fragilidade do CAPTCHA do Lattes. Antes de aplicar a seleção de *templates* no conjunto de validação, foi necessário realizar um pré-processamento na imagem, de forma a remover o fundo e os ruídos, além de selecionar qual método de TM seria utilizado. Após a leitura da imagem pelo *software*, o seu pré-processamento se iniciou com sua transformação de RGB para escala de cinza. Logo após, a mesma foi submetida a uma binarização seguida de duas iterações da operação fechamento, no intuito de remover os pequenos ruídos e preencher os buracos dos caracteres. Por fim, realizou-se uma normalização Min Max para garantir que a imagem estivesse entre 0 e 255. Todo o processo de pré-processamento pode ser visto na Figura 31.

Figura 31: Etapas do Pré-processamento da Imagem na Solução Simples



Fonte: Elaborada pelo autor.

Após a imagem ser pré-processada, ela foi submetida ao processo de TM com os 28 modelos de *template*, para cada caractere, selecionados. Para cada *template*, o maior valor de similaridade foi armazenado, juntamente com o ponto correspondente – x e y –, a largura, altura e o caractere do *template*. Esse tipo de iteração se fez possível pelo fato do CAPTCHA do Lattes não apresentar caracteres repetidos. No fim, foi possível obter um vetor com 28 posições, onde cada posição do vetor representa o representa o percentual de correspondência de um dos *templates* com um fragmento da imagem. Esse vetor foi ordenado de forma decrescente, partindo do *template* com maior até o de menor similaridade. Os quatro primeiros elementos do vetor foram tidos como os caracteres corretos, de forma que ao ordená-los pela coordenada x obtém-se o texto escrito no CAPTCHA.

3.4 Solução Robusta para o CAPTCHA do Lattes

A solução simples para o CAPTCHA do Lattes foi desenvolvida no intuito de demonstrar o quão frágil o mesmo é e que não seria necessário desenvolver uma solução complexa para quebrá-lo. Entretanto, tendo em vista que o esquema de CAPTCHA proposto neste trabalho tem o intuito de não ser quebrado com uma solução simples e que será necessário comparar o desempenho de ambos esquemas de CAPTCHA, fez-se relevante o desenvolvimento de uma solução robusta para o CAPTCHA do Lattes. Portanto, a solução complexa para o CAPTCHA do Lattes tem o intuito de criar um indicador de comparação entre o esquema de CAPTCHA encontrado no Lattes e o desenvolvido nesta pesquisa. Desta forma, a solução proposta será a mesma para o novo esquema de CAPTCHA, tendo como diferença os conjuntos de treinamento, teste e validação.

Essa solução chama-se de robusta pelo fato de mostrar-se mais tolerante a variações

– mudanças de cores e distorções – diferentemente do TM, que é extremamente inflexível a esse tipo de variação. Para que o TM tenha uma robustez tão grande quanto a solução proposta, o mesmo deve ter *templates* que equivalham a todas as possíveis distorções e manipulações para cada caractere da base, o que seria algo complexo de se fazer. Dessa forma, propõe-se a utilização de uma *Faster* R-CNN para a segmentação e classificação dos caracteres de um CAPTCHA.

Para o desenvolvimento dessa solução, foram utilizadas as bibliotecas Tensorflow e NumPy. Além delas, usou-se também o modelo pré-treinado *Faster* R-CNN Inception V2 COCO, disponibilizado pelo Google no repositório online Github (2019). A escolha desse modelo se deu pelo fato do mesmo ter um considerável COCO mAP e apresentar um curto tempo de inferência por imagem. Além disso, ele também utiliza da Inception V2 para classificação, que é uma rede bem menor que uma Inception ResNet V2, a qual demoraria ainda mais para treinar.

Por esse modelo ter sido treinado sobre o *dataset* COCO, ele apresenta uma detecção de *feature* bem diversificada, o que permite a reutilização dele em um outro projeto. Desta forma, será feita a transferência de aprendizado do modelo pré-treinado para o modelo proposto, sendo necessário treinar somente as últimas camadas da Inception V2. Esse estilo de treinamento a partir da transferência de aprendizado, permite que se obtenha um bom modelo demandando menos horas de treino da rede.

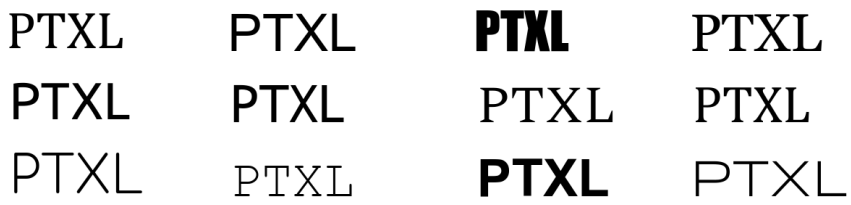
Realizou-se o treinamento da *Faster* R-CNN em um computador equipado com uma 1050 Ti – Nvidia – e foi determinado o número de 22.500 épocas, o equivalente a uma média de três horas de duração do treino. Esse número de épocas foi definido de forma que fosse o mínimo necessário para obter uma alta acurácia, ao submeter o modelo a um conjunto de validação. Todos os hiperparâmetros da rede serão os mesmos para as duas soluções robustas propostas nesse trabalho, de forma a não viesar os resultados obtidos e permitir uma comparação entre eles. Desta forma, utilizou-se da API do Tensorflow específica para treinamento de modelos capazes de detectar objetos.

3.5 Proposta de Esquema de CAPTCHA

Após uma análise detalhada do CAPTCHA do Lattes e suas fragilidades, percebe-se quais pontos devem ser modificados de forma a torná-lo mais robusto. Uma das melhorias planejadas para o esquema proposto é a utilização de 56 fontes diferentes no novo esquema de CAPTCHA, tendo em vista que o do Lattes só utiliza uma, de forma que um caractere raramente seja igual a outro em mais de uma CAPTCHA gerado. Tendo em vista a grande diversidade de fontes existentes na internet, levou-se em consideração a escolha das que não tornariam-se ilegíveis quando submetidas à distorção, pois isso feriria um dos princípios básicos de um CAPTCHA textual, o da legibilidade. Alguns exemplos de fontes utilizadas no

novo esquema podem ser vistos na Figura 32.

