DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Identificação de Textos em Imagens CAPTCHA utilizando conceitos de Aprendizado de Máquina e Redes Neurais Convolucionais

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação da disciplina:

DAS 5511: Projeto de Fim de Curso

Murilo Rodegheri Mendes dos Santos

Florianópolis, Julho de 2018

Identificação de Textos em Imagens CAPTCHA utilizando conceitos de Aprendizado de Máquina e Redes Neurais Convolucionais

Murilo Rodegheri Mendes dos Santos

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Marcelo Ricardo Stemmer

Banca Examinadora:

André Carvalho Bittencourt
Orientador na Empresa

Prof. Marcelo Ricardo Stemmer
Orientador no Curso

Prof. Ricardo José Rabelo
Responsável pela disciplina

Flávio Gabriel Oliveira Barbosa, Avaliador

Guilherme Espindola Winck, Debatedor

Ricardo Carvalho Frantz do Amaral, Debatedor

Agradecimentos

Agradeço à minha mãe Terezinha Rodegheri, ao meu pai Orlisses Mendes dos Santos e ao meu irmão Camilo Rodegheri Mendes dos Santos que sempre estiveram ao meu lado, tanto nos momentos de alegria quanto nos momentos de dificuldades, sempre me deram apoio, conselhos, suporte e nunca duvidaram da minha capacidade de alcançar meus objetivos.

Agradeço aos meus colegas Guilherme Cornelli, Leonardo Quaini, Matheus Ambrosi, Matheus Zardo, Roger Perin e Victor Petrassi por me acompanharem em toda a graduação, seja nas disciplinas, nos projetos, nas noites de estudo, nas atividades extracurriculares, nas festas, entre outros desafios enfrentados para chegar até aqui.

Agradeço aos meus amigos de infância Cássio Schmidt, Daniel Lock, Gabriel Streit, Gabriel Cervo, Guilherme Trevisan, Lucas Nyland por proporcionarem momentos de alegria mesmo a distância na maior parte da caminhada da graduação. Agradeço também de forma especial a minha amiga Sarah Gava que tive o prazer de conhecer, aprender e ter experiências incríveis e revigoradoras durante os momentos de tensão que a conclusão da graduação produz.

Agradeço a todos os mestres do curso de Engenharia de Controle e Automação que compartilharam seus conhecimentos em sala de aula, sempre estiveram dispostos a ajudar e contribuir para um melhor aprendizado e acompanharam a minha jornada enquanto universitário. Sou grato especialmente aos professores Hector Bessa Silveira, Julio Elias Normey Rico, Leandro Buss Becker, Ricardo José Rabelo, Werner Kraus Junior e Marcelo Ricardo Stemmer que foi meu orientador, me auxiliou nas pesquisas e revisou minha monografia.

Agradeço a Universidade Federal de Santa Catarina por proporcionar um ambiente saudável para todos os alunos, além de estimular a criatividade, a interação e a participação nas atividades acadêmicas. Sou grato a todo corpo docente, à direção e administração dessa instituição.

Agradeço à empresa Neoway, que proporcionou a realização deste projeto. Meu muito obrigado aos meus colegas de trabalho Alencar Hornung, Alexandre Albalustro, Claudinei Callegari, Iury Martins, Josué Machado, Leonardo Borges, Matheus Anversa, Paulo Jansen e ao meu orientador na empresa André Carvalho Bittencourt que acompanharam todas as etapas desse projeto, compreenderam a minha dedicação aos estudos e vibraram com a minha felicidade.

Resumo

Diversas aplicações na Internet possuem a política de manter público o acesso a alguns dados e informações. No entanto, para que isso seja possível, é necessário o desenvolvimento de uma aplicação ou um site eletrônico robusto para que os acessos, realizados por usuários, possam ser feitos sem onerar ou derrubar a fonte de dados. Nessas condições, surge o conceito de Big Data que consiste em dados produzidos e captados rapidamente, múltiplas fontes fornecedoras de dados, enorme quantidade de dados, dados e fontes confiáveis e verdadeiros e relevância dos dados para negócios. Do conceito de Big Data, surgiram empresas especializadas que começaram a realizar captação de dados em uma enorme velocidade e quantidade. Com essa necessidade de captação, surgem os Web Crawlers, Web Spiders e a automação da captura de dados. Os Crawlers e Spiders são programas desenvolvidos com a função de automatizar a consulta e adquirir dados de forma rápida, eficaz e repetitiva, fazendo muitas consultas em um pequeno intervalo de tempo. Com a repetitividade excessiva de consultas, a maioria dos sites eletrônicos adotam uma ferramenta para tentar bloquear/dificultar o funcionamento dos programas citados. Esta ferramenta é denominada CAPTCHA (teste de Turing público completamente automatizado para diferenciação entre computadores e humanos) e consiste em imagens contendo textos que devem ser reconhecidos e digitados pelo usuário para completar uma consulta de forma correta. Por fim, tem-se que o presente trabalho é uma implementação de soluções para reconhecimento de textos em imagens CAPTCHA utilizando-se de conceitos de processamento de imagem, aprendizado de máquina e redes neurais convolucionais.

Palavras-chave: Reconhecedor de Textos em Imagens; Redes Neurais Convolucionais; CAPTCHA

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SIT - Secretaria de Inspeção do Trabalho

CPF - Cadastro de Pessoas Físicas

CNPJ - Cadastro Nacional de Pessoas Jurídicas

Bot - Diminutivo de Robot (Robô)

CAPTCHA - Completely Automated Public Turing test to tell Computers and Humans Apart

CNN - Convolutional Neural Network

SVM - Support Vector Machine

ASCII - American Standard Code for Information Interchange

ReLU - Rectified Linear Unit

CRM - Customer Relationship Management

ERP - Enterprise Resource Planning

Sumário

1 Introdução

Não há dúvida de que as indústrias estão em chamas com a enorme erupção de dados. Nenhum dos setores permaneceu intocado por esta mudança drástica em uma década. A tecnologia penetrou dentro de cada área de negócios e tornou-se uma parte essencial deles. Falando especificamente sobre o setor de TI, software e automação são termos essenciais.

As empresas estão se concentrando mais na agilidade e na inovação do que na estabilidade, e a adoção das tecnologias em *Big Data* ajudam as empresas a conseguir isso em pouco tempo. A inserção de inteligência e ciência de dados em *Big Data* não apenas permitiu que as empresas permanecessem atualizadas com a dinâmica variável, mas também permitia que elas previssem as tendências futuras, dando uma vantagem competitiva.

Empresas focadas em *Big Data* possuem uma equipe de desenvolvimento e captura de dados que são essenciais para o funcionamento do conceito como um todo. O conceito de captação de dados existe há quase tanto tempo quanto o surgimento dos sites eletrônicos e tem como motivação ganhar sempre uma vantagem comercial fácil, ou seja, reduzir o preço promocional de um concorrente, roubar *leads*, sequestrar campanhas de marketing, redirecionar APIs e roubar conteúdo e dados.

No entanto, com o aumento da automação da captura de dados de sites eletrônicos surgem meios de validação de acessos humanos ou meios de dificultar e, até mesmo, bloquear os acessos feitos por *bots*. Uma das diversas formas de dificultar o acesso é a implementação de CAPTCHA (teste de Turing público completamente automatizado para diferenciação entre computadores e humanos) para validação do acesso, ou seja, uma imagem que contém um texto qualquer que deve ser interpretado pelo usuário e respondido corretamente para completar o acesso.

Para superar esta dificuldade e continuar com a automação da captura de dados existem técnicas e serviços que conseguem resolver o CAPTCHA. Quando trata-se de um serviço, fala-se de uma empresa que ao solucionar o CAPTCHA, cobra um valor para informar a resposta correta e quando fala-se de uma técnica, tem-se o desenvolvimento de um programa que dado um determinado CAPTCHA, é capaz de interpretá-lo e solucioná-lo. Na maioria dos casos, esse programa é desenvolvido internamente nas empresas e utiliza conceitos de processamento de imagem, aprendizado de máquina e redes neurais. Ressalta-se que estas aplicações não são contra a lei, como já citado até mesmo alguns sites ajudam a resolver CAPTCHAs com ou sem software.

Os conceitos de aprendizado de máquina e redes neurais surgiram em meados dos

anos 60 e formam um conjunto de fórmulas matemáticas e algoritmos que, atualmente, conseguem interpretar e solucionar desafios enfrentados por humanos, sendo o CAPTCHA um deles.

Este documento refere-se a disciplina de Projeto de Conclusão de Curso do Curso de Engenharia de Controle e Automação pela Universidade Federal de Santa Catarina e demonstra um CAPTCHA e os meios utilizados para solucioná-lo, primeiramente utilizando aprendizado de máquina através de SVM (máquina de vetores de suporte) e posteriormente, uma nova solução utilizando redes neurais convolucionais.

1.1 Problemas

Durante o dia a dia de trabalho da equipe de dados da Neoway enfrentam-se os desafios existentes na realização da captura de dados de sites eletrônicos. Dentre estes desafios, um deles é o CAPTCHA.

Para superar este desafio e poder realizar a captura são adotadas algumas práticas. Uma destas práticas é a utilização de serviços pagos que ao receberem um desafio CAPTCHA, conseguem interpretá-lo, solucioná-lo e enviar a resposta correta para a equipe. Outra prática se trata do desenvolvimento de um interpretador, o qual recebe a imagem CAPTCHA e, por meio de algoritmos computacionais, retorna a solução com a resposta correta, sem custo por solução.

No caso do desenvolvimento de um interpretador tem-se a necessidade da dedicação da equipe para que seja alcançado um resultado aceitável, o que consome tempo e gera custos.

O trabalho de conclusão de curso proposto tem a intenção de retratar os passos e desafios enfrentados no desenvolvimento de um interpretador de CAPTCHA, assim como sua eficiência, assertividade em ambientes de teste e produção, e melhorias desenvolvidas utilizando conceitos de redes neurais convolucionais.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um reconhecedor de textos em imagens CAPTCHA utilizando conceitos de aprendizado de máquina e redes neurais convolucionais com o intuito de diminuir custos relacionados a captura de dados públicos da fonte pública do ministério do trabalho.

1.2.2 Objetivos Específicos

- Compreender conceitos de aprendizado de máquina e redes neurais convolucionais para reconhecimento de imagens;
- Investigar estudos e artigos relacionados ao tema e compreender o cenário atual da arte;
- Adequar, treinar, validar e aprimorar um reconhecedor de textos em imagens CAPTCHA.

1.3 Escopo do Trabalho

O escopo deste trabalho engloba somente os estudos e análises do desenvolvimento de um reconhecedor de textos em uma imagem de CAPTCHA específica utilizando conceitos de aprendizado de máquina e redes neurais convolucionais.

1.4 Metodologia

A metodologia deste trabalho pode ser descrita na sequência a seguir:

- Estudar artigos, monografias e trabalhos realizados na área do projeto tratado e disponíveis em fontes públicas.
- Utilizar um material didático e uma ferramenta desenvolvida pela Neoway para solucionar o problema proposto e adquirir maior conhecimento no assunto e poder propor uma solução diferente.
- Contribuir para a evolução da tecnologia interna da Neoway implementando e validando o conceito de redes neurais para o desenvolvimento de soluções para problemas como o tratado neste projeto.

1.5 Estrutura do Trabalho

No intuito de fornecer uma melhor clareza dos conteúdos, optou-se pela divisão e ordenação do trabalho em 7 capítulos, os quais são numerados e dispostos a seguir.

O capítulo 1 aborda a introdução ao tema, os objetivos e explica-se a proposta.

O capítulo 2 trata da apresentação da empresa onde foi realizado o trabalho, contextualizando com o histórico, os produtos e as equipes envolvidas ou afetadas pelo projeto.

No capítulo 3 tem-se a fundamentação teórica, onde são explicitados os conceitos de aprendizado de máquina e redes neurais, assim como algumas abordagens voltadas ao reconhecimento de imagem.

O capítulo 4 contém a proposta de projeto. Neste capítulo são apresentados alguns pré-procedimentos necessários para o desenvolvimento do projeto e alguns resultados esperado.

No capítulo 5 está a etapa que caracteriza o desenvolvimento do projeto, ou seja, são apontados as etapas de pré-processamento da imagem CAPTCHA, arquitetura de rede, configuração da rede, treinamento, testes e a validação do reconhecedor.

No capítulo 6 são apresentados os resultados atingidos com o desenvolvimento do trabalho, assim como a importância do trabalho para a captura de dados dentro da empresa.

Finalizando no capítulo 7 com as conclusões e sugestões para trabalhos futuros aplicados a área.

2 Apresentação da Empresa

2.1 Histórico

A *N*eoway teve início no final de 2002, empresa basilar em um nicho de mercado que já não pode mais ser desconsiderado, soluções em Big Data. Sua logomarca é apresentada na Figura 1.

Enquanto concluía o doutorado na Universidade Federal de Santa Catarina (UFSC), o empreendedor, e hoje CEO da *N*eoway, Jaime de Paula experienciou uma consultoria que o fez ver uma necessidade específica do mercado. Tal necessidade fez com que a companhia focasse em Big Data, conceito que sistematiza e traz velocidade ao acesso, análise e armazenamento de grandes volumes de dados.

Figura 1 – Logomarca da *N*eoway.

2.2 Produtos [1]

Atualmente, a empresa conta com mais de 500 clientes no Brasil e com planos de expansão internacional, refletindo a qualidade de seu produto e o fato de que os seus serviços contribuem diretamente para o aumento do faturamento dos clientes. O SIMM, Sistema Sistema de Inteligência MultiMercado, é uma plataforma que possibilita o conhecimento profundo de clientes e fornecedores, atuando com precisão em áreas sensíveis como concessão de crédito, gestão de clientes e fornecedores, detecção de fraudes e cobrança, entre outras.

O sistema, acessado através de assinatura, foi recebido pelo mercado como uma solução que agrega inteligência de forma automática aos negócios. Dependendo da demanda e do volume de dados envolvidos, uma assinatura mensal pode variar entre R$ 1 mil a R$ 1 milhão.

O SIMM 2.0 é dividido em alguns módulos que contribuem de forma específica para a tomada de decisões dos clientes, sua divisão é dada por:

- SIMM Seach
- SIMM Maps
- SIMM Leads
- SIMM Pathfider

As logos dos módulos citados são apresentadas na Figura 2 de acordo com a ordem citada acima.

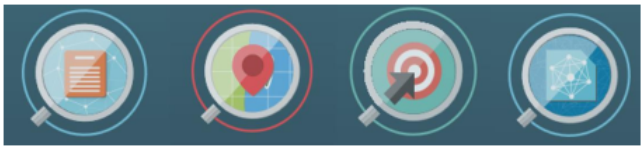

Figura 2 – Logos dos módulos do SIMM 2.0.

2.2.1 SIMM Search

O Search é o principal módulo do SIMM 2.0 e está ligada ao conceito de uma ferramenta de buscas, onde as consultas podem ser segmentadas e filtradas de acordo com o interesse do usuário. Sua base de dados é atualizada e contém mais de 34 milhões de empresas, 194 milhões de indivíduos, seus ativos, processos judiciais, tudo isso trazido em modelagens feitas por cientistas de dados utilizando mais de 7.000 variáveis de inteligência.

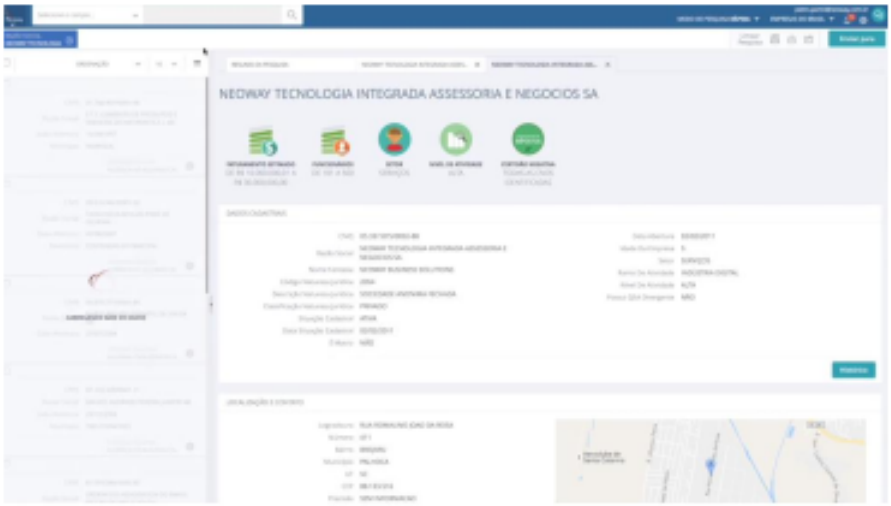

Figura 3 – Layout de busca do SIMM Search.

O Search ajuda os clientes a buscarem e conhecerem seus potenciais clientes de forma profunda e incomparável, fornecendo *"insights"* para tornar as decisões de negócios mais fáceis e assertivas. Sua representação inicial é apresentada na Figura 3.

Algumas informações importantes trazidas pela ferramenta Search são apresentadas a seguir, e referem-se a dados de empresas ou pessoas físicas:

- Dados profundos e de forma granular;
- Histórico da entidade;
- Faturamento;
- Certificados e regularidade;
- Participações societárias;
- Funcionários;
- entre outros.

Além disso, outra importante característica da ferramenta Search é a possibilidade de integração com os sistemas internos do usuário e criação de filtros conforme pode ser visto na Figura 4. Isso faz com que o Search seja um módulo complementar e não substituto nos atuais processos do usuário. Pode-se contar também com as seguintes características:

- Transportar para todas as soluções Neoway;
- Alimentar o CRM e ERP;
- Aplicativo Neoway mobile;
- Integrar com seus sistemas legados;
- Exportar os resultados para usar como preferir.

2.2.2 SIMM Maps

O Maps é uma solução que auxilia a tomada de decisões estratégias e de ida ao mercado, geo-localizando o mercado total, potencial trabalhável e o estabelecimento dos alvos, assim como seus concorrentes, região de atuação, infraestrutura entre outros. Essas características de mapeamento são ilustradas na Figura 5.

