UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO

WAYNER MOYSÉS MARCELINO

SOLUÇÃO DE CAPTCHA BASEADO EM IMAGEM UTILIZANDO CLASSIFICADORES MULTICLASSE E DE CLASSE ÚNICA

VITÓRIA
2021

WAYNER MOYSÉS MARCELINO

Solução de CAPTCHA baseado em imagem utilizando classificadores multiclasse e de classe única

Dissertação apresentada ao Programa de Pós-Graduação em Matemática do Centro de Ciências Exatas da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Mestre em Matématica, na área de concentração Matemática Aplicada e Probabilidade, orientado pelo Prof.º Dr.º Fabiano Petronetto do Carmo e coorientado pelo Prof.º Dr.º Thales Vieira (UFAL).

VITÓRIA
2021

Solução de Captcha Baseado em Imagem Utilizando Classificadores Multiclasse e de Classe Única

Wayner Moysés Marcelino

Dissertação submetida ao Programa de Pós-Graduação em Matemática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Matemática.

Aprovada em 24 de setembro de 2021 por:

Prof. Dr Fabiano Petronetto do Carmo
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Thales Miranda de Almeida Vieira
Instituto de Computação / Universidade Federal de Alagoas
Coorientador

Prof. Dr. Afonso Paiva Neto
Instituto De Ciências Matemáticas e de Computação / Universidade de São Paulo

Prof. Dr. Alcebíades Dal Col Junior
Universidade Federal do Espírito Santo

Universidade Federal do Espírito Santo
Vitória, setembro de 2021

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
ALCEBIADES DAL COL JUNIOR - SIAPE 1027525
Departamento de Matemática - DM/CCE
Em 29/09/2021 às 22:32

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/277037?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
FABIANO PETRONETTO DO CARMO - SIAPE 1721397
Departamento de Matemática - DM/CCE
Em 01/10/2021 às 09:06

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/278359?tipoArquivo=O

AGRADECIMENTOS

Agradeço a Deus pela oportunidade de ter feito essa jornada;

À minha esposa, Renata. Eu sei que não foi fácil, mas sem o seu apoio esse trabalho jamais seria possível;

Também aos meus filhos que pacientemente entenderam que era necessário ficar sem minha companhia em alguns momentos de lazer;

Aos meus colegas do PPGMAT que percorreram toda ou parte dessa jornada. Obrigado pela oportunidade de conviver com vocês;

Aos professores do PPGMAT que com muita competência se dedicaram a minha formação, em particular meu orientador Professor Fabiano Petronetto do Carmo, juntamente com meu co-orientador Professor Thales Vieira da Universidade Federal de Alagoas;

Por fim, a CAPES pelo suporte financeiro.

RESUMO

O uso de CAPTCHA hoje é comum para a proteção de serviços de internet para garantir seu bom e seguro funcionamento, no entanto, pode ser um inconveniente para alguns usuários que queiram acesso à esses serviços de forma correta. Um desses "usuários" são os robôs que, sem nenhuma intenção desonesta, são desenvolvidos para a automação de processos nas empresas. Um dos principais tipos de CAPTCHA utilizados é o baseado em imagens que consiste na escolha das imagens do objeto solicitado dentre imagens distintas, sendo o reCAPTCHA da Google o mais utilizado. O objetivo deste trabalho é dotar um modelo que seja capaz de solucionar esses desafios sem explorar qualquer vulnerabilidade técnica e que possa ser utilizado por ferramentas de automação de processos. Foram estudados e treinados 3 modelos: um multiclasse, implementada numa rede neural convolucional (CNN), e dois de classe única conhecidos como One-Class SVM (OC-SVM) e kernel PCA (kPCA). Os três modelos foram treinados com um banco de imagens disponível na internet e avaliados sobre um conjunto de desafios formados por imagens obtidas no próprio reCAPTCHA. Foram elaborados quatro cenários de avaliação que se diferem pela forma de escolha das imagens e pelo critério para determinar se o desafio foi solucionado. Nos dois primeiros cenários o modelo CNN teve melhor desempenho com 55% dos desafios solucionados. O modelo kPCA e OC-SVM tiveram desempenho semelhante no terceiro cenário resolvendo 46 e 44% dos desafios, respectivamente. No quarto e último cenário o kPCA resolveu 71% dos desafios, enquanto o OC-SVM apenas 65%. Nestes últimos cenários o modelo CNN não foi avaliado por não ser aplicável a eles.

Palavras-chave: reCAPTCHA, CAPTCHA, Redes Neurais Convolucionais, Reconhecimento de Imagem, Visão Computacional, One-Class SVM, kernel PCA.

ABSTRACT

The use of CAPTCHA today is common for the protection of internet services to ensure their smooth and safe operation, however, it can be an inconvenience for some users who want to access these services correctly. One of these “users” are robots that, without any dishonest intention, are developed for the automation of processes in companies. One of the main types of CAPTCHA used is the image-based one, which consists of choosing the images of the requested object among different images, with Google’s reCAPTCHA being the most used. The objective of this work is to provide a model that is capable of solving these challenges without exploiting any technical vulnerability and that can be used by process automation tools. Three models were studied and trained: a multiclass, implemented in a convolutional neural network (CNN), and two of a single class known as One-Class SVM (OC-SVM) and kernel PCA (kPCA). The three models were trained with images availables on the internet and evaluated on a set of challenges formed by images obtained from reCAPTCHA itself. Four evaluation scenarios were elaborated, which differ in the way the images are chosen and in the criteria to determine if the challenge was solved. In the first two scenarios the CNN model performed better with 55% of the challenges solved. The kPCA and OC-SVM model had similar performance in the third scenario, solving 46 and 44% of the challenges, respectively. In the fourth and final scenario, kPCA solved 71% of the challenges, while OC-SVM only 65%. In these last scenarios the CNN model was not evaluated because it isn’t applicable to them.

Keywords: reCAPTCHA, CAPTCHA, Convolutional Neural Networks, Image Recognition, Computer Vision, One-Class SVM, kernel PCA.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

CAPTCHA é um acrônimo da expressão em inglês “Completely Automated Public Turing test to tell Computers and Humans Apart” surgida em 2003 [VABHL03], que em tradução livre é Teste Público de Turing Completamente Automatizado para diferenciar Computadores e Humanos. É usado para impedir o uso automatizado de serviços online. Em síntese, são pequenos desafios facilmente resolvíveis por seres humanos, mas, em tese, difíceis para computadores.

O CAPTCHA é utilizado para a segurança do cenário moderno de serviços de Internet, implantado essencialmente em serviços de autenticação, permitindo distinguir seres humanos autorizados de ataques automatizados.

O uso de CAPTCHAS tornou-se uma necessidade a partir do uso generalizado de robôs automatizados para aumentar a escala de atividades online nefastas. Portanto, são mecanismos de defesa contra fraudadores e são utilizados para prevenir, entre outras coisas, a criação em massa de contas e postagem de mensagens em redes sociais.

O primeiro CAPTCHA, utilizado em 1997, consistia numa imagem contendo um conjunto de letras, portando baseado em texto, que deveria ser lido pelo usuário para verificação do computador. Com o passar do tempo e o aumento da capacidade computacional, foi superado. Para dificultar o reconhecimento dos caracteres por algum algoritmo, utiliza-se a adição de características de anti-segmentação, que busca tornar a segmentação dos caracteres impossível por meio de ruídos e traços na imagem, e de anti-reconhecimento, que consiste no uso de letras com diferentes tamanhos, rotacionadas e deformadas [VABHL03].

Como opção ao CAPTCHA baseado em texto, surgiu um novo tipo baseado em áudio para que pessoas com dificuldades visuais ou de leitura pudessem utilizar os serviços protegidos pelo CAPTCHA. Ao invés do texto, é apresentado ao usuário um áudio com a leitura dos caracteres. Neste tipo também se faz uso de distorções e ruídos para dificultar o reconhecimento por algoritmos [VABHL03].

A junção de CAPTCHA baseados em texto e áudio são comuns e se mostraram bem eficientes, no entanto, um terceiro tipo inicialmente mais complexo foi desenvolvido. Baseado em imagens, ele exibe um conjunto de imagens e o usuário deve escolher quais apresentam o objeto solicitado pelo CAPTCHA. Esse não é o único tipo baseado em imagem. Outros, por exemplo, solicitam que a imagem que esteja fora do contexto das outras imagens seja selecionada, isto é, num conjunto formado por diversas imagens de astronauta e apenas uma de um carro, deve-se selecionar o carro. Esse tipo de desafio não será abordado nesse trabalho.

Apesar da funcionalidade existir para garantir a segurança dos dados e disponibilidade de serviços, alguns usuários consideram o CAPTCHA um incômodo e desafios considerados simples impedem uma quantidade significativa de usuários de visitarem um site ou usarem algum serviço on-line.

Outros “incomodados” pelo CAPTCHA, motivação deste trabalho, são os aqui denominados “robôs do bem”, sem nenhuma intenção criminosa ou nefasta. São robôs desenvolvidos para automação de processos nas empresas, tais como, processo de admissão, pagamento de fornecedores, extração de informação pública e de interesse social, etc, para agilizar processo e reduzir erros e custos desnecessários, além de outros benefícios.

1.1 GOOGLE RECAPTCHA

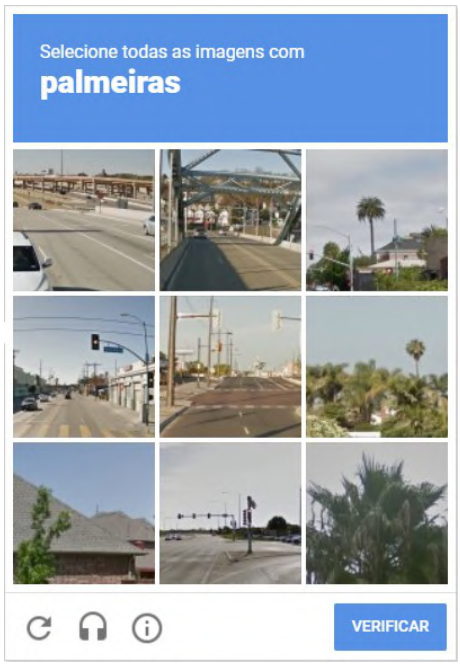
reCAPTCHA é o nome comercial do CAPTCHA da Google. Baseado em imagem, hoje tem duas versões disponíveis: a V2 lançada em 2012 e a V3 em 2018. Elas se diferem pela forma que utilizam para concluir se o agente que interage com a aplicação é um ser humano ou um robô [goo].

Na versão V2 o usuário pode ou não ser submetido ao desafio baseado em imagem após se declarar um humano, clicando em uma caixa de verificação nomeada por “Não sou um robô”. Fica a critério do reCAPTCHA exibir ou não o desafio por meio de uma pontuação calculada com base em informações do usuário, tais como, histórico de navegação, *cookies*, rastreamento de movimento do mouse, etc. O usuário é submetido ao desafio se sua pontuação ficar abaixo do esperado. Já na versão V3, a caixa de verificação inexiste e o dono da aplicação ou *site* define qual é a pontuação esperada e o momento de exibir o desafio, se necessário. Esse recurso é chamado pela Google de “No Captcha reCaptcha”.

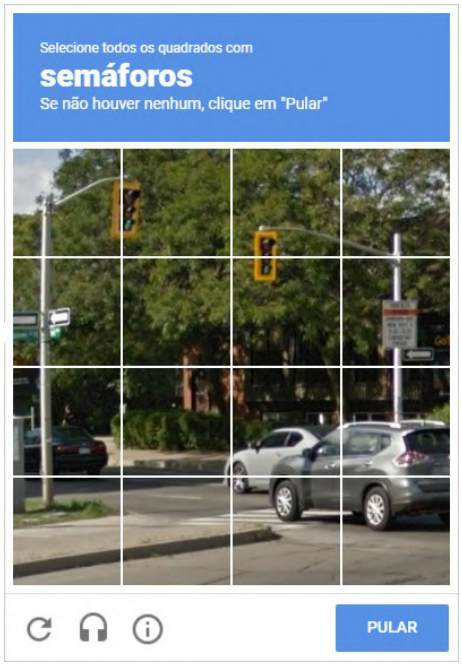
O desafio baseado em imagem consiste i) na escolha de imagens do objeto solicitado numa grade 3x3 com imagens distintas do mesmo tamanho (figura 1(a)), ou ii) na escolha das regiões que contenham partes do objeto solicitado. Neste caso, uma única imagem é dividida em partes do mesmo tamanho e exibida numa grade 4x4 (figura 1(b)).

Ainda no desafio do item i), o CAPTCHA pode exibir uma nova imagem na posição escolhida para ser avaliada (figura 1(c)). Além disso, em qualquer momento a imagem pode ter algum tipo de ruído sem comprometer o entendimento humano (figura 1(d)), mas capaz de dificultar o reconhecimento do objeto por algoritmos [GSS14].

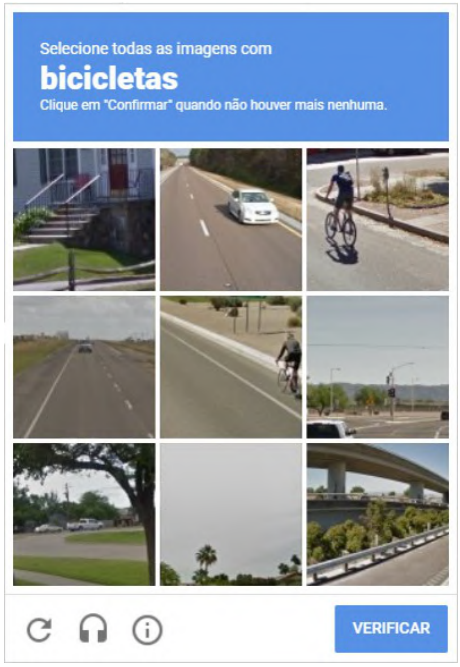
Nas versões iniciais do reCAPTCHA baseado em imagem o usuário não precisa necessariamente atingir 100% de precisão, pois era possível errar algumas imagens e



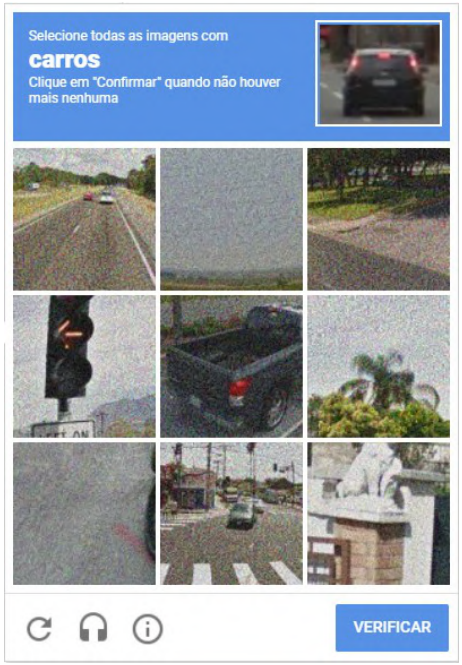
(a) Forma mais simples do desafio, bastando selecionar as três imagens corretas.



(b) Partes das imagens que contém o objeto requerido devem ser selecionadas.



(c) A imagem selecionada é trocada por outra para ser avaliada.



(d) Presença de ruído capaz de dificultar o reconhecimento do objeto por algoritmos.

Figura 1: *Formas de como o reCAPTCHA é exibido.*

ainda assim passar no desafio do CAPTCHA [SPK16], no entanto, atualmente essa flexibilidade não está presente em todas às vezes que o desafio é exibido.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo fim deste trabalho é dotar um modelo que seja capaz de reconhecer os objetos requeridos no reCAPTCHA podendo ser utilizado por uma ferramenta de automação para solucionar os desafio apresentados, avaliando cada imagem para decidir quais selecionará.

1.2.2 Objetivos específicos

1. Estudar e aplicar os conceitos de aprendizado de máquina, mais especificamente sobre a arquitetura e forma de aprendizado das Redes Neurais Artificiais Convolucionais no reconhecimento de objetos.
2. Estudar modelos de classificação de classe única e aplicá-lo no reconhecimento de objetos.
3. Contribuir para a solução robótica do reCAPTCHA.
4. Desenvolver e avaliar uma metodologia para solução de CAPTCHA baseado em imagens.

1.3 TRABALHOS RELACIONADOS

Pesquisas com o intuito de resolver CAPTCHA, sejam baseados em texto ou imagens, são realizadas desde o seu surgimento e possui diversas abordagens. Por exemplo, [KNH17] propõe um método que cria um mapa de calor para localizar os caracteres e usa uma rede neural convolucional para fazer a identificação. Já [Yan20], que aborda o mesmo tipo de CAPTCHA, usa exclusivamente uma rede neural convolucional para identificar os caracteres apresentados.

Abordando desafios baseados em imagem, [SPK16] desenvolveu um classificador que processa a saída de um sistema de anotação de imagens e procura por marcas (palavras) para identificar qual imagem deve ser selecionada. Este trabalho revelou falhas no reCAPTCHA que permitia influenciar o sistema para contornar os desafios propostos. Considerando que esse foi um dos primeiros trabalhos sobre esta versão do reCAPTCHA e que a Google mudou o reCAPTCHA baseado em imagem no mesmo ano, provavelmente a solução proposta está obsoleta.

Em um trabalho mais recente, publicado em 2020, [ZYWB18] propôs imitar o ser humano movimentando o mouse em combinação com a detecção de objetos. A imitação

dos movimentos do mouse foi usada para confundir o reCAPTCHA e tentar evitar o desafio com imagens. Outro ponto importante é que o conjunto de treinamento era formado em parte por imagens obtidas previamente no reCAPTCHA.

1.4 CONTRIBUIÇÕES

Este trabalho é um pouco diferente dos citados, pois não tenta descobrir ou usar qualquer vulnerabilidade do reCAPTCHA para resolvê-lo, propondo uma solução baseada puramente no reconhecimento dos objetos requeridos.

Essa abordagem torna a solução independente de outros critérios de avaliação para determinar se o agente é ou não um ser humano, tal como o movimento do mouse, pois mesmo que o desafio seja proposto, o modelo será capaz de reconhecer o objeto requerido em cada imagem do desafio e, consequentemente, ser solucionado.

Não há a pretensão de estabelecer uma solução fechada para resolver o desafio do reCAPTCHA. O trabalho propõe um modelo de reconhecimento de imagens específicas que pode ser utilizado por qualquer ferramenta ou tecnologia de automação, permitindo a combinação desta com outras soluções para aumentar as chances de sucesso na solução do reCAPTCHA.

2 APRENDIZADO DE MÁQUINA

Desde o início de sua concepção, os computadores eram capazes apenas de executar determinada tarefa seguindo um conjunto de instruções explícitas. A partir de meados do século XX isso começou a mudar com pesquisas em inteligência artificial, mais especificamente em aprendizado de máquina.

O termo Aprendizado de Máquina (do inglês Machine Learning) foi introduzido por Arthur Lee Samuel na década de 50 e a definiu como o "campo de estudo que dá aos computadores a habilidade de aprender sem estar explicitamente programado". Explicitamente programado significa que o computador para realizar determinada tarefa necessita de um algoritmo, que nada mais é que um conjunto finito de instruções, escritas uma após a outra. Um exemplo é um algoritmo para ordenar um conjunto finito de números, conforme abaixo.

```
1  def insertionSort(a):
2      # percorre o vetor a até o fim a partir da posição i
3      for i in range(1,len(a)):
4          pivo = a[i]
5          j = i - 1
6
7          # Move os elementos entre 0 e i-1 maiores que o pivo
8          # para a posição a frente dele
9          while j >=0 and pivo < a[j]:
10             a[j+1] = a[j]
11             j -= 1
12
13         a[j+1] = pivo
```

FONTE: [CLRS01]

Da década de 50 até os dias atuais, várias definições de Aprendizado de Máquina foram apresentadas. Neste trabalho será adotada a definição dada por Tom Mitchell, "Um programa de computador aprende a partir da experiência E em relação a alguma tarefa T e alguma medida de desempenho P, se seu desempenho ao executar T, medido por P, melhora com a experiência E" [M+97]. Em outras palavras, um programa de computador aprende a executar uma tarefa se ele é capaz de executá-la cada vez melhor, de acordo com algum critério de avaliação, à medida que repete sua execução utilizando o que aprendeu anteriormente.