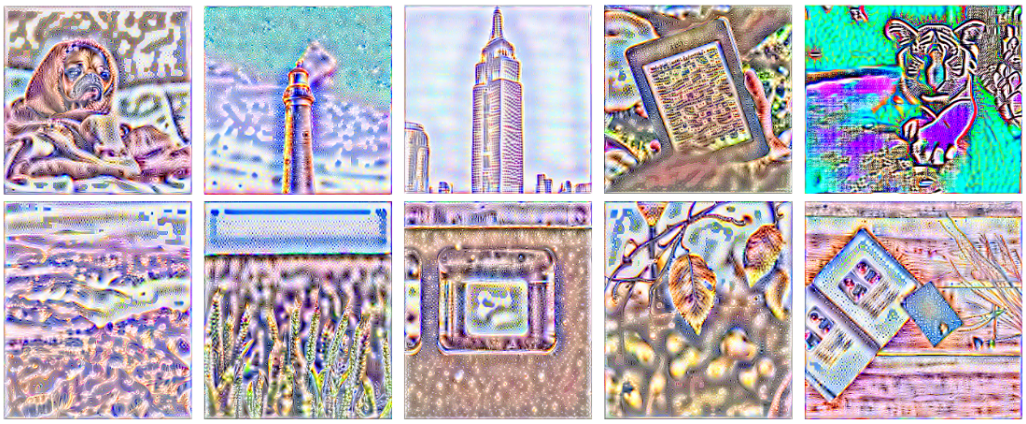
Figura 32: Amostra de fontes utilizadas no novo esquema proposto de CAPTCHA



Fonte: Elaborada pelo autor.

Outra melhoria proposta no novo esquema de CAPTCHA é a utilização de uma diversidade maior de *backgrounds*. Os *backgrounds* utilizados no novo esquema são oriundos de imagens coletadas na internet, mais especificamente 100, que foram distorcidas. Apesar dos *backgrounds* não serem gerados aleatoriamente, apenas um fragmento da *imagem* é utilizado como *background* e esse fragmento é escolhido de forma randômica. Desta forma, 100 imagens de fundo transformam-se em um conjunto muito maior de possibilidades. Uma amostra dos fundos escolhidos pode ser vista na Figura 33.

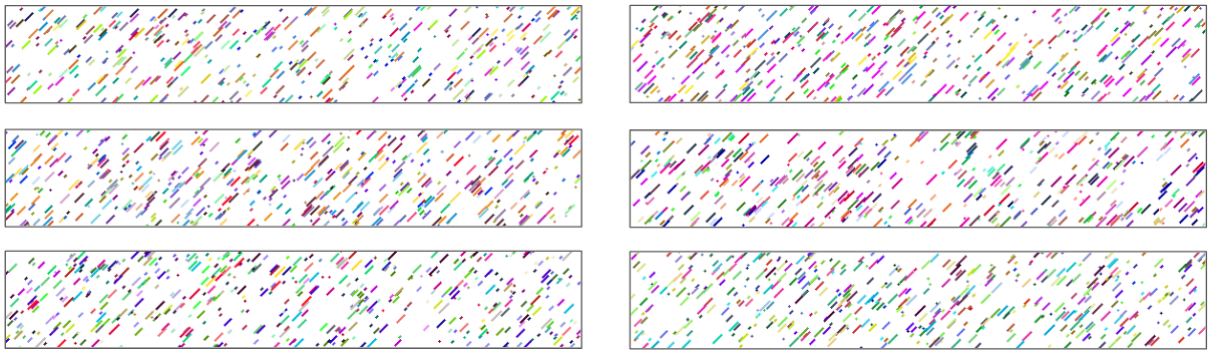
Figura 33: Amostra das imagens dos *backgrounds* escolhidos



Fonte: Elaborada pelo autor.

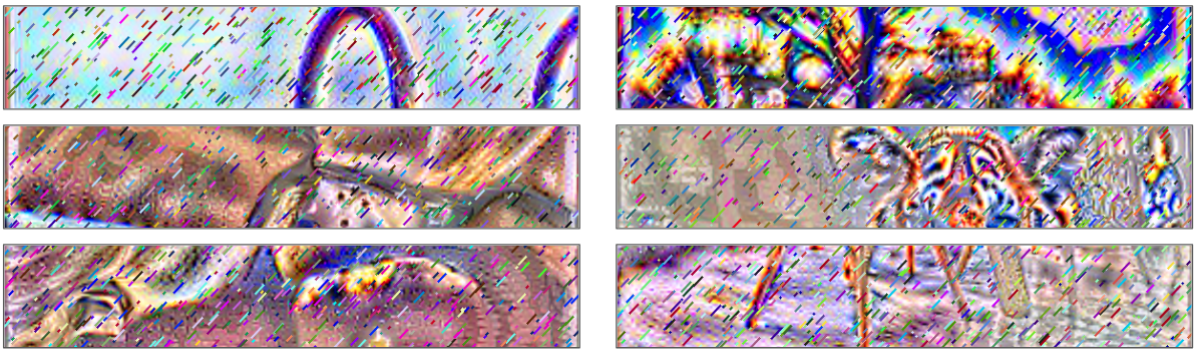
Apesar dos fundos apresentarem uma característica de aleatoriedade proveniente do fragmento escolhido randomicamente da imagem, percebe-se que ainda haveria um conjunto finito de possíveis *backgrounds*. Desta forma, agregou-se mais aleatoriedade ao fundo do CAPTCHA a partir da adição de ruídos coloridos, o que dificulta a extração do fundo e a detecção das *features*. O ruído adicionado é somado ao fundo em posições e cores aleatórias, além disso a densidade do ruído na imagem também é definida aleatoriamente, dentro de um intervalo especificado. Exemplos dos ruídos gerados e do *backgrounds* somados com os ruídos aleatórios podem ser vistos nas Figuras 34 e 35, respectivamente.

Figura 34: Amostra do ruídos aleatório gerado



Fonte: Elaborada pelo autor.

Figura 35: Amostra dos *backgrounds* imbuídos do ruídos aleatório



Fonte: Elaborada pelo autor.

Tendo o fundo do CAPTCHA criado e as fontes definidas, partiu-se para a geração do texto da imagem. Primeiramente, buscou-se corrigir as fragilidades apresentadas no CAPTCHA do Lattes, ou seja, utilizou-se das 56 fontes selecionadas. Além disso, os caracteres agora são apresentados com cores aleatórias e são posicionados em diversas partes da imagem. Eles são individualmente distorcidos – utilizando apenas de redimensionamento e rotação –, tornando-os bem diferentes entre si e entre todos os outros CAPTCHAs gerados. No intuito de evitar ferir a legibilidade do caractere, o redimensionamento e a rotação foram definidos de forma aleatória, mas apresentam um valor máximo e mínimo.