Algumas das características que fazem o MAPS uma ferramenta poderosa de negócio são citadas a seguir:

Figura 4 – Alguns dos filtros da ferramenta Search.

- Alto volume de dados gerando:
  - Inteligência de localização de ultra-volume;
  - Toda a informação útil para montar uma estratégia dividida em camadas;
  - Dados do mercado total, trabalhável e alvo;
  - Fusão de dados censitários, públicos e os seus;
  - Dados públicos de empresas, indivíduos, ativos, propriedades, processos, certidões, regularidade e muito mais.
- Planejamento e ação ao mesmo tempo:
  - Segmentações do total do mercado ao público alvo;
  - Variações de mercado ao longo do tempo;
  - Mapa de calor e outros indicadores importantes.

2.2.3 SIMM Leads

O Leads traz uma solução para planejamento e desenvolvimento de vendas para que o usuário possa encontrar seu próximo cliente, acelerando e aumentando a assertividade dos planos de ações e investimentos, conforme pode ser visto na Figura 6.

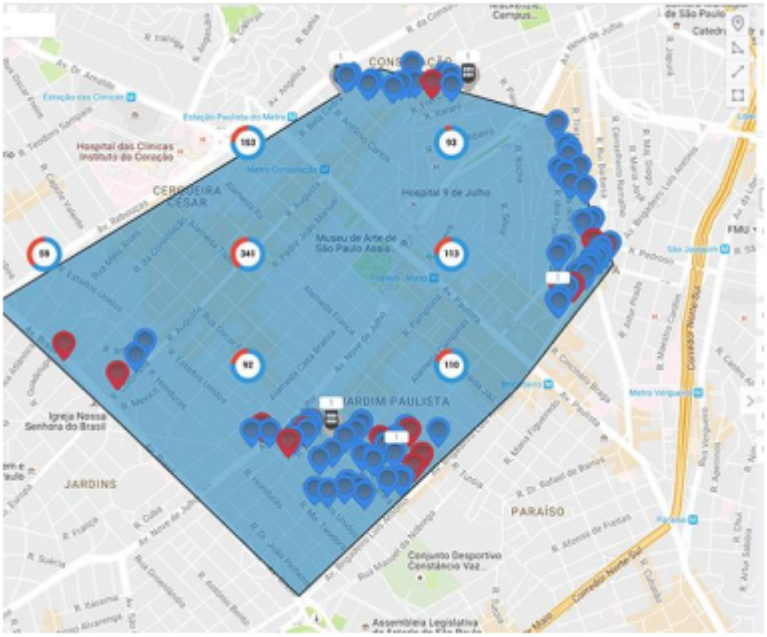

Figura 5 – Demonstração da área de interesse na ferramenta MAPS.

A ferramenta trabalha com aprendizado de máquinas e análise preditiva para descobrir novos clientes, priorizar demandas, gerar *insights* e alcançar resultados diretos e mensuráveis.

Além disso, o Leads possui as seguintes características que melhoram, ainda mais, o processo de negócio do usuário:

- Leads gera efetividade comercial;
- Possui integração à CRMs;
- Disponibilidade mobile;
- Auxilia na organização de tarefas e roteiros, entre outros.

2.2.4 SIMM PathFinder

O Pathfinder é, possivelmente, a ferramenta mais poderosa dentro do SIMM 2.0 e tem potencial enorme para crescimento. A ferramenta relaciona e mapeia os dados ligados a um elemento de busca, identificando riscos, oportunidades, indivíduos, grupos econômicos, empresas, indivíduos, entre outros de forma incomparável.

Como um todo, a ferramenta garante a prevenção de perdas ou aumento do potencial de ganhos, incluindo recuperação de ativos, análise de grupos e tudo que possa ser relacionado.

Figura 6 – Oportunidades na ferramenta LEADS

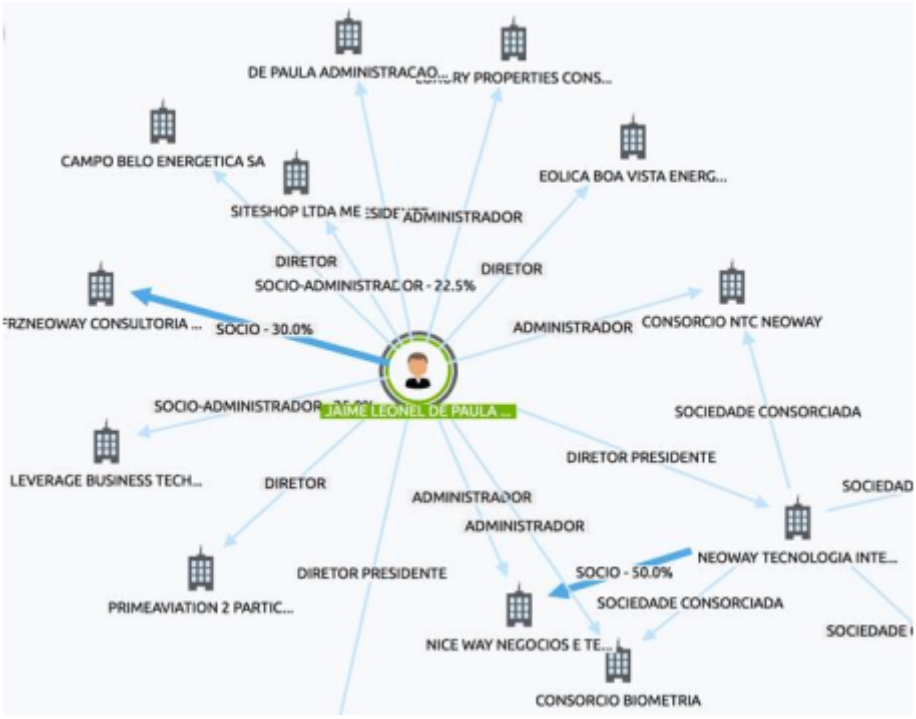

Figura 7 – Relacionamentos na ferramenta PathFinder.

A ferramenta ainda permite o conhecimento do beneficiário real final, como representado na Figura 7, através da soma de dados públicos, proporciona uma análise de lavagem de dinheiro, correlacionando informações com listas restritivas ou lista de atenção e entenda a proximidade com atividades de risco ou fraude.

2.3 Estrutura

O setor de tecnologia tem crescido muito em Florianópolis, que já conta com 900 empresas no setor, e a Neoway é uma dessas empresas, uma empresa multinacional do polo de tecnologia e que tem sua filial em Florianópolis, local onde tudo começou. A Figura 8

ilustra as instalações da empresa em Florianópolis.

Especializada em Big Data, isto é, informações para negócios a Neoway foi fundada por Jaime de Paula, 55 anos, manezinho, e acaba de abrir filial em Nova York.

Aberta em novembro de 2016, a filial de Nova York está focada no mercado financeiro americano, buscando soluções de compliance e conhecimento para o cliente. Dois diretores trabalham por lá, junto com parceiros estratégicos que buscam vender o produto e a plataforma da Neoway.

Outra filial está aberta em San Francisco, na Califórnia, desde 2014. Está foi iniciada após o recebimento de um aporte de um total de de receber os aportes de US 18 milhões de fundos de capital. O objetivo dessa filiar era deixar o pessoal da empresa mais conectado com o que está acontecendo de novidades no Silicon Valley.

Atualmente, a Neoway avalia a possibilidade de entrar nos mercados do México, Colômbia e Portugal. Vale ressaltar que recentemente foi realizada uma parceria com a Microsoft para usar a solução de cloud, a Azure. Em números, a Neoway é:

- Atividade: Inteligência de mercado e Big Data
- Colaboradores: 300
- Sedes: Florianópolis, São Paulo e Nova York
- Processa 50 bilhões de dados e informações
- Pesquisa 34 milhões de empresas no Brasil
- Tem em cadastro 194 milhões de pessoas

2.4 Contextualização das Áreas de Trabalho

A Neoway possui um enorme sistema que está constantemente rastreando a Web e capturando dados úteis sobre empresas e pessoas. Esses dados são usados para ajudar os clientes a encontrar novos mercados e expandir seus negócios. Este sistema também ajuda os clientes na detecção de fraudes, melhorando o lucro e reduzindo a perda.

Para que tudo isso aconteça, é preciso de um grande conjunto de pessoas e times. Alguns times presentes na Neoway são citados a seguir.

2.4.1 Data Source

A equipe de Data Source é composta por 6 pessoas e trabalha de maneira integrada com a equipe de Data Integration, onde a primeira é responsável pela captura e formatação

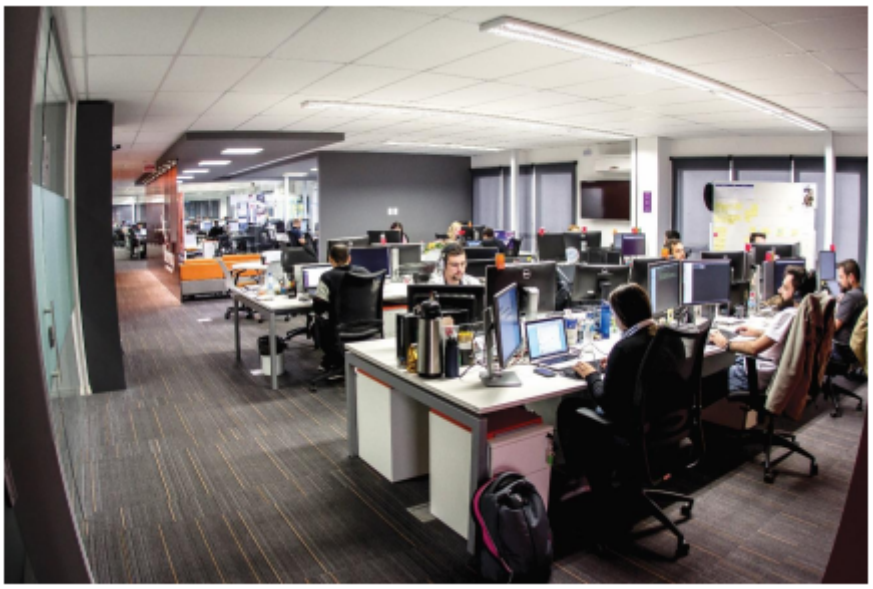

Figura 8 – Sede da Neoway em Florianópolis.

primária dos dados, enquanto a segunda tem o objetivo de fornecer as entradas necessárias para a realização da captura e integração dos dados capturados com o banco de dados.

Os desafios encontrados na rotina da equipe de Data Source são expostos abaixo:

- Captura de dados Web;
- Soluções de Proxy;
- Quebrar CAPTCHA;
- Limpeza de dados.

2.4.2 Data Integration

A equipe Data Integration é responsável por agregar e correlacionar os dados capturados pela equipe de Data Source, monitorar e fornecer dados de entradas para o correto funcionamento dos *bots* de captura de dados e garantir a integridade dos dados capturados para que possam ser inseridos nos bancos de dados para, posteriormente, serem relacionados e utilizados na construção de perfis.

2.4.3 Data Science

Outra equipe que participa ativamente do fluxo de tratamento dos dados é a equipe de Data Science. Ela é responsável pela criação de previsões com base em dados históricos, criação de modelos baseados em dados correlacionados, ou seja, de forma geral, é a equipe responsável por agregar inteligência nos dados tratados.

2.5 Importância do Projeto para a Neoway

A Neoway busca resolver problemas reais de empresas que precisam vender mais ou perder menos. Para isso, utiliza-se Big Data como uma ferramenta de descoberta de oportunidades mercadológicas, tornando o grande volume de dados facilmente navegável, as análises intuitivas e as decisões mais seguras.

Esse grande volume de dados é armazenado em mais de 3.000 bancos de dados e é coletado de mais de 600 fontes diferentes e organizadas em uma PaaS (Plataform as a Service ou Plataforma por Assinatura), que compila, organiza, e oferece uma interface prática e segmentada em aplicativos para o usuário navegar e extrair o máximo das informações sem precisar de grande conhecimento em tecnologia.

Portanto, capturar dados, limpá-los, adequá-los, garantir a integridade deles e depois organizá-los na plataforma são atividades realizadas diariamente dentro da Neoway. Só na parte de captura de dados, são mais de 600 fontes que precisam ser monitoradas e mantidas para que a captura se dê de forma contínua.

A grande maioria destas fontes possuem medidas de segurança, conhecidas como CAPTCHA, para evitar a automação da captura. Para que os dados sejam capturados continuamente, estas medidas de segurança devem ser vencidas pela equipe de Data Source, ou seja, os CAPTCHA devem ser quebrados.

Existem atualmente duas formas de vencer o desafio do CAPTCHA na Neoway, uma delas é através da compra de um serviço privado, o qual gera um custo de acordo com a quantidade de imagens CAPTCHA quebrados, e a outra é desenvolver internamente um reconhecedor de imagens CAPTCHA interno que, dado uma assertividade satisfatória, reduz largamente os custos de captura.

3 Fundamentação Teórica

Em um primeiro plano, é necessária uma revisão da literatura disponível acerca dos principais conceitos envolvidos durante o desenvolvimento do projeto a fim de garantir uma melhor compreensão do mesmo.

3.1 Aprendizado de Máquina (ML - Machine Learning)

O mundo está cheio de dados, entre eles: imagens, músicas, textos, planilhas, vídeos, que são gerados não só por pessoas, mas também por computadores, telefones e outros dispositivos eletrônicos. E não parece que isso vai mudar ou diminuir com o passar do tempo.

Os humanos possuem uma certa capacidade de interpretar estes dados e produzir informações a partir deles, no entanto, quando a quantidade de dados se torna gigantesca, essa capacidade se vê limitada e insuficiente para agregar tudo e, por isso, buscam-se sistemas automatizados que conseguem aprender com os dados e com suas mudanças, adaptando-se conforme a necessidade.

O aprendizado de máquina surge com a promessa de trazer sentido para toda essa gama dados, utilizando-se de ferramentas e tecnologia que buscam responder perguntas e gerar *insights* através do consumo destes dados. O consumo dos dados pelos algoritmos de aprendizado de máquina pode ser chamado de etapa de treinamento, enquanto a geração de respostas e *insights* é chamada de etapa de predição.

O treinamento refere-se a utilização dos dados para criação e parametrização de um modelo preditivo, o qual pode ser utilizado gerar respostas e previsões de resultados para novos dados. Porém, para alcançar um treinamento correto e um resultado preditivo satisfatório, é necessário passar por etapas de aquisição e tratamento de dados. A aquisição de dados é feita através da observação e anotação das informações produzidas por atividades reais ou virtuais de maneira geral. Após adquiridos todos os dados possíveis, deve-se fazer um tratamento nas informações para selecionar quais realmente são importantes para a geração das respostas desejadas. A maior parte do esforço está na etapa de tratamento de informações.

Cada vez mais empresas buscam implementar os conceitos de aprendizado de máquina em suas aplicações e produtos para personalizá-los de maneira automática a cada usuário, fornecendo uma experiência única e voltada aos gostos do cliente. Recomendação de vídeos, identificação de rostos em fotos, pesquisas no Google, detecção de câncer de pele, entre outros são algumas aplicações muito estudadas atualmente.

3.1.1 Aprendizado Supervisionado

No aprendizado supervisionado são utilizados dados rotulados, ou seja, dados corretos para realizar o treinamento de um modelo. Desta forma, esse método se torna muito eficiente, pois o sistema pode trabalhar diretamente com informações corretas.

A utilização de dados rotulados é o conceito mais importante dentro do aprendizado supervisionado, quando os rótulos são contínuos caracteriza-se um problema de regressão e quando os rótulos forem discretos tem-se um problema de classificação.

Para ilustrar, consideramos um problema de classificação em que deve-se predizer o gênero de um indivíduo através de informações dadas de pesos e alturas. Neste exemplo, temos que o gênero é o nosso rótulo e os *inputs* são o peso e a altura. Com os dados já rotulados a priori, dados de peso e altura rotulados com o gênero do indivíduo, e com aprendizado de máquina tem-se que a predição será feita baseada na semelhança dos dados rotulados iniciais.

Em suma, o aprendizado de máquina supervisionado faz o treinamento de um modelo através do histórico dos dados rotulados e realiza a predição de um dado ainda não rotulado apenas com as informações de entrada, que seriam peso e altura para o exemplo ilustrado.

3.1.2 Aprendizado Não Supervisionado

Para o aprendizado não supervisionado tem-se a diferença de que os dados usados para treinamento não possuem rótulos, ou seja, não são fornecidas saídas para as entradas das funções de aprendizado, o que causa incerteza sobre a saída do modelo treinado.

Ilustrando este caso, considera-se um problema de agrupamento ou *clustering*. Para dados de entrada temos os pesos e alturas de cães de raças desconhecidas e, novamente, não temos os rótulos das raças para os dados pesos e alturas. Neste caso, o aprendizado de máquina tentará agrupar os dados similares e a tarefa de classificá-los será do cientista de dados. Essa tarefa de classificação depende muito do conhecimento do cientista na área que está sendo realizado o estudo e o treinamento do modelo.

3.1.3 Aprendizado por Reforço

No aprendizado por reforço o modelo é gerado através de recompensas ou reforços de suas predições. Essas recompensas e reforços podem ser classificados como bons ou ruins e determinam o resultado final.

Os três principais componentes deste aprendizado são: o agente, o ambiente e estrutura de agentes. O agente é definido por aquele que consegue interagir com o ambiente a sua volta a partir de sensores e atuadores [3]. O ambiente pode ser definido como um

conjunto de propriedades, que possui o objetivo de influenciar diretamente nas ações do agente [4]. E a estrutura de agentes é o que os agentes podem fazer.

O aprendizado por reforço é amplamente utilizado para aplicações em que computadores são treinados para jogar videogames, carros autônomos, entre outros.