O objetivo de capacitar um programa de computador a aprender determinada tarefa T se justifica, por exemplo, quando escrever explicitamente um algoritmo para isso seria muito complexo ou até impossível. Escrever um algoritmo para identificar se um e-mail

recebido é *spam*, por exemplo, seria impossível, pois o que é *spam* hoje, pode não ser amanhã ou o que é *spam* para uma pessoa, pode não ser para outra. Na prática, deseja-se que o computador produza automaticamente um algoritmo para esta tarefa.

O entendimento de tarefa T está relacionado com aquilo que se deseja aprender e também pelo exemplo (ou amostra) utilizado no processo de aprendizado. Um exemplo é um conjunto de atributos que o programa de aprendizado de máquina deva aprender a processar. Matematicamente, um exemplo é representado por um vetor $\mathbf{x} \in \mathbb{R}^n$, em que cada entrada x_i do vetor é um atributo. Na tarefa de classificação de e-mails como *spam*, uma amostra seria um e-mail em particular e as informações tais como remetente, assunto e conteúdo produziriam seus atributos.

A medida de desempenho P demonstra o quão bem o algoritmo executa T. Sua escolha deve ser feita com cuidado para que o algoritmo desempenhe a tarefa de maneira satisfatória. Medida de desempenho inadequada pode resultar em situações indesejadas, por exemplo, o algoritmo classificar um e-mail importante como *spam*. Neste caso, é melhor admitir que alguns *spams* não sejam identificados do que emails importantes serem classificados como tal.

A experiência E é um conjunto de informações que irá possibilitar o aprendizado sobre a tarefa T. Assim como a medida de desempenho, o conjunto de experiência E está relacionado à tarefa T e é na prática um conjunto de dados à respeito dos exemplos.

No que se segue, uma classificação sobre os tipos de aprendizado será apresentada. As informações contidas no conjunto de dados e o tipo de tarefa a ser aprendida indicam qual o tipo de aprendizado que será utilizado.

2.1 TIPOS DE APRENDIZADO

1. Supervisionado

   No aprendizado supervisionado cada elemento do conjunto de dados é formado por um par $(\mathbf{x}, \mathbf{y})$, onde $\mathbf{x} \in \mathbb{R}^n$ é o vetor de atributos de um exemplo e $\mathbf{y} \in \mathbb{R}^m$ é o resultado desejado pelo algoritmo, denominado rótulo ou alvo. Dessa forma, o conjunto de dados é representado por $E = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) | \mathbf{x}^{(i)} \in \mathbb{R}^n, \mathbf{y}^{(i)} \in \mathbb{R}^m\}_{i=1}^{N}$.

   O valor de $\mathbf{y}^{(i)}$ é fornecido por um instrutor ou professor que conhece o resultado para cada $\mathbf{x}^{(i)}$, daí a origem do nome supervisionado. Neste tipo de aprendizado, o objetivo é criar um classificador a partir do conjunto E para que ele possa predizer o valor de $\mathbf{y}$ dado um $\mathbf{x}$ desconhecido [GBC16].

2. Não supervisionado

   Ao contrário do supervisionado, não existe neste tipo de aprendizado o papel

do instrutor ou professor, ou seja, o conjunto de dados é formado apenas pelos atributos de cada exemplo. O algoritmo observa o conjunto de dados a fim de descobrir propriedades similares entre os exemplos criando grupos (*clustering*). Também é possível utilizar este tipo de aprendizado para inferir a distribuição dos exemplos no seu domínio (*density estimation*) ou projetar dados de alta dimensão no $\mathbb{R}^2$ ou $\mathbb{R}^3$ para fins de visualização.

3. Semi-supervisionado

   Utiliza-se o aprendizado supervisionado quando há dados rotulados, no entanto, um conjunto de dados de exemplos rotulados pode não estar disponível ou rotulá-lo pode ser caro, tornando inviável a utilização deste tipo de aprendizado. Neste caso, o tipo semi-supervisionado pode ser utilizado, uma vez que considera apenas a parte do conjunto de dados que esteja rotulada para inferir um classificador e determinar o rótulo dos demais exemplos. Esses exemplos recém-rotulados são utilizados com os demais dados rotulados para melhorar o classificador.

4. Por reforço

   Aqui o aprendizado do algoritmo, neste tipo denominado por agente, é construído por tentativa e erro, representado pelo par ação e recompensa, gerando o conjunto de experiência. Diferentemente dos tipos de aprendizados vistos até aqui, o conjunto de experiência E não é fixo, mudando ao longo do processo de aprendizado. O objetivo é encontrar um conjunto de ações de uma dada tarefa que maximize a recompensa, tal como num jogo de xadrez ou um robô num labirinto [Alp20].

   Esta forma de aprendizado é aplicada a problemas de ciclo fechado - onde o resultado das ações tomadas influenciam suas entradas posteriores - e o agente deve descobrir quais ações geram mais recompensas. Em alguns casos, as ações podem afetar não apenas a recompensa imediata, mas também a próxima situação e, por meio dela, todas as recompensas subsequentes [SB18].

2.2 TIPOS DE APLICAÇÃO

O aprendizado de máquina pode ser aplicado em diversas áreas, tais como economia, medicina, manufatura e ciências desempenhando, em geral, duas tarefas: classificação e regressão [Alp20]. Por exemplo, na medicina para inferir o diagnóstico de um paciente (classificação) ou na economia para prever o valor de um ativo financeiro (regressão).

2.2.1 Classificação

É uma tarefa desempenhada com frequência no cotidiano onde se relaciona um exemplo a uma classe ou categoria. Um motorista, por exemplo, olhando a cor e a posição da luz de um semáforo consegue determinar se ele está aberto, prestes a fechar ou fechado. Um médico ao avaliar sintomas e resultados dos exames de um paciente define o diagnóstico. Uma instituição financeira, para decidir se concederá ou não crédito a determinada pessoa, avalia seu salário, reservas financeiras, endividamento, profissão, idade, entre outras informações.

No contexto de aprendizado de máquina é desejado que o programa de computador, com base no conjunto de dados (experiência), seja capaz de produzir uma função discriminante $f : \mathbb{R}^n \rightarrow \{C_1, C_2, ..., C_k\}$, que associa um exemplo $\mathbf{x} \in \mathbb{R}^n$ a uma classe C_i, $i = 1, ..., k$.

Para o caso em que existam apenas duas classes, ou seja, $k = 2$, seu conjunto de dados X é definido por $X = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) | \mathbf{x}^{(t)} \in \mathbb{R}^n, \mathbf{y}^{(t)} \in \{0, 1\}\}_{t=1}^{N}$.

Note que $\mathbf{y}^{(t)}$ assume apenas os valores 0 ou 1, pois são suficientes para distinguir a que classe um determinado $\mathbf{x}$ pertence. Quando $k = 2$, o problema de classificação é chamado de **classificação binária**.

Para o caso onde existam mais de duas classes, ou seja, $k > 2$, $\mathbf{y}^{(t)}$ tem dimensão k e é definido da seguinte forma: $\mathbf{y}_i^{(t)} = 1$, se $\mathbf{x}^{(t)} \in C_i$ ou $\mathbf{y}_i^{(t)} = 0$, se $\mathbf{x}^{(t)} \in C_j$, $j \neq i$. Neste caso, o problema de classificação é chamado de **classificação multiclasse**.

Em ambos os casos, o conjunto de dados possui todos os exemplos rotulados e a quantidade de elementos de cada classe é suficiente para que o classificador possa ser treinado. No entanto, em determinados problemas atender essas duas condições pode ser impossível. Por exemplo, um sistema de detecção de falhas numa indústria deve disparar um alarme caso seja percebido uma condição “anormal” no seu funcionamento por meio do monitoramento de um conjunto de variáveis (atributos). Monitorá-las durante o funcionamento normal é fácil e barato, portanto se tem muitas informações. Ao contrário, obter informações das variáveis monitoradas que represente uma falha é difícil e caro. Difícil, porque dependendo do processo industrial nem se saiba quais as falhas possíveis, e cara, porque uma falha proposital para obter informações pode implicar em danos irreparáveis. Aqui se tem muitos dados a respeito da classe “normal” e poucos dados sobre a classe “anormal”.

Problemas desse tipo são conhecidos como problema de classificação de classe única (OCC, do inglês *One-Class Classification*), pois o objetivo é distinguir os exemplos da classe normal de todos os outros exemplos possíveis pertencentes a classe anormal (não necessariamente conhecida). A distinção entre “normal” e “anormal” é feita pela

comparação entre a pontuação $z : \mathbf{x} \to \mathbb{R}$ (*score*) e um valor limite $k \in \mathbb{R}$ (*threshold*), ambos definidos pelo modelo OCC utilizado. Tanto o modelo quanto o valor limite são determinados no processo de aprendizado observando apenas exemplos “normais” [PCCT14].

O termo positivo e negativo também são utilizados para se referir às classes normal e anormal, respectivamente.

2.2.2 Regressão

Este tipo de tarefa possui formulação matemática muito similar a tarefa de classificação, diferindo desta pelo contra-domínio da função, que passa a ser contínuo. Enquanto que a classificação relaciona um exemplo a uma classe, na regressão associa-se um exemplo a um número real. Na prática, se deseja uma função $f : \mathbb{R}^n \to \mathbb{R}$ que determine um número $y \in \mathbb{R}$ a partir de um exemplo $\mathbf{x} \in \mathbb{R}^n$. A regressão é utilizada, por exemplo, no mercado financeiro para prever o valor de ações em função do cenário econômico e na indústria para estimar o preço do produto em função da demanda.

2.3 CONJUNTO DE DADOS E HIPERPARÂMETROS

Em qualquer processo de aprendizado, independente da tarefa a ser aprendida, do tipo de aprendizado ou do método a ser utilizado, é necessário um conjunto de dados adequado para que o algoritmo possa aprender. Para isso um conjunto de exemplos deve ter a maior variedade possível para que o algoritmo possa aprender todas as possibilidades da tarefa e reduzir ao máximo o erro de generalização, ou seja, aumentar a chance do algoritmo executar a tarefa conforme o esperado quando exposto a um exemplo não visto na etapa de treinamento.

Por exemplo, um algoritmo que sinalize se uma pessoa está ou não fazendo o uso de máscara para se proteger da COVID-19 deve ser exposto a um banco de imagens de pessoas com e sem máscara retratando o maior número de situações de uso no cotidiano para que possa responder corretamente ao observar uma pessoa não vista na etapa de aprendizado.

O conjunto de dados no processo de aprendizado é formado por três conjuntos disjuntos denotados por treinamento, validação e teste. O primeiro conjunto é usado para treinar o modelo, o segundo, no ajuste do modelo, determinando os melhores hiperparâmetros, e o terceiro para avaliação do erro de generalização do modelo final.

Hiperparâmetros são variáveis inerentes ao algoritmo de aprendizado capazes de alterar seu comportamento. O valor dos hiperparâmetros não pode ser aprendido (ou atualizado) na fase de treinamento e para determiná-los utiliza-se o conjunto de validação. Fixados os valores dos hiperparâmetros, o algoritmo é ajustado usando o

conjunto de treinamento e seu desempenho é verificado sobre o conjunto de validação. Esse ciclo é executado para diversos valores de hiperparâmetros na tentativa de encontrar aqueles de tenham um desempenho satisfatório.

2.4 *UNDERFITTING* E *OVERFITTING*

A análise do erro de treinamento e de generalização revelam dois fatores que determinam se um algoritmo conseguirá executar sua tarefa de forma adequada. O primeiro deles é o quão pequeno é o erro de treinamento e o segundo, quão pequena é a diferença entre os erros de treinamento e generalização.

Erro de treinamento grande demonstra que o algoritmo não foi capaz de aprender, fenômeno denominado *underfitting*. Por outro lado, erro de treinamento pequeno, mas com diferença significativa entre os erros de treinamento e generalização indicam uma baixa habilidade de generalizar, conhecido como *overfitting*. Fazendo uma comparação grosseira, o *underfitting* se assemelha ao aluno que sequer entendeu os exercícios da lista. Já o *overfitting* se assemelha ao aluno que a decorou e não adquiriu conhecimento e habilidade para resolver questões novas. Ambos têm baixa probabilidade de sucesso na prova.

2.5 MEDIDAS DE DESEMPENHO

Medidas de desempenho, conforme citado, são utilizadas para medir o quão bem o algoritmo consegue executar a tarefa para a qual foi treinado. A definição da métrica a ser utilizada deve levar em consideração o tipo de tarefa: classificação ou regressão. Nesta seção serão abordadas apenas as métricas para problemas de classificação.

Antes de falar sobre as métricas de classificação, faz-se necessária a explicação de uma ferramenta utilizada por todas elas, denominada matriz de confusão. Ela é resultado da relação entre a classe real e a classe predita pelo algoritmo e tem dimensão K x K, onde K é um número de classes que podem ser preditas. Sem perda de generalidade, a tabela 1 demonstra uma matriz de confusão para duas classes.

Tabela 1: *Matriz de confusão para duas classes [Alp20]*

	Classe Predita		
Rótulo das Classes	Positiva	Negativa	Total
Positiva	verdadeiro positivo (tp)	falso negativo (fn)	$p = tp + fn$
Negativa	falso positivo (fp)	verdadeiro negativo (tn)	$n = fp + tn$
Total	$p' = tp + fp$	$n' = fn + tn$	N

Nela é possível observar quatro possíveis resultados:

1. Verdadeiro Positivo (tp): quando o exemplo e a predição são positivos.

2. Verdadeiro Negativo (tn): quando o exemplo e a predição são negativos.
3. Falso Positivo (fp): quando o exemplo é negativo e a predição é positiva.
4. Falso Negativo (fn): quando o exemplo é positivo e a predição é negativa.

Para cada classe existente, a última coluna totaliza a quantidade de exemplos avaliados (p e n), enquanto que a última linha totaliza o resultado da predição (p' e n').

A partir da matriz de confusão, quatro métricas de desempenho são definidas, a saber:

1. **Acurácia**: é definida como a proporção do número de predições corretas sobre a quantidade total de exemplos.

$$Acc = \frac{tp + tn}{N}, N = tp + tn + fp + fn$$

   O uso da acurácia é indicado apenas para conjuntos balanceados, ou seja, quando a quantidade de exemplos de cada classe estão próximas. Medir acurácia para conjuntos desbalanceados pode dar a falsa sensação que o algoritmo classifica muito bem os exemplos, enquanto que na verdade isso não ocorre. Suponha que um algoritmo seja treinado para detectar uma doença rara que acomete uma pessoa em um milhão. Facilmente o algoritmo pode obter 99.9999% de acurácia apenas determinando que todos os exemplos são negativos, ou seja, dizendo que a doença não foi detectada [GBC16]. Em conjuntos como esses, ou seja, desbalanceados, as métricas precisão e revocação são mais adequadas para avaliação de desempenho.

2. **Precisão**: é definida como o número de verdadeiros positivos dividido pelo número de preditos positivos. Determina a fração de classificações positivas que foram feitas corretamente.

$$Prec = \frac{tp}{p'}, p' = tp + fp$$

   Para os problemas onde há mais de duas classes, a precisão de uma classe i pode ser calculada pela expressão

$$Prec_i = \frac{tp_i}{p'_i} \tag{1}$$

   Neste caso, tp_i se refere ao elemento (i, i) da matriz de confusão e p'_i a soma dos elementos da coluna i.

3. **Revocação** (*Recall)*: é definida como o número de verdadeiros positivos dividido pelo número de positivos de fato. Determina a fração de exemplos positivos que

foram classificados corretamente.

$$Rev = \frac{tp}{p}, p = tp + fn$$

A generalização para os casos onde há mais de duas classes é análoga à realizada na precisão.

4. **Pontuação F_1**: é a ponderação uniforme da Precisão e Revocação em relação a uma classe, neste caso a positiva, e é dada por

$$F_1 = 2\frac{Prec.Rev}{Prec + Rev}$$

Sendo a Precisão e a Revocação limitadas no intervalo $[0, 1]$, a pontuação F_1 também é, uma vez que F_1 tende a zero quando a Precisão ou Revocação tendem a zero e F_1 é igual à 1 quando Precisão e Revocação assumem seu valor máximo. Isso pode ser facilmente notado reescrevendo a expressão de F_1 na forma

$$F_1 = 2\frac{1}{\frac{1}{Prec} + \frac{1}{Rev}}$$

Quanto mais F_1 está próximo de 1, melhor é o desempenho do algoritmo de classificação.

3 REDES NEURAIS ARTIFICIAIS

A história da Inteligência Artificial (IA) ainda está sendo escrita, mas ela se inicia na Antiguidade onde pensadores, poetas e matemáticos já imaginavam máquinas capazes de simular (ou emular) a inteligência e o cérebro humano. Com o desenvolvimento da computação, em torno da década de 1940 se inicia a pesquisa sobre Redes Neurais Artificiais (RNA) - uma área de IA.

3.1 O NEURÔNIO ARTIFICIAL

Em 1943, Warren McCulloch e Walter Pitts abriram o caminho da pesquisa sobre RNA quando publicaram o artigo [MP43] mostrando um modelo que utilizava circuitos elétricos para demonstrar o funcionamento do cérebro humano.

O cérebro humano é formado por bilhões de neurônios responsáveis pela propagação do impulso nervoso, uma corrente elétrica que representa a “mensagem” captada pelos receptores nervosos e é rapidamente propagada por sua estrutura, demonstrada na figura 2.

Basicamente, a propagação do impulso nervoso pelo neurônio se inicia nos dentritos por meio da sinapse, que constitui o processo de comunicação entre os neurônios. O corpo celular combina de forma ponderada esses impulsos e o resultado pode ou não ser enviado ao axônio para ser transmitido a outro neurônio.

Para que haja a transmissão do impulso entre neurônios, o resultado do corpo celular deve atingir determinada intensidade, o chamado limiar de excitação ou de disparo. Caso contrário, ele não prossegue.

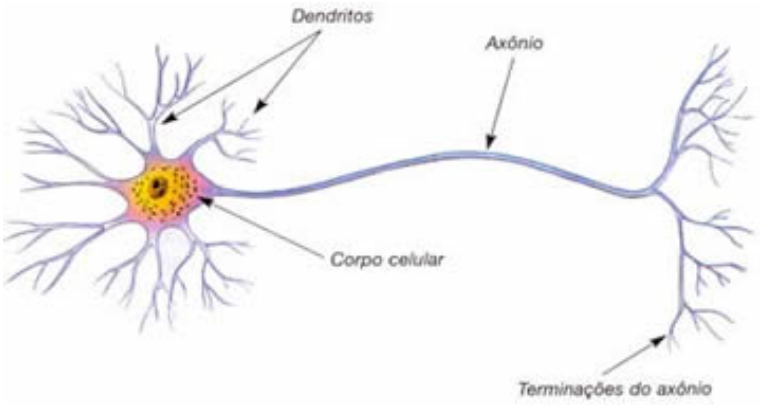


Figura 2: *Visão geral de um neurônio biológico*
Disponível em https://sites.google.com/site/tudoensinomedio/unifei/calendario-1/biologia-3/reinos/fisiologia-animal/sistema-nervoso/celulas-nervosas Acessado em 14/11/2020.