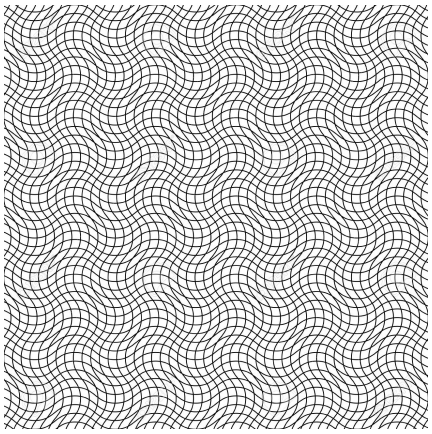
Além da correção das fragilidades encontradas no CAPTCHA do Lattes, foi adicionada uma melhoria que consiste na possibilidade da superposição dos caracteres. Essa melhoria é tida como uma das que mais agregam robustez ao CAPTCHA, em contrapartida, são aquelas que mais reduzem a legibilidade. Desse modo, definiu-se um limite de superposição aceitável e foi permitido que as partes superpostas apresentassem uma opacidade reduzida, fazendo com que o caractere de cima e o de baixo pudessem ser vistos individualmente, em sua totalidade. Exemplos dos textos gerados para o novo esquema de CAPTCHA podem ser vistos na Figura 36.

Figura 36: Amostra dos textos gerados aleatoriamente

Fonte: Elaborada pelo autor.

Com os textos gerados, foi dado início a junção do fundo com os caracteres. Porém, para aumentar a legibilidade dos caracteres, adicionou a eles uma borda preta, que os destacavam do fundo. Isso foi necessário pelo fato do *background* se apresentar muito ruidoso. Entretanto, essa borda acabou por facilitar que um *script* pudesse segmentar os caracteres da imagem. Desta forma, fez-se necessária a adição de um outro ruído que dificultasse essa segmentação. O ruído adicionado na imagem é uma malha distorcida por uma função senoidal, como pode ser visto na Figura 37.

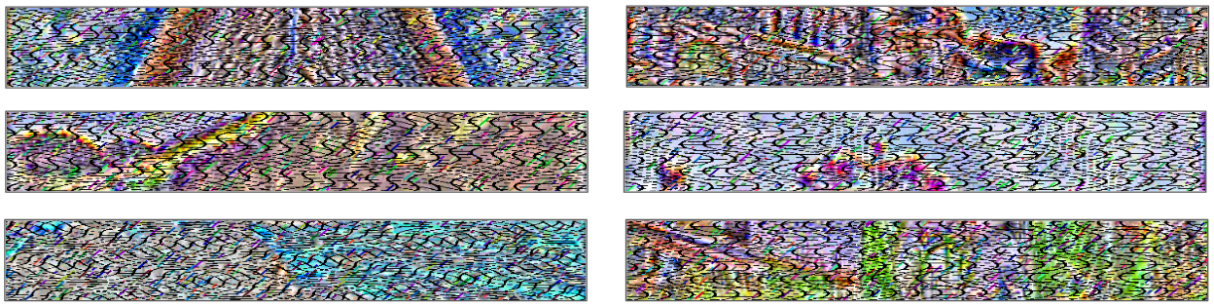
Figura 37: Malha distorcida por uma função senoidal



Fonte: Google Imagens (2019).

Essa malha é rotacionada e redimensionada de forma aleatória, além de ter sua cor definida como preta. A malha apresenta-se sempre nesta mesma cor pelo fato da borda do caractere também ser preta, portanto, ela consegue dificultar a segmentação do caractere através da borda. Alguns exemplos da adição da malha distorcida ao fundo criado para o CAPTCHA podem ser vistos na Figura 38. Ao final, adicionou-se os caracteres gerados ao fundo criado, com todos os ruídos. Desta forma, compõe-se o esquema de CAPTCHA proposto neste trabalho.

Figura 38: Adição da malha distorcida ao fundo criado



Fonte: Elaborada pelo autor.

3.6 Solução Robusta para o Novo Esquema de CAPTCHA

Após a conclusão do novo esquema de CAPTCHA, foram gerados três conjuntos de imagens – treinamento, teste e validação –, permitindo que uma solução robusta fosse treinada para tentar quebrar o novo CAPTCHA. A solução proposta para o novo modelo assemelha-se à apresentada para o do Lattes. Este trabalho propõe um novo esquema de CAPTCHA, para isso, utilizou-se dos mesmos caracteres presentes no *dataset* do Lattes, fazendo com que todos os CAPTCHAs do novo modelo apresentassem sempre quatro caracteres. A intenção deste trabalho é mostrar que o esquema novo é mais robusto à quebra que o anterior, e utilizar de caracteres diferentes poderia tornar o novo CAPTCHA mais forte apenas por ter mais caracteres. É importante salientar que se o número de caracteres fosse definido de forma aleatória, isso iria agregar ainda mais robustez ao CAPTCHA. Tendo o *dataset* do novo CAPTCHA criado, partiu-se para o treinamento de uma *Faster* R-CNN com os mesmos hiperparâmetros e configurações da rede proposta para o CAPTCHA do Lattes.

3.7 Análise da Legibilidade do CAPTCHA Proposto

Com a conclusão do treinamento da solução robusta para o novo esquema de CAPTCHA, fez-se necessário atestar a legibilidade do novo CAPTCHA. Por mais que um CAPTCHA seja mais robusto que outro para com as máquinas, ambos devem ser facilmente lidos por humanos. Um CAPTCHA que não é capaz de ser decifrado por uma máquina e nem por uma pessoa é inadequado, pois a função de separar humanos de robôs não será realizada, ambos estarão no mesmo grupo. Desta forma, não basta provar que um CAPTCHA é mais resistente à quebra que o outro, tem que explicitar também que a legibilidade de ambos persistiu fácil para os humanos. Portanto, foi proposto um questionário, o qual foi submetido à voluntários, responsável por verificar se o novo modelo perdeu muita legibilidade quando comparado com o do Lattes.

O questionário[28] proposto foi criado utilizando da plataforma do Google Forms, pois a mesma facilita a disseminação e a coleta das respostas das questões através internet. O questionário foi preenchido apenas uma vez por pessoa e era dividido em duas partes. A primeira indagava qual a idade do usuário e se o mesmo apresentava alguma deficiência visual. A motivação para a inserção dessa seção no questionário se deu pela necessidade de avaliar o quão invariante o modelo do CAPTCHA proposto era à idade e aos problemas oftalmológicos.

A segunda parte consistia na exposição de 20 CAPTCHAs pré-determinados, no intuito de avaliar a legibilidade do modelo proposto pelos usuários. Dentre os 20 CAPTCHAs selecionados, 10 deles não foram decodificados pela máquina, dez deles foram acertados pela máquina, onde cinco apresentavam-se fáceis de serem lidos por pessoas e os outros cinco difíceis. Essa seleção de CAPTCHAs se deu na tentativa de mostrar que o CAPTCHA pode ser difícil para a máquina e fácil para o humano. Da mesma forma o inverso se aplica.

