Redes Neurais Convolucionais Aplicadas ao Reconhecimento Automático de *CAPTCHAs*

Sérgio Alexandre Arruda Pinheiro
Orientadora: Elloá B. Guedes da Costa

[1]Escola Superior de Tecnologia
Universidade do Estado do Amazonas
Av. Darcy Vargas, 1200 – Manaus – Amazonas

{saap.snf,ebgcosta}@uea.edu.br

1. Introdução

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

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

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

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

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

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

1.1. Objetivos

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

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

1.2. Justificativa

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

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

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

1.3. Metodologia

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

1. Estudos dos conceitos a respeito de CAPTCHAs;
2. Implementação de algoritmo de raspagem para captura de CAPTCHAs do Currículo Lattes;
3. Rotulação manual dos CAPTCHAs coletados;
4. Consolidação da base de dados e descrição das características;
5. Estudo dos conceitos das redes neurais convolucionais;
6. Levantamento de técnicas e tecnologias para implementação de redes neurais convolucionais;
7. Proposição de diferentes redes neurais convolucionais para endereçamento do problema;
8. Definição de métricas de desempenho para avaliação da capacidade de generalização das redes propostas;
9. Definição de técnica de validação cruzada para análise dos resultados;
10. Realização do treinamento das redes neurais convolucionais propostas;
11. Realização dos testes e coleta das métricas de desempenho;
12. Elaboração de artefatos de visualização (gráficos, matrizes de confusão, etc.) para apresentação dos resultados das métricas;
13. Análise e comparação dos resultados obtidos.

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

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

1.4. Cronograma

As atividades descritas na seção anterior serão executadas de acordo com o cronograma apresentado na Tabela 1. Os meses se referem ao ano de 2018.

1.5. Organização do Documento

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

2. Fundamentação Teórica

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

Tabela 1: Cronograma para execução das atividades referentes à este trabalho de conclusão de curso.

	Meses									
Atividades	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**
1	X									
2	X									
3	X									
4		X								
5		X								
6			X							
7			X							
8			X							
9				X						
10				X	X	X	X			
11					X	X	X	X		
12							X	X		
13									X	X
14	X	X	X	X						
15				X						
16					X	X	X	X	X	X
17										X

2.1. CAPTCHA

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

Os testes considerados no CAPTCHA visam distinguir os humanos das máquinas, especialmente os *web bots*, por meio da solicitação da identificação de padrões não-triviais em imagens e sons, elaborados com características dificilmente capturáveis por algoritmos convencionais. Deste modo, a aprovação em um CAPTCHA aumenta as evidências de que o usuário é, de fato, um humano (SANTOS, 2016).

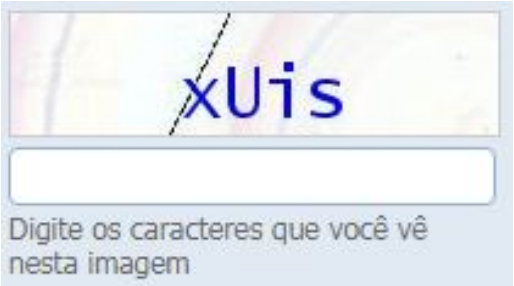
No entanto, os CAPTCHAs estão sujeitos à falsos positivos e negativos. Isso se dá pois este tipo de teste pode variar em seu nível de dificuldade. Em certos tipos de CAPTCHAs baseados em imagens, por exemplo, a pouca distorção ou poluição pode facilitar a sua detecção por um algoritmo de visão computacional,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

Considerando as maneiras de testar as diferenças entre humanos e máquinas, há diversos tipos de CAPTCHAs. Os *CAPTCHAs baseados em texto*, por exemplo, consistem em imagens contendo caracteres (letras, dígitos e símbolos) que devem ser identificados

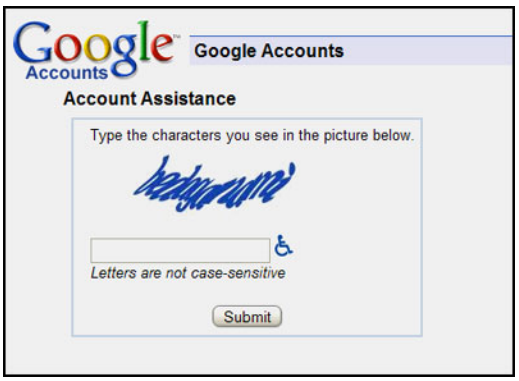
pelo usuário. Porém, esses caracteres são distorcidos de diversas formas a fim de não tornar óbvio o seu entendimento. Essa distorções podem ser ondulações, diminuição da qualidade da imagem, adição de ruído, inserção de planos de fundo confusos, entre outras. A quantidade e qualidade das distorções irão influenciar diretamente a dificuldade da resolução do CAPTCHA. Este tipo de CAPTCHA é ilustrado na Figura 1.

Figura 1: Exemplos de CAPTCHAs de texto. Fonte: (GOOGLE, 2018a)

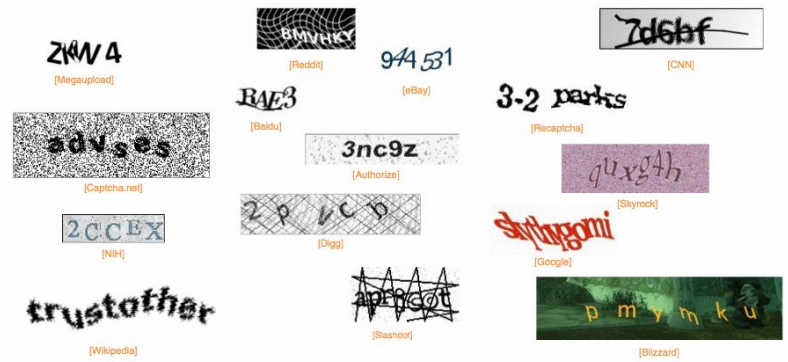
(a) CAPTCHA de texto considerado fácil.



(b) CAPTCHA de texto considerado difícil.



(c) Variações de CAPTCHAs de texto.



Os *CAPTCHAs baseados em imagens*, por sua vez, apresentam uma ou mais imagens ao usuário e, em seguida, solicitam a identificação de um padrão ou característica presente nas mesmas. Por exemplo, no CAPTCHA ilustrado na Figura 2, solicita-se a identificação de todas as partes da imagem que contêm placas de trânsito.