A partir da estrutura e funcionamento do neurônio biológico, Warren McCulloch e Walter Pitts propuseram um modelo matemático batizado de MCP que, em termos gerais,

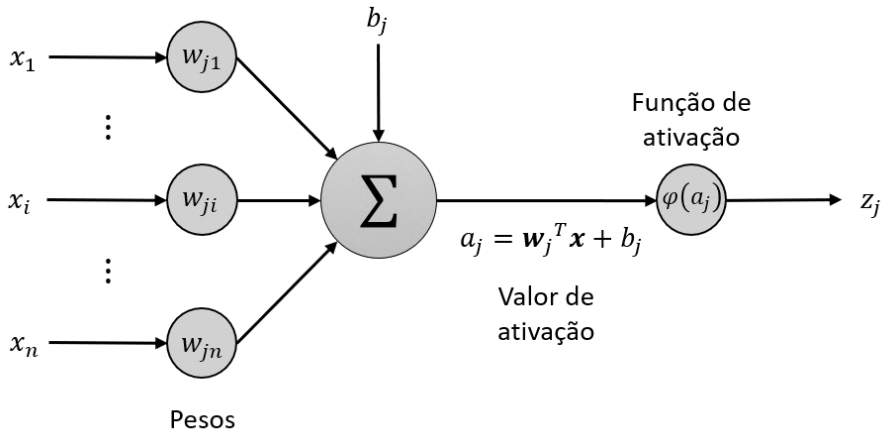


Figura 3: *Modelo matemático de um neurônio*

calcula a soma ponderada dos valores de entrada. Se esse resultado fosse maior ou igual a um valor limite, a saída era não-nula, caso contrário era nula ou negativa. Nesse modelo tanto as entradas, quanto a saída, são valores binários [Kov02].

Frank Rosenblatt, inspirado em McCulloch e Pitts, propôs o modelo perceptron, que se diferenciava do MCP por considerar entradas não binárias [Ros57]. Os dois podem ser interpretados como um caso particular de um modelo mais geral de n entradas e uma saída demonstrado na figura 3. Matematicamente é dado por:

$$z = \varphi(a) = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \varphi\left(\mathbf{w}^T\mathbf{x} + b\right), z \in \mathbb{R} \tag{2}$$

onde $\mathbf{w} = \{w_1, w_2, ..., w_n\} \in \mathbb{R}^n$ é o vetor de pesos associado ao vetor de entrada $\mathbf{x} = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^n$, $b \in \mathbb{R}$ o *bias* ou viés e $\varphi : \mathbb{R} \to \mathbb{R}$ a função de ativação.

O termo viés tem o papel de definir a intensidade do “impulso” que será avaliado pela função de ativação. Ele introduz um grau de liberdade que permite ajustar a capacidade do modelo.

3.2 FUNÇÃO DE ATIVAÇÃO

A função de ativação irá definir o intervalo do valor de saída do neurônio e é escolhida de acordo com a necessidade da aplicação a que se destinará a RNA, mas seu principal objetivo é introduzir no modelo a não-linearidade, permitindo resolver problemas que o modelo linear não consegue, por exemplo, o problema do XOR [MP69]. Note que os modelos MCP e Perceptron utilizam a função degrau, ou limiar, como função de ativação.

Não existe uma regra para escolha da função de ativação, portanto utiliza-se o pro-

cesso de tentativa e erro para verificar qual funciona melhor [GBC16], ou seja, a função de ativação é um exemplo de um hiperparâmetro, apresentado no capítulo 2.

Muitas funções de ativação já foram testadas, mas poucas deram algum resultado prático [Neg05]. Abaixo estão descritas algumas funções utilizadas.

1. **Função de Ativação por Limiar**

$$\varphi(a) = \begin{cases} 0 & , a < 0 \\ 1 & , a \geq 0 \end{cases} \qquad a \in \mathbb{R} \tag{3}$$

$$\varphi'(a) = 0 \tag{4}$$

É também conhecida como função degrau, devido a forma de seu gráfico que está exemplificado na figura 4(a). São geralmente usadas em problemas de reconhecimento de padrões e classificação, mas é impossível ser treinada utilizando o gradiente descendente, que será apresentado posteriormente, pois sua derivada é nula em todo domínio.

2. **Sigmoid**

$$\varphi(a) = \frac{1}{1 + e^{-a}} \qquad a \in \mathbb{R} \tag{5}$$

$$\varphi'(a) = \varphi(a)(1 - \varphi(a)) \tag{6}$$

A imagem da função sigmoid tem forma de um S, conforme mostrado na figura 4(b). Ela transforma $a \in \mathbb{R}$ no intervalo $(0, 1)$ de forma não linear. Sua derivada é contínua uma vez que é produto de funções contínuas e sua imagem tem valor abaixo de 1 e tende a zero quando $|a|$ cresce.

3. **Tangente Hiperbólica**

$$\varphi(a) = \tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{a}} \qquad a \in \mathbb{R} \tag{7}$$

$$\varphi'(a) = 1 - \varphi^2(a) \tag{8}$$

O gráfico da função tangente hiperbólica tem forma parecida com a função sig-

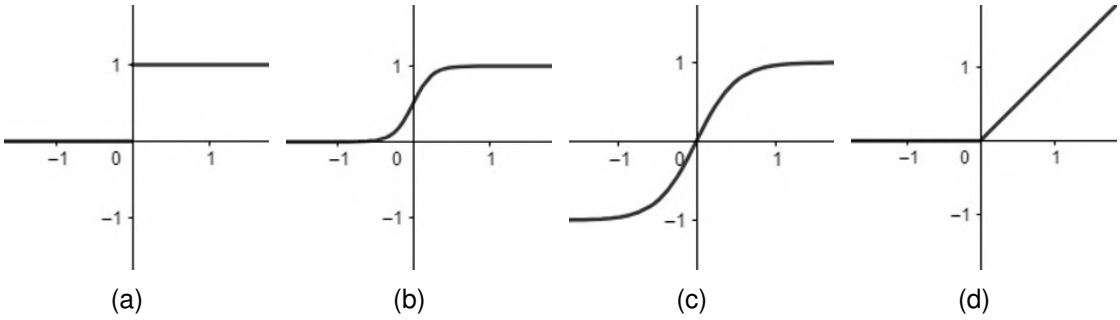


Figura 4: *Funções de ativação mais comumente utilizadas*

moid (figura 4(c)), com a vantagem de ser uma função ímpar e, portanto, simétrica em relação a origem. Ela permite a obtenção de valores positivos e negativos na saída do neurônio.

4. **Unidade Linear Retificada (ReLU)**

$$\varphi(a) = max\{0, a\} = \frac{|a| + a}{2} \qquad a \in \mathbb{R} \tag{9}$$

$$\varphi'(a) = \begin{cases} 0 & , a < 0 \\ 1 & , a \geq 0 \end{cases} \tag{10}$$

Tem comportamento idêntico à função Identidade para valores positivos de a e é nula para valores negativos (figura 4(d)). A principal característica que a diferencia das outras funções é a estabilidade do valor da derivada, sendo sempre igual à 0 ou 1. Note que $\varphi(a)$ em $a = 0$ não possui derivada, pois suas derivadas à direita e à esquerda não são iguais, no entanto, define-se $\varphi'(0) = 0$ ou $\varphi'(0) = 1$, conforme equação (10).

5. **SoftMax**

A função SoftMax transforma um vetor n-dimensional com coordenadas reais arbitrárias em outro n-dimensional com coordenadas no intervalo $(0, 1)$, tal que a soma do valor dessas n coordenadas seja igual à 1.

$$\begin{array}{rccc} \varphi & : & \mathbb{R}^n & \to & \mathbb{R}^n \\ & & \mathbf{a} & & (\varphi_1(\mathbf{a}), \ldots, \varphi_n(\mathbf{a})) \end{array}$$

onde cada função coordenada φ_i é dada por

$$\varphi_i(\mathbf{a}) = \frac{e^{a_i}}{\sum_{i=1}^{m} e^{a_i}} \tag{11}$$

Ela é utilizada em problemas de classificação com mais de duas classes, onde a i-ésima coordenada da imagem de $\varphi(\mathbf{a})$ representa a probabilidade do exemplo, representado por **a**, pertencer a i-ésima classe.

3.3 FUNÇÃO DE CUSTO

Essa função é responsável por dizer quão longe o valor obtido está do valor desejado. Quantifica o "custo" ou "perda" ao aceitar o valor obtido pelo modelo. É nela que será aplicado o gradiente descendente para o treinamento da RNA.

A medida de erro mais comum é o erro quadrático médio (MSE - *Mean Square Error*), dada pela equação (12), onde y_i é o valor desejado, $\hat{y}_i$ o valor predito e m a quantidade de exemplos do conjunto de treinamento. O fator 2 é inserido por conveniência para simplificar o gradiente do MSE. Essa função é usado nos problemas de regressão.

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} ||y_i - \hat{y}_i||^2 \tag{12}$$

Uma segunda função de custo, utilizada nos problemas de classificação, é a medida de entropia cruzada que tem sido utilizada como alternativa ao erro quadrático médio, principalmente quando a saída representa hipóteses independentes e indica a probabilidade de cada uma das hipóteses ser verdadeira.

Por exemplo, num problema de classificação de classes A, B e C, suponha que o valor obtido pelo modelo ao avaliar um exemplo **x** seja $\hat{\mathbf{y}} = (0.85, 0.05, 0.10)$. Isso significa que **x** tem 85% de chance de pertencer à classe A, 5% de pertencer à B e 10% de pertencer à C.

A entropia cruzada, definida pela equação (13), compara duas distribuições de probabilidade, esperado $\mathbf{y} = (y_1, y_2, ..., y_m)$ *versus* predito $\mathbf{f}(\mathbf{x}) = (f_1(x), f_2(x), ..., f_m(x))$, e determina o quanto elas estão próximas.

$$H(x, y) \quad = \quad -\sum_{i=1}^{m} y_i \log(f_i(x)) \tag{13}$$

No exemplo acima, considerando a notação *one-hot encoded* para o resultado esperado, se **x** pertence à classe A, então a saída esperada do modelo é o vetor $\mathbf{y} =$

$(1, 0, 0)$. Portanto, tem-se:

$$\begin{aligned} H &= -1\log(0,85) - 0\log(0,05) - 0\log(0,10) \\ &= -\log(0,85) \approx 0,2345 \end{aligned}$$

A entropia cruzada tem duas propriedades desejadas em uma função de custo. A primeira, que ela é sempre positiva, uma vez que todos os termos do somatório são negativos - note que o argumento do $\log$ está no intervalo $(0, 1)$. A segunda, que quanto menor a diferença entre o valor do modelo e o valor real, mais a entropia cruzada se aproxima de zero.

Note que a saída da função de ativação SoftMax é uma distribuição de probabilidade e devido a isso as duas funções (SofMax e Entropia Cruzada) são usadas em conjunto em problemas de classificação.

3.4 O MÉTODO DO GRADIENTE

O método do gradiente é um método de otimização utilizado para localizar um ponto de mínimo de uma determinada função $f(x) : A \to B$ denotada por função objetivo, que no contexto de RNA, é chamada de **função de custo**. O objetivo é determinar para qual valor de $x \in A$ a função f assume seu menor valor em B. Geralmente, essa busca visa determinar o mínimo global, isto é, o ponto $a \in A$ tal que $f(a) <= f(x),\ \forall\, x \in A$, no entanto, em alguns problemas essa busca pode ser inviável ou determinar um mínimo local seja suficiente. O ponto $a \in V \subset A$ é um mínimo local se $f(a) <= f(x),\ \forall\, x \in V$, onde V é um aberto.

Esse método é um processo iterativo que explora uma propriedade importante do gradiente de uma função: a maior taxa de variação de uma função ocorre na direção e sentido do vetor gradiente.

Sua definição na forma mais simples está descrita na equação (14) que será usada para uma simples introdução. No entanto, existem variações que não serão abordadas aqui, tal como o método do gradiente estocástico e suas implementações.

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}) \tag{14}$$

onde:

- k é o número da iteração;
- $\alpha^{(k)}$ a taxa de aprendizado;

- $\nabla f(x^{(k)}) = \left(\frac{\partial f}{\partial x_1^{(k)}}, \ldots, \frac{\partial f}{\partial x_n^{(k)}} \right)$ o gradiente de f.

Os fatores $x^{(0)}$ e $\alpha^{(k)}$ na equação (14) são hiperparâmetros importantes que podem determinar o sucesso ou fracasso do método. Dependendo do valor de $x^{(0)}$, o método pode ter um comportamento indesejado, podendo ficar "preso" em um mínimo local, demorar um tempo demasiado para convergir ou não convergir.

O processo é simples: calcular o gradiente da função de custo f em uma solução atual $x^{(k)}$ e depois atualizar de forma iterativa as soluções ao longo do "caminho" de descida mais rápido de f. Ele termina quando uma das duas condições abaixo é satisfeita:

1. O número máximo de iterações é atingido ou;
2. O método converge, de acordo com algum critério.

A condição de parada do número máximo de iterações é necessária por uma questão prática. Não há garantia que a segunda condição seja satisfeita, tornando a execução do método interminável.

Já o hiperparâmetro $\alpha^{(k)}$ determina o tamanho do deslocamento na direção oposta ao gradiente. Ela determina a velocidade do aprendizado, mas pode causar alguns problemas se mal escolhida. Taxa de aprendizado muito pequena pode tornar o processo de otimização muito demorado, enquanto taxa de aprendizado muito grande pode fazer com o que o método não atenda a condição 2.

Para um dado conjunto de treinamento, a equação (14) pode ser calculada de duas formas básicas. Na primeira forma, conhecida como forma sequencial ou estocástica, $x^{(k+1)}$ é calculado após a apresentação de cada exemplo de treinamento (ou de um pequeno conjunto de exemplos, denominado *mini-batch*). A outra forma é calcular $x^{(k+1)}$ após a apresentação de todos os exemplos de treinamento que constituem uma época [Hay07].

3.5 ALGORITMO DE RETROPROPAGAÇÃO (*BACKPROPAGATION*)

A apresentação deste algoritmo foi feita por [RHW86] em 1986 permitindo o treinamento de redes neurais com um número de camadas até então não utilizado e contribuiu para a retomada da pesquisa em redes neurais após o "apagão" iniciado na década de 1970.

Este algoritmo é utilizado para calcular o gradiente da função de custo no método de gradiente descente para o treinamento de redes neurais. Ele se baseia, como o nome sugere, na propagação do erro da rede no sentido contrário da informação, ou seja, da saída para a entrada.

Para demonstrar o funcionamento do algoritmo e suas equações, considere a RNA da figura 5 com L camadas. Na primeira camada a entrada x não sofre alteração. Nas camadas subsequentes a entrada é ponderada pela matriz de pesos $W^l = [w^l_{jk}]$ e a função de ativação φ é aplicada. Cada entrada da matriz W^l é um valor real que indica o peso da conexão entre o k-ésimo neurônio da camada $l-1$ e o j-ésimo neurônio da camada l.

3.5.1 Equações

Por definição, o erro do j-ésimo neurônio da camada $l = 2, 3, ..., L$ é dado pela expressão

$$\delta^l_j = \frac{\partial C}{\partial a^l_j} \tag{15}$$

onde C é a função de custo e

$$a^l_j = W^l_j . z^{l-1}_j = \sum_{k=1}^{k_{l-1}} w^l_{jk} . \varphi(a^{l-1}_k) \tag{16}$$

Para simplificar, δ^l pode ser escrito como um vetor de k_l entradas, cada uma correspondendo ao erro do j-ésimo neurônio na camada l, ou seja, $\delta^l = [\delta^l_j]$, com $j = 1, 2, ...k_l$.

Além disso, o termo b_j que aparece no modelo da figura 3 não é explicitado, mas ele está sendo considerado. Basta para isso acrescentar, de forma adequada, mais uma coordenada de valor 1 ao vetor de entrada e mais uma coluna $b^l = [b^j]$ na matriz de pesos W^l.

Pela regra da cadeia, a equação (15) pode ser reescrita em função de $\varphi'(a^L_j)$, conforme abaixo, para obter o erro da camada L.

$$\delta^L_j = \sum_{k=1}^{k_L} \frac{\partial C}{\partial z^L_k} \frac{\partial z^L_k}{\partial a^L_j} = \frac{\partial C}{\partial z^L_j} \frac{\partial z^L_j}{\partial a^L_j} = \frac{\partial C}{\partial z^L_j} \varphi'(a^L_j) \tag{17}$$

Note que os termos do somatório são todos nulos quando $k \neq j$. Novamente aplicando a regra da cadeia na equação (15), obtêm-se a equação

$$\delta^l_j = \sum_{k=1}^{k_l} \frac{\partial C}{\partial a^{l+1}_k} \frac{\partial a^{l+1}_k}{\partial a^l_j} = \sum_{k=1}^{k_l} \frac{\partial a^{l+1}_k}{\partial a^l_j} \delta^{l+1}_k \tag{18}$$

Derivando a equação (16) na camada $l+1$ em relação a a^l_j, temos

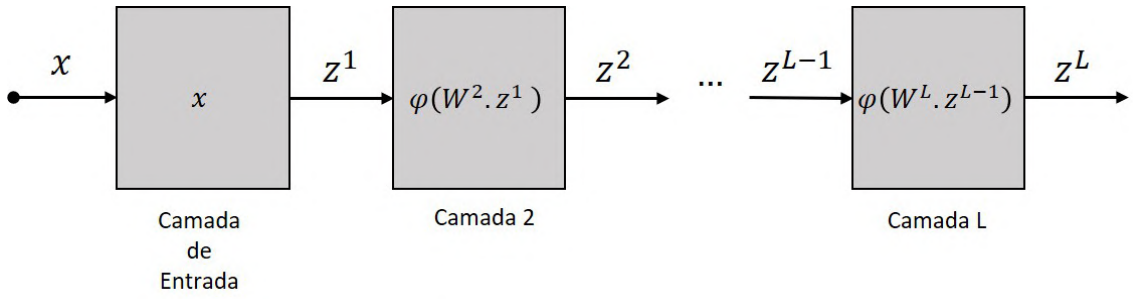


Figura 5: *Rede neural feedforward com L camadas.*

$$\frac{\partial a_k^{l+1}}{\partial a_j^l} = w_{kj}^{l+1}\varphi'(a_j^l) \tag{19}$$

A equação (18), usando o resultado da equação (19), fornece o erro de uma camada em função do erro da camada seguinte.

$$\delta_j^l = \sum_{k=1}^{k_l} w_{kj}^{l+1}\delta_k^{l+1}\varphi'(a_j^l) \tag{20}$$

Com as equações (17) e (20), consegue-se agora obter o gradiente da função de custo.

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial a_j^l}\frac{\partial a_j^l}{\partial w_{jk}^l} = \delta_j^l z_k^{l-1} \tag{21}$$

3.5.2 Algoritmo

Usando a notação vetorial para as equações obtidas, o algoritmo de retropropagação pode ser implementado da seguinte forma:

1. Para cada camada $l = 2, 3, ..., L$ compute o valor da saída $z^l = [z_j^l]$.
2. Calcule o erro da rede $\delta^L = [\delta_j^L]$ com a equação (17).
3. Para cada camada $l = L-1, L-2, ..., 2$ compute o valor do erro $\delta^l = [\delta_j^l]$ com a equação (18).
4. Calcule o gradiente da função de custo $\nabla C = \left[\frac{\delta C}{\delta w_{jk}^l}\right]$ com a equação (21).

3.6 ARQUITETURA DAS REDES NEURAIS ARTIFICIAIS

Uma rede neural artificial é um modelo matemático inspirado no funcionamento do cérebro humano e é formada pela conexão entre seus neurônios. A disposição dos

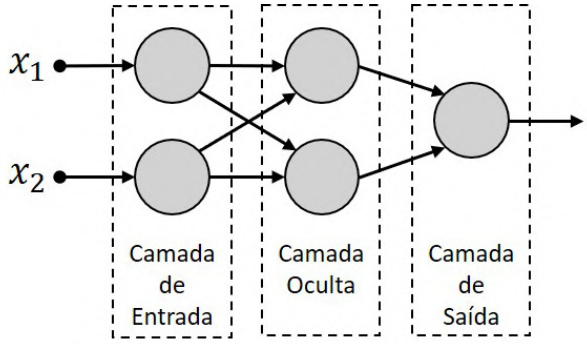


Figura 6: *Exemplo de uma rede neural feedforward demonstrando suas camadas*

neurônios nas camadas, bem como sua quantidade, e a forma de conexão entre eles definem a arquitetura da RNA. Os assuntos abordados até o momento apresentam elementos suficientes para a construção e treinamento da maioria das arquiteturas existentes, senão todas.