Na medida em que o resultado proveniente das respostas do questionário foi se concretizando, desenvolveu-se uma análise comparativa entre a legibilidade do novo esquema de CAPTCHA e o do Lattes. Entretanto, assumiu-se que o do Lattes apresenta uma legibilidade ideal, ou seja, que o mesmo pode ser lido e terá seu conteúdo entendido por 100% da amostra. Foi realizada essa simplificação pelo fato de que os humanos apresentam uma capacidade de entendimento melhor que as máquinas e as mesmas obtiveram 100% de acurácia ao tentar quebrá-lo, portanto, foi interessante assumir que os humanos também teriam.

[28]Para ter acesso ao questionário, clique no link a seguir. Formulário disponível em <https://forms.gle/YVCFt3mLW3guuY8z5>.

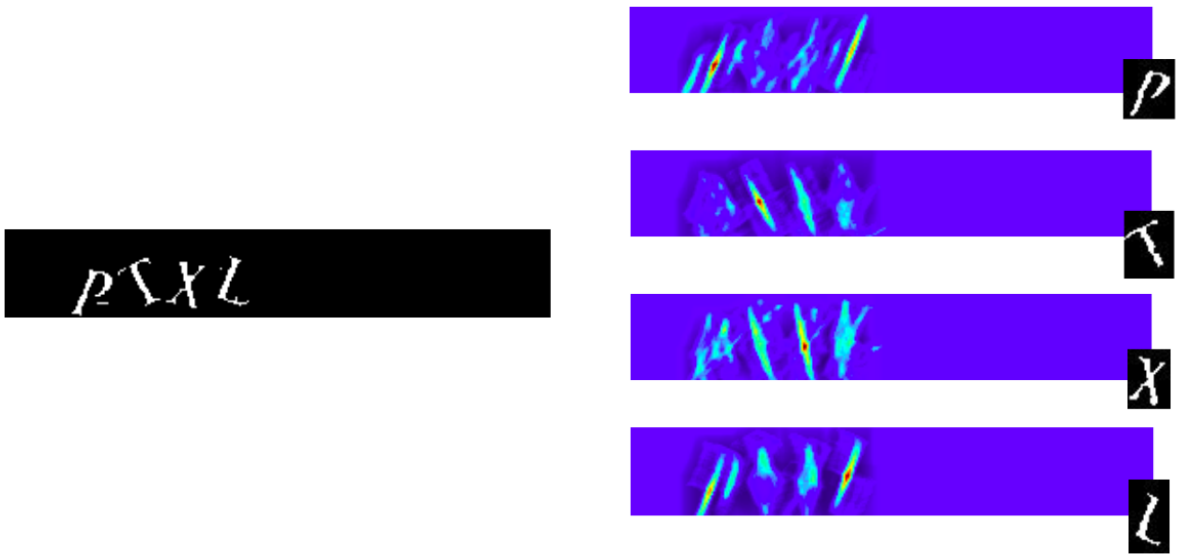
4

Resultados e Discussões

4.1 Resultado da Solução Simples para o CAPTCHA do Lattes

Ao realizar a solução simplificada em todo o conjunto de validação do Lattes, obteve-se 95,1% de acurácia. Analisando as predições incorretas, percebe-se que muitas delas se deram por caracteres que apresentam *templates* similares, como o T e o 7 ou o V e o W, como pode ser visto na Figura 30. É possível observar na Figura 39, a resposta da operação de TM ao executá-la utilizando dos *templates* P, T, X e L. Observa-se que o ponto de maior similaridade aproxima-se da cor vermelha.

Figura 39: Exemplo da aplicação do TM em um CAPTCHA do Lattes

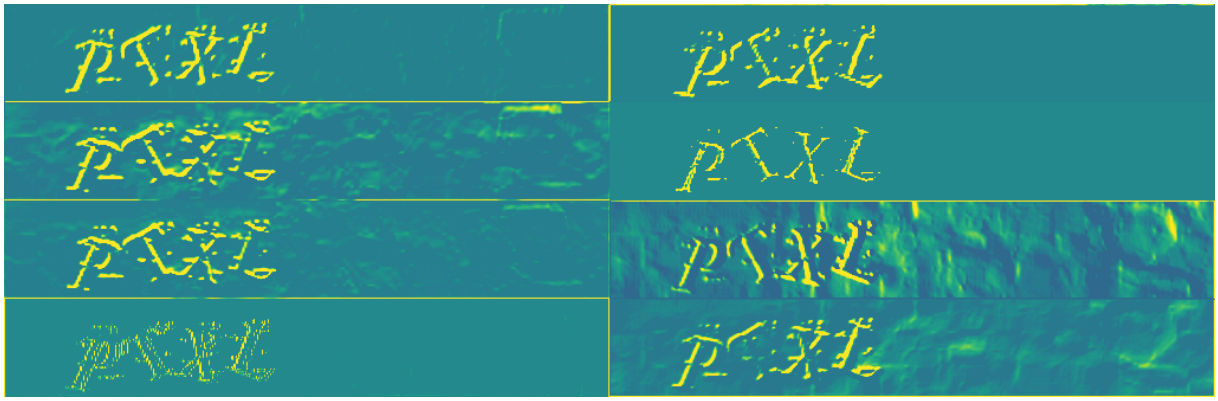


Fonte: Elaborada pelo autor.

4.2 Resultado da Solução Robusta para o CAPTCHA do Lattes

Ao realizar a solução robusta em todo o conjunto de validação do Lattes, obteve-se 100% de acurácia. Esse resultado era esperado, pois apesar dos caracteres e os fundos serem diferentes em cada CAPTCHA, um padrão muito evidente era estabelecido entre eles. A solução robusta se mostrou mais do que eficaz na resolução dos CAPTCHAs do Lattes, salientando, mais uma vez, a fragilidade existente no seu modelo de CAPTCHA. Ao analisar alguns canais da última camada de ativação do modelo treinado para quebrar o CAPTCHA do Lattes, percebeu-se que uma grande parte da imagem já era desconsiderada pela rede e que a mesma reconheceu exatamente as *features* que definiram os caracteres desenhados na imagem, como pode ser visto na Figura 40.

Figura 40: Alguns canais da última camada de ativação da solução robusta do Lattes

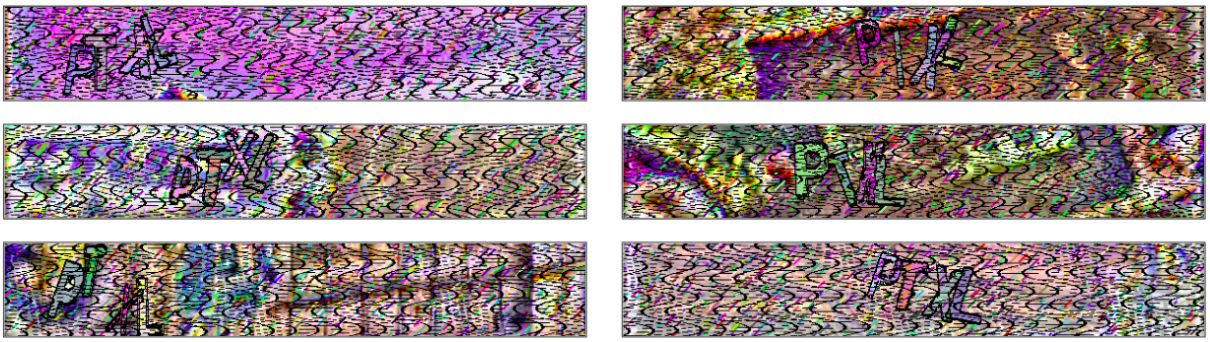


Fonte: Elaborada pelo autor.