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

Considerando maneira não-triviais de identificar usuários humanos, os *CAPTCHAs baseados em questão* demandam a resolução de algum problema proposto. Os tipos mais comuns de problemas são de natureza matemática envolvendo operações aritméticas simples, mas podem conter questões de conhecimentos gerais. Este tipo de CAPTCHA é ilustrado na Figura 3.

Figura 2: CAPTCHA baseado em imagem. Fonte: (GOOGLE, 2018a)

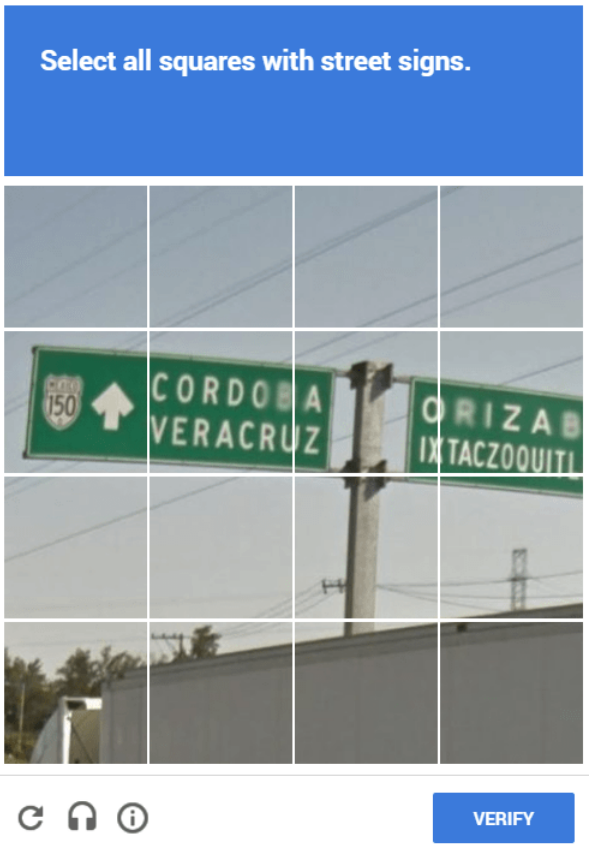
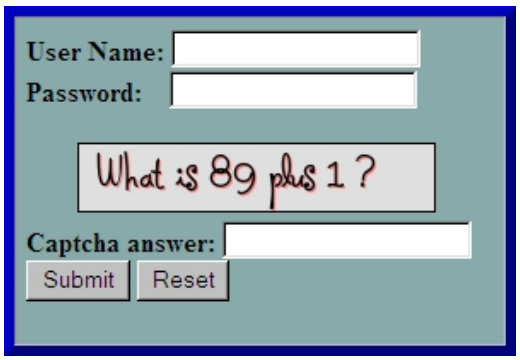
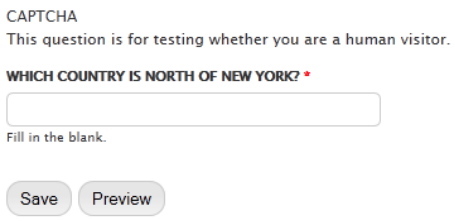


Figura 3: Exemplos de CAPTCHAs de questão. Fonte: (GOOGLE, 2018a)

(a) CAPTCHA de questão matemática.

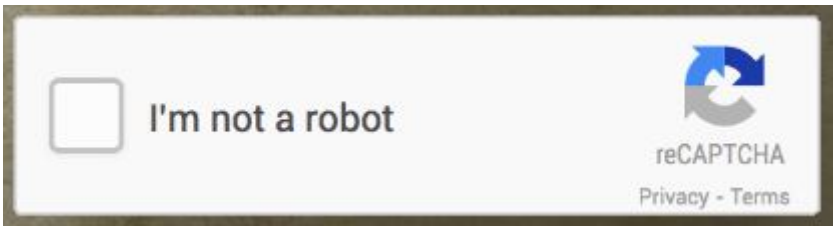


(b) CAPTCHA de questão geográfica.



O modelo *reCAPTCHA*, desenvolvido pela empresa Google, consegue classificar o usuário em humano ou máquina a partir de padrões de navegação antes do teste ser ativado, a exemplo de tempo gasto por página, quantidade e velocidade de cliques, histórico de navegação, dentre outros. Considerando estas informações, o teste é reduzido a um único clique, conforme ilustra a Figura 4, que valida a classificação feita anteriormente. Esta é uma tentativa de tornar os CAPTCHAs mais agradáveis aos humanos, mas mantendo uma boa detecção de *bots* e outros algoritmos similares.

Figura 4: Exemplo de caixa de diálogo de validação no reCAPTCHA. Fonte: (GOOGLE, 2018b)



2.2. Aprendizado de Máquina

Aprendizado de Máquina, ou *Machine Learning*, é a área da Computação responsável pelo estudo de algoritmos e sistemas que aprimoram seu conhecimento ou performance com base na experiência (FLACH, 2012). Também é possível definir como sistemas com a capacidade de modificar ou adaptar suas ações para que elas constantemente melhorem sua precisão (MARSLAND, 2015). Em outras palavras, um algoritmo aprende quando modifica suas ações para se adaptar à um cenário com a finalidade de exercer uma tarefa de forma cada vez mais próxima à um estado considerado otimizado, baseando-se em situações observadas no passado.

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

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

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

- **Aprendizado Supervisionado**. Neste paradigma, o aprendizado é guiado pelas respostas corretas associadas à cada exemplo, os chamados atributos alvos. Assim, este paradigma é preferível em cenários em que há uma quantidade substancial de exemplos com suas respectivas resoluções ideais;
- **Aprendizado Não Supervisionado**. Neste caso, não se objetiva prever ou descobrir uma relação do tipo entrada e saída, mas sim efetuar uma descrição dos dados a partir de suas características intrínsecas, a exemplo de similaridades;
- **Aprendizado por Reforço**. Está no limiar entre o aprendizado supervisionado e o não supervisionado. Neste paradigma, é informado quando a resposta está errada, mas não quando está correta. Assim, tenta-se uma abordagem diferente até que uma resposta aceitável seja produzida;
- **Aprendizado Evolucionário**. É baseado nas ideias de evolução biológica, na qual é feita a seleção de um ou mais configurações, que obtiveram os resultados mais

próximos do ideal. Em seguida, uma adaptação é efetuada, pois continuamente são inseridos fatores de aleatoriedade e soluções com desempenho ruim são eliminadas (FLACH, 2012).