Não é o objetivo deste trabalho demonstrá-las em detalhes, muito menos determinar uma taxonomia, mas as arquiteturas existentes, de forma geral, podem ser divididas em dois grupos: com ou sem realimentação.

As arquiteturas sem realimentação ou redes em avanço (*feedforward*) têm seus neurônios agrupados em camadas que são dispostas uma após a outra. A informação a ser processada percorre a rede em uma única direção, da camada de entrada para a camada de saída. Dentre elas pode-se citar as redes Perceptron de Múltiplas Camadas (figura 6) e a Convolucional (figura 7), utilizadas neste trabalho.

Do ponto de vista algébrico, a rede em avanço é uma composição de funções, cada uma representando a saída de uma camada. Considerando a rede da figura 5, tem-se

$$\begin{aligned} Z^1 &= \mathbf{x} \\ Z^2 &= \varphi(Z^1) \\ &\vdots \\ Z^L &= \varphi(Z^{L-1}) \end{aligned} \tag{22}$$

Redes com essa arquitetura podem ser treinadas pelo método do gradiente juntamente com a retropropagação desde que a função de ativação seja diferenciável (pelo menos por partes) e seu gradiente seja não nulo em alguma região do domínio.

Nas arquiteturas com realimentação ou recorrentes, mostrada na figura 8, ao contrário da arquitetura *feedforward*, a saída de um ou mais neurônios pode ser ligada a outros

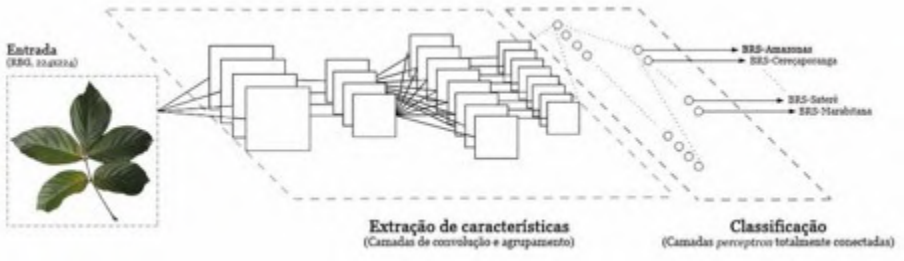


Figura 7: *Exemplo de uma rede neural convolucional ilustrando o processo de classificação de cultivares de guaraná. Fonte: [SS17]*

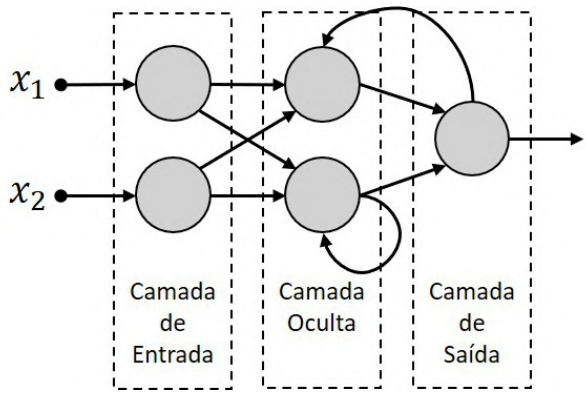


Figura 8: *Exemplo de uma rede neural recorrente demonstrando a realimentação*

neurônios da mesma camada (às vezes, a ele mesmo) ou de camadas anteriores. Esse tipo de arquitetura não será abordada neste trabalho, mas tem larga aplicação em problemas relacionados a dados sequenciais, por exemplo, no processamento de linguagem natural [GBC16].

3.7 REDES PERCEPTRON DE MÚLTIPLAS CAMADAS (MLP)

A rede Perceptron de Múltiplas Camadas é uma rede *feedfoward* totalmente conectada, ou seja, qualquer neurônio tem como entrada todos os neurônios da camada anterior. Ela foi introduzida para superar as limitações do perceptron descritas em [Ros57].

Matematicamente, o perceptron é descrito pela equação (2) utilizando a função de ativação por limiar citada em 3.2. É um modelo não-linear, que geometricamente, pode ser interpretado como um hiperplano que separa o espaço de entrada em duas regiões permitindo classificar os exemplos de acordo com sua posição relativa a esse hiperplano [Hay07].

Um exemplo é a função lógica AND com suas entradas e saídas representadas na figura 9. Nela é possível visualizar que a reta $x_1 + x_2 = 1,5$ separa as entradas em duas regiões: acima da reta está a entrada $(1, 1)$ que produz a saída 1 e abaixo dela as demais que produzem a saída 0.

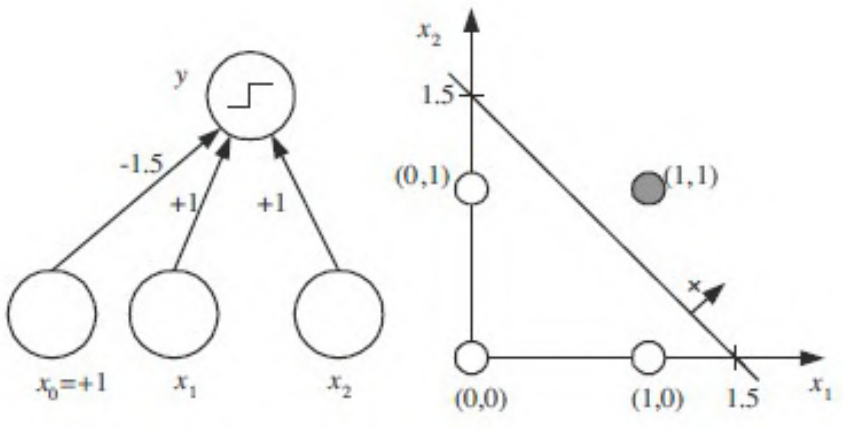


Figura 9: *Perceptron que implementa a função lógica AND e sua interpretação geométrica. Fonte: [Alp20]*

Apesar de ser um modelo não-linear, a aplicação do perceptron fica limitada à problemas linearmente separáveis, ou seja, os exemplos a serem classificados devem estar em regiões que possam ser separadas por um hiperplano [Hay07].

Para superar essas limitações, a rede MLP que é formada por mais de uma camada, cada uma delas com um ou mais neurônios perceptron, é capaz de extrair, com suas camadas ocultas, propriedades relevantes que caracterizam os dados de treinamento. Isso é feito com a transformação não-linear dos dados de entrada para um espaço de características (ou espaço oculto) [Hay07]. É exatamente por essa capacidade que a função lógica XOR pode ser implementada usando uma MLP.

Considere a MLP da figura 10. Semelhantemente ao perceptron da função AND, ela possui duas entradas e uma saída, mas tem uma camada oculta com dois neurônios que torna o problema linearmente separável. Neste caso, a função de ativação ReLU foi utilizada, mas ela não é a única que pode desempenhar esse papel. Outras formas de implementação do XOR podem ser encontradas em [Hay07] e [Alp20]. A transformação do espaço de entrada é ilustrado na figura 11.

3.8 REDES NEURAIS CONVOLUCIONAIS (CNN)

A rede neural convolucional, também chamada de rede convolucional, é uma rede *feedforward* que utiliza alguns dos elementos vistos até agora, tais como, funções de ativação não lineares, gradiente descendente e retropropagação. A novidade é a presença de camadas convolucionais e de *pooling* responsáveis pela geração dos mapas de características e redução da dimensão do espaço de entrada, respectivamente.

A ideia de extrair características dos exemplos de entrada já é utilizada nas redes MLP totalmente conectadas, veja o exemplo do XOR discutido na seção 3.7. A diferença

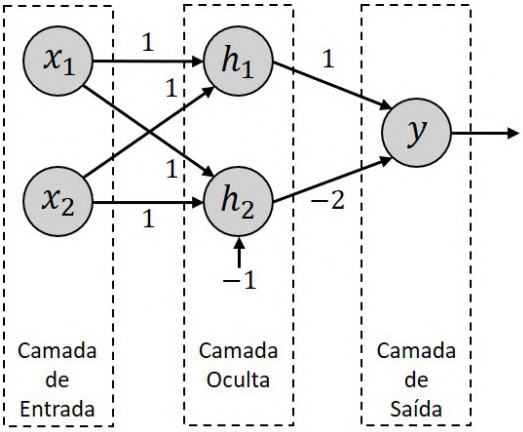


Figura 10: *MLP com uma camada oculta (h) que implementa a função lógica XOR.*

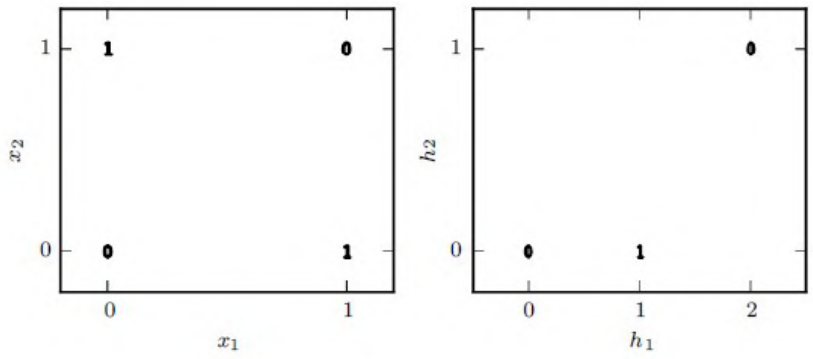


Figura 11: *Na esquerda, o espaço de entrada. Na direita, o espaço oculto representado pelas características extraídas pela camada oculta. Os dois pontos com saída 1 no espaço de entrada, que possuem característica não linear, são mapeados para o mesmo ponto no espaço oculto. Fonte: [GBC16]*

é que a CNN considera a topologia (estrutura espacial ou temporal) dos dados de entrada. Por exemplo, a MLP para classificação de imagens em preto e branco de 28x28 pixels “enxerga” a entrada como um vetor de 784 posições, enquanto que para a CNN a imagem é uma matriz de dimensão 28x28.

A consideração da estrutura espacial da entrada permite a extração de características que só fazem sentido no contexto espacial. As tiras de uma foto 3x4 de uma pessoa colocadas uma ao lado da outra formando uma tira única certamente prejudica o entendimento da foto e dificulta a percepção de suas características, tais como, se a pessoa está usando brinco, qual a cor do olho, etc.

3.8.1 Motivação

As redes convolucionais foram inspiradas nos processos biológicos da visão. A conexão entre os neurônios artificiais está baseada na organização do córtex visual. Neurônios do córtex respondem a estímulos apenas em regiões restritas do campo de visão conhecidas como campos receptivos, que se sobrepõem parcialmente para cobrir todo o campo de visão [HW68].

Baseado nesse processo de visão, a CNN utiliza três ideias que trazem grande vantagem no processo de reconhecimento de imagens em relação a uma rede MLP. A primeira ideia é a **interação** ou **conexão esparsa** entre cada camada da rede. Numa CNN cada unidade interage apenas com uma pequena parcela das unidades da camada anterior, denotada por **campo receptivo local**. Isso diminui drasticamente o número de parâmetros que devem ser ajustados no treinamento, reduzindo o tempo de treino e consumo de memória [GBC16].

A segunda, é o **compartilhamento dos parâmetros** ou **pesos** que refere-se ao uso do mesmo conjunto de parâmetros em todos os campos receptivos locais, ou seja, o valor do peso aplicado a uma região da entrada é o mesmo que será aplicado em outra região, reduzindo ainda mais os requisitos de memória do modelo [GBC16].

Por fim, a terceira ideia é a **subamostragem** (*pooling*). Ela reduz a dimensão da saída da camada convolucional, diminuindo mais ainda a quantidade de parâmetros da rede e a demanda por memória. A motivação por trás dessa ideia é que uma vez identificada uma característica de interesse, sua localização na estrutura de entrada pode não ser importante, sendo suficiente apenas sua presença para a interpretação da entrada.

Esses três conceitos asseguram a esse tipo de RNA uma invariância em relação a entrada. Um objeto numa imagem, por exemplo, não se transforma em outro objeto porque foi transladado, rotacionado (no plano da imagem) ou redimensionado. Matematicamente significa dizer que uma variação na entrada produz a mesma variação na saída [GBC16].

3.8.2 A operação de Convolução

A convolução é um operador linear que realiza a soma do produto entre duas funções, aqui denotadas por f e g, na região em que se sobrepõem no deslocamento de uma delas. É uma ferramenta matemática largamente utilizada nas ciências exatas. Considerando que f e g são funções contínuas, logo integráveis a Riemann, a expressão da convolução é dada por

$$s(t) = (f * g)(t) = \int_{-\infty}^{+\infty} f(a)g(t-a)da \tag{23}$$

Para facilitar o entendimento e ilustrar uma aplicação da convolução, considere um sensor que mede a temperatura em sua cidade, representado pela função f. Suponha que devido à fatores inerentes ao instrumento exista a presença de ruído na leitura realizada. Com o objetivo de reduzir o efeito do ruído na medida final pode-se fazer uma média dos valores medidos numa janela de tempo a, onde t significa o instante de tempo que se deseja saber o valor da temperatura e a o deslocamento de tempo em relação à t para o cálculo da média. Neste exemplo, a função g tem o papel de ponderar a relevância de cada medida de f no domínio $[t-a, t]$.

No mundo computacional, os dados em função do tempo são discretizados, portanto a expressão (23) da convolução pode ser reescrita como um somatório finito.

$$s(t) = (f * g)(t) = \sum_{a} f(a)g(t-a) \tag{24}$$

As funções f e g na terminologia das redes convolucionais são identificadas por entrada (*input*) e filtro (*kernel*), respectivamente. O resultado da convolução é chamado de mapa de características.

No exemplo dado, a entrada da convolução é uma função real, portanto, o conjunto dos dados obtidos é um vetor unidimensional. Acontece que nos problemas onde se aplicam as rede convolucionais, os dados de entrada geralmente tem estrutura multidimensional chamada de tensor, tal como imagens, que são representadas por uma estrutura bi ou tridimensional. Nestes casos a expressão (24) deve considerar todas as dimensões da entrada. No caso de uma imagem bidimensional, tem-se

$$s(x, y) = (f * g)(x, y) = \sum_{i} \sum_{j} f(i, j)g(x-i, y-j) \tag{25}$$

Independente da dimensão da entrada, observe que quando o argumento da função

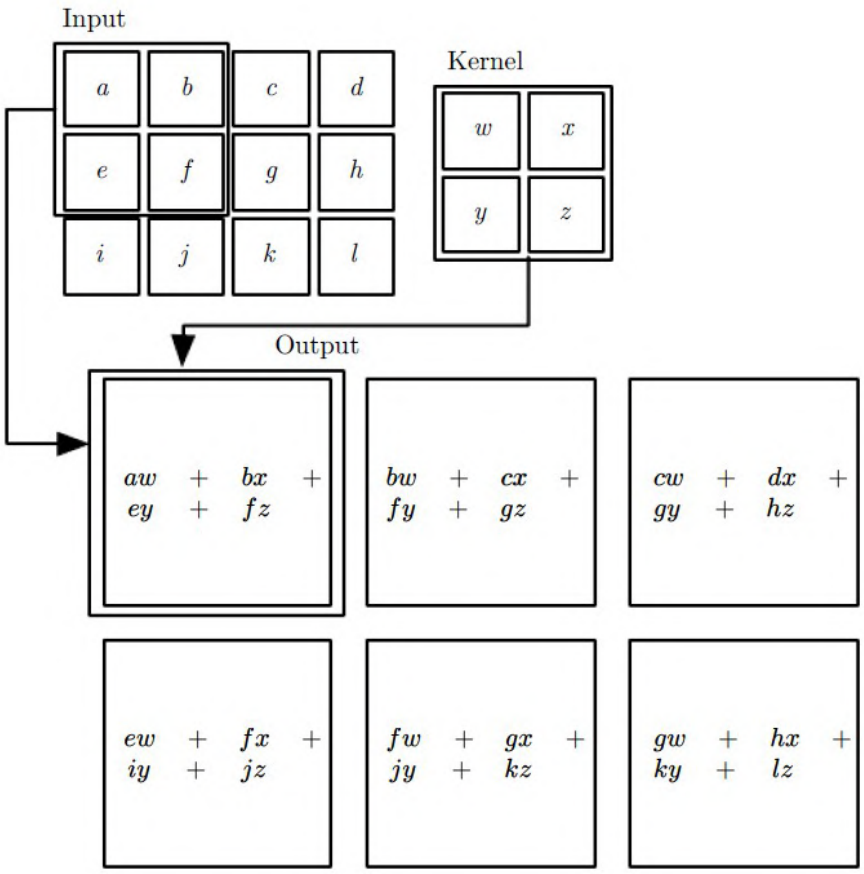


Figura 12: *Exemplo de uma "convolução" sobre uma entrada de dimensão 3x4 e um filtro de dimensão 2x2. Operação restrita à região onde o filtro está inteiramente contido na entrada. Fonte: [GBC16]*

de entrada aumenta, o argumento do filtro diminui. Isso, na prática, provoca uma rotação de 180 graus do filtro, tornando-se um inconveniente para a implementação computacional. Além disso, o controle sobre os índices do somatório são mais complicados.

Essa rotação do filtro é evitada implementando a função *cross-correlation* ao invés da convolução e é definida por

$$s(x,y) = (f * g)(x,y) = \sum_i \sum_j f(x+i, y+j) g(i,j) \tag{26}$$

Apesar de não ser uma convolução propriamente dita, muitos a implementam como se fosse. Na figura 12 é apresentada a forma como a função *cross-correlation* atua sobre uma matriz.

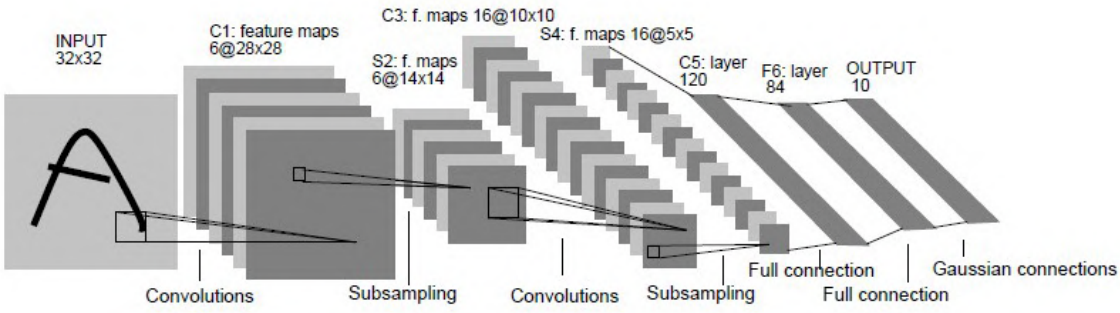


Figura 13: *Arquitetura da CNN utilizada para reconhecimento de dígitos escritos à mão. Fonte: [LBBH98]*

3.8.3 Arquitetura

A arquitetura de uma CNN é formada por uma camada de entrada, seguida de uma ou mais camadas convolucionais e *pooling*. Por fim, uma ou mais camadas MLP totalmente conectadas formam o classificador que fará a interpretação da entrada de acordo com as características identificadas pelas camadas convolucionais e de *pooling*.

A figura 13 é um exemplo onde é possível ver cada camada citada e que serão detalhadas abaixo.

1. **Camadas Convolucionais**

   Como não poderia deixar de ser, a camada convolucional é o principal componente desse tipo de rede. Não apenas dá o nome, mas sua presença em pelo menos uma camada da RNA a define como uma CNN [GBC16].

   Ela é formada por um conjunto de filtros que são aplicados à entrada pela operação de convolução. O número de filtros por camada, denotado por k, e o tamanho do campo receptivo local são os primeiros hiperparâmetros dessa arquitetura.

   Tomando como exemplo, uma camada convolucional composta por 3 filtros de dimensão 3x3 aplicados a uma imagem representada por uma matriz de dimensão 28x28 produz 3 mapas de características, cada um correspondendo a um filtro.