3.2 Máquinas de Vetor de Suporte (SVM)

No aprendizado de máquina, as máquinas de vetores de suporte (SVMs) são modelos de aprendizado supervisionados associados com algoritmos de aprendizado que analisam os dados usados para classificação e regressão [5].

O algoritmo SVM original foi inventado por Vladimir N. Vapnik e Alexey Ya. Chervonenkis em 1963. Em 1992, Bernhard E. Boser, Isabelle M. Guyon e Vladimir N. Vapnik sugeriram uma maneira de criar classificadores não-lineares aplicando o truque do kernel em hiperplanos de margem máxima [6].

A classificação de dados é uma tarefa comum no aprendizado de máquina. Uma técnica de SVM constrói um hiperplano ou um conjunto de hiperplanos em um plano de alta ou infinita dimensão e pode ser usada para classificação, regressão ou outras tarefas.

Basicamente o funcionamento de uma SVM pode ser descrito da seguinte forma: dadas duas classes e um conjunto de pontos que pertencem a essas classes, uma SVM determina o hiperplano que separa os pontos de forma a colocar o maior número de pontos da mesma classe do mesmo lado, enquanto maximiza a distância de cada classe a esse hiperplano. A distância de uma classe a um hiperplano é a menor distância entre ele e os pontos dessa classe e é chamada de margem de separação. O hiperplano gerado pela SVM é determinado por um subconjunto dos pontos das duas classes, chamados de vetor de suporte [7].

De uma forma prática, o SVM pode ser explicado através de um exemplo de separação binária entre dois grupos de dados. Na Figura 9, os grupos de dados estão dispostos em um plano onde um grupo é representado por círculos, enquanto o outro por quadrados.

O objetivo do exemplo é traçar o melhor hiperplano que classifique todos os vetores de treinamento em duas classes. Em outras palavras, uma boa separação é obtida pelo hiperplano que tem a maior distância (maior margem) até o ponto de dados de treinamento mais próximo de qualquer classe. Uma margem maior induz menor erro de generalização do classificador [8]. Na Figura 10 pode-se observar que o melhor hiperplano é o z2, uma vez que ele possui a maior margem até os dados.

Com isso podemos definir o melhor hiperplano para realizar a classificação dos dados e diminuir problemas comuns aos algoritmos de SVM, como outliers e exemplos

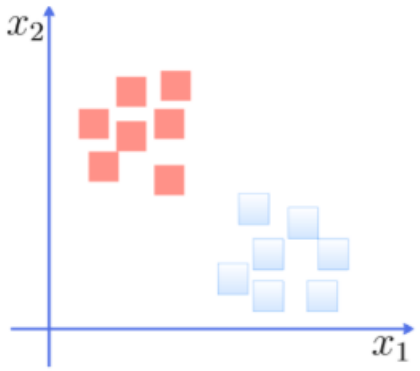

Figura 9 – Exemplo SVM com os grupos de dados

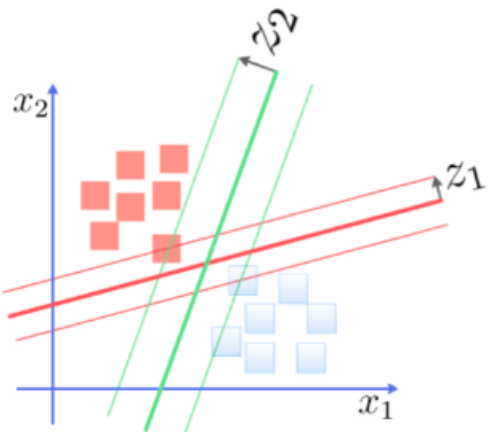

Figura 10 – Hiperplanos z1 e z2 e suas margens de distância.

rotulados erroneamente. Os outliers são exemplos muito distintos dos demais presentes no conjunto de dados. Esses dados podem ser ruídos ou casos muito particulares, raramente presentes no domínio [9].

Quanto trata-se de classificação de imagens e o reconhecimento de caracteres escritos à mão, os algoritmos SVMs podem e são utilizados com frequência. O algoritmo SVM também tem sido amplamente aplicado na área de ciências biológicas, onde seu uso serve, dentre outros, para classificar proteínas [10]. As SVMs também são robustas diante de dados de grande dimensão, sobre os quais outras técnicas de aprendizado de máquina comumente obtêm classificadores super ou sub ajustados.

Entre as principais limitações das SVMs encontram-se a sua sensibilidade a escolhas de valores de parâmetros e a dificuldade de interpretação do modelo gerado por essa técnica.

3.3 Redes Neurais Artificiais (RNA)

Redes neurais artificiais tem base em estudos biológicos, mais especificamente no estudo do neurônio, também conhecido como perceptron. O neurônio é composto por

dendritos, corpo e axônios, e funciona, de maneira simplificada, através de descargas elétricas que passam pelos dendritos até o corpo de irá sair um único sinal elétrico que será passado através do axônio e que, provavelmente, irá passar para um próximo neurônio.

Um neurônio artificial, utilizado em redes neurais, tenta imitar o comportamento de um neurônio biológico, possuindo, da mesma forma, entrada de sinais e saída de um sinal. O exemplo de um modelo de perceptron pode ser observado na Figura 11.

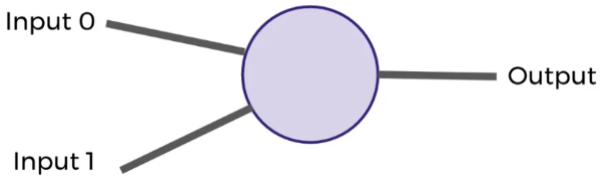

Figura 11 – Perceptron com suas representações de entrada e saída.

O funcionamento do perceptron pode ser exemplificado da seguinte forma: consideramos a existência da entrada 0, da entrada 1, do corpo e de uma única saída. Imaginamos que estas entradas recebem os valores 12 e 4, respectivamente, e que estes valores serão multiplicados por pesos, que definem a importância da entrada perante ao que o corpo, o qual, através do que chamamos de função de ativação, que será explicada posteriormente, irá realizar um cálculo e produzir uma saída única. O funcionamento do perceptron pode ser visualizado através da Figura 12.

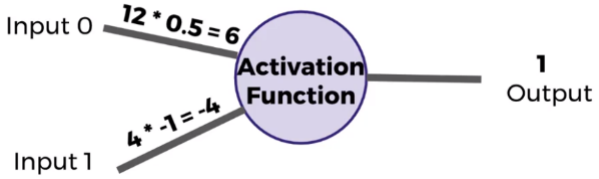

Figura 12 – Representação do funcionamento do perceptron.

Além disso, segundo Mackay [11] as redes neurais são, geralmente, especificadas por 3 condições:

- Arquitetura: A arquitetura especifica quais variáveis estão envolvidas na rede e suas relações topológicas - por exemplo, as variáveis envolvidas em uma rede neural podem ser os pesos das conexões entre os neurônios, juntamente com as atividades deles.
- Regra de atividade: A maioria dos modelos de redes neurais tem uma dinâmica de escala de tempo curta: as regras locais definem como as atividades dos neurônios mudam em resposta uma à outra. Normalmente, a regra de atividade depende dos pesos (os parâmetros) na rede.

- Regra de aprendizado: A regra de aprendizado especifica a maneira pela qual os pesos da rede neural mudam com o tempo. Esse aprendizado é geralmente visto como ocorrendo em uma escala de tempo maior do que a escala de tempo da dinâmica sob a regra de atividade. Normalmente, a regra de aprendizado dependerá das atividades dos neurônios. Pode também depender dos valores dos valores-alvo fornecidos por um professor e do valor atual dos pesos

Considerando a utilização do conceito de redes neurais para o reconhecimento de imagens ou de textos em imagens, como é o caso do projeto aqui documentado, tem-se como entrada da rede o conjunto de píxeis da imagem, com atribuição de pesos para cada pixel que, com ajuda da função de ativação, irão determinar quais neurônios da rede serão ativados. Posteriormente, estes pesos serão reajustados através de uma regra de aprendizado determinada até que o resultado obtido encontre a condição definida pelo criador da rede.

3.3.1 Redes de Múltiplos Perceptrons

Uma rede de múltiplos perceptrons é caracterizada pela união de entradas e saídas ligadas a múltiplos perceptrons de forma a construir diferentes camadas, sendo que cada camada recebe uma determinada nomenclatura. De acordo com a Figura 13, pode-se observar na cor roxa a camada de entradas, nas cores azul e verde as camadas intermediárias, também chamadas de *hidden layers* e em vermelho a camada de saída.

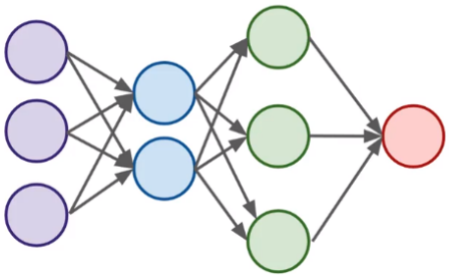

Figura 13 – Exemplo de Rede de Múltiplos Perceptrons.

As camadas de entrada, *input layers*, são responsáveis por receber os valores reais, também chamados de dados. As camadas intermediárias, *hidden layers*, estão entre as camadas de entrada e saída, e quando compostas por 3 ou mais camadas tornam-se uma rede profunda, também chamada de *deep network*. Por fim, a camada de saída é onde gera-se o valor estimado final.

3.3.2 Funções de Ativação

O funcionamento de neurônio artificial consiste, de maneira simplificada, em calcular uma soma ponderada das entradas com um bias para decidir se será "disparado" ou não. O cálculo citado anteriormente é exemplificado na equação 3.1 a seguir:

$$w * x + b = y \tag{3.1}$$

Na equação 3.1, x é o vetor das entradas, w os pesos, b é o termo tendencioso (*bias*) e y corresponde ao vetor de pontuação para cada classe.

O treinamento da rede neural atua nos pesos e no *bias*, ou seja, tenta-se encontrar valores para ambos que terão uma boa performance em fazer predições para as entradas.

No entanto, pode-se perceber facilmente que o valor de y pode variar de infinito positivo para infinito negativo. Essas representações limitam a ativação do neurônio, uma vez que o mesmo não reconhece o limite destes valores.

As funções de ativação são um elemento extremamente importante da arquitetura das redes neurais artificiais. Elas basicamente decidem se um neurônio deve ser ativado ou não. Algumas funções de ativação são exemplificadas e conceituadas a seguir.

3.3.2.1 Função de Etapa Binária

Trata-se da representação mais simples que pode-se pensar em quesito de ativação, ou seja, atingindo-se um valor limiar o neurônio será ativado, caso contrário o neurônio não receberá uma ativação. Matematicamente, a representação da função de etapa binária se dá pelas equações 3.2 e 3.3 e é representada graficamente pela Figura 14.

$$f(x) = 1, x >= 0 \tag{3.2}$$

$$f(x) = 0, x < 0 \tag{3.3}$$

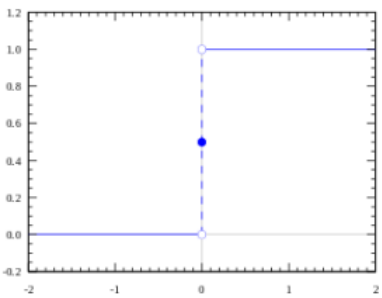

Figura 14 – Gráfico da Função de Etapa Binária.

A função não costuma ser utilizada na prática, pois, na maioria dos casos, realiza-se classificações em várias classes do que apenas uma única classe. A função de etapa não seria capaz de fazer isso.

Além disso, o gradiente da função de etapa é zero. Isso faz com que a função de etapa não seja tão útil durante o *backpropagation* quando os gradientes das funções de ativação são enviados para cálculos de erro para melhorar e otimizar os resultados. O gradiente da função de etapa reduz tudo para zero e a melhoria dos modelos realmente não acontece [12].

3.3.2.2 Função Sigmoide

A função não-linear sigmoide possui a fórmula matemática descrita em 3.4 é apresentada no gráfico da Figura 15. Ela recebe valores reais e colocá-os na região [0, 1] com um rápido crescimento em $-2.5 < x < 2.5$. Esse intervalo é chamado de regime linear, pois a sigmoide comporta-se aproximadamente como uma função linear nesse intervalo. Isso a torna conveniente para problemas de classificação, visto que possui uma interpretação lógica e pode ser interpretada como uma probabilidade. Recentemente, a função tem sido menos utilizada devido a dois problemas inerentes:

- A sigmoide satura e anula os gradientes que serão explicados posteriormente.
- A saída da sigmóide não está centrada em zero.
- A saturação na saída da sigmóide impede que ela treine caso seus pesos atuais sejam muito altos ou muito baixos, já o fato dela não estar centrada em zero pode fazer com que todos os pesos w se tornem ou positivos ou negativos.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

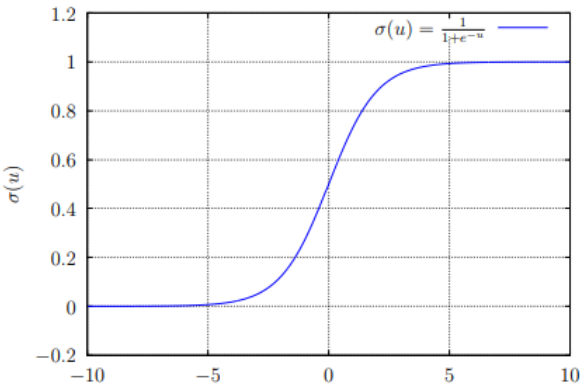

Figura 15 – Gráfico da Função Sigmóide.

3.3.2.3 Função Tangente Hiperbólica

A função tangente hiperbólica recebe valores reais e os força para uma região [-1, 1]. Como o neurônio sigmoide, sua ativação satura, porém a sua saída é centrada em zero. Na prática, a tanh é preferível à sigmoide. Sua definição é dada pela equação 3.5 e o seu comportamento é mostrado na Figura 16.

$$tanh(x) = 2\sigma(2x) - 1 \tag{3.5}$$

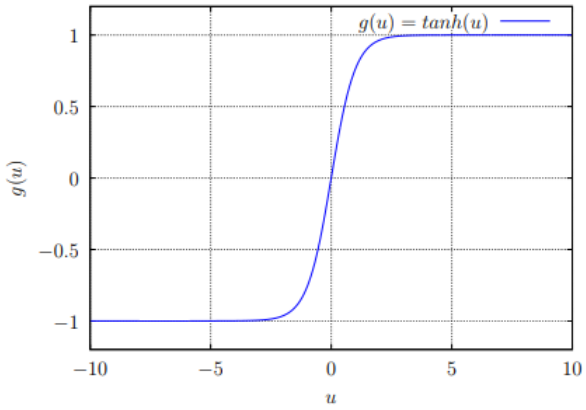

Figura 16 – Gráfico da Função Tangente Hiperbólica.

3.3.2.4 Função Unidade Linear Retificada (ReLU) [2]

É sem dúvida uma das funções de ativação mais populares nos últimos anos, principalmente em aprendizado profundo. Ela resolve $f(x) = max(0, x)$, ou seja, a ativação se dá simplesmente por um limiar em zero 3.6. Seus prós e contras podem ser resumidos em:

- Acelera consideravelmente a convergência do Gradiente descendente quando comparado às funções sigmoide e tanh.
- É implementada com funções bem menos custosas que a sigmoide e a tanh, que utilizam exponenciais.
- Ela é capaz de inativar neurônios quando a soma ponderada de todas suas entradas é negativa. Portanto, a ReLU não propaga erro durante a fase de *backpropagation*, quando isso acontece.

$$ReLU(x) = \begin{cases} 0 \text{ if } x < 0 \\ x \text{ else} \end{cases} \tag{3.6}$$

O comportamento da ReLu pode ser observado na Figura 17.

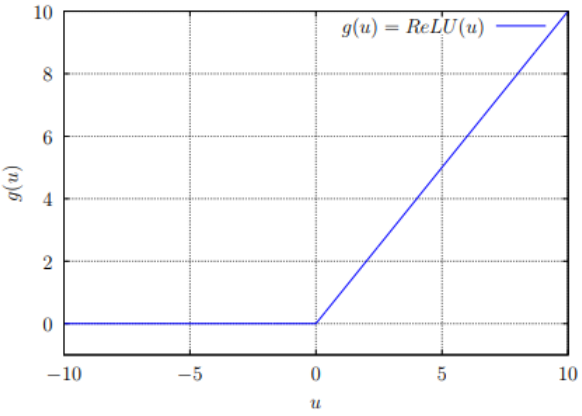

Figura 17 – Gráfico da Função ReLU.

3.3.2.5 Função SoftMax

A função de ativação softmax é usada em redes neurais de classificação. Ela força a saída de uma rede neural a representar a probabilidade dos dados serem de uma das classes definidas. Sem ela as saídas dos neurônios são simplesmente valores numéricos onde o maior indica a classe vencedora [13].

Sua equação é definida em 3.7, onde j representa o índice do neurônio de saída $\varnothing$ sendo calculado e k representa os índices de todos os neurônios de um nível. A variável z designa o vetor de neurônios de saída. É importante ressaltar que nesta função de ativação a saída de um neurônio depende dos outros neurônios de saída.

$$\varnothing(z)_j = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.7}$$

3.3.3 Codificação One-Hot

A codificação *one-hot* é uma forma matemática utilizada para representar os rótulos de cada entrada à rede neural. Cada rótulo será identificado por um vetor de tamanho igual ao número de classes possíveis, assim como o vetor de probabilidades. No caso do vetor, será atribuído o valor de 1.0 para a posição referente a classe correta daquele exemplo e 0.0 para todas as outras posições. Com esta técnica torna-se possível medir a eficiência do treinamento apenas comparando dois vetores [14].

Um exemplo de utilização da codificação One-Hot pode ser visualizado na Tabela 1, onde três categorias são representadas por um vetor único que segue a lógica descrita anteriormente.