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

2.3. Redes Neurais Artificiais

As *Rede Neurais Artificiais* (RNAs), inspiradas no cérebro humano, são sistemas paralelos distribuídos compostos por unidades de processamento simples, denominados *neurônios artificiais*, que calculam determinadas funções matemáticas (normalmente não lineares) (BRAGA; CARVALHO; LUDERMIR, 2007). As capacidades de aprender e generalizar são fatores que tornam as RNAs atrativas para a resolução de problemas, com aplicações em diversos domínios.

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

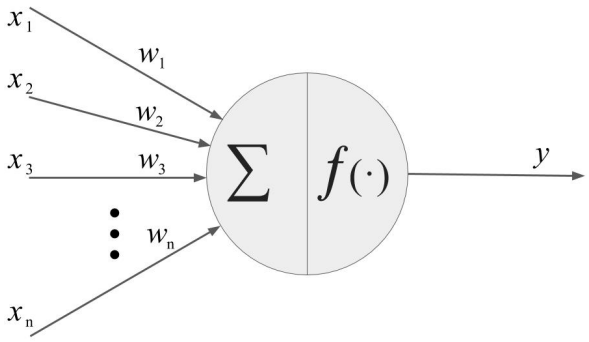
As seções a seguir detalham os conceitos relativos às RNAs utilizados, elementares para o desenvolvimento da solução proposta neste trabalho. Os conceitos relativos aos neurônios artificiais e a combinação destes em redes estruturalmente simples são apresentados na Seção 2.3.1. As RNAs Perceptron de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

2.3.1. Neurônio Artificial e Redes Perceptron

Em 1943, McCulloch e Pitts desenvolveram um modelo de simplificação do neurônio baseado no que se sabia até então a respeito dessa estrutura biológica. Esta descrição matemática resultou em um modelo, denominado *neurônio MCP* e ilustrado na figura 5, com n terminais de entrada, que representavam os dendritos de um neurônio biológico, recebendo os valores $\mathbf{X} = (x_1, x_2, ..., x_n)$, e produzindo apenas um valor de saída y, análogo ao axônio. Com o intuito de representar o comportamento das sinapses, foram atribuídos valores $\mathbf{W} = (w_1, w_2, ..., w_n)$ como pesos associados às entradas, podendo ser positivos ou negativos. O efeito de uma sinapse particular em um neurônio pós-sináptico é dada por $\sum_i x_i \cdot w_i$.

A função dos pesos em um neurônio artificial é de mensurar o quanto a entrada associada deve ser considerada para a ativação daquele neurônio. Dessa maneira, consi-

Figura 5: Modelo do neurônio de McCulloch e Pitts (BRAGA; CARVALHO; LUDERMIR, 2007)



derando as várias entradas, o neurônio processa essa informação por meio de uma soma ponderada:

$$u = \sum_{i=1}^{n} x_i \cdot w_i. \tag{1}$$

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

A função de ativação é responsável por determinar a saída y do neurônio a partir dos valores de entrada $\mathbf{X}$ e seus determinados pesos $\mathbf{W}$. No caso do MCP, a função de ativação é do tipo degrau deslocada do limiar de ativação θ em relação à origem, como mostrado na Equação 2.

$$f(u) = \begin{cases} 1, & \text{se } u \geq \theta, \\ 0, & \text{caso contrário.} \end{cases} \tag{2}$$

Além da função degrau deslocada, outras funções podem ser utilizadas na formulação de neurônios artificiais, desde que sejam diferenciáveis. As funções linear, sigmoidal e ReLu (retificada linear), juntamente com a função degrau, são ilustradas na Figura 6.

Fazendo uso do modelo de neurônios MCP, em 1958 foi publicado o modelo *perceptron*, composto de três camadas (retina, associação e resposta), dispostas conforme ilustrado na Figura 7. A camada retina era composta de unidades sensoras, aptas a receberem a informação do mundo externo; a camada de associação, por sua vez, era formada por neurônios MCP possuindo pesos fixos definidos previamente; e a camada de resposta, por sua vez, possuindo um neurônio MCP responsável por disponibilizar o processamento produzido pela rede para o mundo externo (ROSENBLATT, 1953).

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

Figura 6: Exemplos de funções que podem ser utilizadas como função de ativação.

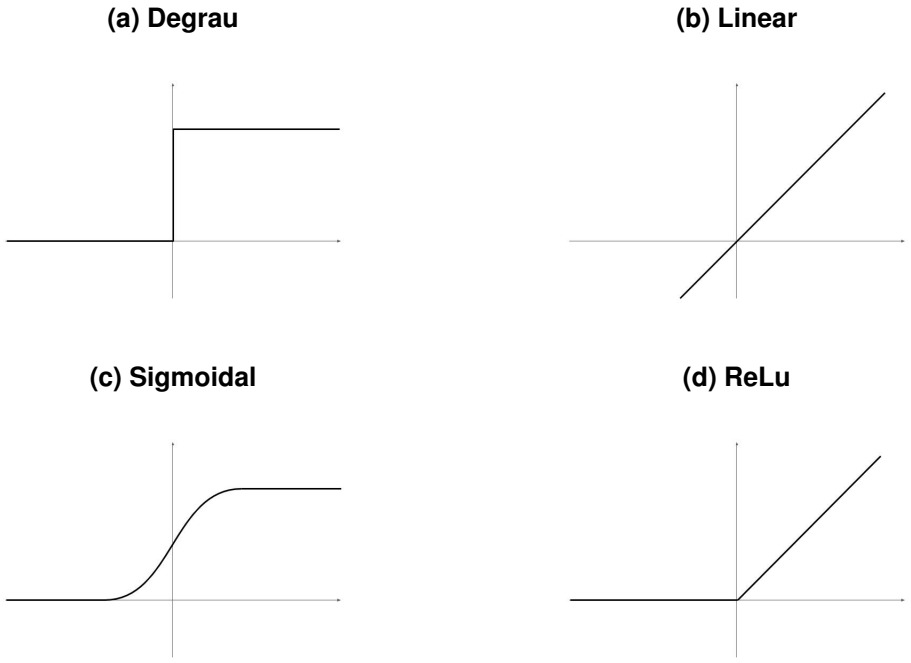
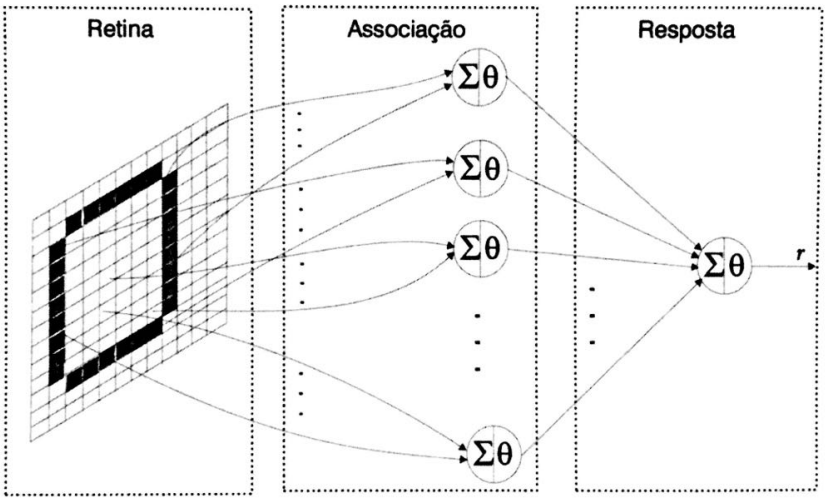