   A dimensão do mapa de características depende de k e de mais dois hiperparametros: o passo do filtro (*stride*), denotado por s, que estabelece quanto o filtro será deslocado entre convoluções, e do preenchimento com zeros (*zero-padding*), denotado por p, que é utilizada no aumento da dimensão da entrada.

   No exemplo acima, aplicando os 3 filtros com campo receptivo de dimensão 5x5, com passo unitário e sem realizar o preenchimento com zeros, obtêm-se uma

saída de dimensão 24x24x3, ou seja, 3 mapas de características de dimensão 24x24.

A relação do volume de saída da camada convolucional com os hiperparâmetros é bem simples. Para cada dimensão da entrada vale a seguinte expressão [DV18]:

$$O = \frac{(I - F - 2p)}{s} + 1 \tag{27}$$

onde I é a dimensão da entrada e F a dimensão do filtro.

Os filtros aplicados na camada de convolução são os mesmos filtros utilizados em técnicas de processamento de imagem utilizadas em visão artificial, por exemplo o filtro de detecção de bordas. A diferença entre a CNN e as técnicas "clássicas" de processamento de imagem é que as CNN são treinadas para aprender quais filtros devem ser aplicados em cada camada. Não há nenhuma escolha prévia desses filtros, a não ser sua quantidade por camadas e tamanho do campo receptivo.

2. **Camada de *Pooling***

   Os mapas de características gerados pela camada convolucional são geralmente submetidos ao processo de *pooling*. Sua função é condensar os mapas de características para reduzir o número de parâmetros de entrada da camada seguinte.

   Cada unidade na camada de *pooling*, por exemplo, reduz uma região de 2x2 neurônios da camada anterior para 1 neurônio. Com isso, um mapa 24x24 é reduzido para 12x12.

   Uma forma comum de realizar o *pooling* é conhecida como *max pooling*, onde se preserva a ativação máxima na região de entrada, em outras palavras, ele mantém apenas o maior valor na região onde é aplicado [ZC88]. Outras técnicas utilizadas são a média dos valores da região de entrada, conhecido como *average pooling*, a norma L^2 dos valores de ativação ou uma média ponderada baseada na distância do elemento central da região [GBC16].

   A técnica que melhor se ajusta ao problema a ser resolvido é um hiperparâmetro e, como tal, deve ser escolhida com base no resultado da validação.

Ao final da convolução, aplica-se uma função de ativação não linear, geralmente a ReLU, sobre o resultado, estabelecendo o seguinte padrão de *layout* da arquitetura de

uma CNN.

$$\text{Entrada} \rightarrow [[\text{Convolução}]+ \rightarrow [\text{Pooling}]?]+ \rightarrow [\text{TC}]+$$

onde $[\cdot]+$ indica ocorrência de $\cdot$ uma ou mais vezes, $[\cdot]?$ indica a ocorrência de $\cdot$ no máximo uma vez e TC um camada totalmente conectada.

4 METODOLOGIA

4.1 VISÃO GERAL

O objetivo principal deste trabalho é dar a um modelo a capacidade de reconhecer o objeto requerido nas imagens apresentadas pelo reCAPTCHA. Sua utilização, na prática, consiste em avaliar cada imagem do reCAPTCHA e indicar quais contém o objeto requerido. Na figura 14, por exemplo, o modelo depois de treinado deve indicar que as imagens em destaque devem ser selecionadas.

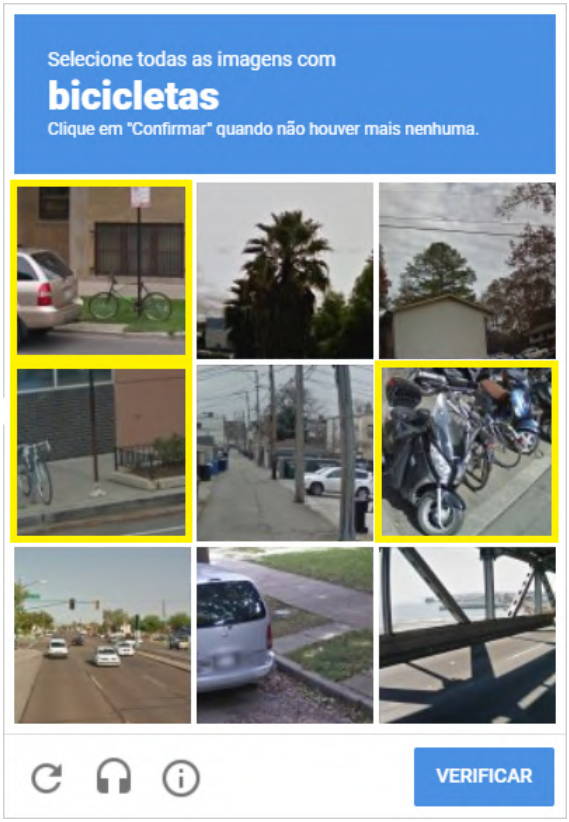


Figura 14: *O destaque em amarelo indica que a imagem devem ser selecionada para que o desafio seja resolvido.*

A estratégia utilizada para atingir o objetivo destacado, ilustrada na figura 15, consistiu, numa primeira etapa, coletar um conjunto de imagens das classes de interesse - bicicletas, carros, ônibus e semáforos - e treinar o modelo para classificá-las.

Já na etapa seguinte, de avaliação do modelo, uma nova atividade de coleta de imagens foi realizada, mas agora com objetivo de formar um conjunto de imagens do reCAPTCHA que posteriormente foi submetido ao modelo já treinado para avaliação de seu desempenho.

As atividades descritas nas etapas 1 e 2, foram subdivididas em outras, conforme listadas abaixo.

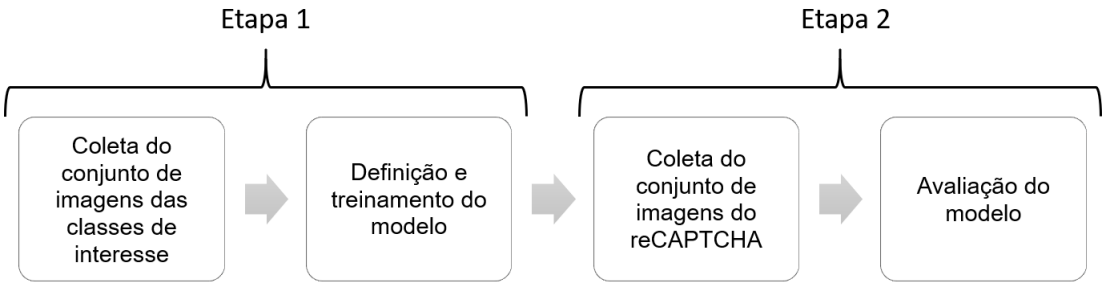


Figura 15: *Etapas desenvolvidas para capacitar o modelo a resolver o reCAPTCHA.*

1. Coleta do conjunto de imagens das classes de interesse

   (a) Coleta de imagens de interesse

   Uma busca manual na internet por bancos de imagens públicos foi realizada com o objetivo de identificar quais teriam as imagens de interesse em qualidade e quantidade suficientes para serem utilizadas. As imagens foram armazenadas localmente por meio de um script desenvolvido para tal.

   (b) Seleção das imagens apropriadas

   Para cada classe de interesse foram coletadas milhares de imagens e muitas delas não eram apropriadas para serem utilizadas no treinamento do modelo, pois não tinham as mesmas características das imagens do reCAPTCHA, tal como mostrado na figura 16. Foram selecionadas manualmente 1000 imagens de cada classe observando se eram semelhantes às imagens do reCAPTCHA quanto ao contexto dos objetos de interesse.

(a) Bicicleta (b) Carro (c) Ônibus (d) Semáforo

Figura 16: *Exemplos de imagens retiradas do conjunto de imagens obtido. As figuras em 16(a), 16(b), 16(c) e 16(d) estavam presentes no conjunto de imagens das classes bicicleta, carro, ônibus e semáforo, respectivamente.*

2. Definição e treino dos modelos de classificação

   (a) Definição dos modelos

Devido ao sucesso das arquiteturas convolucionais em aplicações de visão computacional, optou-se inicialmente por um modelo de rede neural convolucional clássico, ou seja, um conjunto de camadas convolucionais seguido de camadas densas (i.e., totalmente conectadas), responsáveis pela classificação da imagem.

No entanto, o classificador desse modelo requer um aprendizado de todas as classes possíveis de imagens que poderiam aparecer no reCAPTCHA. Isto torna-se inviável, pois as imagens negativas podem exibir inúmeros cenários e objetos. Além disso, a inclusão de novos objetos requeridos pelo reCAPTCHA demandaria um novo treinamento do modelo.

A fim de mitigar essas limitações, optou-se por utilizar também os classificadores de classe única: OC-SVM e kernel PCA. Nestes modelos, a inclusão de novos objetos não impactaria no reconhecimento dos objetos já conhecidos, uma vez que cada classe utilizaria um classificador treinado para reconhecê-la.

Independentemente do classificador, os modelos tem a mesma arquitetura, ou seja, eles recebem uma imagem representada por uma matriz, extraem suas características nas camadas convolucionais e, com base nessas características, as camadas densas decidem a que classe a imagem pertence, no caso do modelo com a RNA totalmente conectada, ou se pertence ou não a determinada classe, no caso dos modelos com classificadores de classe única.

(b) Treinamento do modelos

A extração de características precedendo à etapa de classificação é comum a todos os modelos, por isso todas as imagens utilizadas no treinamento tiveram, antes de tudo, suas características extraídas e armazenadas num arquivo onde cada coluna representa um atributo e cada linha uma imagem. Isso possibilitou uma maior velocidade e flexibilidade no treinamento dos classificadores.

Tanto a arquitetura quanto os pesos das camadas convolucionais utilizadas na extração das características foram aproveitadas de uma rede neural já treinada, prática conhecida como (*transfer learning*), no entanto, as camadas densas foram treinadas separadamente.

Os hiperparâmetros de cada modelo foram definidos variando seus valores e avaliando o resultado sobre um conjunto de validação.

3. Obtenção dos desafios do reCAPTCHA

(a) Busca e armazenamento

Para a avaliação dos modelos optou-se por armazenar localmente as imagens de diversos desafios do reCAPTCHA relativos aos objetos que os modelos foram treinados para reconhecerem. A busca e armazenamento desses desafios foi realiza utilizando ferramenta de automação de tarefas[1], que ao identificar que o desafio se refere a uma classe treinada, armazena as nove imagens exibidas.

(b) Extração de características

Semelhante ao que foi realizado no treinamento dos modelos, as características das imagens foram extraídas utilizando os blocos convolucionais, comum aos modelos, e armazenadas localmente. Isso permitiu comparar o resultado do teste de cada modelo, pois o conjunto de imagens é o mesmo para todos. Outro ponto é que todos os mecanismos de detecção de robô utilizados pelo reCAPTCHA que poderiam interferir no resultado, tal como o movimento do mouse, foram eliminados, focando apenas na capacidade do modelo de reconhecer as imagens.

4. Avaliação dos desafios pelos modelos treinados

A avaliação dos modelos foi realizada considerando a quantidade de desafios resolvidos. A partir do conjunto de características obtido dos desafios, os modelos classificaram cada imagem individualmente, gerando um conjunto de imagens com sua classificação. Uma função de avaliação foi aplicada sobre esse conjunto, distinguindo as imagens de cada desafio e contabilizando quantos foram resolvidos.

Dois critérios foram estabelecidos para medir a quantidade de desafios resolvidos: o primeiro admite como resolvido apenas os desafios que tenham 100% de acerto, ou seja, nenhuma imagem foi selecionada indevidamente ou deixada de ser selecionada quando era para ser; o segundo, admite um falso-positivo, ou seja, apenas uma imagem foi selecionada indevidamente e todas as imagens corretas foram selecionadas.

O segundo critério explora uma flexibilidade do reCAPTCHA que, segundo trabalhos relacionados, foi removida pela Google, no entanto, ela ainda está presente em alguns momentos aleatoriamente, conforme verificado interagindo com o reCAPTCHA manualmente. Não foi possível perceber algum padrão ou critério para determinar quando essa flexibilidade aparece, portanto, no segundo critério se considerou que ela estava presente em todos os desafios avaliados.

[1] Maiores detalhes podem ser encontrados em http://www.uipath.com

4.2 EXTRAÇÃO DE CARACTERÍSTICAS PROFUNDAS

Extração de características é uma técnica empregada para reduzir a quantidade de informação à respeito da entrada, mantendo apenas o suficiente para continuar a representando com certa precisão.

Se as características extraídas forem selecionadas adequadamente se espera que esse conjunto represente a parte relevante da informação para se executar a tarefa desejada ao invés de se usar os dados de entrada na integra, ou seja, com todos os seus atributos.

Métodos de extração de características determina um subespaço apropriado de dimensionalidade m (conjunto contendo m características) a partir de um espaço de dimensionalidade d, sendo m $\leq$ d [JDM00].

A definição do método de extração de características é um dos fatores mais importantes para a construção de um sistema de visão ou reconhecimento de padrões. O desempenho de um sistema de visão está diretamente relacionado ao poder de discriminação do conjunto de características escolhido. As características devem destacar as diferenças entre objetos de classes distintas (interclasses) e minimizar possíveis diferenças de objetos de mesma classe (intraclasse).

Para a extração de características das imagens do CAPTCHA utilizou-se uma rede neural já treinada denominada MobileNet [HZC$^+$17]. Ela foi escolhida por ser uma rede leve e por ter desempenho satisfatório para esta aplicação.

Originalmente é uma rede neural convolucional com 28 camadas de convoluções separáveis profundas, que divide uma convolução padrão em uma convolução profunda e uma convolução 1x1, chamada de convolução pontual.

Uma convolução padrão filtra e combina entradas em um novo conjunto de saídas em uma única etapa. A convolução separável profunda divide isso em duas camadas, uma camada para filtrar e uma camada para combinar. Essa fatoração tem o efeito de reduzir drasticamente a computação e o tamanho do modelo. Ao final das camadas de convoluções separáveis é realizado um *average pooling* cuja saída é conectada a uma camada totalmente conectada fazendo a função do classificador [HZC$^+$17].

Neste trabalho o *average pooling* foi substituído pelo *max pooling* e adicionada uma camada totalmente conectada para redução da dimensão do espaço de características de 1024 para $D = 8; 16; 32; 64; 128; 256; 512$, conforme tabela 2.

A figura 17 mostra a projeção no $\mathbb{R}^2$ do espaço de características no $\mathbb{R}^{1024}$ de 50 imagens contendo bicicletas e 50 imagens contendo ônibus extraídas pela MobileNet. Essa projeção foi gerada com a técnica t-Distributed Stochastic Neighbour Embed-

Tabela 2: *Arquitetura da MobileNet utilizada para extração das características.*

	Tipo de Camada	Tamanho do passo	Dimensão do Filtro	Dimensão da Entrada
1	Convolucional	2	3 x 3 x 3 x 32	128 x 128 x 3
2	Convolucional Prof.	1	3 x 3 x 32 Prof.	64 x 64 x 32
3	Convolucional Pontual	1	1 x 1 x 32 x 64	64 x 64 x 32
4	Convolucional Prof.	2	3 x 3 x 64 Prof.	64 x 64 x 64
5	Convolucional Pontual	1	1 x 1 x 64 x 128	32 x 32 x 64
6	Convolucional Prof.	1	3 x 3 x 128 Prof.	32 x 32 x 128
7	Convolucional Pontual	1	1 x 1 x 128 x 128	32 x 32 x 128
8	Convolucional Prof.	2	3 x 3 x 128 Prof.	32 x 32 x 128
9	Convolucional Pontual	1	1 x 1 x 128 x 256	16 x 16 x 128
10	Convolucional Prof.	1	3 x 3 x 256 Prof.	16 x 16 x 256
11	Convolucional Pontual	1	1 x 1 x 256 x 256	16 x 16 x 256
12	Convolucional Prof.	2	3 x 3 x 256 Prof.	16 x 16 x 256
13	Convolucional Pontual	1	1 x 1 x 256 x 512	8 x 8 x 256
14	Convolucional Prof.	1	3 x 3 x 512 Prof.	8 x 8 x 512
15	Convolucional Pontual	1	1 x 1 x 512 x 512	8 x 8 x 512
16	Convolucional Prof.	1	3 x 3 x 512 Prof.	8 x 8 x 512
17	Convolucional Pontual	1	1 x 1 x 512 x 512	8 x 8 x 512
18	Convolucional Prof.	1	3 x 3 x 512 Prof.	8 x 8 x 512
19	Convolucional Pontual	1	1 x 1 x 512 x 512	8 x 8 x 512
20	Convolucional Prof.	1	3 x 3 x 512 Prof.	8 x 8 x 512
21	Convolucional Pontual	1	1 x 1 x 512 x 512	8 x 8 x 512
22	Convolucional Prof.	1	3 x 3 x 512 Prof.	8 x 8 x 512
23	Convolucional Pontual	1	1 x 1 x 512 x 512	8 x 8 x 512
24	Convolucional Prof.	2	3 x 3 x 512 Prof.	8 x 8 x 512
25	Convolucional Pontual	1	1 x 1 x 512 x 1024	4 x 4 x 512
26	Convolucional Prof.	2	3 x 3 x 1024 Prof.	4 x 4 x 1024
27	Convolucional Pontual	1	1 x 1 x 1024 x 1024	4 x 4 x 1024
28	Max Pooling	1	7 x 7	7 x 7 x 1024
29	Totalmente Conectada		1024 x D	1 x 1 x 1024

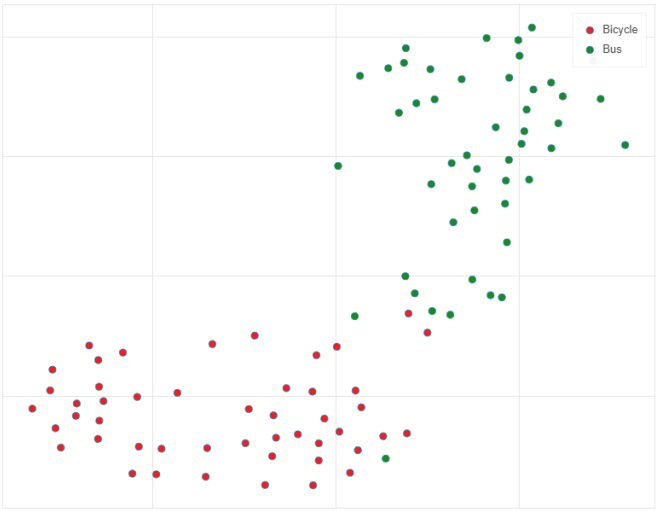


Figura 17: *Projeção no $\mathbb{R}^2$, usando t-SNE, do espaço de características de imagens contendo bicicletas e ônibus.*

ding (t-SNE) que reduz a dimensão dos dados (em alta dimensão), permitindo sua visualização em um espaço de duas ou três dimensões [MH08].

Note que objetos da mesma classe, de maneira geral, ficam próximos indicando possuírem as mesmas características, tais como forma e o contexto (cenário onde estão). Isso demonstra a capacidade da MobileNet para este propósito.

4.3 CLASSIFICAÇÃO DE IMAGEM

Conforme visto na seção 2.2.1, um problema de classificação nada mais é que encontrar uma função discriminante $f : \mathbb{R}^n \rightarrow \{C_1, C_2, ..., C_k\}$, que associa um exemplo $\mathbf{x} \in \mathbb{R}^n$ a uma classe C_i, $i = 1, ..., k$.

Nos problemas de classificação em geral, as classes C_k são disjuntas, ou seja, cada exemplo pertence a uma única classe. Com isso, faz todo sentido dividir o espaço do domínio em regiões disjuntas por meio de fronteiras (de decisão), onde cada região representa uma classe.

Essas fronteiras de decisão são definidas ou suportadas pelos exemplos existentes de cada classe. É necessário que uma quantidade suficiente de exemplos de cada classe esteja disponível e que esteja balanceada, pois classificadores multiclasse não tem bom desempenho quando qualquer classe tem poucos exemplos.