Categoria	Valor
Carros	001
Motos	010
Aviões	100

Tabela 1 – Exemplo da codificação One-Hot

3.3.4 Inicialização de Pesos

Os pesos precisam ser inicializados de alguma forma para poderem passar pelos processos de aprendizagem nas redes neurais artificiais. O mais comum é utilizar uma distribuição aleatória [15]. Esta ideia foi adotada no projeto aqui proposto.

3.3.5 Função de Perda Cross-Entropy

A cross entropy mede o desempenho de um modelo de classificação cuja saída é um valor de probabilidade entre 0 e 1. Ela aumenta à medida que a probabilidade prevista diverge do rótulo real. Portanto, prever uma probabilidade de 0,012 quando o rótulo de observação real é 1 seria ruim e resultaria em um alto valor de perda.

A cross entropy é a forma mais comum de se medir a distância entre o vetor de probabilidades e o vetor correspondente ao rótulo.

3.3.6 Método do Gradiente Descendente

O método do gradiente descendente é um algoritmo de otimização muito utilizado em tarefas de redes neurais. Ele é uma ferramenta utilizada que otimiza funções complexas de forma iterativa para encontrar um valor mínimo [12].

Simplificando, o método do gradiente descendente é utilizado para encontrar um conjunto de pesos e *bias* de uma função que minimizam uma função de custo. Para o treinamento funcionar o método é executado dezenas ou centenas de vezes até encontrar os valores mínimos de pesos e bias.

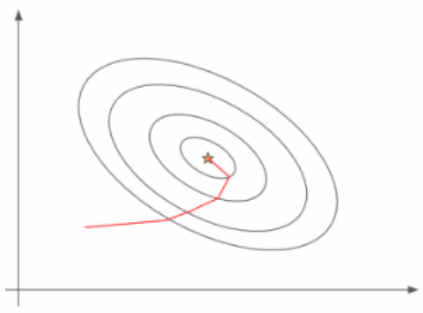

Figura 18 – Representação do método do gradiente descendente.

Na Figura 18, as funções de perda são representadas pelos círculos, onde a área dos círculos varia de acordo com os parâmetros de pesos e bias. Com o objetivo de encontrar os melhores pesos, o método do gradiente descendente vai calculando a derivada da perda. A derivada é um conceito de Cálculo e refere-se à inclinação da função em um determinado ponto. Conhecendo a inclinação,consegue-se descobrir a direção em que os valores dos coeficientes devem se mover a fim de obter um custo menor na próxima iteração.

3.3.7 Função de Otimização Adam

A função de otimização Adam, de acordo com [16], é um método para otimização estocástica eficiente que requer apenas gradientes de primeira ordem com pouca necessidade de memória. O método calcula as taxas de aprendizagem para diferentes parâmetros a partir das estimativas do primeiro e segundo momentum dos gradientes; o nome Adam é derivado da estimação de momento adaptativa.

3.3.8 Momentum

Momentum é um termo na função de otimização que assume um valor entre 0 e 1 e tem como finalidade aumentar o tamanho das iterações que buscam o mínimo erro. Se o valor do momentum for grande, a taxa de aprendizado deverá ser mantida menor.

Com isso, a utilização de um valor alto de momentum terá que a convergência ocorrerá rapidamente. No entanto, se ambos o momentum e a taxa de aprendizado forem mantidos com valor alto as iterações terão intervalos maiores e pode-se pular o mínimo da função.

Por outro lado, um valor pequeno para o momentum também não garante que o mínimo da função seja encontrado e ainda pode deixar lento o treinamento do modelo.

Contudo, o momentum ajuda na suavização de variações da função de otimização e pode ter ser valor ideal alcançado através de tentativa e erro ou utilizando funções de perda.

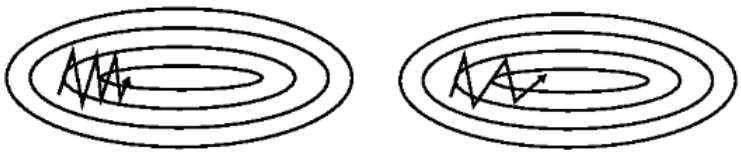

Figura 19 – Efeito de suavização do momentum no método de gradiente descendente.

A Figura 19 mostra, à esquerda, as etapas realizadas pelo método do gradiente descendente, enquanto que, à direita, tem-se ilustrado o efeito de amortecimento das oscilações e da aceleração da convergência, ambos causados pelo momentum.

3.3.9 Overfitting

Segundo Goodfellow [17], no aprendizado de máquina há dois desafios centrais para os pesquisadores: underfitting e overfitting.

O underfitting ocorre, segundo [18], quando a rede não é treinada suficientemente para se tornar capaz de produzir resultados satisfatórios, ou seja, quando há um número muito baixo de neurônios ou de camadas para o problema que está sendo abordado. Pode ocorrer também underfitting no treinamento, quando a rede neural é treinada com uma quantidade de épocas abaixo do necessário.

Já o overfitting acontece quando a diferença é muito grande entre o valor de perda para o conjunto de treinamento e o valor de perda para o conjunto de teste [14]. Em outras palavras, o overfitting é observado quando a rede memoriza os padrões de treino e perde a capacidade de generalizar, deixando de prever as saídas de forma correta.

3.3.10 Dropout

O termo dropout, segundo [19], refere-se a eliminação aleatória de neurônios durante o processo de aprendizagem, evitando a ocorrência de overfitting.

De forma mais técnica, o dropout elimina nodos individuais em cada etapa de treinamento deixando a rede neural reduzida, com menos conexões, parâmetros e neurônios.

Seu principal motivo de utilização é o alcance de maior robustez à rede neural para previsões de informações populacionais ao invés e características amostrais. O dropout tem alcançado o estado da arte e desempenho em muitas bases de dados de referência [20].

Um exemplo do dropout pode ser observado através da Figura 20, que mostra uma rede neural densamente conectada e a mesma rede após a aplicação da técnica. Percebe-se que a rede neural passa a ter menos nodos e menos conexões ativas.

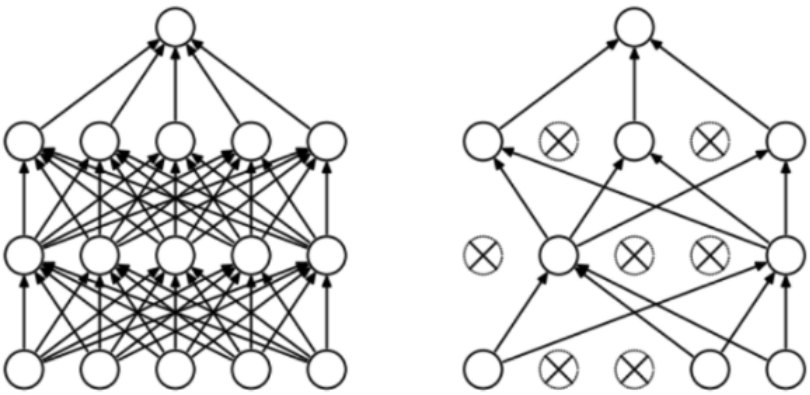

Figura 20 – Rede neural sem e com a técnica de dropout.

3.4 Redes Neurais Convolucionais (CNN)

As Redes Neurais Convolucionais (*Convolutional Neural Networks*, CNNs). foram primeiramente propostas em 1998 por [21], na qual os autores desenvolveram uma arquitetura neural conhecida como LeNet5, utilizada para reconhecer dígitos escritos à mão.

As CNNs são similares a redes neurais tradicionais, ou seja, ambas são compostas por neurônios que possuem pesos e bias que necessitam ser treinados. Cada neurônio recebe algumas entradas, aplica o produto escalar das entradas e pesos além de uma função não-linear. Além disso, ambas possuem a última camada toda conectada e todos os artifícios utilizados para melhorar a rede neural tradicional também são aplicados nesta camada [22].

A grande vantagem das CNNs está para as entradas do tipo imagem, permitindo codificar algumas propriedades na arquitetura da rede, diferentemente das redes neurais tradicionais que, por produzirem um número muito alto de pesos a serem treinados, se tornam não escaláveis para o tipo de entrada citado.

3.4.1 Arquitetura

A composição de uma CNN é feita por uma sequência de camadas. A primeira camada é definida pelo dado de entrada, normalmente uma imagem com largura, altura e profundidade. Após a camada de entrada, outros três tipos de camadas compõem uma CNN, sendo elas: camada convolucional, camada de *pooling* e camada totalmente conectada. Além disso, é comum existir uma camada de ativação após uma camada de convolução. Uma representação genérica de uma rede neural convolucional é ilustrada na Figura 21.

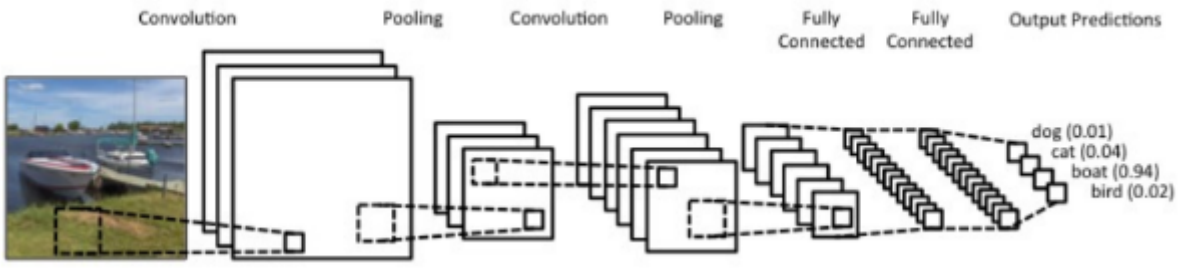

Figura 21 – Arquitetura de uma rede neural convolucional.

As redes convolucionais são projetadas especificamente para reconhecer formas bidimensionais com um alto grau de invariância quanto a translação, escalonamento, inclinação e outras formas de distorção [23].

3.4.2 Camada Convolucional

A camada convolucional é a camada mais importante da rede. Nela é realizada a parte mais pesada do processamento computacional [26].

Uma característica básica de uma camada convolucional é que cada neurônio não está ligado a todas as entradas da rede. Outra característica é a composição da camada por um conjunto de filtros, que são pequenas matrizes de valores reais capazes de aprender de acordo com um treinamento. No treinamento destes filtros, ocorrem operações de convolução, exemplificadas na Figura 22, para reconhecer a imagem de entrada e criar um mapa de características. Se o conjunto de valores de um filtro for capaz de descrever bem uma região de uma imagem, a rede entende que estes mesmos valores poderão ser capazes de reconhecer outra região da entrada se aplicados em outro filtro. Ainda assim, os valores dos filtros se alteram ao longo do treinamento para fazer correções, adaptações e realizar uma melhor extração de características da imagem de entrada.

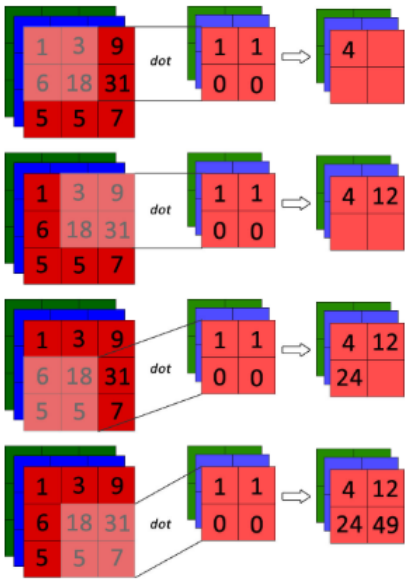

Figura 22 – Sequência da operação de convolução e strides em matrizes de entrada.

Na Figura 22 também pode-se observar a ocorrência de um deslizamento do quadro que seleciona os valores, esse efeito é controlado pelo parâmetro stride. Neste exemplo o filtro desliza na imagem deslocando 1 pixel para lado ou para baixo. O tamanho do deslocamento do quadro pode variar, desde que esteja dentro dos limites da imagem de entrada.

Existe a possibilidade do tamanho do filtro e do stride não se adequem à imagem, sendo necessário utilizar um conceito denominado por zero-padding. Este nada mais é do que adicionar zeros na borda da imagem para tornar o deslizamento do filtro possível [26].

De forma geral, uma camada convolucional dedica vários neurônios para mapear a entrada, conforme exibido na Figura 23. O fato de dedicar uma quantidade limitada de

neurônios diminui drasticamente o número de conexões da rede e facilita a otimização dos parâmetros internos.

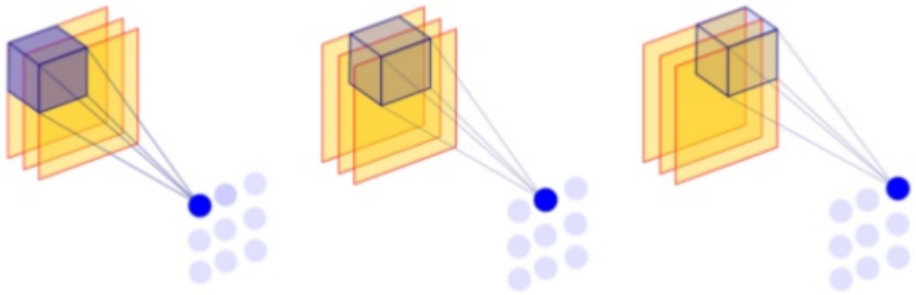

Figura 23 – Conexões locais em uma camada convolucional.

3.4.3 Camada de Pooling

A camada de pooling é utilizada com objetivo de reduzir o tamanho espacial das matrizes resultantes da convolução. Com isso, são reduzidas as quantidades de parâmetros necessários para o aprendizado da rede e o controle de overfitting é melhorado.

O funcionamento da camada de pooling pode ser observado na Figura 24, onde tem-se a operação independente da camada, que age utilizando os conceitos de filtros e stride citados anteriormente para reduzir o tamanho da matriz de entrada. Na Figura 24, tem-se um exemplo de uma matriz de entrada 4x4x3 sendo filtrada por uma configuração 2x2 com stride igual a 2 e com a função de max pooling para produzir o resultado final. O max pooling nada mais é do que uma técnica que seleciona o maior valor da matriz que está sendo filtrada.

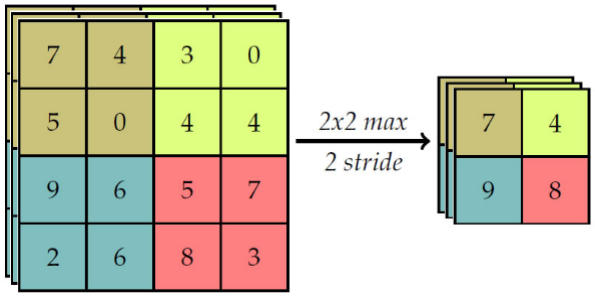

Figura 24 – Exemplo de Camada de Pooling utilizando max pooling.

Como pode ser observado, a camada de pooling não reduz a quantidade de canais e sim a quantidade de elementos em cada canal. Além do max pooling, pode ser utilizadas outras funções, como por exemplo a média dos valores. Todavia, a função max vem obtendo melhores resultados, uma vez que pixels vizinhos são altamente correlacionados [26].

3.4.4 Camada Totalmente Conectada

A camada totalmente conectada tem como característica principal a completude de conexões com a camada que a antecede. Comumente elas são as últimas camadas de uma rede neural convolucional, pois atuam da mesma forma que em redes neurais tradicionais. Desta forma se torna possível aplicar as mesmas técnicas para melhorar o desempenho da camada, o dropout é um exemplo.

É importante ressaltar que, como a camada totalmente conectada é alocada após uma camada convolucional ou uma camada de pooling, é essencial conectar cada elemento das matrizes vindas da camada anterior em um neurônio de entrada. Essa condição pode ser visualizada na Figura 25, onde fica exemplificada a camada anterior com 48 mapas no formato 4x4 conectada com uma camada totalmente conectada de 768 entradas, que possui 500 neurônios ocultos e resultam em 10 saídas da rede.

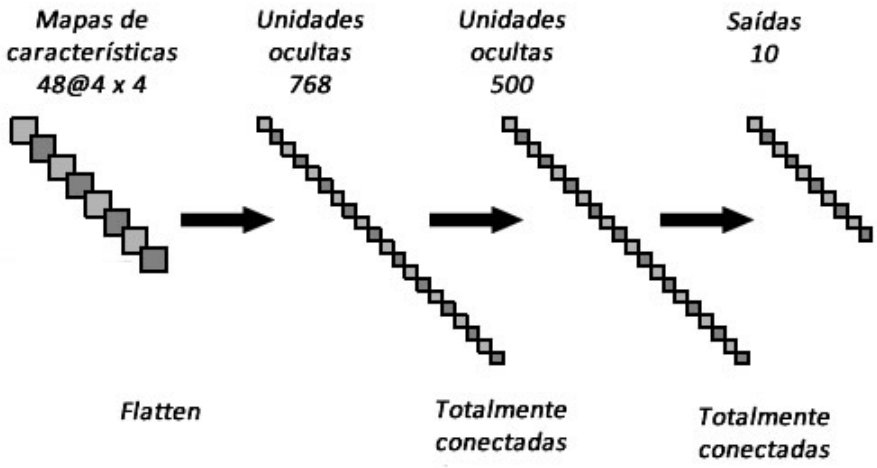

Figura 25 – Ilustração de uma camada totalmente conectada.

Nas unidades de saída, que no exemplo da Figura 25 são 10, é utilizada uma função softmax para se obter a probabilidade de dada entrada pertencer a uma classe. Neste ponto é realizado o algoritmo de treinamento supervisionado backpropagation, assim como em uma rede neural tradicional. O erro obtido nesta camada é propagado para que os pesos dos filtros das camadas convolucionais sejam ajustados. Dessa forma, os valores dos pesos compartilhados são aprendidos ao longo do treinamento.

3.5 CAPTCHA

Desde os primórdios da Internet, os usuários querem tornar o texto ilegível para os computadores. Os primeiros a discutir sobre o assunto podem ter sido os hackers, pois eles achavam que estavam sendo monitorados automaticamente por palavras-chave por estarem postando tópicos sensíveis em fóruns da Internet.