Figura 7: Modelo perceptron. Fonte: (BRAGA; CARVALHO; LUDERMIR, 2007)



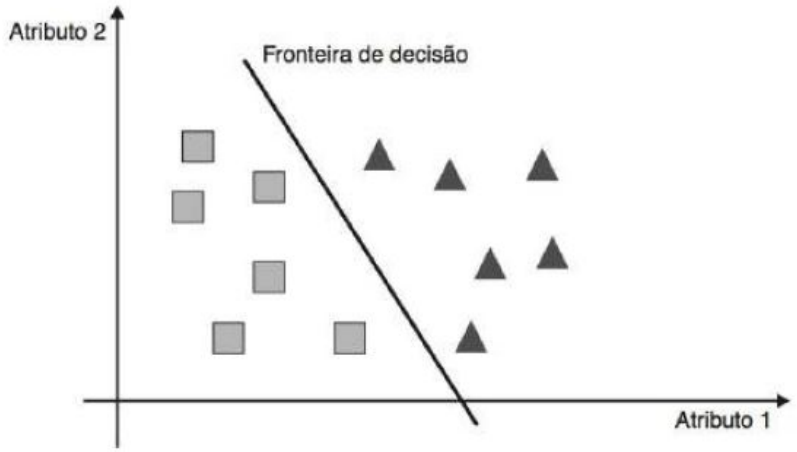
$$\mathbf{W}(t+1) = \mathbf{W}(t) + \Delta\mathbf{W}(t). \tag{3}$$

A diferença $\Delta\mathbf{W}(t)$ é proporcional a um valor η, denominado *taxa de aprendizado*. Esta taxa está intimamente ligada ao tempo necessário para a convergência da rede. Caso esse valor seja muito pequeno, será necessária uma grande quantidade de ciclos para obter uma solução aceitável. Por outro lado, um valor muito grande pode provocar oscilações que dificultam a convergência. Uma possível medida para amenizar esse problema é a introdução do termo *momentum* α que tem a função de quantificar o grau de importância da variação de peso do ciclo anterior em relação ao ciclo atual. Esses parâmetros são

importantes no ajuste do processo de aprendizado do modelo em questão.

Apesar do modelo perceptron ser dividido em três camadas, ele é conhecido como *modelo perceptron de camada única*, justamente por apresentar propriedades adaptativas somente na camada de saída. Isso restringe sua capacidade de solução à um único tipo de problema, aqueles linearmente separáveis (FACELI et al., 2015). Neste tipo de problema, uma reta é capaz de distinguir entre objetos de duas classes distintas, conforme ilustrado na Figura 8.

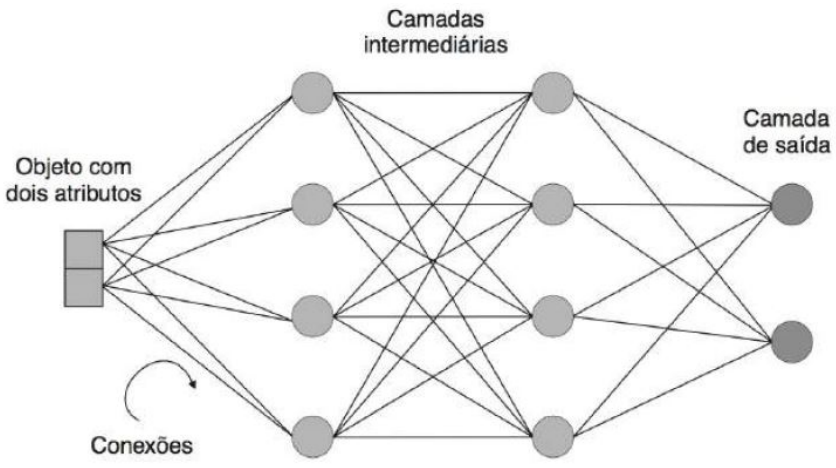
Figura 8: Objetos lineramente separáveis (FACELI et al., 2015)



2.3.2. Redes Perceptron de Múltiplas Camadas

O escopo dos problemas linearmente separáveis é pequeno e, para o tratamento de problemas com um perfil mais complexo, mais passíveis de ocorrência em cenários reais, faz-se necessária a adição de mais camadas de neurônios às redes neurais artificiais (FACELI et al., 2015). Essas camadas, denominadas *camadas ocultas* ou intermediárias, localizam-se entre as camadas de entrada e de saída, conforme mostrado na Figura 9.

Figura 9: Exemplo de rede neural artificial com camadas ocultas. Fonte: (FACELI et al., 2015).