4.3 Resultados do Novo Esquema de CAPTCHA

Após a adição do fundo criado, com os ruídos e caracteres gerados, o novo esquema de CAPTCHA proposto foi finalizado e o resultado alcançado. Isso ocorreu, pois a utilização de técnicas como TM não podem quebrá-lo, pelo fato do mesmo apresentar diversas componentes aleatórias em sua geração. Tendo em vista que foi proposto um *software* capaz de gerar esses CAPTCHAs, criou-se uma base de dados para a realização do treinamento, entretanto, diferentemente da base do Lattes, essa já foi gerada e anotada automaticamente, não sendo necessário realizar a etapa morosa de sua anotação. Algumas amostras de CAPTCHAs gerados, utilizando do novo esquema, podem ser vistas na Figura 41.

Figura 41: Amostras de CAPTCHAs gerados a partir do esquema proposto

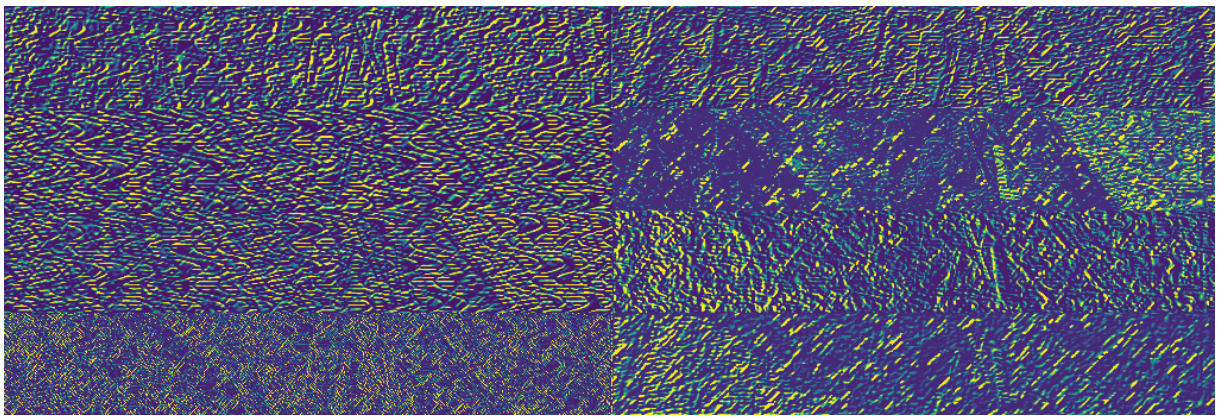


Fonte: Elaborada pelo autor.

4.4 Resultado da Solução Robusta para o novo esquema de CAPTCHA

Após a aplicação do novo modelo, treinado para solucionar os CAPTCHA do novo esquema, no conjunto de validação, foi obtida uma acurácia 55,7%. Isso denota uma melhora na segurança, provida pelo CAPTCHA do novo esquema, quando comparado com o do Lattes. Esse resultado poderia ser melhorado, entretanto, houve uma preocupação com relação a legibilidade do CAPTCHA. Diferentemente do CAPTCHA do Lattes, o novo apresenta muito mais textura, o que acaba por confundir a solução robusta. Como pode ser visto na Figura 42, alguns canais da última camada de ativação analisam *features* difusas, que não representam com qualidade os caracteres desenhados na imagem. Isso é um indício de que a CNN não conseguiu aprender ao certo as *features* que descrevem os caracteres. Um dos motivos para isso acontecer é o fato dos caracteres, fundos e ruídos serem muito aleatórios e não haverem padrões muito bem estabelecidos para se analisar.

Figura 42: Canais da última camada de ativação da solução robusta do novo CAPTCHA

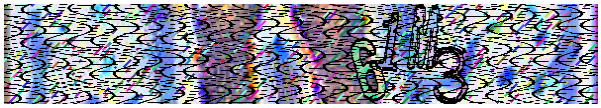


Fonte: Elaborada pelo autor.

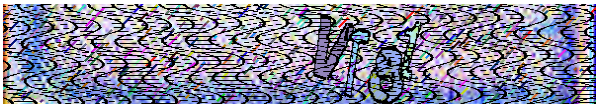
4.5 Resultado do Questionário de Legibilidade

Percebeu-se que o esquema de CAPTCHA proposto nesse trabalho mostra-se mais resistente à quebra por uma solução robusta que o modelo do Lattes, entretanto, ainda resta avaliar se o mesmo apresenta uma boa legibilidade para com humanos. O questionário aplicado apresentava 20 CAPTCHAs para os voluntários responderem o que enxergavam, porém, das 20 questões cinco delas eram tidas como fáceis para humanos e máquinas, outras cinco foram consideradas como difíceis para os humanos e fáceis para as máquinas e, em relação as 10 restantes, a máquina não soube responder. Os CAPTCHAs fáceis compunham as perguntas 1,4,7,10 e 13, os difíceis os números 3,6,9,12 e 15 e as demais exibiam os CAPTCHAs que a máquina errou. Os esquemas que foram usados no questionário podem ser vistos nas Figuras 43, 44 e 45.

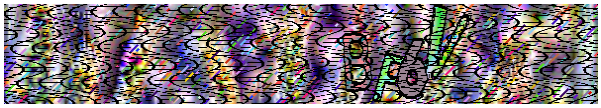
Figura 43: Seleção de CAPTCHAs Fáceis



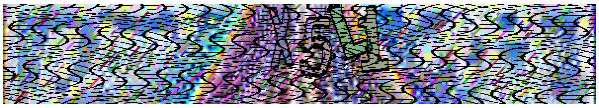
(a) Questão 1



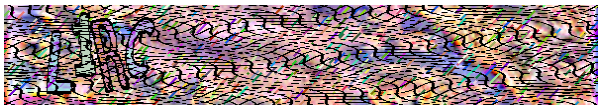
(b) Questão 4



(c) Questão 7



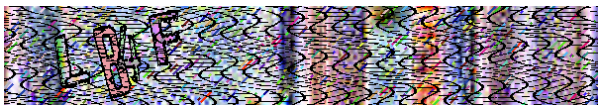
(d) Questão 10



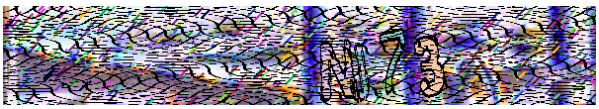
(e) Questão 13

Fonte: Elaborada pelo autor.