Um dos primeiros usos comerciais dos CAPTCHAs foi no teste de Gausebeck-Levchin. Em 2000, o idrive.com começou a proteger sua página de inscrição com um CAPTCHA e se preparou para registrar uma patente sobre essa técnica aparentemente nova. Em 2001, o PayPal usou esses testes como parte de uma estratégia de prevenção de fraudes, na qual pedia aos humanos que "redigitassem texto distorcido que os programas têm dificuldade em reconhecer". O cofundador do PayPal e CTO Max Levchin ajudou a comercializar esse uso antecipado [24].

O acrônimo CAPTCHA significa Teste de Turing Público Completamente Automatizado para Diferenciação entre Computadores e Humanos, e foi cunhado em 2000 por Luis Von Ahn, Manuel Blum, Nicholas J. Hopper [25].

Nada mais são do que distorções de textos ou imagens, de modo que elas ainda são reconhecíveis para a maioria dos humanos, mas podem se tornar difíceis para um computador reconhecer. Atualmente, é um mecanismo de segurança estabelecido para impedir postagens automatizadas nos fóruns da Internet, votações em pesquisas on-line, download de arquivos em grandes quantidades e muitos outros usos abusivos de serviços da web, ou seja, são desafios para garantir que os usuários sejam de fato humanos.

Um tipo comum de CAPTCHA usado na maioria dos sites exige que os usuários insiram a sequência de caracteres que aparecem de forma distorcida na tela. Os CAPTCHAs são usados pelo fato de que é difícil para os computadores extraírem o texto de uma imagem tão distorcida, enquanto é relativamente fácil para um humano entender o texto oculto por trás das distorções. Portanto, supõe-se que a resposta correta a um desafio CAPTCHA venha de um ser humano e o usuário tenha permissão para entrar no site [26].

Algumas outras aplicações reais da utilização de CAPTCHAs são encontradas para evitar spam de comentários em blogs, confirmações em solicitações online e, o mais comum e explorado neste documento, proteger os registros e bases de dados de sites.

A lógica de acesso a uma determinada informação em um site eletrônico e protegida por CAPTCHA pode ser observada na Figura 26 abaixo.

3.5.1 Tipos de CAPTCHA

3.5.1.1 CAPTCHA Textual

O primeiro uso do captcha foi em 1997 pela empresa de software Alta-Vista, que buscava uma maneira de evitar envios automáticos para o mecanismo de busca. Foi um teste simples baseado em texto que foi suficiente para aquela época, mas foi rapidamente provado ineficaz quando as taxas de sucesso de reconhecimento de caracteres de computador melhoraram. As técnicas mais usadas para impedir o reconhecimento automático podem ser divididas em dois grupos, denominados recursos de antirreconhecimento e recursos de antissegmentação [27].

Os recursos anti-reconhecimento, como tamanhos e fontes diferentes de caracteres ou rotação, foram um primeiro passo na construção de CAPTCHAs mais sofisticados. Todas essas características são bem aceitas pelos humanos, já que aprendemos várias formas de letras desde a infância. Uma maneira efetiva de reduzir a precisão do classificador é uma distorção. Distorção é uma técnica na qual ondulações e deformações são adicionadas à imagem. Mas a distorção excessiva pode tornar isso muito difícil até para os humanos e, assim, o uso desse recurso desaparece lentamente, sendo substituído por recursos anti-segmentação [27].

Os recursos de antissegmentação não são projetados para complicar o reconhecimento de um único caractere, mas tentam tornar a segmentação automática da imagem do captcha incontrolável. Os dois primeiros recursos utilizados para esse fim foram adicionados ruído e fundo confuso. Mas mostrou-se que ambos são maiores obstáculos para os seres humanos do que para os computadores e, portanto, substituem as linhas de oclusão, um exemplo pode ser visto na Figura 27. O mais recente recurso de anti-segmentação é chamado de kerning negativo. Isso significa que os personagens vizinhos são movidos tão próximos uns dos outros que podem eventualmente se sobrepor. Ele mostrou que os seres humanos ainda são capazes de ler o texto sobreposto com apenas uma pequena taxa de erro, mas para computadores é quase impossível encontrar uma segmentação correta [27].

Estes são os formatos de CAPTCHA mais utilizados até hoje, eles estão presentes na maioria dos sites eletrônicos que fornecem algum tipo de informação e possuem um nível de segurança CAPTCHA. Existem uma gama enorme de CAPTCHAs imagem-texto, com padrões de dificuldade mais variados possíveis, alguns exemplos podem ser observados na Figura 28.

Um CAPTCHA baseado em imagem-texto é utilizado para o desenvolvimento deste projeto e será ilustrado no Capítulo 4.

3.5.1.2 CAPTCHA em Áudio

Embora amplamente fornecidos por razões de acessibilidade, os CAPTCHAs de áudio receberam atenção científica substancialmente menor [27].

Os CAPTCHAs de áudio típicos consistem em um ou vários alto-falantes dizendo letras ou dígitos em intervalos aleatoriamente espaçados. Um usuário deve identificar corretamente os dígitos ou caracteres falados no arquivo de áudio para passar o CAPTCHA. Para dificultar esse teste para sistemas de computador atuais, especificamente programas de reconhecimento automático de fala (ASR), o ruído de fundo é injetado nos arquivos de áudio [28].

Atualmente este estilo de CAPTCHA é uma opção alternativa ao padrão de CAPTCHA baseado em imagem-texto e é encontrado em uma quantidade irrelevante de

sites eletrônicos. Muitos são os casos em que, uma vez fornecido o CAPTCHA sonoro, o CAPTCHA imagem-texto também estará presente, cabendo ao usuário definir qual o método de preferência para realizar o reconhecimento.

O exemplo da Figura 29 abaixo é o de um CAPTCHA sonoro implementado no site eletrônico authorize.net, que consiste em cinco letras ou dígitos proferidos por uma voz feminina.

Pode-se observar os formatos de onda que representam as letras do CAPTCHA sonoro e também o padrão de intervalos entre a representação sonora de um caractere e outro. Com a identificação dos padrões e segmentação das faixas que representam os caracteres, pode-se criar um reconhecedor de CAPTCHA sonoro e realizar a quebra de maneira automatizada.

3.5.1.3 CAPTCHA em Imagem

CAPTCHAs baseados em imagens são testes nos quais os usuários têm que escolher aquelas imagens que possuem algumas propriedades similares [27].

A maioria dos CAPTCHAs baseados em imagem usa um grande banco de dados de imagens fotográficas e animadas de objetos do cotidiano (um bebê, uma cadeira, um cachorro, um pássaro, etc). A Figura 30 representa várias amostras.

Normalmente, um conjunto de imagens é mostrado ao usuário, todas relacionadas com o mesmo conceito ou objeto. Para efetuar a quebra do CAPTCHA, o usuário deve selecionar o conceito ou objeto ao qual todas as imagens pertencem de forma correta.

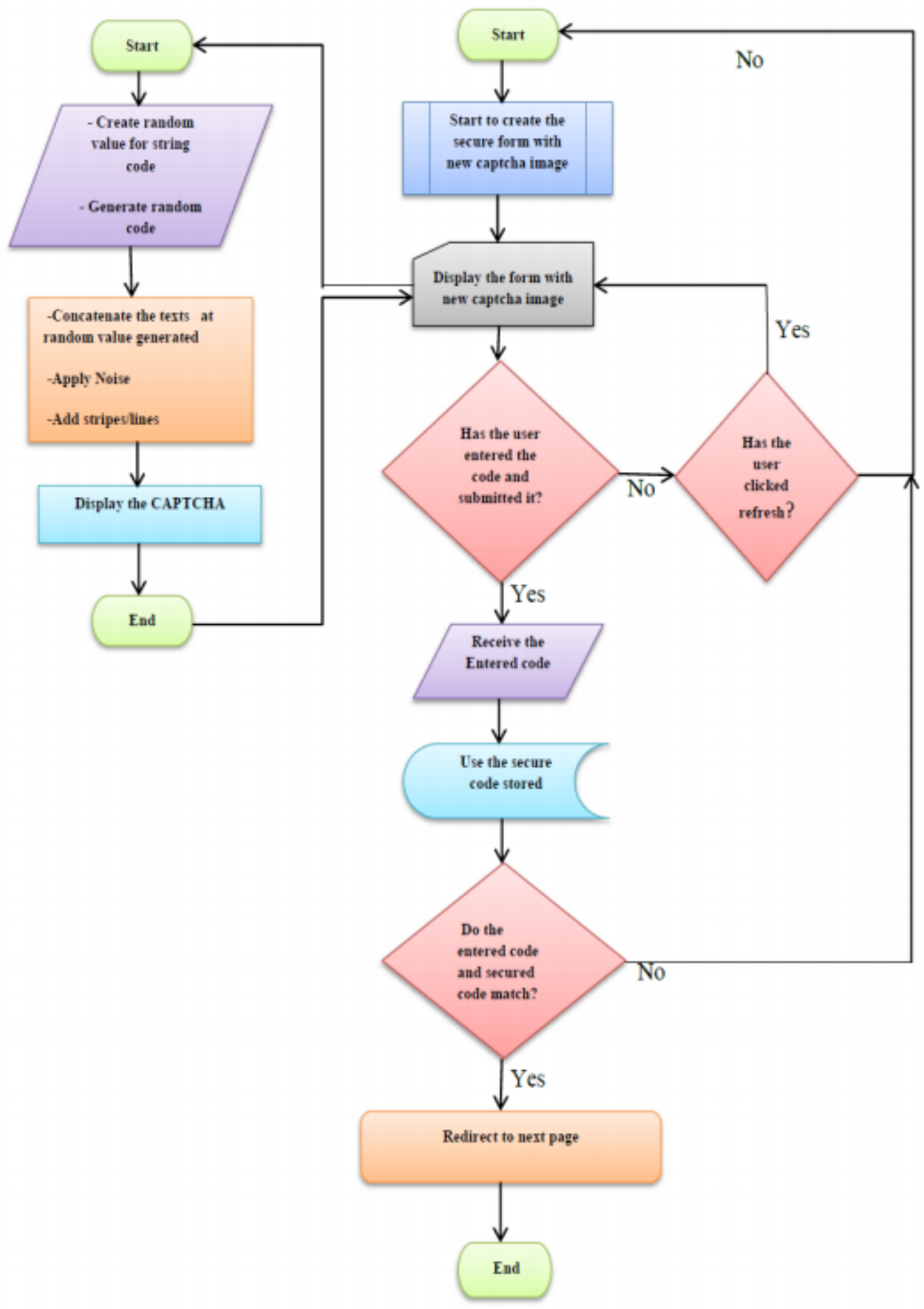

Figura 26 – Fluxo de criação de CAPTCHA.

Figura 27 – Exemplo mais antigo do Google reCaptcha.

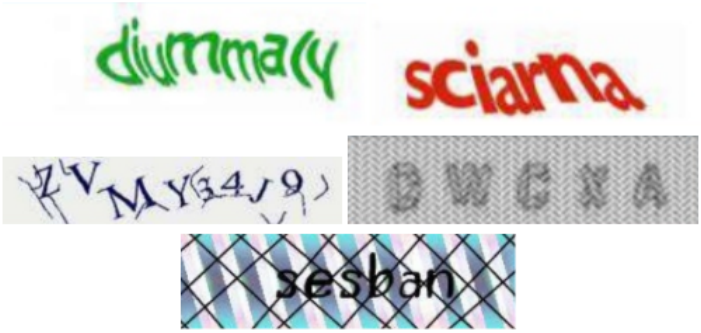

Figura 28 – Exemplos variados de CAPTCHA.

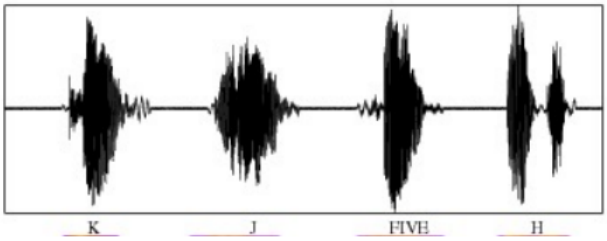

Figura 29 – Exemplo de CAPTCHA sonoro utilizado em authorize.net.

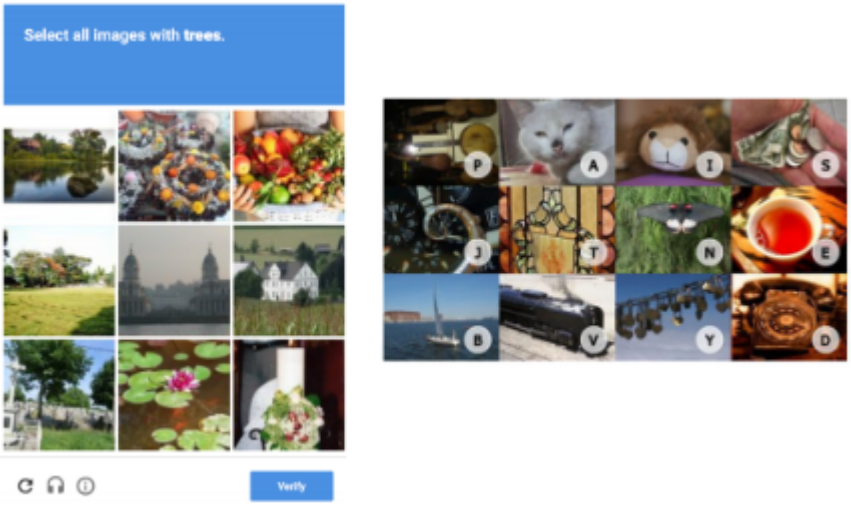

Figura 30 – Representação de CAPTCHAs baseados em imagens.

4 Proposta de Projeto

O intuito básico do projeto foi desenvolver um reconhecedor de textos em uma imagem CAPTCHA de assertividade razoável. Para isso foram efetuadas algumas etapas básicas e comuns à abordagem de reconhecimento de imagens utilizando aprendizado de máquinas e redes neurais. Primeiramente foi realizada a captura das imagens para a construção de um *dataset* necessário na fase de treinamento dos modelos. Em seguida, as imagens passaram por uma etapa de pré-processamento para facilitar o reconhecimento. Após o pré-processamento, tem-se a configuração e treinamento dos modelos. Para finalizar, calculou-se a acurácia do modelo e, uma vez validado, o modelo foi colocado em produção.

4.1 Captura de Imagens

Ao solicitar uma emissão de certidão negativa de débitos através do site eletrônico do SIT é exigido o preenchimento de um formulário com dados de CPF ou CNPJ do usuário e a solução da imagem CAPTCHA. A fim de capturar dados da fonte pública citada, a equipe de dados da Neoway desenvolveu um bot que fornece os dados de CPF e CNPJ, captura a imagem, salva essa imagem em um banco de dados, envia a imagem para um serviço externo pago, recebe do serviço externo a resposta correta da imagem e realiza corretamente o acesso.

A partir do funcionamento correto deste *bot*, pode-se capturar muitas imagens CAPTCHA juntamente com o texto reconhecido de forma correta, o qual foi determinado pelo serviço externo. Dado um intervalo de tempo deste bot estar em ambiente de produção foi alcançado uma quantidade de 93,468 imagens, um exemplo do CAPTCHA pode ser observado na Figura 31 abaixo.

Figura 31 – Exemplo de imagem CAPTCHA utilizado no projeto.

Com esta quantidade de imagens foi possível ter um dataset de tamanho suficiente para o desenvolvimento do projeto.

4.2 Pré-Processamento

Na área de processamento de imagem e reconhecimento ótico de caracteres, a segmentação é uma das etapas que desempenham um papel importante no tratamento de imagens de texto off-line e on-line. Segmentação de caracteres significa dividir uma imagem com palavra em uma sequência de caracteres. Uma perspectiva ampla de segmentação está na segmentação dos caracteres em CAPTCHA. A segmentação de caracteres atua como base para analisar a força do CAPTCHA. Quanto mais forte o CAPTCHA, mais difícil é quebrar. As operações que podem ser realizadas em qualquer imagem são: conversão de escala de cinza, remoção de ponto, remoção de linha, correção de inclinação, extração de cor, desbaste.

Na proposta do pré-processamento da imagem estava a ideia de utilizar o mínimo de operações, filtros e segmentação para atingir o resultado da identificação de textos. Esta abordagem tem problemas e vantagens. Ao passo em que a menor preocupação com a limpeza e o pré-processamento da imagem gera custos em processamento, complexidade de arquitetura e a necessidade de um dataset gigantesco, uma maior preocupação requer o emprego do uso de muitos filtros, limpezas, segmentação, identificação, rotulamento, entre outros.

4.3 Reconhecedor através de SVM

A proposta determinada para o desenvolvimento do reconhecedor de textos utilizando os conceitos de SVM está inteiramente ligada ao uso de uma ferramenta inteira desenvolvida na Neoway.

Nesta ferramenta alguns passos são indispensáveis para que um resultado satisfatório seja alcançado. Dentre estes passos estão a análise da imagem, a determinação do ruído de imagem, a seleção de filtros para limpeza dos ruídos, a definição de algoritmos de segmentação para separar os caracteres da imagem, a configuração do reconhecimento, o treinamento do reconhecedor, testes locais e, por fim, envio para o ambiente de produção.

4.4 Reconhecedor através de Redes Neurais Convolucionais

Para o desenvolvimento utilizando o conceito de redes neurais convolucionais utilizou-se uma abordagem semelhante à do método relatado anteriormente.

4.4.1 Pré-Processamento da Imagem CAPTCHA

A primeira etapa contou com um pré-processamento da imagem. No entanto, preferiu-se simplificar o máximo possível nos quesitos de limpeza e formatação da imagem

a fim de comprovar o desempenho do conceito aplicado. Apesar de simplificar nos pontos citados, foi necessário depreender uma atenção maior na formatação da imagem para usá-la como entrada da rede.