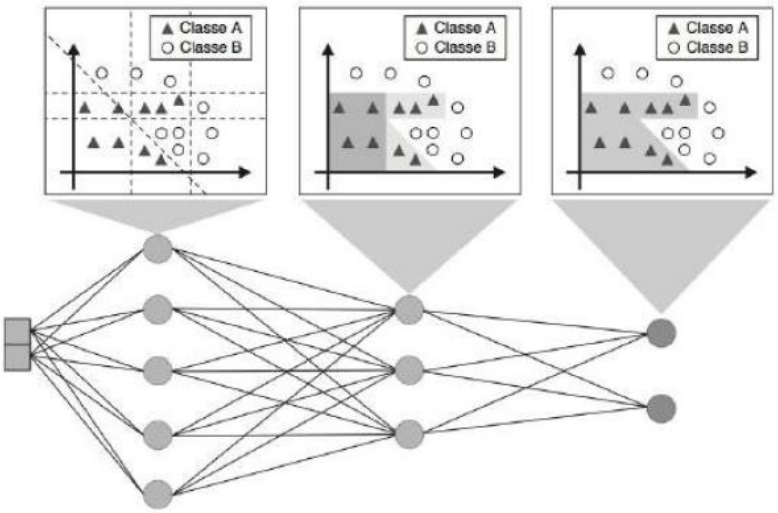


As camadas de uma RNA são conectadas de forma que os valores gerados na saída de um neurônio sejam aplicados como valor de entrada de um neurônio da próxima camada. Com isso, existem três padrões de conexões para uma RNA de múltiplas camadas: a *completamente conectada*, a *parcialmente conectada* e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

- Uma rede **completamente conectada** tem como característica o fato de que as saídas de todos neurônios de uma camada ℓ estão conectados às entradas de todos os neurônios da próxima camada $\ell + 1$;
- Em uma rede **parcialmente conectada** tem-se que apenas alguns neurônios de uma camada ℓ estão conectados à apenas alguns da próxima camada $\ell + 1$;
- Uma rede **localmente conectada** é uma rede parcialmente conectada, porém suas conexões entre camadas são feitas em áreas bem definidas.

Da-se o nome de *Rede Neural Perceptron de Múltiplas Camadas* (MLP, do inglês *Multilayer Perceptron*) à rede neural artificial completamente conectada e composta de pelo menos uma camada oculta contendo neurônios com funções de ativação sigmoidal (BRAGA; CARVALHO; LUDERMIR, 2007). Graças à existência das camadas ocultas, este tipo de RNA é capaz de resolver problemas não-linearmente separáveis, da seguinte forma: na primeira camada, cada neurônio é responsável pelo aprendizado de uma função linearmente separável, definindo hiperplanos; os neurônios da camada posterior combinam os resultados recebidos como entrada e passam a ser capazes de reconhecer regiões convexas; a partir daí, as camadas seguintes são capazes de reconhecer regiões de formato arbitrário (FACELI et al., 2015), conforme apresentado na Figura 10.

Figura 10: Papel desempenhado pelos neurônios nas diferentes camadas de uma rede MLP. Fonte: (FACELI et al., 2015).



No processo de aprendizado supervisionado de uma rede neural é necessário efetuar o ajuste de pesos em decorrência do erro, isto é, da diferença entre a saída produzida e a saída desejada. No caso de uma RNA de camada única, este erro é obtido de maneira trivial. Com este valor do erro, pode-se então fazer ajustes no vetor de pesos $\mathbf{W}$, conforme descrito anteriormente na Equação 3. Contudo, em uma RNA MLP, esse processo só pode ser aplicado à última camada, visto que não existem saídas desejadas definidas para as camadas ocultas (BRAGA; CARVALHO; LUDERMIR, 2007).

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

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

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

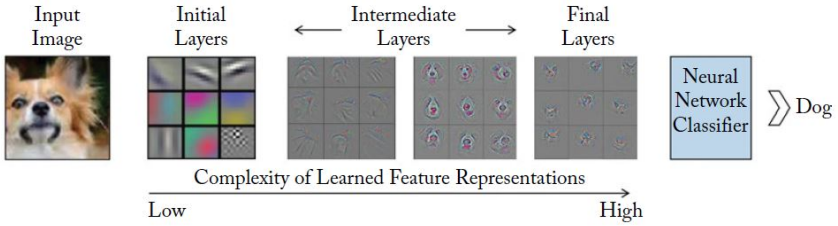
Considerando a capacidade de endereçar problemas não-linearmente separáveis, as RNAs MLP são amplamente utilizadas em diversos domínios. Previsão do valor de ações no mercado financeiro, reconhecimento de fonemas em sinais de voz e identificação de assinaturas manuscritas são apenas alguns exemplos dentre um vasto corpo de soluções utilizando este modelo de Aprendizado de Máquina (BRAGA; CARVALHO; LUDERMIR, 2007).

2.3.3. Redes Neurais Convolucionais

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

A diferença entre uma CNN e uma RNA convencional é que cada unidade em uma determinada camada é um filtro de duas ou mais dimensões. Esses filtros incorporam um contexto espacial, mas menor, da mídia de entrada e usam compartilhamento de parâmetros para reduzir significativamente o número de possíveis variáveis a serem processadas (KHAN et al., 2018). Em outras palavras, cada neurônio responde à pequenos campos receptivos da imagem relacionados a sua posição espacial e com sobreposição dos campos receptivos de seus vizinhos. A partir desses filtros, diferentes mapas são produzidos, variando de acordo com o posição e níveis de abstração que acontecem com o aprofundamento dos níveis de neurônios, conforme ilustrado na Figura 11.

Figura 11: Processo em que uma CNN inicia aprendendo características simples e a medida que aprofunda suas camadas, torna o padrão mais completo, por fim faz a classificação (KHAN et al., 2018).

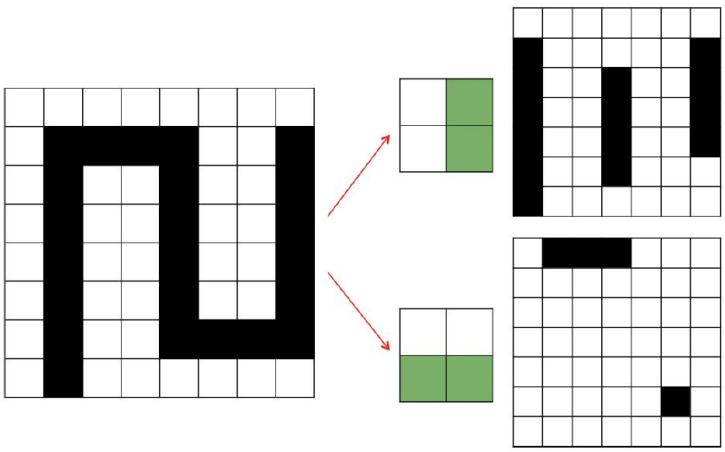


Uma CNN é basicamente composta por tipos específicos de camadas, são elas: camadas convolucionais, camadas de *pooling*, camadas de ativação, camadas completamente conectadas e, por fim, uma camada de saída.