Definir essa fronteira para classificadores de classe única (OCC), descrito também na seção 2.2.1, é um problema muito mais difícil de ser solucionado, pois a ausência ou a insuficiência de exemplos negativos possibilita apenas a determinação de um limite da fronteira que separa os exemplares positivos e negativos. O desafio é definir uma fronteira de classificação em torno da classe positiva, de modo que aceite os

exemplos positivos, enquanto minimiza a chance de aceitar os exemplos da classe negativa. Para isso é necessário definir quais os exemplos devem ser considerados e o limite em cada direção ao redor deles.

4.3.1 CNN como um classificador multiclasse

Diversos modelos foram desenvolvidos ao longo de décadas para encontrar uma função discriminante adequada para os problema de classificação multiclasse, não existindo um que seja melhor que o outro de forma geral. O desempenho de cada modelo depende do problema que se queria resolver.

Em problemas de classificação de imagens é utilizada a rede neural convolucional clássica vista na seção 3.8, onde as características extraídas pelas camadas convolucionais alimentam uma sequência de camadas totalmente conectadas para realizar a classificação desejada.

A imagem à cores no sistema RGB [2] é representada por 3 matrizes bi-dimensionais $a \times l$, onde cada matriz representa um canal. Em outras palavras, uma entrada (i, j, k) com $k = 1, 2, 3$ corresponde a intensidade do canal k.

Essa representação tridimensional preserva a relação local entre os elementos da imagem e permite que os campos receptivos locais, por meio das operações de convolução e *pooling* descritas na seção 3.8, extraiam suas características.

As características são representadas por um vetor e alimentam a rede em avanço totalmente conectada que utiliza a Softmax como função de saída e a entropia cruzada como função de custo a ser minimizada.

4.3.2 Kernel PCA para detecção de novidades

O Kernel PCA estende o PCA padrão para distribuições de dados não lineares, mapeando pontos do espaço de entrada em um espaço de características de dimensão superior.

Para um conjunto de exemplos $\mathbf{x}_k \in \mathbb{R}^n$, com $k = 1, \ldots, M$, as operações originalmente lineares do PCA são executadas sobre imagem da aplicação não linear $\mathbf{\Phi}(\mathbf{x}_k)$, ou seja, no espaço de características F de dimensão $d > n$, conforme exemplificado na figura 18.

Importante destacar que na aplicação do PCA, os exemplos são centrados em torno da origem, fazendo com que a média seja nula, no entanto, não há garantia que isso

[2] RGB é a sigla do sistema de cores formado pelas iniciais das cores em inglês Red, Green e Blue. As cores são obtidas de acordo com a intensidade de cada uma dessas cores, que denotaremos por canal, em quantidades determinadas que variam entre os inteiros 0 e 255 ou no intervalo [0, 1].

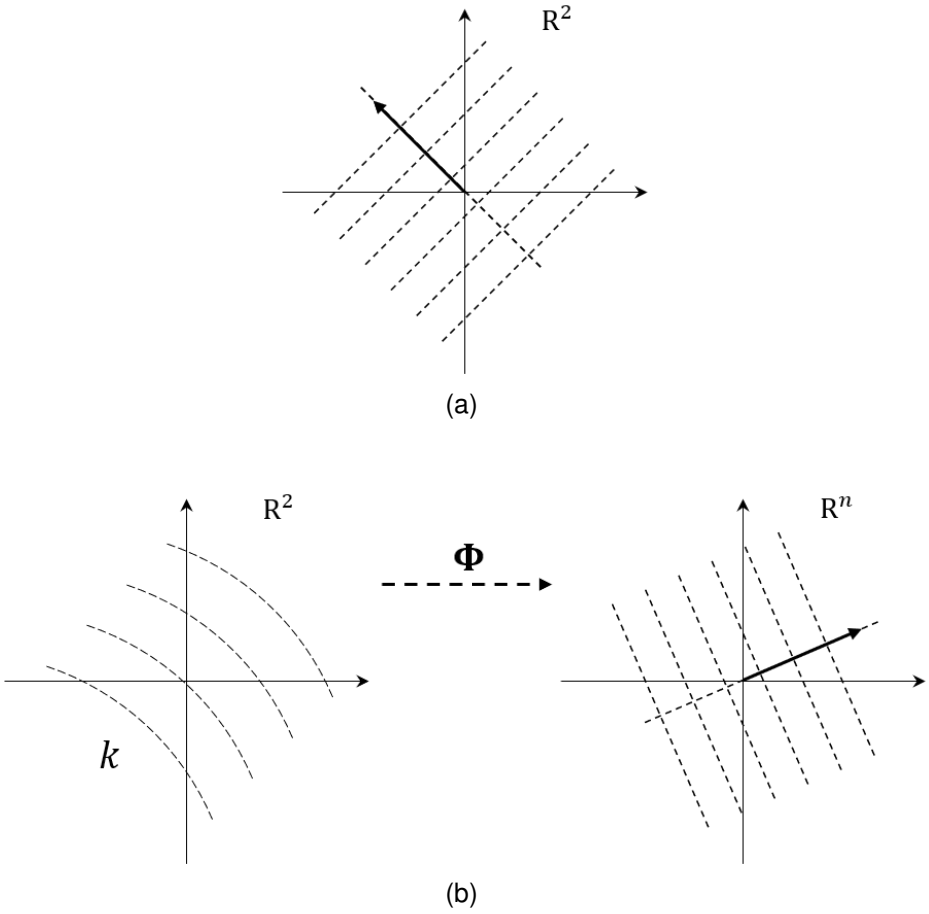


Figura 18: *Ideia básica do kernel PCA. (a) PCA aplicado ao espaço de entrada. (b) PCA aplicado ao espaço de características gerado por uma aplicação não linear.*

ocorra com os pontos $\mathbf{\Phi}(\mathbf{x}_k)$.

De forma semelhante ao PCA, isso é feito subtraindo cada ponto $\mathbf{\Phi}(\mathbf{x}_k)$ da média $\frac{1}{M}\sum \mathbf{\Phi}(\mathbf{x}_k)$, mas isso, como será demonstrado, é desnecessário. Por hora, considere que a imagem de $\mathbf{\Phi}$ está centrada na origem.

Neste sentido, a matriz de covariância em F é dada por

$$\mathbf{C} = \frac{1}{M}\sum_{j=1}^{M} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^T \tag{28}$$

O problema neste momento é encontrar autovetores de $\mathbf{C}$, ou seja, $\mathbf{V} = (V_1, \ldots, V_d) \in F\backslash\{0\}$ e $\lambda \geq 0$, tais que

$$\lambda\mathbf{V} = \mathbf{C}\mathbf{V} \tag{29}$$

Pela equação (28), tem-se

$$\mathbf{CV} = \frac{1}{M}\sum_{k=1}^{M}(\Phi(\mathbf{x}_k)^T\mathbf{V})\Phi(\mathbf{x}_k) \tag{30}$$

Substituindo a equação (30) na equação (29), conclui-se que $\mathbf{V}$ está no espaço gerado por $\mathbf{\Phi}(\mathbf{x}_k)$ para $k = 1, \ldots, M$, logo existem α_i ($i = 1, \ldots, M$) tais que

$$\mathbf{V} = \sum_{i=1}^{M}\alpha_i\mathbf{\Phi}(\mathbf{x}_i) \tag{31}$$

Realizando o produto interno entre $\mathbf{\Phi}(\mathbf{x}_k)$ para $k = 1, \ldots, M$ e os dois membros da equação (29), obtem-se

$$\lambda(\mathbf{\Phi}(\mathbf{x}_k).\mathbf{V}) = (\mathbf{\Phi}(\mathbf{x}_k).\mathbf{CV}) \tag{32}$$

Substituindo as equações (28) e (31) na equação (32)

$$\begin{aligned}
\lambda\sum_{i=1}^{M}\alpha_i(\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_i)) &= \mathbf{\Phi}(\mathbf{x}_k)\cdot\left[\frac{1}{M}\sum_{j=1}^{M}\mathbf{\Phi}(\mathbf{x}_j)\cdot\left(\sum_{i=1}^{M}\alpha_i\mathbf{\Phi}(\mathbf{x}_i)\right)\mathbf{\Phi}(\mathbf{x}_j)\right] \\
&= \frac{1}{M}\mathbf{\Phi}(\mathbf{x}_k)\cdot\left[\sum_{j=1}^{M}\sum_{i=1}^{M}\alpha_i(\mathbf{\Phi}(\mathbf{x}_j)\cdot\mathbf{\Phi}(\mathbf{x}_i))\mathbf{\Phi}(\mathbf{x}_j)\right] \\
&= \frac{1}{M}\mathbf{\Phi}(\mathbf{x}_k)\cdot\left[\sum_{j=1}^{M}\mathbf{\Phi}(\mathbf{x}_j)\sum_{i=1}^{M}\alpha_i(\mathbf{\Phi}(\mathbf{x}_j)\cdot\mathbf{\Phi}(\mathbf{x}_i))\right] \\
&= \frac{1}{M}\sum_{j=1}^{M}\mathbf{\Phi}(\mathbf{x}_k)\cdot\mathbf{\Phi}(\mathbf{x}_j)\sum_{i=1}^{M}\alpha_i(\mathbf{\Phi}(\mathbf{x}_j)\cdot\mathbf{\Phi}(\mathbf{x}_i))
\end{aligned} \tag{33}$$

Substituindo a função kernel $k_{nm} = \mathbf{\Phi}(\mathbf{x}_n)\cdot\mathbf{\Phi}(\mathbf{x}_m)$ na equação (33), tem-se

$$M\lambda\sum_{i=1}^{M}\alpha_i k_{ki} = k(\mathbf{x}_k, \mathbf{x}_i) = \sum_{j=1}^{M}k_{kj}\sum_{i=1}^{M}\alpha_i k_{ij} \tag{34}$$

que em notação matricial pode ser escrita como

$$M\lambda K\alpha = K^2\alpha \tag{35}$$

onde $K = [k_{ij}]$ é uma matriz $M \times M$ simétrica, portanto se recai em outro problema

de encontrar autovalores, mas agora para a matriz K, denotada por Matriz de Gram.

$$K\alpha = M\lambda\alpha \tag{36}$$

Finalmente, a projeção de um ponto $\mathbf{x}$ no espaço de entrada sobre um autovetor V é dada por

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{V} \\ &= \mathbf{\Phi}(\mathbf{x}) \cdot \sum_{i=1}^{M} \alpha_i \mathbf{\Phi}(\mathbf{x}_i) \\ &= \sum_{i=1}^{M} \alpha_i (\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x}_i)) \\ &= \sum_{i=1}^{M} \alpha_i k(\mathbf{x}, \mathbf{x}_i) \end{aligned} \tag{37}$$

A demonstração do kernel PCA se baseou no fato da imagem da $\mathbf{\Phi}(\mathbf{x}_k)$ ter média zero, no entanto, de forma geral isso não é garantido. A imagem centralizada é dada pela expressão (38), que não é interessante, pois não se deseja calcular a imagem da $\mathbf{\Phi}$ para os pontos do espaço de entrada. Todo o esforço feito até aqui seria em vão.

$$\widetilde{\mathbf{\Phi}}(\mathbf{x}_k) = \mathbf{\Phi}(\mathbf{x}_k) - \frac{1}{M} \sum_{i=1}^{M} \mathbf{\Phi}(\mathbf{x}_i) \tag{38}$$

Essa dificuldade é superada colocando a matriz de Gram dos pontos centralizados em função da matriz de Gram dos pontos em suas posições originais.

$$\begin{aligned} \widetilde{k}_{nm} &= \widetilde{\mathbf{\Phi}}(\mathbf{x}_n) \cdot \widetilde{\mathbf{\Phi}}(\mathbf{x}_m) \\ &= \left(\mathbf{\Phi}(\mathbf{x}_n) - \frac{1}{M} \sum_{i=1}^{M} \mathbf{\Phi}(\mathbf{x}_i) \right) \cdot \left(\mathbf{\Phi}(\mathbf{x}_m) - \frac{1}{M} \sum_{j=1}^{M} \mathbf{\Phi}(\mathbf{x}_j) \right) \\ &= \mathbf{\Phi}(\mathbf{x}_n) \cdot \mathbf{\Phi}(\mathbf{x}_m) - \frac{1}{M} \sum_{j=1}^{M} \mathbf{\Phi}(\mathbf{x}_n) \cdot \mathbf{\Phi}(\mathbf{x}_j) - \\ &\quad - \frac{1}{M} \sum_{i=1}^{M} \mathbf{\Phi}(\mathbf{x}_m) \cdot \mathbf{\Phi}(\mathbf{x}_i) + \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j) \end{aligned} \tag{39}$$

Em notação matricial, tem-se

$$\widetilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_M \mathbf{K} - \mathbf{K}\mathbf{1}_M + \mathbf{1}_M \mathbf{K} \mathbf{1}_M \tag{40}$$

onde $\mathbf{1}_M = \left[\frac{1}{M}\right]_{ij}$ é uma matriz $M \times M$. Dessa forma pode-se calcular $\widetilde{\mathbf{K}}$ usando o kernel e depois determinar seus autovetores e autovalores.

Essa formulação para o kernel PCA é baseada na maximização da variância da projeção do espaço de entrada no espaço de característica. É a mesma ideia para a formulação do PCA linear, mas não é a única.

Uma alternativa é a minimização do erro de reconstrução do espaço de entrada, mas que em certa altura enfrenta o mesmo problema de encontrar autovetores para a matriz de covariância [Bis06].

A reconstrução do espaço de entrada m-dimensional é a operação "inversa" da projeção, que é uma combinação linear dos n autovetores obtidos pelo PCA. Como n é menor que m (devido a redução de dimensionalidade), o espaço reconstruído possui um erro associado.

Na detecção de novidades, ou seja, na classificação de um exemplo como anormal, espera-se que o erro de reconstrução para exemplos anormais, sob certo sentido, seja suficientemente grande. Na prática, o erro de reconstrução do exemplo avaliado é comparado com o maior erro obtido sobre o conjunto de treinamento, denominado de *threshold*.

O erro de reconstrução utilizado neste trabalho é o mesmo utilizado em [Hof07] e é dado por

$$p(\widetilde{\mathbf{\Phi}}) = (\widetilde{\mathbf{\Phi}} \cdot \widetilde{\mathbf{\Phi}}) - (\mathbf{W}\widetilde{\mathbf{\Phi}} \cdot \mathbf{W}\widetilde{\mathbf{\Phi}}) \tag{41}$$

onde $\widetilde{\mathbf{\Phi}}$ é dada pela equação (38) e $\mathbf{W}$ a matriz cujas linhas são os q autovetores selecionados para gerar a projeção.

Sendo α_i as coordenadas do i-ésimo autovetor no espaço de características, conforme equação (31), tem-se $\mathbf{W}\widetilde{\mathbf{\Phi}} = [\widetilde{\mathbf{K}}\alpha_i]_i$ para $i = 1, \ldots, q$.

Note que $\widetilde{\mathbf{\Phi}} \cdot \widetilde{\mathbf{\Phi}}$ pode ser escrito em função apenas de $\mathbf{x}$ usando o kernel.

$$\widetilde{\mathbf{\Phi}}(\mathbf{x}) \cdot \widetilde{\mathbf{\Phi}}(\mathbf{x}) = \widetilde{\mathbf{k}}(\mathbf{x}, \mathbf{x}) \tag{42}$$

Portanto, a equação (41) pode ser escrita na forma

$$p(\mathbf{x}) = \widetilde{\mathbf{k}}(\mathbf{x}, \mathbf{x}) - \sum_{i=1}^{q} \widetilde{\mathbf{K}}\alpha_i \tag{43}$$

4.3.3 One-Class SVM

Antes de explicitar o One-Class SVM, vale relembrar que o SVM (*Suport Vector Machine*) é um algoritmo de classificação largamente utilizado para distinguir entre duas ou mais classes.

O SVM implementa essa classificação utilizando um ou mais hiperplanos, a depender da quantidade de classes, cuja distância para determinados elementos de cada classe é a maior possível. Essa distância é conhecida como margem (ou semi-margem) e esses elementos recebem o nome de vetores de suporte, justamente porque definem a equação desses hiperplanos.

Como na maioria dos problemas de classificação é impossível realizar a separação de classes com hiperplanos no espaço dos dados de entrada, o SVM utiliza um mapeamento Φ não-linear do espaço de entrada para o espaço de características de alta dimensão com a intenção de tornar esses dados linearmente separáveis.

Semelhante ao kernel PCA, a função Φ não precisa ser explicitada, sendo necessário apenas definir a função kernel, que depende somente dos elementos do espaço de entrada.

O OC-SVM também realiza a classificação baseado em um hiperplano, no entanto, a origem é considerada como uma segunda classe e o hiperplano procurado é definido como sendo aquele que possui máxima distância da origem, conforme representado na figura 19.

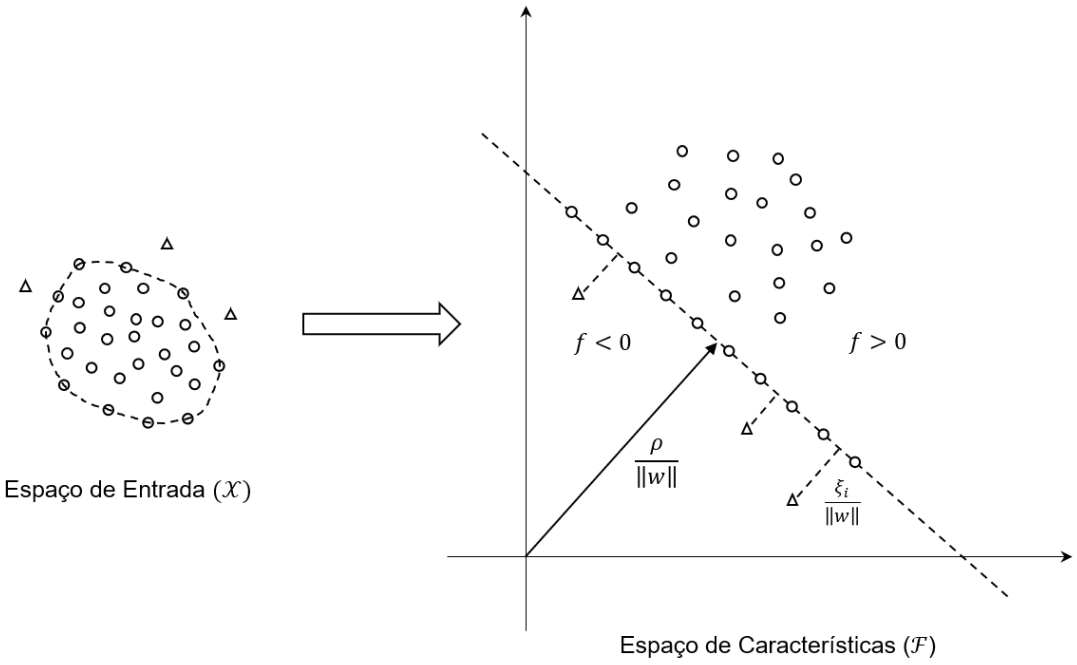


Figura 19: *Representação do One-Class SVM demonstrando os dados no espaço de entrada e hiperplano separando-os da origem no espaço de características.*

Esse hiperplano é obtido encontrando os valores de $\mathbf{w} \in \mathcal{F}$ e $\xi_i, \rho \in \mathbb{R}$ que minimizam a equação

$$\frac{1}{2}||w||^2 + \frac{1}{\nu l}\sum_i \xi_i - \rho \tag{44}$$

sujeito à condições

$$(w \cdot \Phi(x_i)) \geq \rho - \xi_i \tag{45}$$

$$\xi_i \geq 0 \tag{46}$$

onde $\xi_\mathbf{i} \in \mathbb{R}$ são variáveis de folga representando o erro associado a x_i e ν é a fração máxima permitida de elementos que poderão ser considerados como anormais.