4.4.2 Conjunto de Dados

m dos passos mais importantes no desenvolvimento de uma rede é a separação das imagens em um conjunto de treinamento e um conjunto de testes. Para evitar problemas na construção dos dois conjuntos, optou-se por aumentar a quantidade de imagens CAPTCHA no dataset. Desta forma, as imagens utilizadas no desenvolvimento com SVM foram deixadas de lado e novas imagens foram adquiridas. Com uma maior quantidade de imagens, a possibilidade de divisão em conjuntos se tornou mais fácil.

Além dessa divisão, costuma-se usar uma subdivisão do conjunto de treinamento, criando um conjunto de validação, utilizado para verificar a eficiência da rede quanto a sua capacidade de generalização durante o treinamento, e podendo ser empregado como critério de parada do treinamento.

Depois de determinados estes conjuntos, eles são, geralmente, colocados em ordem aleatória para prevenção de tendências associadas à ordem de apresentação dos dados.

4.4.3 Configuração da Rede

Normalmente feito de forma empírica, a configuração da rede é uma etapa essencial no desenvolvimento da rede e é considerada uma arte por muitos, requisitando uma grande experiência do desenvolvedor. A definição da configuração pode ser segmentada em três etapas, segundo [29]:

1. Seleção do paradigma neural apropriado à aplicação.
2. Determinação da topologia da rede a ser utilizada - o número de camadas, o número de unidades em cada camada, etc.
3. Determinação de parâmetros do algoritmo de treinamento e funções de ativação. Este passo tem um grande impacto na performance do sistema resultante.

Neste projeto, optou-se pela simplicidade na configuração inicial da rede e, conforme o resultado do treinamento e do teste, aumentar a complexidade em busca e melhores resultados.

4.4.4 Treinamento

Nesta fase, serão investigados os valores para a inicialização da rede, o modo de treinamento e o tempo de treinamento através do ajuste dos pesos das conexões.

A escolha de um bom método de inicialização de pesos produz um resultado direto sob o tempo de treinamento, enquanto que uma escolha errada levará a uma saturação prematura.

Quanto ao modo de treinamento, utilizou-se o modo batch para ter uma melhor estimativa do vetor gradiente, o que torna o treinamento mais estável.

Em relação ao tempo de treinamento, não houve uma preocupação exagerada, buscou-se um bom desempenho, porém a configuração do dispositivo computacional utilizado para o desenvolvimento do projeto foi, por muitas tentativas, um ponto crucial para a quantidade de tempo utilizada. Porém, com o uso do critério de parada por um número máximo de ciclos foi possível realizar treinamentos mais rápidos ou mais demorados.

Quando a rede apresentou uma boa capacidade de generalização e quando a taxa de erro mostrou-se suficientemente pequena o treinamento foi encerrado.

4.4.5 Teste

Para finalizar, é realizado o teste da rede. Durante esta fase, utiliza-se o conjunto de teste, separado anteriormente, para determinar a performance da rede com dados não utilizados no treinamento. Nesta etapa pode-se ter uma boa perspectiva da performance real da rede.

4.5 Infraestrutura

Os algoritmos de *Machine Learning* e de Redes Neurais em geral exigem uma certa capacidade de processamento dos dispositivos computacionais. Para realizar a execução de códigos destas áreas é comum utilizar máquinas alugadas em nuvem, as quais possuem componentes dedicados para estes tipos de aplicação.

Inicialmente, para fins de introdução e desenvolvimento de *toy examples*, utilizou-se um notebook pessoal, o qual possuía a seguinte configuração:

- CPU: Intel Core I5-6200U (2.3 GHz com Turbo Boost até 2.8GHz)
- Memória: 8GB DDR4
- GPU: GeForce 940MX (2GB de VRAM dedicada)

Com o equipamento descrito acima, foi possível fazer a validação de alguns conhecimentos descritos ao longo do documento e também serviu como fonte de aprendizado sobre a necessidade de utilizar uma máquina dedicada e com mais desempenho de processamento e gráfico.

Após algumas pesquisas foi possível encontrar a ferramenta de desenvolvimento gratuita chamada Google Colaboratory. Ela foi desenvolvida utilizando o projeto open source chamado Jupyter, o qual é comumente utilizado nos estudos de *Data Science* e *Machine Learning*.

O Colaboratory é marcado pela sua característica educacional e de auxílio a pesquisas. Com ele, tem-se um ambiente interativo para desenvolvimento e com o seguinte poder computacional:

- CPU: 2 x Intel(R) Xeon(R) CPU @ 2.20GHz
- Memória: 13 GB
- GPU: Tesla K80 GPU (24 GB DDR5)

Com isso, pode-se obter resultados muito mais rápidos na execução dos códigos e algoritmos desenvolvidos, fazendo com que este se torna-se o dispositivo principal para a implementação do projeto desenvolvido e aqui documentado.

4.6 Bibliotecas Utilizadas

NNs podem ser utilizadas de maneira bem simples e rápida, por meio do uso de frameworks. Alguns de uso mais rápido, só escolher os parâmetros, inserir os dados de treinamento e executar, e outros que necessitam um pouco mais de elaboração. Os frameworks utilizados são citados a seguir:

- Keras: Framework de uso simples e direto. Estes fatores fazem com que ele seja menos maleável a modificações. É importante ressaltar que ele roda em cima do framework TensorFlow, sendo necessário a instalação de ambos.
- TensorFlow [30]: TensorFlow ™ é uma biblioteca de software de código aberto para computação numérica de alto desempenho. Sua arquitetura flexível permite a fácil implantação de computação em várias plataformas (CPUs, GPUs, TPUs) e de desktops a clusters de servidores para dispositivos móveis e periféricos. Originalmente desenvolvido por pesquisadores e engenheiros da equipe do Google Brain na organização de IA do Google, ele oferece um forte suporte para aprendizado de máquina e aprendizado profundo, e o núcleo flexível de computação numérica é usado em muitos outros domínios científicos.

Além deles, optou-se por utilizar a linguagem de programação Python e outras bibliotecas para auxiliar no processamento de imagens. Estas são citadas abaixo:

- NumPy [31]: O NumPy é o pacote básico da linguagem Python que permite trabalhar com arranjos, vetores e matrizes de N dimensões, de uma forma comparável e com uma sintaxe semelhante ao software proprietário Matlab, mas com muito mais eficiência, e com toda a expressividade da linguagem.

- OpenCV [32]: O OpenCV (Open Source Computer Vision Library) é uma biblioteca de software de visão computacional e aprendizado de máquina em código aberto. O OpenCV foi construído para fornecer uma infra-estrutura comum para aplicações de visão computacional e para acelerar o uso da percepção da máquina nos produtos comerciais. Sendo um produto licenciado pela BSD, o OpenCV torna fácil para as empresas utilizar e modificar o código. A biblioteca tem mais de 2500 algoritmos otimizados, o que inclui um conjunto abrangente de algoritmos de visão computacional e de aprendizado de máquina clássicos e de última geração.

5 Atividades Desenvolvidas

Neste capítulo é descrito o desenvolvimento das atividades do projeto proposto. Aqui constam, de forma mais detalhada, os passos e códigos implementados na construção dos reconhecedores de textos em imagem CAPTCHA.

Na primeira etapa, desenvolveu-se o reconhecedor baseado nos conceitos de SVM partindo da utilização de um framework chamado Decaptcher, o qual foi desenvolvido na própria Neoway e transformado em um serviço que é utilizado na captura de dados realizada pelos bots. As tarefas realizadas para construção deste reconhecedor foram:

- Análise da imagem
- Limpeza
- Segmentação
- Reconhecimento
- Treinamento
- Testes
- Envio para a Produção

Na segunda etapa, optou-se por validar os conceitos de redes neurais convolucionais para o desenvolvimento de um novo reconhecedor para o mesmo CAPTCHA. Buscou-se obter uma assertividade maior com um esforço menor. Para isso foram realizados os seguintes passos:

- Desenvolvimento do leitor e processador do conjunto de dados.
- Desenvolvimento do processador de imagem.
- Desenvolvimento da função que monta a CNN.
- Configuração da CNN.
- Desenvolvimento sessão de treinamento da CNN.
- Desenvolvimento da validação e testes do modelo gerado.

Na terceira etapa tentou-se simplificar ainda mais a implementação da CNN através da remoção do processamento de imagem. Com isso, evitaria-se a preocupação de filtragem, redução de ruídos e segmentação de uma imagem.

Em ambas as etapas fundamentadas nos conceitos de redes neurais foram utilizados como base códigos já existentes para o reconhecedor de dígitos em imagens. A partir deles foi construído o reconhecedor do projeto proposto.

5.1 Atualização do Bot

Antes de começar o desenvolvimento dos reconhecedores foi preciso realizar uma atualização no bot mte-cnd, o qual é o responsável por capturar as imagens CAPTCHA utilizadas neste projeto. O bot não estava configurado corretamente e, com isso, não estava realizando a captura e o armazenamento das imagens. Realizada a atualização, foi necessário aguardar algumas horas para que o bot arrecadasse uma quantidade mínima de imagens para o início do desenvolvimento dos reconhecedores.

5.2 Reconhecedor SVM

Por utilizar uma ferramenta desenvolvida e disponibilizada pela Neoway foi desenvolvido um reconhecedor utilizando o conceito de SVM no primeiro momento.

A ferramenta possui uma documentação que auxilia através de um passo a passo de tarefas que devem ser estudadas e efetuadas para o correto desenvolvimento do reconhecedor.

A seguir são elencados e descritos os passos executados para atingir a solução do problema proposto.

5.2.1 Análise da imagem

Com o conhecimento da imagem CAPTCHA para a qual será desenvolvido o reconhecedor é mandatório fazer observações a respeito dos padrões de ruídos utilizados na construção da imagem.

Nas imagens utilizadas neste projeto Figura 32, evidenciou-se os seguintes pontos:

- Tamanho em pixels de 120 de largura por 30 de altura.
- Presença das tonalidades de cores acinzentadas.
- Ruídos em formatos circulares e em posições aleatórias.
- Textos compostos por 5 caracteres .

- Caracteres representados por dígitos variando de 0 a 9 em ordem aleatória.
- Distanciamento e posicionamento uniforme entre os números.
- Bordas sem a presença de conteúdo.

Figura 32 – CAPTCHA MTE.

Feito este levantamento, os próximos passos desse desenvolvimento envolvem a limpeza dos ruídos, mudança da escala de cores, segmentação da imagem e extração dos caracteres de forma individual.

5.2.2 Limpeza

Esta etapa é a mais importante dentro do processo de desenvolvimento do reconhecedor por SVM. A limpeza de ruídos deve ser feita de forma a deixar os caracteres mais visíveis e delineados possível para que o algoritmo de segmentação consiga dividir os caracteres de forma correta.

Para este reconhecedor foram utilizadas as técnicas abaixo:

- Threshold: geralmente é o primeiro algoritmo a ser aplicado, pois deixa a imagem em preto e branco. Sem este passo, a maioria dos algoritmos citados não poderiam ser executados. O Threshold calcula o valor de todos os pixels, e os que forem maiores que um determinado parâmetro serão passados para preto e os menores para branco.
- Chopping: um algoritmo que tenta diminuir os ruídos da imagem como se estivesse realizando um corte ou picando os traços mais finos e contínuos.
- Remove small objects: uma das técnicas mais utilizada nas limpezas dos CAPTCHAs, já que o ruído muitas vezes vem em pequenos objetos em que a quantidade de pixels é muito baixa. Assim é possível remover elementos da imagem que não são identificados como letras.

Observa-se na Figura 33 a aplicação das Funções de limpeza na imagem CAPTCHA.

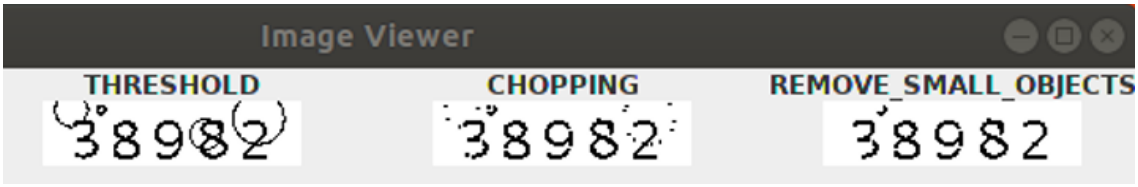

Figura 33 – Funções de Limpeza.

5.2.3 Segmentação

Para possibilitar a identificação do reconhecedor, os caracteres colados, conectados ou de outra maneira juntados precisam ser separados.

Atualmente o DecaptcherFramework possui 6 algoritmos de segmentação. Todos os algoritmos levam em consideração o tamanho mínimo e máximo das letras assim como a quantidade mínima e máxima de letras que a palavra do CAPTCHA possui.

Na configuração desenvolvida, optou-se por utilizar três algoritmos de segmentação presentes no framework. Desta forma, era possível identificar alguns casos em que um algorítmo se mostrava mais eficiente que o outro. Os algoritmos usados são descritos a seguir:

- Histogram: o algoritmo do histograma faz a contagem de pixels pretos para cada região no eixo das abscissas, montando assim um gráfico da imagem mostrando as regiões onde há poucos pixels e portanto mais probabilidade de ser um espaço entre as letras. O histograma então fará a divisão nos vales da linha do gráfico.

- Hough Transform: essa técnica é comumente utilizada na análise de imagens. No caso do Decaptcher, o HoughSegmentator procura por linhas retas predominantemente brancas entre as letras. Essas linhas podem conter alguns pixels pretos assim como podem ser inclinadas mas nunca curvas. Após detectar as linhas, a divisão é feita nesse caminho.

- AutoSegmentator: consiste em um controle geral para os algoritmos de segmentação. Baseado no tamanho médio das letras e no número de caracteres, o AutoSegmentator receberá propostas de divisão dos outros algoritmos de segmentação, e assim escolherá o que melhor se encaixa aos parâmetros do CAPTCHA.

5.2.4 Reconhecimento

O reconhecedor utilizado é do tipo SVM, o modelo do reconhecedor precisa ter sido treinado previamente para ser usado na parte final da tarefa de decifrar o CAPTCHA. Os caracteres segmentados são recebidos pela classe AbstractRecognizer na forma de um array de BufferedImages. Cada carácter então passa por três processos:

- Resize: consiste em mudar o tamanho da imagem do carácter para um tamanho padrão. Conhecido o tamanho, a imagem fica mais fácil de ser vetorizada.
- Vectorize: converte a imagem de um BufferedImage em um array de números float. Depois dessa conversão, o vetor irá ser alimentado ao reconhecedor propriamente.
- Predict probability: o array então é passado para a classe svm da biblioteca libsvm, que possui o método *s*vm-predict-probability. O resultado da classe vem por código *a*scii, portanto convertemos o código em uma string que representa a resposta da predição dada pelo modelo svm.

Uma vez determinados os impedimentos e implementado o necessário, uma configuração é feita que determina, passo a passo, as operações a ser aplicadas à imagem. O resultado é uma lista de pequenas imagens, obtidas do CAPTCHA, em qual cada imagem representa exatamente um caractere sem ruído.

5.2.5 Treinamento

Esta etapa consiste em utilizar as letras obtidas na segmentação para treinar um reconhecedor, geralmente do tipo SVM. Com isso completa-se toda a cadeia do processo de quebra de CAPTCHA.

5.2.6 Testes

Com o Decaptcher pronto, ele é testado localmente a fim de verificar sua assertividade. Para isto, são obtidas 100 novos CAPTCHAs, na qual é medido o percentual acertado. Por serem novas imagens, garante-se que a assertividade obtida não sofre nenhum viés, o que não acontece caso as imagens sejam utilizadas para realizar as configurações e/ou treinar o reconhecedor.

5.2.7 Envio para Ambiente de Produção

Com a assertividade do Decaptcher verificada, o framework é recompilado (caso hajam modificações) e enviado com o reconhecedor e as configurações para a equipe de produção.

5.3 Reconhecedor CNN

Uma vez desenvolvido o reconhecedor por SVM com auxílio do framework Decaptcher disponibilizado pela Neoway, decidiu-se aprimorar os conhecimentos e conceitos na área de solução de CAPTCHAs. Por esse motivo, buscou-se ferramentas e o desenvolvimento de um reconhecedor utilizando-se de redes neurais para validação de conceito.

Conforme citado anteriormente, com o auxílio de materiais e estudos realizados teve-se contato com os frameworks TensorFlow e Keras. Essas duas ferramentas abriram os caminhos para a utilização de redes neurais na solução de CAPTCHA e foram essenciais para alcançar os resultados deste projeto.

A utilização de redes neurais permite diversas abordagens na solução de CAPTCHA, porém optou-se por duas neste projeto. A primeira abordagem segue a mesma linha de desenvolvimento utilizada com SVM, a qual realiza a segmentação da imagem, extração de caracteres de forma individual e treinamento do modelo a partir destas imagens segmentadas. Na segunda abordagem tentou-se retirar a etapa de segmentação, de forma que o treinamento do modelo fosse realizado a partir da imagem com todos os caracteres.

Na sequência serão descritas as etapas desenvolvidas em cada abordagem.

5.3.1 Código Utilizado como Base

De forma a entender melhor os conceitos de redes neurais e a forma de implementação das mesmas foram estudados exemplos de códigos abertos do repositório de códigos do TensorFlow e exemplos de implementação apresentados pelo curso "Complete Guide to TensorFlow for Deep Learning with Python" disponível na plataforma de cursos Udemy [33].

A partir destas fontes, conheceu-se o reconhecedor de dígitos da base de dados MNIST e utilizou-se diversos conceitos do mesmo para compreendimento das ferramentas e implementação de redes neurais.

Com o estudo deste caso e a observação das implementações utilizando o framework TensorFlow percebeu-se que existia uma ferramenta de mais alto nível, o Keras, para a construção de CNNs. Com isso, o projeto desenvolvido foi baseado em cima do Keras.

5.3.2 Primeira Abordagem

5.3.2.1 Leitor e Pré-Processamento da Imagem

Depois de passar pelo pré-processamento de imagem para o desenvolvimento do reconhecedor utilizando SVM e compreender a dificuldade de limpeza e adaptação dos dados para alcançar um resultado satisfatório, estabeleceu-se que o tratamento das imagens utilizando CNN deveria ser o mínimo possível, de forma a fazer com que a rede reconheça e aprenda as dificuldades da imagem por si só.