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

Na Figura 12 a realização da operação de convolução é ilustrada em detalhes. A imagem à esquerda é fornecida como entrada, a qual será sujeita à convolução por dois filtros distintos. Os filtros possuem dimensão 2×2 e contém apenas valores binários, em que os pixels na cor verde correspondem ao valor 1. O resultado da convolução da entrada com cada um dos filtros é mostrada mais à direita, onde percebe-se o mapa de características que foi produzido como resposta. É importante salientar que os mapas produzidos possuem dimensões menores que da imagem original (BUDUMA, 2017).

Figura 12: Exemplo de convolução aplicada a uma entrada. Fonte: (BUDUMA, 2017).



Ter um número grande de filtros pequenos é uma boa forma de conseguir uma representação fiel das características da imagem com uma quantidade pequena de parâmetros. Também é sugerido que os filtros se movam apenas um pixel por vez (parâmetro *stride* igual a 1) sem usar nenhum tipo de preenchimento (parâmetro *padding* igual a 0), para que o mapa de saída mantenha a proporção da imagem original, mas com um número

de parâmetros significativamente menor (BUDUMA, 2017).

A *Camada de Pooling* recebe como entrada os mapas de características tipicamente gerados pelas camadas de convolução que a antecedem. Nela, são aplicadas funções com o objetivo de reduzir drasticamente as dimensões dos mapas de características recebidos como entrada, a fim de torná-los compactos e invariantes perante pequenas mudanças da imagem de entrada. Essas funções variam de acordo com a aplicação, mas a função *Max Pooling*, por exemplo, gera como saída um mapa com os valores mais altos encontrados em cada iteração da varredura. Já a *Average Pooling*, por sua vez, faz o mesmo que a *Max Pooling*, mas com a média dos valores (KHAN et al., 2018). A *Camada de Ativação* recebe um valor de entrada e transforma em uma saída com uma extensão pré definida, a exemplo do intervalo $[0, 1]$. O papel dessa camada é atuar como um mecanismo que decide quando e o quanto um neurônio deve ser disparado, dada determinada entrada. Assim como nas MLPs, as funções dessa camada devem ser contínuas e diferenciáveis, como a sigmoidal ou a ReLu, mostradas anteriormente na Figura 6.

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

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

Tendo em vista as camadas apresentadas, pode-se organizá-las de diferentes maneiras com vistas a compor a arquitetura da CNN desejada, desde mais simples até mais complexa. Algumas arquiteturas de CNNs são ilustradas na Figura 13.

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

A arquitetura *LeNet* é uma das primeiras e mais básicas arquiteturas de CNNs aplicada na identificação de imagens contendo dígitos escritos à mão. Uma versão dessa arquitetura que apresenta bons resultados neste problema é a *LeNet-5* que é composta por dois blocos, cada um contendo uma camada convolucional seguida de uma camada de *Max Pooling*. Ao final destes dois blocos, segue-se uma camada convolucional e duas camadas completamente conectadas que agem como o classificador das características. Na Figura 14, é mostrada uma CNN LeNet-5 sendo usada para reconhecimento de um dígito manuscrito.

O processo de treinamento de uma CNN começa com a atribuição de pesos aleatórios aos neurônios que a compõem. Uma restrição importante nesta atribuição inicial é que os pesos não sejam iniciados todos iguais à zero, pois isto dificultará o ajuste de parâmetros necessário ao aprendizado. No mais, a apresentação de exemplos, estimativa de erro e ajuste de parâmetros é feito de maneira análoga às MLPs, por meio do algo-

Figura 13: Exemplos de arquiteturas de CNN variando em sua complexidade. O último exemplo consiste em uma arquitetura VGG. Fonte: (BUDUMA, 2017).

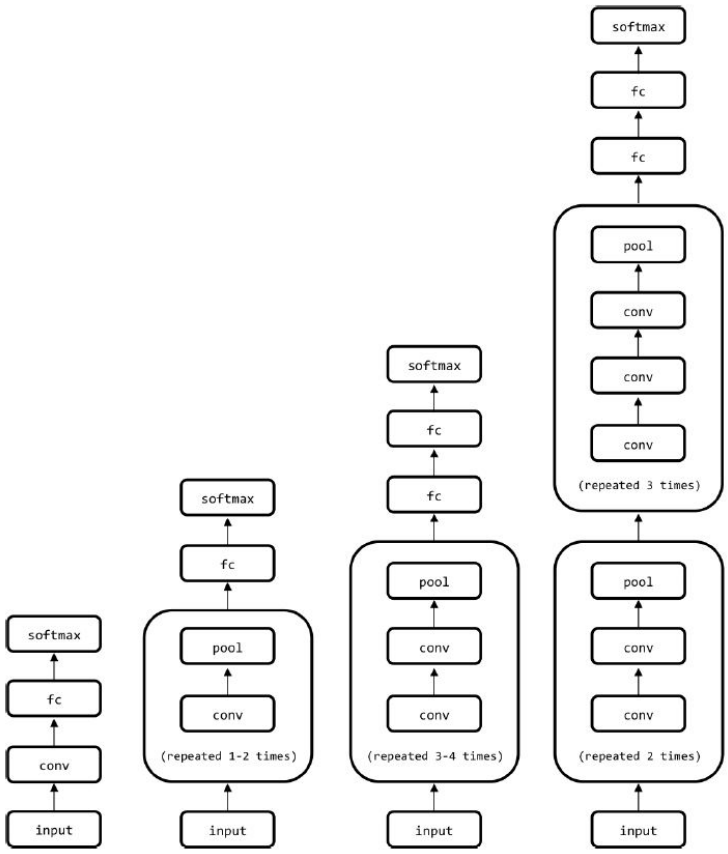
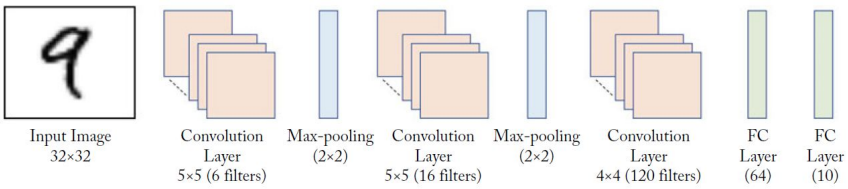


Figura 14: Arquitetura do modelo LeNet-5 sendo usada em um digito do dataset MNIST (KHAN et al., 2018).



ritmo *back-propagation*, apresentado anteriormente. Diversas técnicas de otimização de busca no gradiente descendente podem ser usadas neste algoritmo com vistas a acelerar o treinamento, como: diferenciação numérica, diferenciação analítica, entre outras.