Utilizando multiplicadores de Lagrange pode-se determinar os valores de $\mathbf{w}$ e ρ, e, juntamente com a função kernel k, obter a função de classificação

$$f(x) = \begin{cases} -1, & \sum_i \alpha_i k(x_i, x) - \rho < 0 \\ +1, & \sum_i \alpha_i k(x_i, x) - \rho \geq 0 \end{cases} \tag{47}$$

Sendo k o kernel Gaussiano, a imagem de $\mathbf{\Phi}(\mathbf{x})$ está na superfície de uma hiperesfera em $\mathcal{F}$, pois $k(\mathbf{x}, \mathbf{x}) = \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{x}) = 1$, logo $||\mathbf{\Phi}(\mathbf{x})|| = 1$. Dessa forma, o OC-SVM é melhor representado pela figura 20.

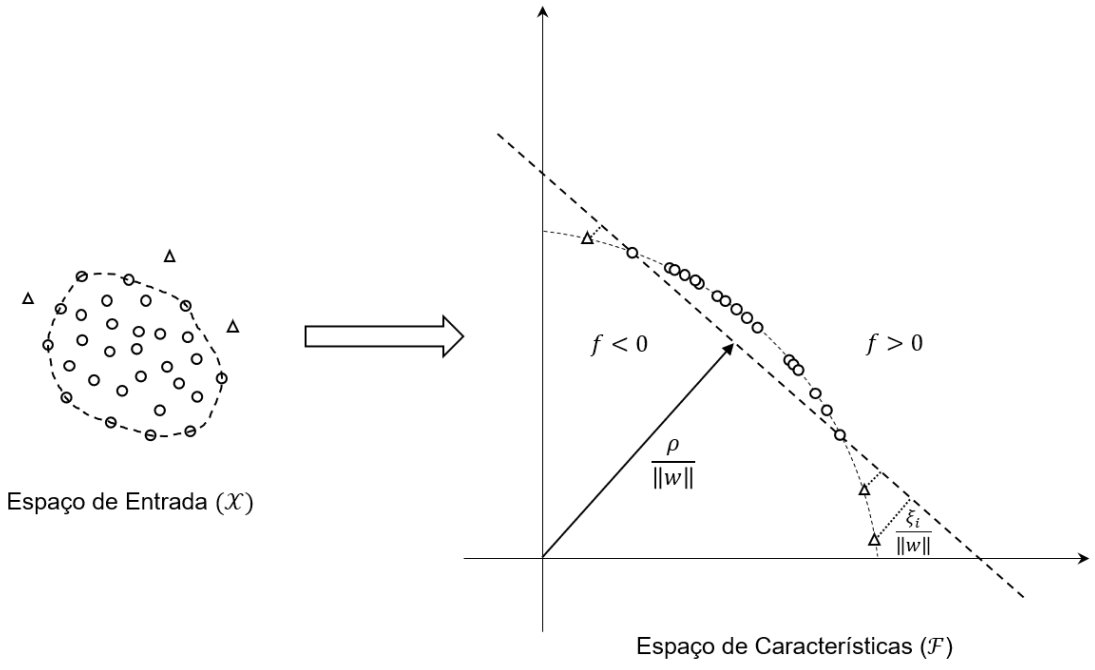


Figura 20: *Representação do One-Class SVM demonstrando os dados no espaço de entrada e hiperplano separando-os da origem no espaço de características.*

5 RESULTADOS

5.1 CONJUNTO DE DADOS DE TREINAMENTO

O conjunto de dados utilizado para o treinamento dos modelos é um banco de imagens aberto e disponível em https://appen.com/datasets/open-images-annotated-with-bounding-boxes que contém 9 milhões de imagens rotuladas para serem utilizadas em problemas de classificação, detecção e segmentação de objetos.

Dentre as imagens disponíveis no conjunto de dados foram selecionadas manualmente 1100 imagens de 4 classes - bicicleta, ônibus, carro e semáforo - sendo 1000 imagens para treino e validação, na proporção 80/20, e 100 imagens para teste.

As imagens selecionadas foram utilizadas sem qualquer alteração de dimensão, cor ou de qualquer propriedade aparente, tais como, brilho, contraste e saturação.

5.2 EXPERIMENTOS

Três modelos de classificação foram elaborados para realização desta tarefa, sendo um multiclasse, representado por uma rede neural convolucional clássica, e dois classe única, representados pelo OC-SVM e kernel PCA (kPCA).

A estratégia é capacitar os modelos para reconhecer o objeto requerido analisando as 9 imagens do reCAPTCHA individualmente, portanto o treinamento será feito sobre um conjunto de imagens, e não sobre um conjunto de desafios, na expectativa que ele seja capaz de sinalizar quais imagens devem ser selecionadas.

A implementação dos algoritmos foi feita em Python, utilizando os pacotes TensorFlow/Keras e Scikit Learn.

5.2.1 Modelo multiclasse

Conforme mencionado, o modelo multiclasse é uma CNN construída por um conjunto de camadas convolucionais seguido por um conjunto de camadas totalmente conectadas. Esse último conjunto tem a função de reduzir a dimensão do espaço de características e de classificar a imagem dentre as categorias definidas.

As camadas convolucionais foram aproveitadas da rede convolucional MobileNet disponível no pacote TensorFlow. Essa rede foi previamente treinada a partir de imagens disponíveis na ImageNet (http://www.imagenet.org)[3] para reconhecer 1000 classes de imagens, portanto, o treinamento dessas camadas não foi necessário.

[3] ImageNet é um banco de dados de imagens disponível aos pesquisadores da área de tratamento de imagem e visão artificial.

Tabela 3: *Acurácia obtida sobre o conjunto de validação variando os hiperparâmetros D e tamanho do lote.*

	Tamanho do lote				
D	30	40	50	100	200
8	86,50	85,25	85,50	87,12	83,88
16	86,75	87,25	86,50	86,62	87,75
32	**88,00**	87,75	86,87	87,37	86,75
64	87,00	86,87	87,12	87,25	86,50
128	**88,00**	87,25	87,25	87,12	87,37
256	87,37	87,88	86,87	87,75	87,37
512	87,12	87,50	87,50	87,75	87,75

As camadas totalmente conectadas, que têm a função de classificar a imagem a partir das características extraídas, foram treinadas com o conjunto de imagens descrito na seção 5.1 usando o algoritmo de retro-propagação na forma sequencial com o tamanho do lote igual à 30. Esse conjunto de camadas é formado por uma camada de entrada, uma oculta e uma de saída com 1024, 32 e 4 neurônios, respectivamente. A ReLU foi utilizada como função de ativação da camada oculta e a SoftMax na camada de saída.

O tamanho do lote e a quantidade de neurônios D na camada oculta foram definidos por meio de um *grid search* sobre o conjunto de validação utilizando a métrica de acurácia, conforme relacionado na tabela 3. De maneira geral, os valores da acurácia estão próximos entre si e, em tese, qualquer um poderia ser adotado, no entanto, optou-se pela configuração que resultou no maior deles.

A maior acurácia obtida foi de 88,00% com tamanho de lote igual à 30 e $D \in \{32, 128\}$, sendo o valor 32 escolhido para o hiperparâmetro D, porque resulta numa quantidade menor de parâmetros a serem treinados.

O modelo aplicado ao conjunto de teste obteve uma acurácia de 84,50%. A matriz de confusão está demonstrada na tabela 4 e a projeção do conjunto de teste com classificação de cada imagem pode ser visualizada na figura 21. Note que no canto inferior esquerdo aparece um ponto azul, corretamente classificado pelo modelo como Carro, mas que pela projeção possui características da classe Ônibus. Isso ocorre devido a imprecisão inerente ao método de projeção.

5.2.2 Modelos de classe única

O treinamento dos modelos de classe única kPCA e OC-SVM foi realizado de forma semelhante entre eles. Ambos possuem 3 hiperparâmetros e foram determinados da mesma forma que os hiperparâmetros do modelo multiclasse, ou seja, verificando a acurácia sobre o conjunto de validação variando valores dos hiperparâmetros.

Tabela 4: *Matriz de confusão gerado pelo modelo obtido na avaliação do conjunto de teste formado por 100 imagens de cada classe.*

	bicicleta	ônibus	carro	semáforo
bicicleta	86	2	5	7
ônibus	0	90	5	5
carro	7	4	81	8
semáforo	7	5	7	81

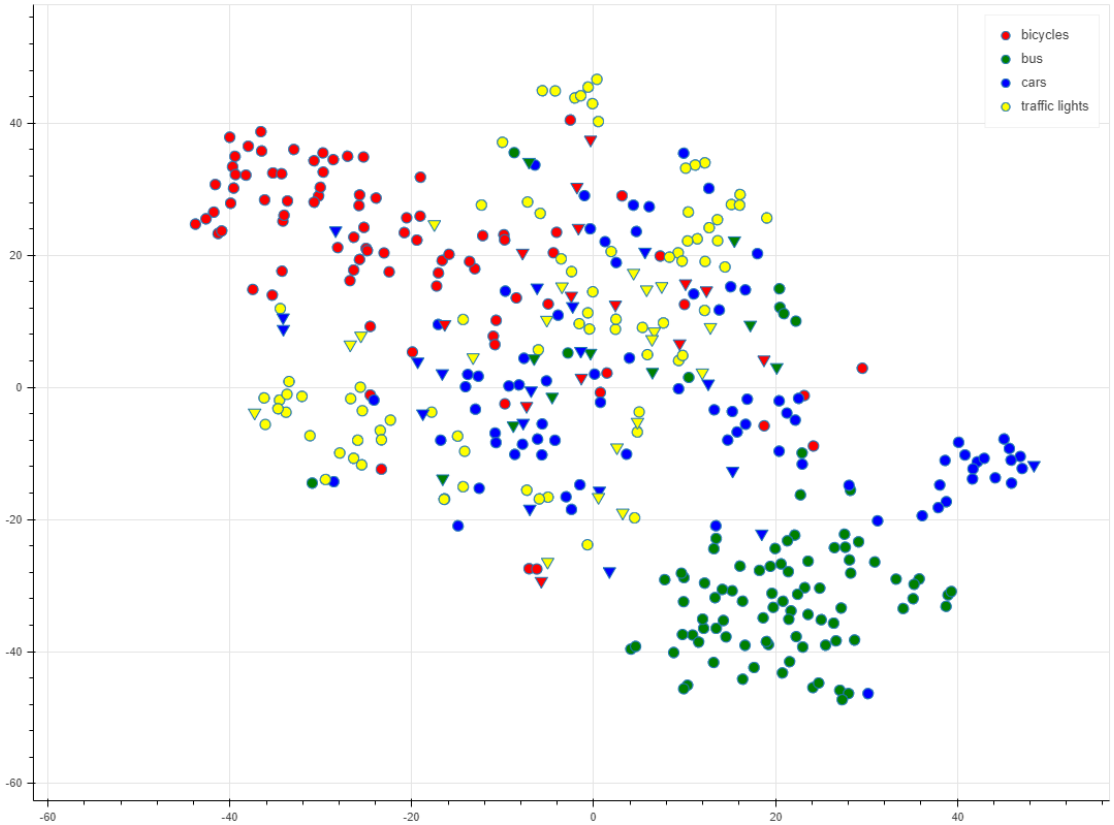


Figura 21: *Projeção em $\mathbb{R}^2$ (t-SNE) do espaço de características do conjunto de testes avaliado pelo modelo. Cada cor representa o rótulo da classe indicada na legenda. Os círculos indicam que o modelo fez a classificação corretamente, enquanto que os triângulos invertidos indicam um erro na classificação.*

Tabela 5: *Descrição dos hiperparâmetros intrínsecos aos modelos de classe única utilizados.*

Modelo	Hiperparâmetro	
kPCA	q	Quantidade de autovetores utilizados na reconstrução do espaço de entrada
OC-SVM	ν	Limite superior da fração de elementos que podem ser considerados como anormais

Tabela 6: *Acurácia obtida sobre o conjunto de validação variando os hiperparâmetros de cada modelo.*

	kPCA				OC-SVM			
Classe	D	σ	q	Acurácia	D	γ	ν	Acurácia
bicicleta	16	1,5	150	97,50	128	0,0001	0,05	97,01
ônibus	8	5	150	95,01	256	0,0001	0,05	91,54
carro	16	1,5	50	92,77	16	0,01	0,1	93,23
semáforo	16	2	1	91,54	16	0,0001	0,1	92,79

Apesar do objetivo do modelo ser classificar imagens - que tem uma representação espacial - a entrada de ambos os modelos consiste num vetor $\mathbf{x} \in \mathbb{R}^D$, com $D \in \{8, 16, 32, 64, 128, 256, 512\}$, que representa as características extraídas da imagem pela rede neural descrita na seção 4.2. O parâmetro D é o primeiro hiperparâmetro dos modelos.

O segundo hiperparâmetro está relacionado com o kernel $k(\mathbf{x}, \mathbf{y}) = e^{-\gamma||\mathbf{x}-\mathbf{y}||^2}$, conhecido como kernel RBF, utilizado nos dois modelos.

Na implementação do kPCA a constante γ é definida por

$$\gamma = \frac{1}{2\sigma}$$

onde σ representa a largura do kernel. Na prática, ele influencia na abrangência da fronteira de decisão, portanto, para este modelo o hiperparâmetro será o σ. Já para o modelo OC-SVM o segundo hiperparâmetro será diretamente o valor de γ.

Por fim, o terceiro hiperparâmetro é intrínseco a cada modelo e está descrito na tabela 5. A influência de cada um deles sobre seus respectivos modelos está descrita nas seções 4.3.2 e 4.3.3.

Conforme mencionado, os hiperparâmetros foram determinados utilizando a técnica de *grid search* sobre o conjunto de validação. Os hiperparâmetros que resultaram numa melhor acurácia, e portanto, os escolhidos para compor os modelos, estão listados na tabela 6.

Na tabela 7 está a acurácia obtida no teste dos dois modelos para cada classe utili-

Tabela 7: *Acurácia obtida sobre o conjunto de teste utilizando os hiperparâmetros obtidos para cada modelo.*

Classe	kPCA	OC-SVM
bicicleta	85,93	90,95
ônibus	82,91	87,93
carro	78,89	78,89
semáforo	85,93	83,41

Tabela 8: *Matriz de confusão gerada com a predição dos métodos kPCA e OC-SVM usando o conjunto de teste.*

kPCA

	bicicleta	outros
bicicleta	78	22
outros	6	93

	ônibus	outros
ônibus	71	29
outros	5	94

	carro	outros
carro	75	25
outros	17	82

	semáforo	outros
semáforo	88	12
outros	16	83

OC-SVM

	bicicleta	outros
bicicleta	84	16
outros	2	97

	ônibus	outros
ônibus	87	13
outros	11	88

	carro	outros
carro	72	28
outros	14	85

	semáforo	outros
semáforo	89	11
outros	22	77

zando os hiperparâmetros encontrados. Apesar dos resultados, as matrizes de confusão de cada classe de ambos os métodos, demonstradas na tabela 8, mostram uma quantidade de falsos negativos superior ao de falsos-positivos, o que pode comprometer a taxa de sucesso do modelo, uma vez que o reCAPTCHA não aceita falsos-negativos.

5.3 QUEBRA DO RECAPTCHA

O objetivo dos modelos propostos é reconhecer a presença dos objetos requeridos pelo reCAPTCHA em uma imagem, ficando livre sua utilização por alguma ferramenta de automação. Quatro cenários de testes foram elaborados para verificar a eficácia dos modelos, onde a forma de avaliação das imagens e o critério para determinar se o desafio foi solucionado podem ser diferentes em cada um deles.

O primeiro e segundo cenário foram aplicados aos três modelos e diferem entre si no critério de avaliação. Enquanto o primeiro considera que a quebra do reCAPTCHA, ou seja, a solução do desafio, foi obtida somente quando exatamente as três imagens foram selecionadas corretamente, o segundo permite a ocorrência de um falso positivo, ou seja, a seleção de quatro imagens com um único falso positivo resolverá o desafio.

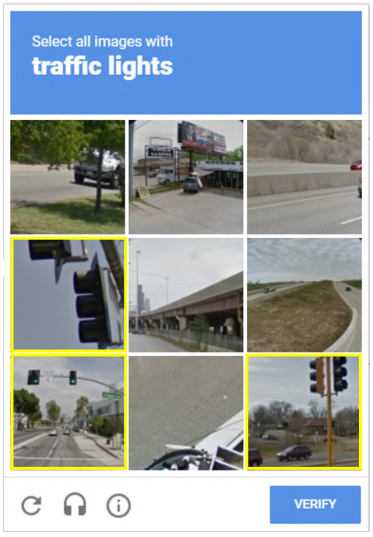
Os outros dois cenários foram realizados somente para modelos de classe única e se diferenciam na forma como indicam as imagens a serem selecionadas. Em ambos os cenários, dado um desafio, os modelos geram um vetor com as respectivas pontuações, no entanto, aqui surge a diferença entre eles. Em um, as imagens com as três pontuações mais adequadas (a depender do modelo) são selecionadas, enquanto que no outro, são quatro. O critério de avaliação do terceiro cenário é idêntico ao primeiro e o critério do quarto idêntico ao segundo. A figura 22 ilustra a diferença entre eles.

Apesar de trabalhos anteriores afirmarem que a possibilidade de escolha de um falso positivo tenha sido retirada nas versões mais atuais do reCAPTCHA, essa possibilidade de escolha foi percebida durante o desenvolvimento deste trabalho e, por isso o segundo e quarto cenário foram elaborados. Ela não está presente em todas as vezes que os desafios são apresentados e não foi possível determinar um padrão de ocorrência dessa flexibilidade para determinar quando considerá-la na avaliação do desafio.

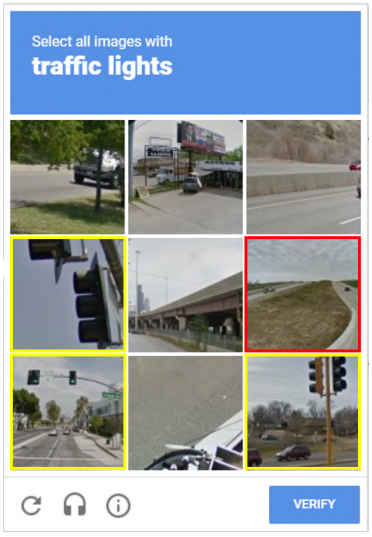
Em todos os cenários foram considerados 20 desafios para cada classe treinada, ou seja, 80 desafios, totalizando 720 imagens. Todos os desafios utilizados foram propostos pelo reCAPTCHA e armazenados localmente, devidamente separados e identificados.

As imagens do objeto requerido de cada desafio foram rotuladas com o nome de sua respectiva classe, ou seja, se o desafio requeria a seleção de imagens que continham bicicleta, então as três imagens que as continham foram rotuladas com a classe bicicleta, se requeria ônibus, foram rotuladas com ônibus, e assim, por diante. As imagens do desafio que não possuíam o objeto requerido foram deixadas sem rótulo.

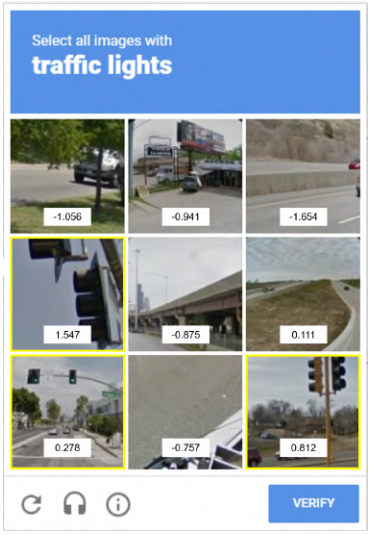
A rotulação de cada imagem foi necessária para comparar com a previsão dos modelos obtida em cada cenário e, consequentemente, contabilizar o número de desafios



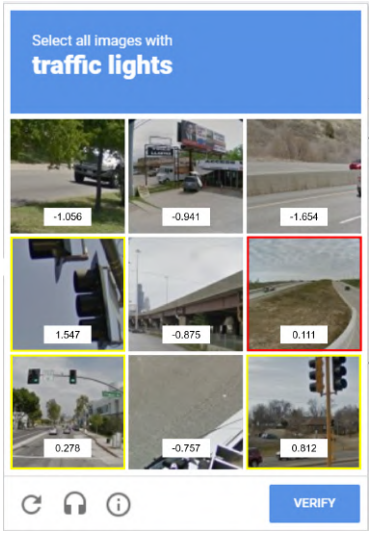
(a) Primeiro cenário: selecão das imagens feita exclusivamente pelo modelo. O desafio será resolvido se as três imagens corretas forem escolhidas.