Nesta etapa, foi desenvolvido um script em Python que cuida de forma sequencial do pré-processamento, da leitura da imagem, da filtragem, da segmentação, da extração de caracteres e da rotulagem.

5.3.2.2 Leitura das Imagens

As imagens são carregadas de um arquivo local para a memória RAM do computador utilizando a biblioteca OpenCV através do método *i*mread.

5.3.2.3 Filtragem

Para este desenvolvimento, buscou-se a simplificação dos métodos de filtragem e limpeza da imagem. Por isso, foram usados dois métodos que são descritos a seguir:

- Escala de Cinza: Ao transformar a imagem em um array representativo considera-se apenas um valor de escala de cinza e padroniza-se os valores entre 0 e 1 de acordo com a intensidade de pixels. A escala de cinza de uma imagem representa para cada valor de pixel uma média dos valores de cor RGB da imagem. Para cada pixel é somado o valor de vermelho com os valores de verde e azul e dividido por 3. Este método utiliza-se da biblioteca OpenCV e de sua função *C*OLOR-BGR2GRAY.
- Recorte: Com o array representativo da imagem, realizou-se por tentativa e erro um recorte da imagem a fim de retirar campos da imagem sem caracteres. Com isso o tamanho da imagem, em pixels, passou de 120 de largura por 30 de altura para 96 de largura por 20 de altura.

5.3.2.4 Segmentação e Extração de Caracteres

Após um longo convívio e análise da imagem ficou nítida uma padronização do espaçamento dos caracteres. Com isso, criou-se uma rotina que realiza a segmentação da imagem de acordo com um número definido de pixels.

Essa rotina de segmentação permitiu obter os caracteres de forma individual. Teve-se apenas o cuidado de repassar o rótulo correto da imagem original ao caractere segmentado para que, posteriormente, pudesse ser armazenado na classe correta e realizado o treinamento supervisionado.

5.3.2.5 Rotulagem

Conforme falado anteriormente, as imagens já estão rotuladas corretamente, ou seja, o nome do arquivo da imagem corresponde a sua solução correta.

No entanto, como as imagens foram segmentadas por dígitos e redistribuídas em pastas fez-se necessário a criação do vetor do rótulo através do algoritmo de codificação *O*ne-hot. O fato das pastas serem nomeadas pelo dígito que continham ajudou a validar a presença de 10 dígitos (0 à 9) e na criação dos vetores para classificação, cada um com 10 posições. Todas as posições são completadas com 0 e em seguida é atribuído o número 1 para a posição referente ao dígito.

5.3.2.6 Conjunto de Dados de Treino e de Teste

Para o desenvolvimento desta solução definiu-se, para o treinamento, um conjunto composto de 400 imagens. Este número é pequeno se considerarmos estas imagens como sendo as entradas de treinamento da rede, porém estas 400 imagens são compostas por 5 caracteres cada. Dada essa condição e realizando a segmentação de dígitos, passou-se a ter um conjunto de 2 mil imagens, cada qual com apenas um caractere. Essa quantidade foi considerada razoável para início e demonstrou-se suficiente após a obtenção dos resultados.

Para o teste teve-se o cuidado de utilizar imagens diferentes das usadas na etapa de treinamento. Inicialmente separou-se 2 mil imagens para o teste de acurácia, porém com o resultado obtido foram separadas mais 20 mil imagens novas para teste.

5.3.2.7 Arquitetura da Rede Neural

A arquitetura implementada é uma adaptação dos estudos realizados [34]. Ela pode ser visualizada através da Figura 34 e é formada por uma camada de entrada, 2 camadas convolucionais, 1 camada completamente conectada e 1 camada completamente conectada de saída. Entre as camadas convolucionais existe uma camada de ativação e uma camada de *pooling* e antes das camadas completamente conectadas há uma camada de Flatten. A camada de Flatten é responsável por transformar matrizes n-dimensionais em matrizes de uma dimensão.

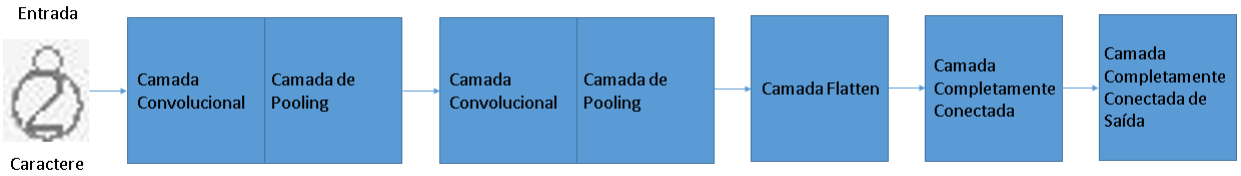

Figura 34 – Arquitetura da Rede Neural.

5.3.2.8 Entradas

A camada de entrada recebe dois parâmetros, sendo eles as imagens de entrada e os rótulos correspondentes.

5.3.2.9 Camadas

As camadas descritas na arquitetura serão expostas, em ordem, e tem seu funcionamento explicado a seguir:

1. Camada de entrada: array multidimensional no formato 20x20x1 alimentado pelos valores das imagens segmentadas em dígitos.

2. Camada Convolucional: recebe a imagem da camada de entrada com valor 1 de profundidade. Realiza convoluções nas entradas com um *kernel* de formato 5x5 e 20 valores de profundidade. Tem valor de *stride* igual a 1x1 e usa *padding* do tipo *same padding*.

3. Camada Oculta: tem como entrada a camada convolucional anterior e utiliza a função de ativação ReLU.

4. Camada de *Pooling*: tem como entrada as imagens que passaram pela convolução e pela função de ativação. Realiza a operação de *max pooling* com um *kernel* de formato 2x2 e *stride* igual a 2x2. Esta camada reduz o tamanho da imagem pela metade.

5. Camada Convolucional: recebe a imagem da camada anterior com valor 64 de profundidade. Realiza convoluções na imagem com um kernel de formato 5x5 e 50 valores de profundidade. Tem valor de *stride* igual a 1x1 e usa *padding* to tipo *same padding*.

6. Camada Oculta: tem como entrada a camada convolucional anterior e utiliza a função de ativação ReLU.

7. Camada de *Pooling*: tem como entrada as imagens que passaram pela convolução e pela função de ativação. Realiza mesma operação da camada de *pooling* citada no item 4 e possui as mesmas configurações.

8. Camada de *Flatten*: recebe como entrada a imagem da camada de *pooling* e reduz a profundidade para 1, adaptando a imagem para a camada seguinte. Com o TensorFlow, haveria a necessidade de descobrir o tamanho do vetor de saída das camadas convolucionais para redimensioná-lo, porém com o uso do Keras não é preciso se preocupar com isso.

9. Camada Completamente Conectada: tem como entrada todas as ativações das camadas anteriores devidamente formatadas pela camada de *flatten*. Esta camada utiliza a função de ativação ReLU.

10. Camada Completamente Conectada de Saída: esta camada utiliza a função softmax para classificação das imagens recebidas como entrada. Além disso, ela recebe o valor da quantidade de classes possíveis, no caso do projeto esse número é 10, que representa 10 possíveis dígitos.

Com a utilização do Keras essa implementação se torna muito simples e requer apenas 8 linhas de código. Como citado anteriormente, o Keras é um framework que roda em cima do TensorFlow e abstrai, de forma automática, muitas preocupações que o modelista da rede precisaria ter.

5.3.2.10 Configuração

Desenvolvida a arquitetura da CNN com o Keras, ainda falta especificar a função de perda e o tipo do otimizador a ser utilizado. Em Keras, isso pode ser executado em um comando:

model.compile(loss="categorical-crossentropy", optimizer="adam", metrics=["accuracy"])

Neste projeto, usou-se a função de perda cross entropy para classificação de classe categórica. Para o tipo do otimizador optou-se pelo Adam. Também pode-se especificar uma métrica que será calculada quando executado o método evaluate no modelo. O Keras também torna isso fácil.

5.3.2.11 Treinamento

Em seguida, deseja-se treinar o modelo. Isso é feito através da execução de um único comando em Keras, apresentado a seguir:

model.fit(X-train, Y-train, validation-data=(X-test, Y-test), batch-size=32, epochs=10, verbose=1)

Primeiramente são fornecidos todos os dados de treinamento - representados por x-train e y-train. Após isso, é passado o argumento do tamanho do lote. Neste caso, definiu-se um tamanho de lote de 32. O tamanho do lote é definido pelo número de exemplos do conjunto total passados por etapa de treino. Em seguida, definiu-se o número de epoch em 10. O epoch é a quantidade de vezes que o conjunto de dados inteiro passa pela rede neural. O verbose, definido como 1, especifica o desejo que informações detalhadas sejam impressas no console sobre o andamento do treinamento. Durante o treinamento, se verbose estiver definido como 1, será enviado para o console os textos apresentados pela Figura 35

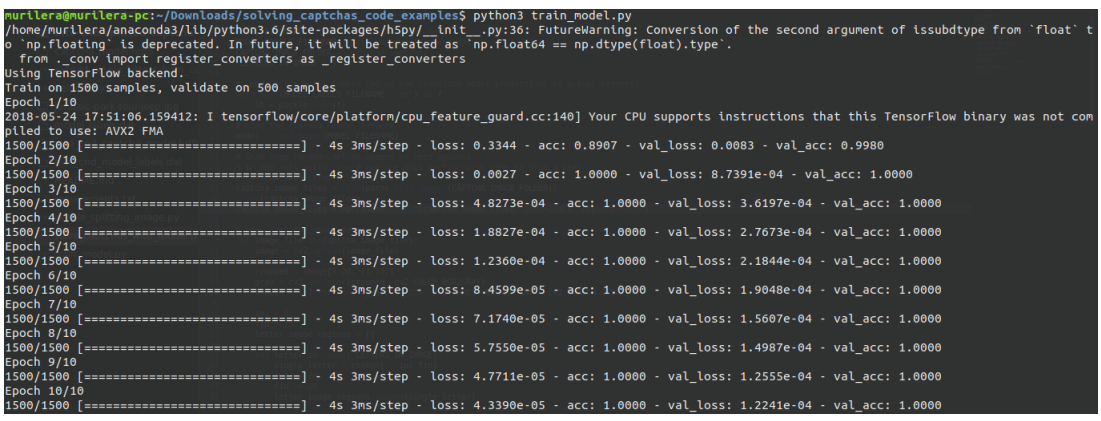

Figura 35 – Passos de Treinamento visualizados com verbose igual a 1.

5.3.2.12 Teste

Ao final do treinamento e da otimização do modelo desenvolveu-se um script para realizar o teste de acurácia. Esse script recebe como entrada o conjunto de dados de treino separados anteriormente e os rótulos das imagens. Em seguida, são realizadas as etapas de filtragem e segmentação das imagens. Com isso, as imagens segmentadas são enviadas como entrada para o modelo gerado anteriormente através do método model.predict, que retorna o valor da predição do modelo. Este valor recebe o tratamento necessário de codificação para formar a resposta predita pelo modelo da imagem de entrada.

5.3.3 Segunda Abordagem

Após o desenvolvimento realizado pela primeira abordagem foi possível validar a eficiência da rede neural aplicada ao reconhecimento de textos em imagens CAPTCHA. Contudo, surgiu o desejo de facilitar ainda mais os processos de desenvolvimento a ponto de retirar as etapas de pré processamento de imagem, filtragem, limpeza e segmentação.

Com a ausência destas etapas, a rede passa a ter novas necessidades para realizar o reconhecimento. A primeira dificuldade advém da quantidade de imagens necessárias no conjunto de dados, uma vez que não há mais segmentação por caracteres. A segunda dificuldade se dá pela configuração da rede, que passa a ter mais camadas e mais parâmetros de controle. Quantidade de ativações, tamanho do kernel, taxa de aprendizado, momentum, dropout e número de iterações são alguns desses parâmetros, além da necessidade de maior poder de processamento computacional.

Mesmo as dificuldades citadas, um script foi desenvolvido e uma rede foi construída para realizar o treinamento a partir de imagens sem segmentação. Inicialmente foi utilizado como base o desenvolvimento da primeira abordagem, com a mesma configuração da rede neural, porém com as devidas alterações e simplificações de limpeza, filtragem e segmentação.

Em sequência, visto o desempenho insatisfatório da rede para este desafio, procurou-se aumentar o número de camadas convolucionais e os tamanhos das profundidades das camadas. Esta alteração também foi ineficaz.

Outras várias alterações foram feitas e testadas de maneira empírica na tentativa de alcançar um resultado minimamente satisfatório, porém até o momento não foi possível.

Por estes e outros motivos a segunda abordagem é um estudo em aberto, com muito potencial para evolução. Na mesma medida que facilita nas questões de pré processamento de imagem, a abordagem se mostra dificultosa na construção da rede neural e na parametrização da mesma.

6 Resultados

Neste capítulo serão apresentados os resultados obtidos com as atividades desenvolvidas do projeto proposto. Estes resultados advêm das etapas de treinamento e de testes de acurácia desenvolvidos para os reconhecedores apresentados neste documento.

6.1 Reconhecedor por SVM

Este reconhecedor utilizou-se de um framework disponibilizado pela Neoway e tem um relatório de informações que são apresentadas ao término do desenvolvimento. O relatório traz uma série de informações que podem ser vistas na Figura 36.

Da Figura 36 obtêm-se informações a respeito da performance de assertividade por classe, ou seja, para cada dígito é possível verificar o número de reconhecimentos corretos realizados pelo modelo desenvolvido e também a porcentagem desta assertividade. Estes valores são reproduzidos na Tabela 2.

Classe	Corretos	Incorretos	Porcentagem de Acerto
0	27	5	84,4%
1	186	3	98,4%
2	120	4	96,8%
3	123	7	94,6%
4	139	1	99,3%
5	204	11	94,9%
6	134	3	97,8%
7	154	1	99,4%
8	78	1	98,7%
9	47	2	95,9%

Tabela 2 – Valores de assertividade por Classe

Além disso, é possível verificar a quantidade de reconhecimentos completamente e parcialmente corretos. Os valores dessa verificação são apresentados na Tabela 3.

Por fim, o relatório traz a informação da assertividade do modelo, o qual atingiu o valor de 88%. Este valor foi determinado como satisfatório e o modelo foi inserido no ambiente de produção o mais rápido possível.

Além da redução de custos causada pela substituição do serviço externo de quebra de CAPTCHA pelo modelo desenvolvido, outro fator que se destaca é do ganho em tempo de resposta. As requisições feitas pelo bot para a captura de dados não precisam mais estabelecer uma comunicação com um serviço externo, esperar eles gerarem a resposta

```
 ### CHARACTER PERFORMANCE ###
char -> correctAnswers/occurrences ..... correctAnswersPerc
0  ->   27/32   .....  84.4%
1  ->   186/189 .....  98.4%
2  ->   120/124 .....  96.8%
3  ->   123/130 .....  94.6%
4  ->   139/140 .....  99.3%
5  ->   204/215 .....  94.9%
6  ->   134/137 .....  97.8%
7  ->   154/155 .....  99.4%
8  ->   78/79   .....  98.7%
9  ->   47/49   .....  95.9%

 ##############################


 ### PARTIAL AGREE BY CAPTCHA ###
numOfCharCorrect -> numOfCaptchas/total ..... percentageOfCaptchaInThisCategory
1   ->   2/250  .....  0.8%
3   ->   2/250  .....  0.8%
4   ->   26/250 .....  10.4%
5   ->   220/250        .....  88.0%

 ##############################


identified/correctedSegmented/total:    220/248/250
max assertiveness of segmentation:      99.20%
max assertiveness of recognition:       88.71%
general assertiveness of process:       88.00%
average time of processing per captcha: 22.77 ms
average velocity of processing:         43.91 captchas/sec

Bom dia,
Segue o Decaptcher do mteCnd-1.0

Conf @ /DecaptcherFramework/cfg/mteCnd-1.0.dcj
Rec @ /DecaptcherFramework/../decaptcher-resources/recognizers/mteCnd-1.0/16-16-mteCnd-1.0.svm
Assertividade 88% +-6%


##############################

BUILD SUCCESSFUL

Total time: 12.133 secs
2018-04-03 18:00:56 UTC [     info] Cleaning up. Done
```

Figura 36 – Relatório gerado pelo framework Decaptcher.

e receber a solução deles, apenas precisa chamar a execução de um serviço interno que executa no mesmo ambiente e fornece a solução de uma forma mais rápida, acelerando o processo de captura.

Na Figura 37 é demonstrado alguns casos em que o reconhecedor acertou o texto representado na imagem.

Conclui-se, portanto, que o objetivo foi alcançado e os resultados para este reconhecedor foram satisfatórios.

Quantidade de Caracteres Corretamente Reconhecidos	Quantidade de Ocorrências	Porcentagem de Ocorrência
1	2	0,8%
3	2	0,8%
4	26	10,4%
5	220	88,0%

Tabela 3 – Valores de Reconhecimentos Totais ou Parciais

Figura 37 – Imagens e soluções geradas pelo reconhecedor SVM.

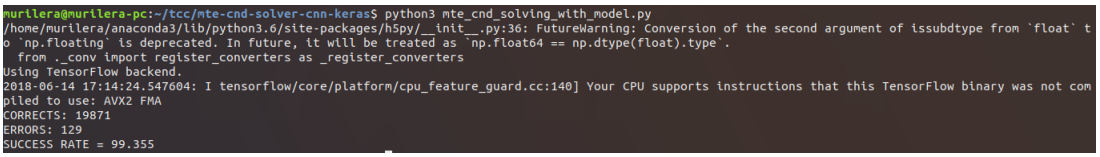

Figura 38 – Resumo de assertividade do reconhecedor por CNN.