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

Uma aplicação bastante consolidada e que retrata a importância prática das CNNs

diz respeito ao problema do reconhecimento de dígitos manuscritos. Neste problema, são dadas como entrada imagens monocromáticas de dimensão 28×28 *pixels*, contendo representações de dígitos escritos à mão. O objetivo do problema é classificar as representações encontradas na imagem, identificando o número ou letra correspondente ao que está apresentado. A base de dados mais conhecida para esta tarefa é o MNIST, contendo 60.000 exemplos de imagens para treinamento e 10.000 para os testes. O estado da arte desse problema, segundo Benenson (BENENSON, 2018), é alcançado por CNNs, com taxa de erro de $0,21\%$. Este resultado, juntamente com diversos outros disponíveis na literatura, reforçam a importância deste modelo de Aprendizado de Máquina e o seu potencial de uso em aplicações práticas.

2.4. Tecnologias Utilizadas

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

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

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

3. Solução Proposta

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

[1]<http://jupyter.org/>
[2]<http://scikit-learn.org>
[3]<http://keras.io>
[4]<https://opencv.org/>

3.1. Consolidação da Base de Dados

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

Primeiramente, ao acessar a URL <http://buscatextual.cnpq.br/buscatextual/visualizacv.do> foi possível identificar um *web service* provedor das imagens de CAPTCHA. Por meio da linguagem de programação Python, foi construído um script para consumo deste *web service* de maneira automática, o qual era responsável por fazer requisições sequenciais de conexão com o endereço provedor do serviço para realizar o download das imagens. O código-fonte deste script encontra-se na Figura 15.

Figura 15: Script para download dos CAPTCHAs da Plataforma Lattes.

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

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

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

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

keep_working = True
count = 0

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

Para possibilitar esta coleta, um cuidado particular se fez necessário na implementação do script: aumentar o espaçamento temporal entre duas requisições sequenciais, o

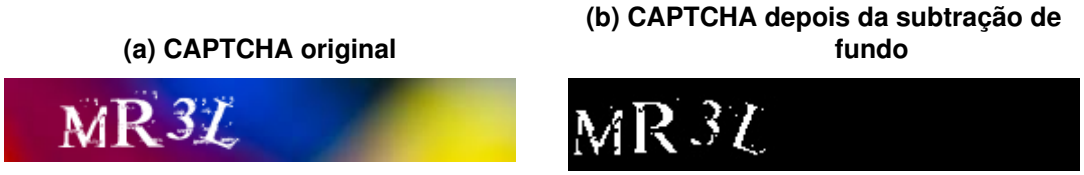
que culminaria em evitar uma grande quantidade de acessos rápidos, com vistas a não prejudicar o funcionamento do serviço e nem tampouco causar o bloqueio das requisições, que poderiam ser confundidas com um ataque do tipo *Deny of Service*. Ao ser executado, o script produzido também gerava um arquivo de *log*, registrando o histórico de solicitações e seus resultados, possibilitando análises detalhadas de tempo e recuperação em caso de eventuais problemas.

Como resultado da execução deste script durante cerca de 10 horas, foram coletados 6.000 CAPTCHAs de texto gerados pela Plataforma Lattes, tais como os ilustrados na Figura 16.

Figura 16: Exemplos de CAPTCHAs fornecidos pela Plataforma Lattes.

A partir de uma inspeção visual nas imagens coletadas, foi possível perceber uma característica em comum: toda a informação textual encontrava-se majoritariamente na cor branca, enquanto o fundo da imagem encontrava-se com padrões de cores variados. Isso ensejou a realização da subtração de fundo, tarefa de processamento digital de imagens com vista a eliminar informações irrelevantes. O processo de subtração de fundo, exemplificado na Figura 17, foi aplicado em todas as imagens previamente coletadas.

Figura 17: CAPTCHA antes e depois da subtração de fundo.



Após a etapa de subtração de fundo, foi verificado que alguns caracteres possuíam partes com linhas muito estreitas, que poderiam dificultar a identificação dos caracteres correspondentes. Para amenizar esse problema, foi utilizado um algoritmo de dilatação por iteração, disponível na biblioteca *OpenCV*. Esse algoritmo torna os elementos visuais de uma imagem mais espessos. De maneira análoga, todos os exemplos da base de dados foram submetidos à esse processamento, exemplificado na Figura 18.

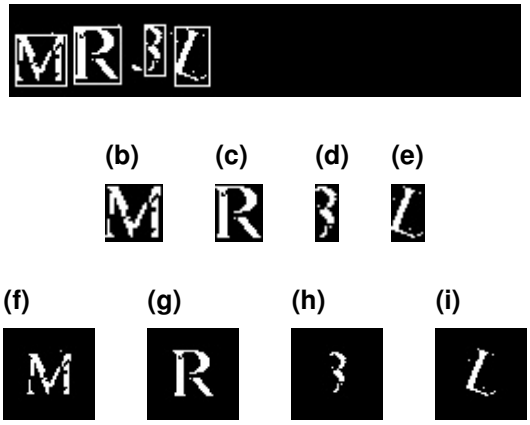
Também a partir da inspeção visual foi verificado que todos os CAPTCHAS consistiam na identificação de um número fixo de quatro caracteres. A partir disto foi consolidado o próximo passo, que tratou da separação dos caracteres individuais que compunham o CAPTCHA textual. Esse processo, exemplificado na Figura 19, também foi realizado com o auxílio da biblioteca *OpenCV*, por meio de um algoritmo de segmentação de estruturas. Por fim, foi acrescentado às imagens dos caracteres, bordas para torna-las com dimensões fixas.

Figura 18: CAPTCHA antes e depois da dilatação.

(a) CAPTCHA original **(b) CAPTCHA depois da dilatação**

Figura 19: Processo de segmentação do CAPTCHA.

(a) CAPTCHA com marcações onde foram encontradas as estruturas



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

3.2. Visão Geral da Base de Dados

Após a etapa anterior, foi obtida uma base de dados contendo 24 mil exemplos, que foram categorizados manualmente em pastas rotuladas de A a Z e de 0 a 9. Dentre o total de exemplos rotulados, observou-se $70,68\%$ correspondiam a caracteres alfabéticos e que o restante correspondia a caracteres numéricos, distribuídos conforme o histograma da Figura 20.