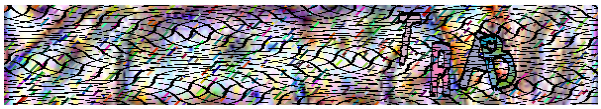
Figura 44: Seleção de CAPTCHAs Difíceis



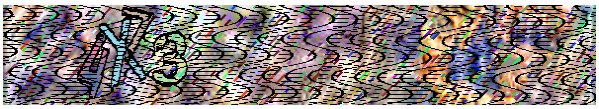
(a) Questão 3



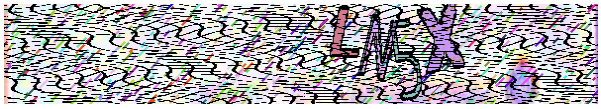
(b) Questão 6



(c) Questão 9



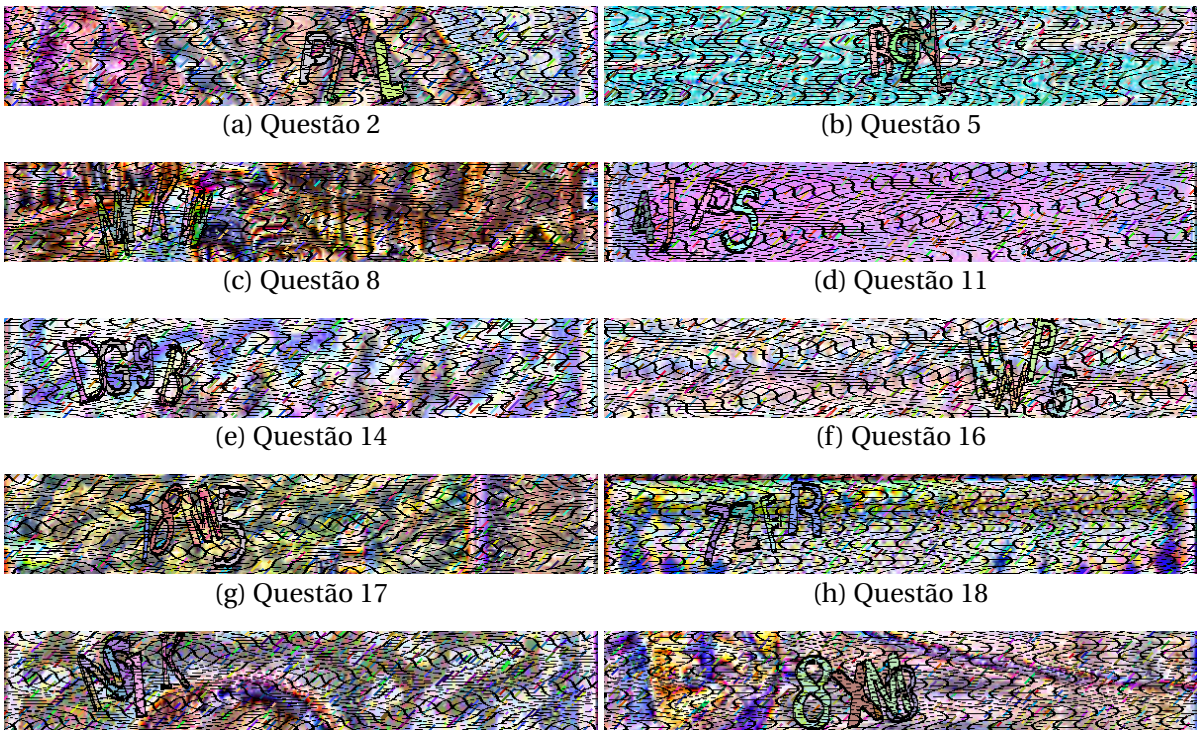
(d) Questão 12



(e) Questão 15

Fonte: Elaborada pelo autor.

Figura 45: Seleção de CAPTCHAs que a Máquina não conseguiu solucionar



(a) Questão 2 (b) Questão 5

(c) Questão 8 (d) Questão 11

(e) Questão 14 (f) Questão 16

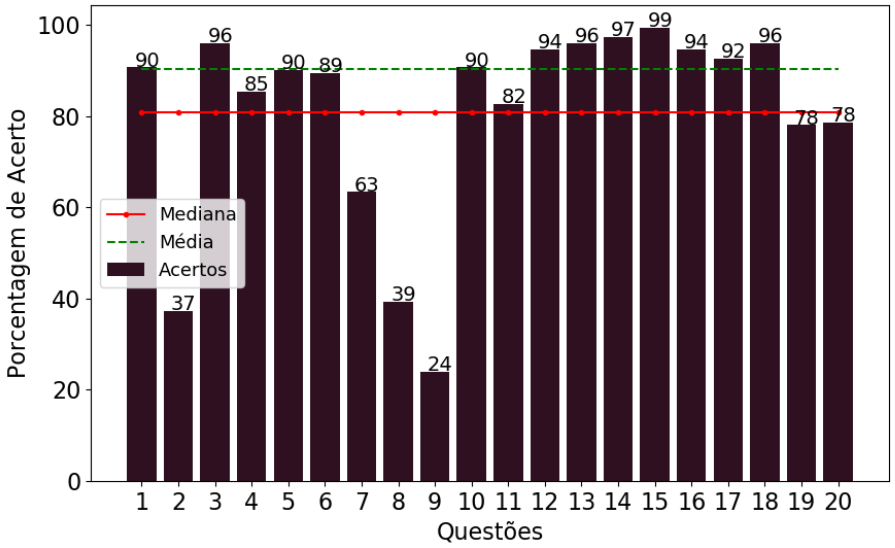
(g) Questão 17 (h) Questão 18

(i) Questão 19 (j) Questão 20

Fonte: Elaborada pelo autor.

Tendo em vista como ficou estruturado o questionário, o mesmo foi aplicado para 150 pessoas, que o responderam individualmente. Das respostas coletadas, foram obtidos os percentuais de respostas corretas por questão, como pode ser visto na Figura 46.