6.2 Reconhecedor por CNN

Mesmo com o resultado satisfatório obtido com o desenvolvimento do reconhecedor por SVM, optou-se por buscar um aprimoramento na assertividade e também nos conceitos utilizados para solucionar o problema enfrentado. Dedicou-se tempo, esforços e recursos para o aprendizado dos conceitos de redes neurais e das ferramentas necessárias para o desenvolvimento deste reconhecedor no intuito de validar, documentar e acrescentar nas tecnologias exploradas dentro da Neoway.

Como esperado, o reconhecedor por CNN mostrou-se ser uma solução mais efetiva para o problema proposto e atingiu um resultado mais expressivo com um menor esforço de desenvolvimento. Os valores de assertividade por classe deste reconhecedor podem ser observados na Tabela 4.

Na Figura 38 são apresentados alguns resultados obtidos com este reconhecedor. É possível verificar que a assertividade atingiu 99,35% em um teste realizado com 20 mil imagens, onde 19.871 imagens foram classificadas da forma correta

Este aumento de assertividade proporcionado pelo uso de redes neurais na construção do reconhecedor traz benefícios de produtividade e reduz a necessidade de refazer

Classe	Corretos	Incorretos	Porcentagem de Acerto
0	6496	4	99,93%
1	8614	1	99,98%
2	8702	8	99,90%
3	8484	73	99,14%
4	8846	0	100%
5	8589	42	99,51%
6	8555	14	99,83%
7	7056	2	99,97%
8	8659	32	99,74%
9	7949	16	99,79%

Tabela 4 – Valores de Assertividade do Reconhecedor por CNN

consultas por envio da solução errada da imagem solicitada. Em outras palavras, ao fazer uma requisição é preciso, conforme citado anteriormente, preencher informações de CNPJ ou CPF e a solução correta da imagem CAPTCHA. Uma vez que a solução da imagem é enviada de forma incorreta, faz-se necessário o envio de uma nova requisição e isso gera custos de retrabalho, pode-se dizer.

Alguns reconhecimentos realizados corretamente pelo modelo podem ser observados na Figura 39, enquanto alguns casos onde o modelo não determinou a resposta podem ser vistos na Figura 40.

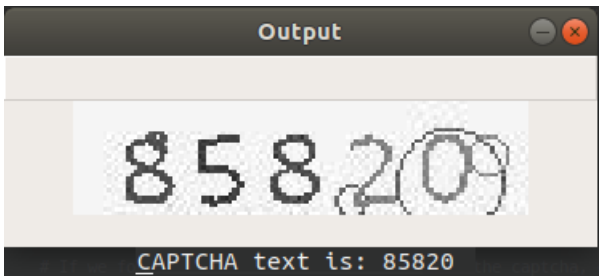

Figura 39 – Reconhecimentos corretos por CNN.

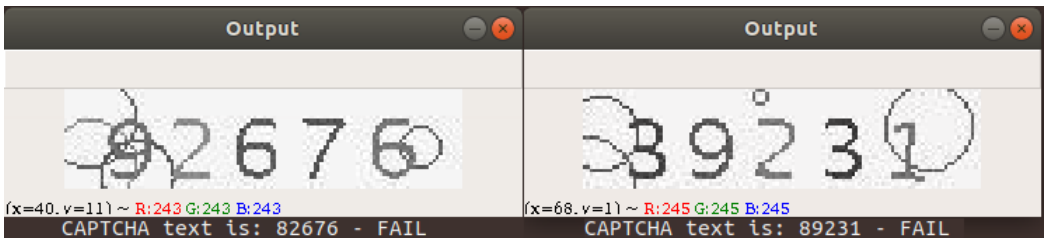

Figura 40 – Reconhecimentos incorretos por CNN.

6.3 Comparativo

Para efeitos de comparação entre os reconhecedores desenvolvidos apresenta-se a Tabela 5 a seguir.

	Reconhecedor por SVM	Reconhecedor por CNN
Acertos	220	19871
Erros	30	129
Assertividade	86%	99,355%

Tabela 5 – Comparativo dos Reconhecedores

7 Conclusões e Perspectivas

Este documento apresentou o desenvolvimento de dois reconhecedores de textos em imagens CAPTCHA através dos conceitos de SVM e redes neurais, os quais foram utilizados no contexto do estágio realizado na Neoway.

O objetivo desejado com o projeto foi alcançado. Com o uso de ferramentas internas e ferramentas focadas em tarefas de aprendizado de máquina foi possível chegar a dois resultados satisfatórios.

No primeiro momento, utilizando o framework Decaptchar para o desenvolvimento do reconhecedor foi possível alcançar uma assertividade de 88%. Esse valor, por si só, já foi considerado surpreendente dentro do repositório de reconhecedores desenvolvidos pela Neoway.

No segundo momento, desenvolveu-se um novo reconhecedor utilizando conceitos de redes neurais convolucionais, o qual atingiu uma assertividade de 99,35%. Este mostrou-se melhor que a abordagem por SVM e serviu para validar a capacidade desta abordagem como uma solução alternativa.

Além dos ótimos resultados obtidos, tem-se um ganho financeiro com o desenvolvimento destas soluções. Conforme explicitado anteriormente, todos os desenvolvimentos foram feitos em cima da imagem CAPTCHA do site eletrônico do MTE. Através de informações de captura fornecidas pela Neoway, sabe-se que foram processadas 625.606 consultas até o momento. Para essas consultas é preciso solucionar, de forma ideal, 625.606 imagens CAPTCHA que gera um custo de $940,00 quando utilizado um serviço externo. Contudo, com os reconhecedores desenvolvidos, principalmente o que utiliza redes neurais, evita-se todo o custo e tem-se uma assertividade praticamente ideal.

Do ponto de vista conceitual das imagens CAPTCHA fica provada a ineficiência neste caso em proteger o site eletrônico de consultas automatizadas. Por esse motivo, o site eletrônico está também vulnerável a um volume muito grande de consultas, o que pode causar instabilidades e, até mesmo, a queda do serviço fornecido. Além disso, pode-se criar uma discussão a respeito da produção de imagens CAPTCHA mais difíceis de serem reconhecidas através de mais ruídos e distorções. Contudo, tornar mais difíceis as imagens pode dificultar tanto o acesso automatizado quanto o acesso de humanos, ou seja, vai contra os propósitos iniciais do CAPTCHA. Portanto, deve-se realizar estudos para novas formas de proteção contra acessos automatizados.

7.1 Perspectivas Futuras

Após todos os desafios enfrentados no desenvolvimento do projeto, alguns pontos ficaram em abertos e são citados a seguir como melhorias futuras:

- Incorporar o reconhecedor desenvolvido com o uso de redes neurais na infraestrutura de produção da Neoway.
- Aprimorar o reconhecedor, que não se utiliza de pré-processamento de imagens, a fim de atingir um resultado que pode ser usado.
- Estender a rede neural construída para realizar o reconhecimento de outros tipos de CAPTCHA de outros sites eletrônicos.
- Desenvolver um script generalista que realiza o pré-processamento de imagens e a correta segmentação de qualquer imagem a ele fornecido.
- Desenvolver uma arquitetura de múltiplas redes neurais que se adaptam e são ativadas de forma automática pelo nível do desafio enfrentado.

Novamente, ressalta-se que os conceitos de redes neurais são extremamente eficazes para este tipo de problema e incentiva-se o avanço dos estudos relacionados a esta área.

Referências

1 NEOWAY Produtos. Acesso em: 20 jan. 2018. Disponível em: <https://www.neoway.com.br/>. Citado 3 vezes nas páginas 19, 21 e 23.

2 RUSSEL, S.; NORVIG, P. *Inteligência Artificial: Uma abordagem moderna*. 3a. ed. [S.l.: s.n.], 2009. Citado na página 30.

3 REHM, F. e. a. *Aplicações da Inteligência Artificial em Jogos*. [S.l.: s.n.], 2009. Citado na página 31.

4 MACHINE, S. vector (Ed.). Acesso em: 30 maio 2018. Disponível em: <https://en.wikipedia.org/wiki/Support_vector_machine>. Citado na página 31.

5 BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory – COLT '92*, 1992. Citado na página 31.

6 GEVERT, V. G. et al. Modelos de regressão logística, redes neurais e support vector machine (svms) na análise de crédito a pessoas jurídicas. *Vanessa Terezinha Ales*, 2010. Citado na página 31.

7 PIYUSH, R. Hyperplane based classification: Perceptron and (intro to) support vector machines. *CS5350/6350: Machine Learning*, 2011. Citado na página 31.

8 LORENA, A. C.; CARVALHO, A. C. P. L. F. de. Uma introdução às support vector machines. *RITA*, 2007. Citado na página 32.

9 GAONKAR, B.; DAVATZIKOS, C. Analytic estimation of statistical significance maps for support vector machine based multi-variate image analysis and classification. Citado na página 32.

10 MACKAY, D. J. C. *Information Theory, Inference and Learning Algorithms*. [S.l.]: Cambridge University Press, 2015. ISBN 9780521670517. Citado na página 33.

11 ACADEMY, D. S. *Deep Learning Book*. Disponível em: <http://deeplearningbook.com.br/>. Citado 2 vezes nas páginas 36 e 39.

12 UJJWALKARN. *A Quick Introduction to Neural Networks*. 2016. Disponível em: <https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/>. Citado na página 37.

13 REIS, B. *Redes Neurais – Funções De Ativação*. 2016. Disponível em: <http://www.decom.ufop.br/imobilis/redes-neurais-funcoes-de-ativacao/>. Citado na página 38.

14 PINTO, V. A. Redes neurais convolucionais de profundidade para reconhecimento de textos em imagens de captcha. 2016. Citado 2 vezes nas páginas 38 e 41.

15 MARTINS, P. B. de M. Aplicação de redes neurais geradoras adversárias para colorização de imagens em preto e branco. 2017. Citado na página 39.

16 KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. *ICLR*, 2015. Citado na página 40.

17 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org>. Citado na página 41.

18 ANDRADE, L. N. de. Redes neurais artificiais aplicadas na identificação automática de áreas cafeeiras em imagens de satélite. 2011. Citado na página 41.

19 SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014. Citado na página 41.

20 BALDI, P.; SADOWSKI, P. Understanding dropout. *Working Paper*, 2014. Citado na página 41.

21 LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. Citado na página 42.

22 PACHECO, A. *Redes Neurais Convolutivas – CNN*. 2017. Disponível em: <http://www.computacaointeligente.com.br/artigos/redes-neurais-convolutivas-cnn/>. Citado na página 42.

23 HAYKIN, S. *Redes Neurais: Princípios e Prática*. [S.l.: s.n.], 2007. Citado na página 42.

24 WIKIPEDIA. *CAPTCHA*. Disponível em: <https://en.wikipedia.org/wiki/CAPTCHA>. Citado na página 46.

25 KOPP, M.; NIKL, M.; HOLENA, M. Breaking captchas with convolutional neural networks. *CEUR Workshop Proceedings*, 2017. Citado na página 46.

26 BURSZTEIN, E. et al. The failure of noise-based non-continuous audio captchas. *IEEE Symposium on Security and Privacy*, 2011. Citado na página 46.

27 KAUR, K.; BEHAL, S. Captcha and its techniques: A review. *International Journal of Computer Science and Information Technologies*, 2014. Citado 3 vezes nas páginas 46, 47 e 48.

28 TAM, J. et al. Breaking audio captchas. Disponível em: <https://papers.nips.cc/paper/3443-breaking-audio-captchas.pdf>. Citado na página 47.

29 CARVALHO, A. P. de Leon F. de. *Desenvolvimento de Aplicações*. Disponível em: <http://conteudo.icmc.usp.br/pessoas/andre/research/neural/desenv.htm>. Citado na página 53.

30 TENSORFLOW. Disponível em: <https://www.tensorflow.org/>. Citado na página 55.

31 PYSCIENCE-BRASIL. *Numpy*. Disponível em: <http://pyscience-brasil.wikidot.com/module:numpy>. Citado na página 56.

32 OPENCV. *About*. Disponível em: <https://opencv.org/about.html>. Citado na página 56.

33 PORTILLA, J. *Complete Guide to TensorFlow for Deep Learning with Python*. Disponível em: <https://www.udemy.com/complete-guide-to-tensorflow-for-deep-learning-with-python/>. Citado na página 62.

34 KERAS. *Keras: Deep Learning for humans*. Disponível em: <https://github.com/keras-team/keras/tree/master/examples>. Citado na página 64.

Apêndices

APÊNDICE A
– Mte-cnd-splitting-image.py

```
import os
import os.path
import cv2
import glob

_CAPTCHAS_FOLDER = "mte-cnd-captchas"
_OUTPUT_FOLDER = "mte-cnd-splitted-captcha-letters"
_LETTERS_IN_IMAGE = 5

captcha_image_files = glob.glob(os.path.join(_CAPTCHAS_FOLDER, "*"))
counts = {}

for (i, captcha_image_file) in enumerate(captcha_image_files):
    print("[INFO] processing image {}/{}".format(i + 1, len(
        captcha_image_files)))

    filename = os.path.basename(captcha_image_file)
    captcha_correct_text = os.path.splitext(filename)[0]

    image = cv2.imread(captcha_image_file)
    cropped = image[8:28, 11:107]
    gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV | cv2.
        THRESH_OTSU)[1]

    ini = 0
    fin = 15
    letter_image_regions = []
    for letter in range(_LETTERS_IN_IMAGE):
        single_letter = thresh[:, ini:fin]
        ini += 20
        fin += 20
        letter_image_regions.append(single_letter)
        cv2.imshow("a", single_letter)
        cv2.waitKey(0)

    for letter_image, letter_text in zip(letter_image_regions,
        captcha_correct_text):
        save_path = os.path.join(_OUTPUT_FOLDER, letter_text)
        if not os.path.exists(save_path):
```

```
            os.makedirs(save_path)
39
        count = counts.get(letter_text, 1)
41      p = os.path.join(save_path, "{}.png".format(str(count).zfill(6)))
        cv2.imwrite(p, letter_image)
43
        counts[letter_text] = count + 1
```

APÊNDICE B – Mte-cnd-solving-with-model.py

```
1
import os
3 import os.path
from keras.models import load_model
5 from helpers import resize_to_fit
from imutils import paths
7 import numpy as np
import cv2
9 import pickle


11
MODEL_FILENAME = "mte_cnd_captcha_model.hdf5"
13 MODEL_LABELS_FILENAME = "mte_cnd_model_labels.dat"
CAPTCHA_IMAGE_FOLDER = "mte-cnd-no-format"
15 _LETTERS_IN_IMAGE = 5

17 with open(MODEL_LABELS_FILENAME, "rb") as f:
    lb = pickle.load(f)
19
model = load_model(MODEL_FILENAME)
21
captcha_image_files = list(paths.list_images(CAPTCHA_IMAGE_FOLDER))
23 captcha_image_files = np.random.choice(captcha_image_files, size=(20000,),
    replace=False)

25 errors = 0
corrects = 0
27
for image_file in captcha_image_files:
29     filename = os.path.basename(image_file)
    captcha_correct_text = os.path.splitext(filename)[0]
31
    image = cv2.imread(image_file)
33     if image is not None:
        cropped = image[8:28, 11:107]
35         gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
        thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV | cv2.
            THRESH_OTSU)[1]
37
        ini = 0
```

```
        fin = 15
        letter_image_regions = []

        for letter in range(_LETTERS_IN_IMAGE):
            single_letter = thresh[:, ini:fin]
            ini += 20
            fin += 20
            letter_image_regions.append(single_letter)

        if len(letter_image_regions) != 5:
            continue

        output = image
        predictions = []

    for letter_image in letter_image_regions:
        letter_image = resize_to_fit(letter_image, 20, 20)

        letter_image = np.expand_dims(letter_image, axis=2)
        letter_image = np.expand_dims(letter_image, axis=0)

        prediction = model.predict(letter_image)

        letter = lb.inverse_transform(prediction)[0]
        predictions.append(letter)

    captcha_text = "".join(predictions)

    if captcha_text == captcha_correct_text:
        print("CAPTCHA text is: {} - SUCCESS".format(captcha_text))
        corrects = corrects + 1
    else:
        print("CAPTCHA text is: {} - FAIL".format(captcha_text))
        errors = errors + 1

print("CORRECTS: {}".format(corrects))
print("ERRORS: {}".format(errors))
print("SUCCESS RATE = {}".format((corrects * 100) / 20000))
```

APÊNDICE C – Mte-cnd-train-model.py

```
import cv2
import pickle
import os.path
import numpy as np
from imutils import paths
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Flatten, Dense
from helpers import resize_to_fit


LETTER_IMAGES_FOLDER = "mte-cnd-splitted-captcha-letters"
MODEL_FILENAME = "mte_cnd_captcha_model.hdf5"
MODEL_LABELS_FILENAME = "mte_cnd_model_labels.dat"

data = []
labels = []

for image_file in paths.list_images(LETTER_IMAGES_FOLDER):
    image = cv2.imread(image_file)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    image = resize_to_fit(image, 20, 20)
    image = np.expand_dims(image, axis=2)

    label = image_file.split(os.path.sep)[-2]

    data.append(image)
    labels.append(label)


data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)

(X_train, X_test, Y_train, Y_test) = train_test_split(data, labels,
    test_size=0.25, random_state=0) #random_state

lb = LabelBinarizer().fit(Y_train)
Y_train = lb.transform(Y_train)
Y_test = lb.transform(Y_test)
```

```
43 with open(MODEL_LABELS_FILENAME, "wb") as f:
       pickle.dump(lb, f)
45
   model = Sequential()
47
   model.add(Conv2D(20, (5, 5), padding="same", input_shape=(20, 20, 1),
       activation="relu"))
49 model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

51 model.add(Conv2D(50, (5, 5), padding="same", activation="relu"))
   model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
53
   model.add(Flatten())
55 model.add(Dense(500, activation="relu"))

57 model.add(Dense(10, activation="softmax"))

59 model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["
       accuracy"])

61 model.fit(X_train, Y_train, validation_data=(X_test, Y_test), batch_size
       =10, epochs=10, verbose=1)

63 model.save(MODEL_FILENAME)
```