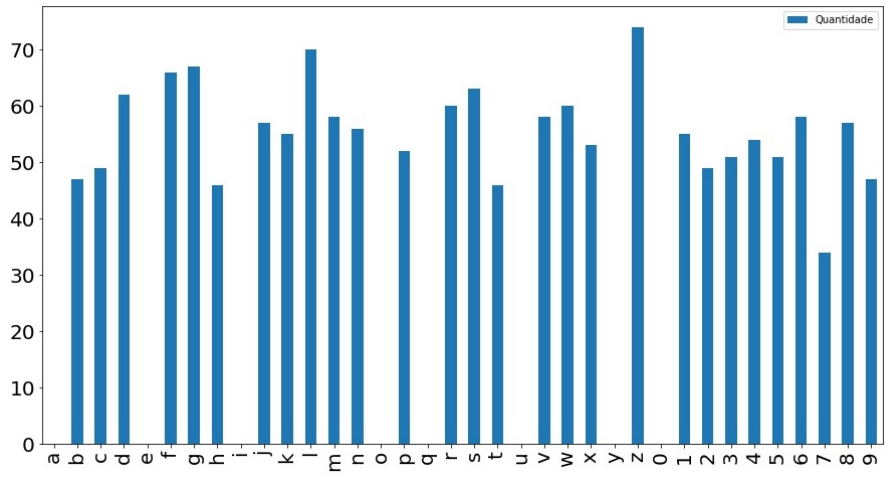
A partir desta análise da base de dados, foi possível perceber que determinados caracteres, tais como o 0 e as vogais, não ocorreram no total de amostras coletadas. Dentre os demais caracteres, observou-se uma ocorrência heterogênea, de modo que a base de dados pode ser vista como não balanceada.

3.3. Tarefa de Classificação e Métricas de Desempenho

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

Os 24 mil exemplos da base de dados serão particionados de maneira aleatória em dois conjuntos: conjunto de treinamento, com 70% das amostras; e conjunto de teste,

Figura 20: Histograma da quantidade de dígitos classificados.



com as demais amostras. O conjunto de treinamento, contendo as imagens e seus respectivos rótulos, serão utilizadas para ajuste de parâmetros e aprendizado de padrões pelos modelos, enquanto a outra fração dos dados será utilizada para aferir o poder de generalização. Estes procedimentos obedecem à técnica de *holdout* para validação cruzada (BRINK; RICHARDS; FETHEROLF, 2017).

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

3.4. Modelos Propostos

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

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

4. Trabalhos Relacionados

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

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

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

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

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

5. Considerações Parciais

Esta proposta de trabalho de conclusão de curso tem por objetivo endereçar o problema de reconhecimento automático do CAPTCHA de texto da Plataforma Lattes, colaborando para o livre acesso e transparência das informações a respeito dos pesquisadores brasileiros e seus trabalhos no âmbito acadêmico.

A solução proposta considera a perspectiva do Aprendizado de Máquina e a utilização de redes neurais convolucionais. Para treinar e testar modelos, foi primeiramente necessário consolidar uma base de dados. Esta foi feita a partir da aquisição automática de vários exemplos de CAPTCHAs e posterior processamento das imagens. A linguagem de programação Python foi essencial para a realização dessas tarefas, automatizando procedimentos e permitindo, por meio da biblioteca *OpenCV*, o processamento das estruturas gráficas dos caracteres componentes do CAPTCHA. Assim, ao final desta etapa, foi consolidada uma base de dados contendo cerca de 6 mil exemplos.

Para as etapas posteriores, já foi efetuada a fundamentação teórica e tecnológica das redes neurais convolucionais. Restando apenas as etapas de implementação, treino e teste dos modelos, considerando diferentes parâmetros. As métricas de desempenho obtidas serão comparadas e os modelos mais aptos para o problema proposto serão identificados.

O desenvolvimento deste trabalho de conclusão de curso tem proporcionado a aplicação de conhecimentos obtidos nas disciplinas da grade do curso de Sistemas de Informação, com vistas a ajudar na transparência e disseminação de informações relevantes para a socidades. Disciplinas como Linguagem de Programação e Inteligência Artificial foram cruciais no desenvolvimento do mesmo, entretando, conhecimentos específicos houveram de ser adquiridos de maneira independente, como, por exemplo, o funcionamento da API *OpenCV*.

Referências

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

BENENSON, R. *What is the class of this image?* 2018. <http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html>. Acessado em 15 de junho de 2018.

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

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

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

DELICATO, F. C.; PIRES, P. F.; SILVEIRA, I. F. *JAI - Jornadas de Atualização em informática*. 1. ed. [S.l.]: Sociedade Brasileira de Computação - SBC, 2017.

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

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

GOOGLE. *Google Imagens*. 2018. <https://images.google.com/>. Acessado em 15 de junho de 2018.

GOOGLE. *What is reCAPTCHA?* 2018. <https://developers.google.com/recaptcha/>. Acessado em 15 de junho de 2018.

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

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

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

KHAN, S. et al. *A Guide to Convolutional Neural Networks for Computer Vision*. [S.l.]: Morgan and Claypool, 2018.

KUBAT, M. *An Introduction to Machine Learning*. Estados Unidos: Springer, 2015.

MARSLAND, S. *Machine Learning - An algoritmic perspective*. 2. ed. Estados Unidos: CRC Press, 2015.

MATIAS, P. Resolução automática de captchas de voz. 2011.

PINTO, V. A. Redes neurais convolucionais de profundidade para reconhecimento de textos em imagens captcha. Florianópolis, Brasil, 2016.

ROSENBLATT, F. *The perceptron: a probabilistic model for information storage and organization in the brain*. [S.l.: s.n.], 1953.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.

SANTA-ROSA, J. G.; LIBERADO, J. R. G. V. Recurso de interface CAPTCHA: Segurança e usabilidade na distinção entre humano e máquina. *Ergodesign & HCI*, v. 1, p. 34–43, 2013.

SANTOS, V. D. dos. Análise de tecnologias de reconhecimento para quebra de captchas. 2016.

YAN, J.; AHMAD, A. S. E. Usability of captchas or usability issues in captcha design. Reino Unido, 2008.