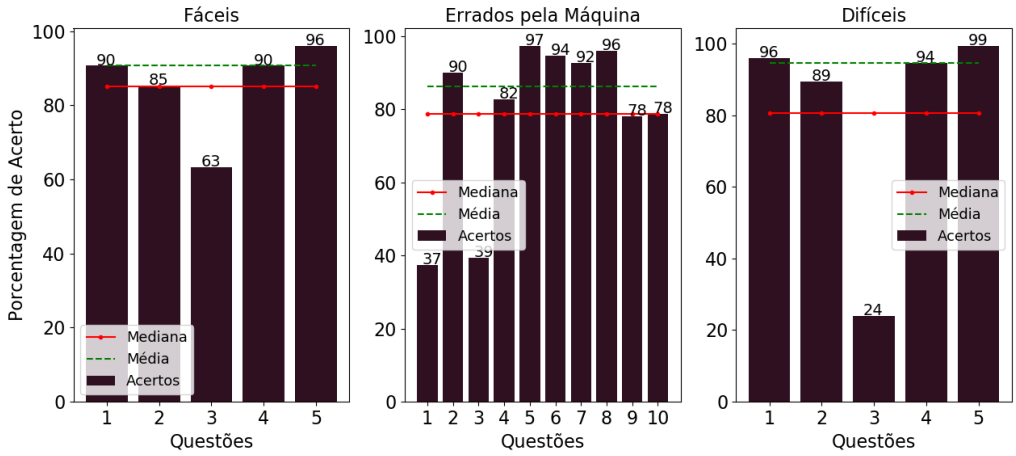
Figura 46: Percentual de respostas corretas por questão



Fonte: Elaborada pelo autor.

Observou-se que os CAPTCHAs que as pessoas mais erraram – questões 2,7,8 e 9 – apresentam seus caracteres muito distorcidos e, alguns, superpostos. Isso dificultou muito a leitura do usuário, que acabou por confundir os caracteres. Apesar dessas quatro questões apresentarem um percentual de acerto bem abaixo da média, percebeu-se que apenas duas das 10 questões que a máquina errou foram difíceis para os humanos, ou seja, apenas uma questão teve um baixo percentual de acertos no conjunto de CAPTCHAs fáceis e uma no de difíceis, como pode ser visto na Figura 47. Desta forma, a média de acertos foi de 80%.

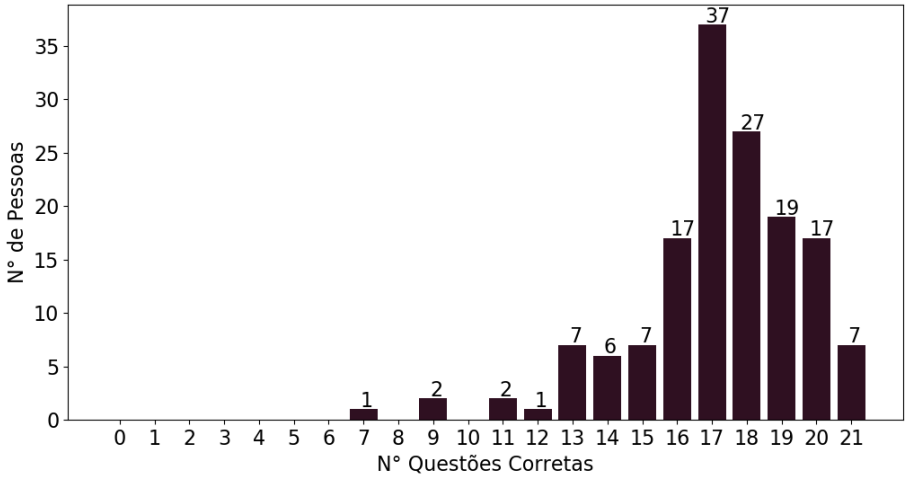
Figura 47: Percentual de respostas corretas por conjunto de questões



Fonte: Elaborada pelo autor.

Outro resultado a ser analisado foi o número de acertos por pessoa, onde 82,6% dos usuários acertaram mais de 15 questões, como pode ser visto na Figura 48.

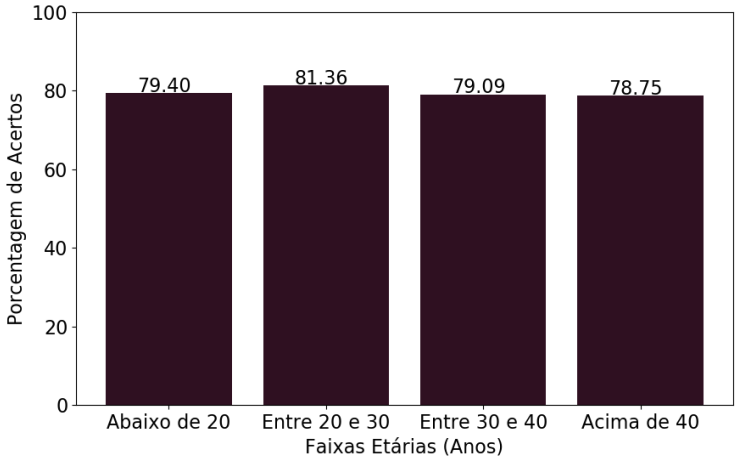
Figura 48: Número de questões corretas por pessoa



Fonte: Elaborada pelo autor.

Tendo em vista que 67,3% da amostra alegou ter algum tipo de deficiência visual e que 34% apresentava-se fora da faixa etária dos 20 a 30 anos, buscou-se analisar o quão invariante a legibilidade era a idade e a problemas oftalmológicos. Desta forma, analisou-se a média de acertos por faixas etárias e percebeu-se que as mesmas apresentavam médias muito próximas da média de todos os usuários – 80% –, como pode ser visto na Figura 49.

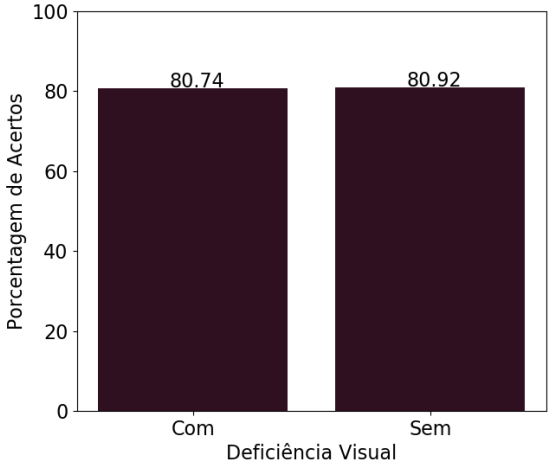
Figura 49: Percentual médio de acerto por faixa etária



Fonte: Elaborada pelo autor.

Ao final, de modo semelhante à faixa etária, promoveu-se uma análise para avaliar a invariância sobre problemas oftalmológicos e percebeu-se os usuários com e sem dificuldades apresentaram uma média de acertos muito similar à média geral dos usuários, conforme apresentado na imagem a seguir.

Figura 50: Percentual médio de acerto para usuários com e sem deficiência visual



Fonte: Elaborada pelo autor.

5

Conclusão

A partir dos resultados coletados, foi mostrado que apesar do CAPTCHA do Lattes apresentar uma boa legibilidade, ele não cumpria com um dos princípios de um CAPTCHA textual, que é ser capaz de diferenciar se o usuário é uma máquina ou um humano. Isso pode ser afirmado, pois o mesmo foi quebrado com 100% de acurácia, utilizando de uma abordagem robusta, e 95,1%, em uma abordagem simples. Dessa forma, foi proposto um esquema de CAPTCHA mais robusto à tentativa de quebra, que se apresentou legível para as pessoas e com uma capacidade de discernimento entre humanos e máquinas. Além do esquema, foi desenvolvido um *software* capaz de gerar tais CAPTCHAs, postado na plataforma Github[29].

Nesse contexto, é possível perceber que o esquema de CAPTCHA proposto apresenta uma característica invariante à idade e a pessoas com deficiências visuais, tendo em vista que a média de acertos prevaleceu para todos os testes realizados com esses grupos, conforme apresentado nas figuras 49 e 50. A maioria dos CAPTCHAs que foram errados pela máquina obtiveram uma taxa de acertos por partes das pessoas, o que mostra que o CAPTCHA realiza o seu papel de ser simples para os humanos e difícil para as máquinas. Umas das características que mais dificulta a máquina de quebrar o CAPTCHA é a superposição dos caracteres, pois ela modifica sua estrutura, que acaba por se tornar bem diferente da que a máquina aprendeu. Essa característica também dificulta a vida do humano, mas não apresenta um impacto tão negativo quando comparado com as máquinas.

Desta forma, é possível observar que todos os objetivos específicos descritos para a realização desse trabalho foram concluídos, pois foram apresentadas às fragilidades do CAPTCHA do Lattes, as duas soluções capazes de quebrá-lo foram propostas, um esquema de novo CAPTCHA foi desenvolvido e provou-se que o mesmo apresenta uma resistência à quebra maior que o proposto pelo Lattes. Por fim, foi elucidado que o mesmo, apesar de apresentar muito mais componentes visuais dispostos em suas imagens, não perdeu tanta legibilidade, tornando-o um CAPTCHA ainda mais completo que o do Lattes. Nesse sentido,

[29] Para ter acesso ao questionário, clique no link a seguir. Disponível <>.

conclui-se que o CAPTCHA proposto atendeu todos os requisitos necessários para enunciá-lo como CAPTCHA textual, onde o objetivo geral também foi alcançado.

5.1 Trabalhos Futuros

Como considerações finais, faz-se interessante elucidar algumas possíveis melhorias futuras que viriam a enriquecer esse trabalho. A primeira delas constitui na utilização de mais caracteres por CAPTCHA, não se limitando a apenas quatro, fazendo com que esse número possa ser mais um componente aleatório do esquema. Isso agregaria mais robustez ao CAPTCHA e apresentaria ainda mais dificuldade à maquina de decifrá-lo. Vale lembrar de sempre tomar cuidado ao propor um CAPTCHA, afim de preservar sua legibilidade para com os humanos.

Outra possível melhoria seria utilizar de *Generative Adversarial Networks*[30] para a geração dos CAPTCHA, podendo gerar os caracteres e fundos ao mesmo tempo. Um possível trabalho futuro é a utilização de outros métodos para tentar quebrar o CAPTCHA proposto e assim mostrar as fragilidades do mesmo. Por fim, uma última melhoria seria a utilização de mais fontes, fundos, ruídos diferentes e outras técnicas como *Hollow Letters*[31], que são vistas na academia como pontos fortes em CAPTCHAs que as utilizam.

[30] Em português, "Redes Generativas Adversárias". (Tradução Livre).
[31] Em português, "Letras Ocas". (Tradução Livre).

Referências Bibliográficas

ABU-MOSTAFA, Y. S.; PSALTIS, D. Image normalization by complex moments. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, IEEE, n. 1, p. 46–55, 1985.

BELK, M. et al. Do human cognitive differences in information processing affect preference and performance of captcha? *International Journal of Human-Computer Studies*, Elsevier, v. 84, p. 1–18, 2015.

CHEN, J. et al. A survey on breaking technique of text-based captcha. *Security and Communication Networks*, Hindawi, v. 2017, 2017.

CHOLLET, F. *Deep learning with Python*. [S.l.]: Manning Publications Company, 2017.

GAFNI, R.; NAGAR, I. Captcha–security affecting user experience. *Issues in Informing Science and Information Technology*, Directory of Open Access Journals, v. 13, n. unknown, p. 063–077, 2016.

GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 580–587.

GITHUB. *Tensorflow Object Detection Zoo*. 2019. Acessado em abril/2019. Disponível em: <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>.

GONZALEZ, R. C.; WOODS, R. E. et al. *Digital image processing*. [S.l.]: Prentice hall Upper Saddle River, NJ, 2002.

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

HARALICK, R. M.; SHAPIRO, L. G. *Computer and robot vision*. [S.l.]: Addison-wesley Reading, 2001. v. 1.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

LI, F.-f.; JOHNSON, J.; YEUNG, S. Cs231n convolutional neural networks for visual recognition. *Neural networks*, v. 1, 2017.

LIU RONG ZHANG, K. Q. K. *CNN for breaking text-based CAPTCHA with noise*. 2017. Disponível em: <https://doi.org/10.1117/12.2281743>.

MITCHELL, T. M. *Machine learning*. McGraw-Hill, 1997. (McGraw Hill series in computer science). ISBN 978-0-07-042807-2. Disponível em: <http://www.worldcat.org/oclc/61321007>.

PRATT, W. K. *Digital Image Processing: PIKS Inside*. 4rd. ed. New York, NY, USA: John Wiley & Sons, Inc., 2007. ISBN 0471374075.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 91–99.

SZEGEDY, C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: *ICLR 2016 Workshop*. [s.n.], 2016. Disponível em: <https://arxiv.org/abs/1602.07261>.

SZEGEDY, C. et al. Going deeper with convolutions. In: *Computer Vision and Pattern Recognition (CVPR)*. [s.n.], 2015. Disponível em: <http://arxiv.org/abs/1409.4842>.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,*. [s.n.], 2016. Disponível em: <http://arxiv.org/abs/1512.00567>.

USMANI, A. et al. New text-based user authentication scheme using captcha. In: *Information and Communication Technology for Competitive Strategies*. [S.l.]: Springer, 2019. p. 313–322.

YAN, J.; AHMAD, A. S. E. Usability of captchas or usability issues in captcha design. In: ACM. *Proceedings of the 4th symposium on Usable privacy and security*. [S.l.], 2008. p. 44–52.

YE, G. et al. Yet another text captcha solver: A generative adversarial network based approach. In: ACM. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. [S.l.], 2018. p. 332–348.