(b) Segundo cenário: selecão das imagens feita exclusivamente pelo modelo. O desafio será resolvido se as três imagens corretas forem escolhidas, no entanto, é tolerável a escolha de um falso positivo.



(c) Terceiro cenário: selecão das imagens feita considerando as imagens com melhor pontuação (neste caso as três maiores). O desafio será resolvido se as três imagens corretas forem escolhidas.



(d) Quarto cenário: selecão das imagens feita considerando as quatro imagens com melhor pontuação (neste caso as quatro maiores). O desafio será resolvido se as três imagens corretas forem escolhidas. Semelhante ao segundo cenário, é tolerável a escolha de um falso positivo.

Figura 22: *Exemplo do critério de avaliação entre os cenários: o primeiro considera que apenas a solução indicada na figura 22(a) resolve o desafio, enquanto que no segundo, na figura 22(b), as duas soluções funcionariam, apesar de haver uma imagem selecionada indevidamente (destacada em vermelho). As figuras 22(c) e 22(d) exemplificam, respectivamente, o terceiro e quarto cenário avaliados pelo OC-SVM, onde pontuação positiva indica que a imagem pertence a classe do objeto procurado.*

resolvidos.

Nas subseções seguintes são apresentados os resultados de cada cenário. O primeiro resultado apresentado é a quantidade de desafios resolvidos em cada classe por modelo. Note que desafios resolvidos por um modelo também podem ter sido resolvidos por outro. Isso está demonstrado no diagrama logo após esses resultados. Em seguida, em alguns cenários, são exibidos exemplos de desafios com a classificação de cada imagem dada por cada modelo.

Com propósito didático, a figura 23 mostra como será representada a classificação de cada imagem. Abaixo dela estão três quadrados com as cores verde e vermelho combinadas com os sinais de + ou -. A cor indica se a imagem foi corretamente classificada (verde) ou não (vermelho). Os sinais de + ou - complementam a interpretação da classificação indicando se é uma classificação positiva ou negativa. Da esquerda para a direita, os quadrados representam o modelo CNN, OC-SVM e kPCA, respectivamente.

Ainda fazendo referência a figura 23, supondo ela que faça parte de um desafio onde é solicitado a seleção de imagens que tenham bicicletas, o modelo CNN e OC-SVM acertaram a classificação indicando a presença da bicicleta na imagem. Por outro lado, o modelo kPCA cometeu um errou ao indicar que não havia bicicleta na imagem.

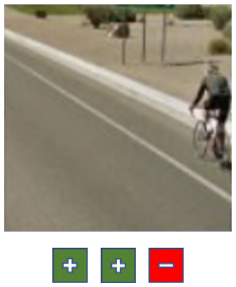


Figura 23: *Representação utilizada na classificação de cada imagem do desafio. Da esquerda para a direita, os quadrados representam o resultado da classificação do modelo CNN, OC-SVM e kPCA, nesta ordem. A cor verde indica uma classificação correta, enquanto o vermelho indica o contrário. Os sinais de + e - complementam a classificação informando se é uma classificação positiva ou negativa.*

5.3.1 Primeiro Cenário

Neste cenário tanto o modelo multiclasse quanto os de classe única foram avaliados considerando que a quebra do reCAPTCHA foi obtida somente quando as três imagens foram selecionadas corretamente. Os resultados para os três modelos estão na tabela 9. Do total de desafios apresentados, a CNN foi o modelo que obteve o melhor resultado geral, com 29% dos desafios resolvidos. Esse resultado foi alcançado pelo bom desempenho no objeto Semáforo, com 55% dos desafios resolvidos. Para os demais objetos, a CNN teve resultado similar ou pior que os demais. Outro resultado de

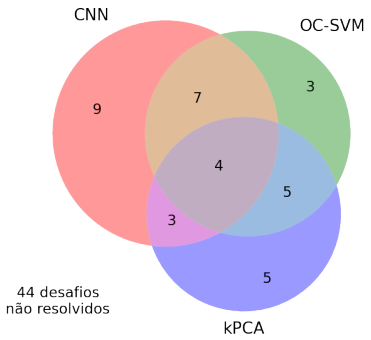


Figura 24: *Quantidade de desafios resolvidos por cada modelo e o total de não resolvidos no cenário 1. As cores vermelho, verde e azul representam os modelos CNN, OC-SVM e kPCA, respectivamente.*

destaque é a pouca ou nenhuma capacidade dos modelos de resolverem os desafios referentes à carros. Das 180 imagens que formam os desafios que requerem a seleção de carros, 133 possuem ruído, ou seja, praticamente todos os desafios possuem pelo menos uma imagem com algum grau de distorção, o que pode justificar esse resultado tão ruim.

Tabela 9: *Quantidade de desafios resolvidos por cada modelo no primeiro cenário de avaliação. A CNN foi o modelo que obteve o melhor resultado geral, apesar de ter tido um resultado similar ou pior que os outros modelos para três objetos requeridos.*

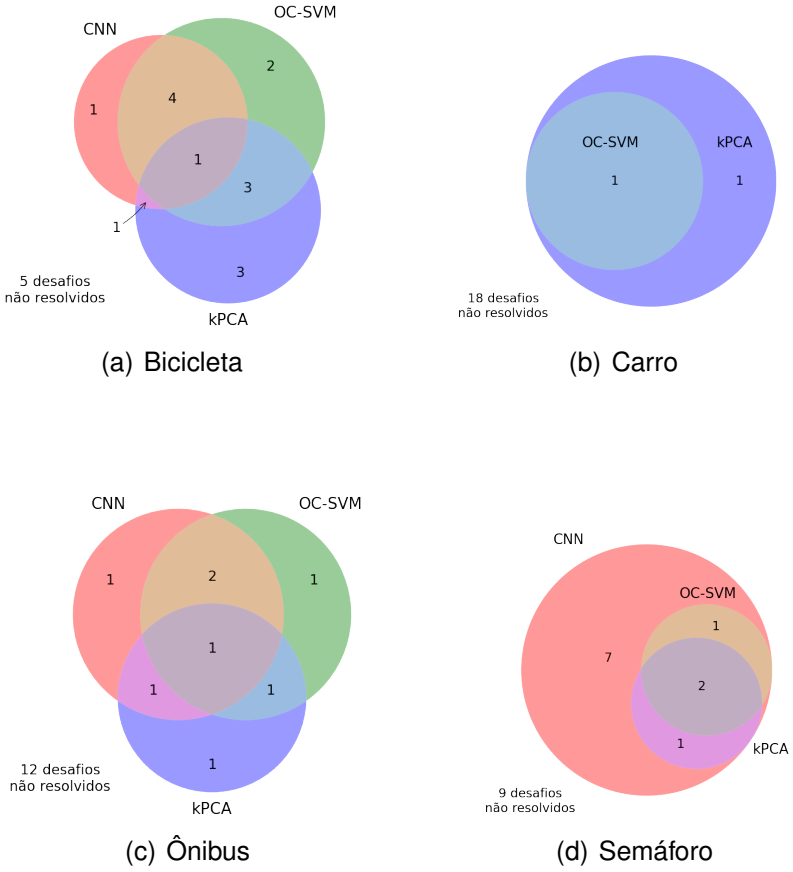
	CNN		OC-SVM		kPCA	
	Resolvidos	%	Resolvidos	%	Resolvidos	%
Bicicleta	7	35	10	50	8	40
Carro	0	0	1	5	2	10
Ônibus	5	25	5	25	4	20
Semáforo	11	55	3	15	3	15
Total	23	29	19	24	17	21

Do total de 80 desafios propostos, 36 foram resolvidos por algum modelo e 44 não, conforme ilustrado na figura 24. Isso quer dizer que juntos, os modelos resolveram 45% dos desafios.

A figura 25 apresenta a quantidade de reCAPTCHA resolvidos separados por objeto requerido, onde é possível verificar que os modelos não são equivalentes, ou seja, o conjunto de desafios resolvidos por um modelo é diferente do resolvido por outro. Além disso, não existe modelo que generalize os outros, com exceção da CNN nos desafios que solicitaram a escolha de semáforos (figura 25(d)).

5.3.2 Segundo Cenário

Semelhante ao primeiro cenário, todos os modelos foram avaliados, mas agora considerando que a quebra do reCAPTCHA acontece quando exatamente as três ima-



(a) Bicicleta (b) Carro

(c) Ônibus (d) Semáforo

Figura 25: *Quantidade de desafios resolvidos por cada modelo no cenário 1 separados por objeto requerido. Em 25(a), 25(b), 25(c) e 25(d) estão os desafios referentes aos objetos Bicicleta, Carro, Ônibus e Semáforo, respectivamente.*

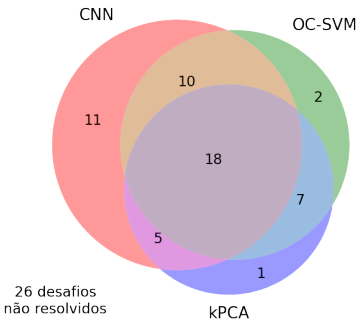


Figura 26: *Quantidade de desafios resolvidos por cada modelo e o total de não resolvidos no cenário 2. As cores vermelho, verde e azul representam os modelos CNN, OC-SVM e kPCA, respectivamente.*

gens foram selecionadas corretamente ou quando quatro imagens foram selecionadas, sendo três verdadeiros positivos e um falso positivo. Com isso, os desafios solucionados no primeiro cenário também serão solucionados neste, justificando o resultado melhor apresentado na tabela 10.

Novamente a CNN foi o modelo que teve melhor desempenho. Do total de desafios apresentados, a CNN resolveu 44 desafios, obtendo um percentual de solução de 55%. Com exceção do objeto Carro, a CNN obteve melhor resultado em todos os outros. Apesar deste cenário permitir a ocorrência de um falso positivo, novamente todos os modelos tiveram pouca capacidade de resolver os desafios referentes à carros.

Tabela 10: *Quantidade de desafios resolvidos por cada modelo no segundo cenário de avaliação.*

	CNN		OC-SVM		kPCA	
	Resolvidos	%	Resolvidos	%	Resolvidos	%
Bicicleta	16	80	16	80	14	70
Carro	1	5	2	10	3	15
Ônibus	11	55	10	50	5	25
Semáforo	16	80	9	45	9	45
Total	44	55	37	46	31	39

Do total de 80 desafios propostos, 54 foram resolvidos por algum modelo, correspondendo a 67.50% dos desafios, conforme ilustrado na figura 26. Em comparação com o primeiro, foram resolvidos 18 desafios a mais, indicando que dos 44 não resolvidos no cenário 1, 26 possuem algum falso negativo ou no mínimo dois falsos positivos na avaliação de algum modelo. Um desafio resolvido no segundo cenário, mas não resolvido no primeiro e outro não resolvido em ambos, estão mostrados na figura 27.

Também neste cenário, verifica-se pela figura 28 que os modelos não são equivalentes, no mesmo sentido do primeiro cenário. Na figura 28(b), o kPCA generaliza os outros modelos para o objeto requerido Carro, mas a quantidade de desafios solucio-

(a) (b)

Figura 27: *Exemplo de um desafio referente à Bicicleta resolvido no segundo cenário, mas não resolvido no primeiro (27(a)) e outro não resolvido em ambos referente à Ônibus (27(b)).*

nados é muito pequena.

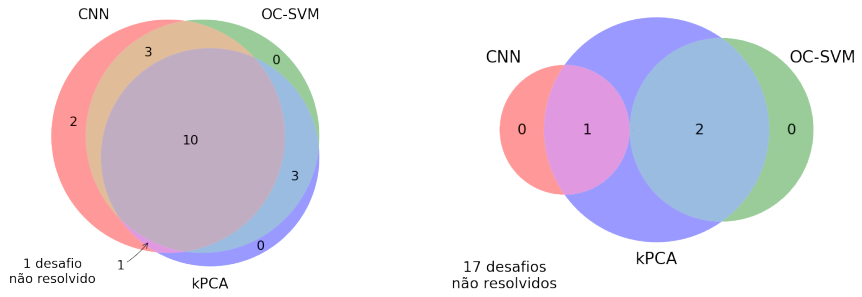
5.3.3 Terceiro Cenário

Nos cenários anteriores a avaliação das imagens do desafio foi feita de forma sequencial e independente, ou seja, uma imagem só é avaliada após a anterior e sua escolha não é influenciada por nenhuma outra. Neste terceiro cenário, os modelos avaliam todas as imagens do desafio e depois decidem quais serão selecionadas de acordo com a pontuação obtida em cada imagem.

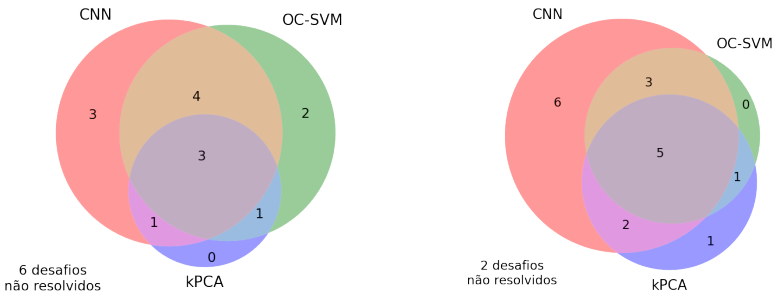
De maneira geral, os modelos de classe única produzem, ao avaliar um exemplar, uma pontuação que é utilizada para classificá-lo. No caso do OC-SVM essa pontuação é a distância com sinal do hiperplano que separa os exemplares positivos e negativos, já no kPCA a pontuação se refere ao erro de reconstrução no espaço de características.

Em particular, fixado um desafio, o resultado produzido pelos modelos de classe única após a avaliação das imagens é um vetor com a pontuação de cada imagem. A partir desse resultado foram selecionadas as três imagens com maior pontuação para o OC-SVM e as três imagens com menor pontuação para o kPCA. Por se tratar de um modelo de múltiplas classes, o CNN não será avaliado nesse cenário, portanto todas as análises e comparações realizadas neste cenário não considerará este modelo.

O critério aplicado para determinar se o desafio foi resolvido é idêntico ao do primeiro cenário, isto é, o desafio será resolvido se selecionar somente as três imagens do



(a) Bicicleta (b) Carro



(c) Ônibus (d) Semáforo

Figura 28: *Quantidade de desafios resolvidos por cada modelo no cenário 2 separados por objeto requerido. Em 28(a), 28(b), 28(c) e 28(d) estão os desafios referentes ao objeto Bicicleta, Carro, Ônibus e Semáforo, respectivamente.*

objeto requerido.

Pode ser visto na tabela 11 que os dois modelos de classe única resolveram mais desafios que no primeiro cenário, com um aumento maior para o kPCA. Importante lembrar que a escolha das imagens no primeiro cenário é livre, ou seja, ela é feita de acordo com a indicação do modelo, podendo selecionar menos ou mais de três imagens.

De volta aos resultados, o OC-SVM resolveu 35 desafios, 16 a mais em relação ao primeiro. Já o kPCA resolveu 37, resultando num aumento de 20. Juntos, os modelos resolveram 46 desafios, correspondendo a 57.5% do total (figura 29).

Comparando com o resultado do segundo cenário, figura 30, pode-se concluir pelo critério de avaliação dos dois cenários que 4 desafios apresentam um falso positivo com uma pontuação entre as três menores (ou maiores, no caso do OC-SVM), pois foi resolvido no segundo cenário, mas não no terceiro. É o caso do desafio referente ao objeto Semáforo mostrado na figura 31.

Por raciocínio análogo, 7 desafios apresentaram pelo menos dois falsos positivos, pois não foram resolvidos no segundo cenário. E mais, suas pontuações não estão entre as

Tabela 11: *Quantidade de desafios resolvidos por cada modelo no terceiro cenário de avaliação.*

	OC-SVM		kPCA	
	Resolvidos	%	Resolvidos	%
Bicicleta	15	80	13	65
Carro	2	10	2	10
Ônibus	10	50	10	50
Semáforo	8	40	12	60
Total	35	44	37	46

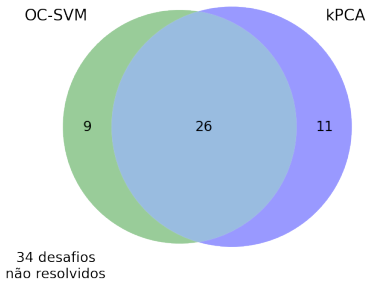


Figura 29: *Quantidade de desafios resolvidos por cada modelo e o total de não resolvidos no cenário 3. As cores verde e azul representam os modelos OC-SVM e kPCA, respectivamente.*

três melhores, pois foram resolvidos neste cenário. A figura 32 demonstra um desafio do objeto Semáforo que exemplifica esta situação.

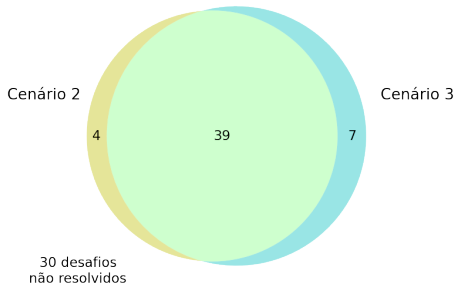


Figura 30: *Desafios resolvidos pelos modelos OC-SVM e kPCA no segundo e terceiros cenários. Em amarelo estão os desafios resolvidos no cenário 2 e em azul claro os desafios resolvidos no cenário 3.*

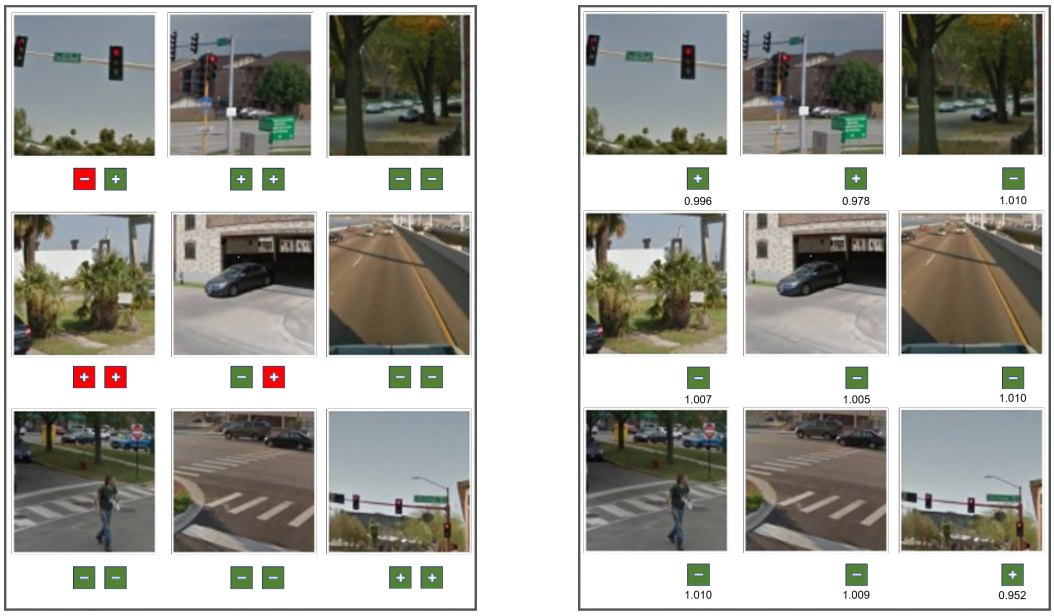
Semelhante ao apresentado nos outros cenários, a figura 33 mostra a quantidade de desafios resolvidos em cada classe pelos modelos, com destaque para Bicicleta que foi o desafio mais solucionado por ambos. Novamente os desafios de Carro apresentaram pior resultado.

5.3.4 Quarto Cenário

O quarto e último cenário é análogo ao segundo em relação ao critério de avaliação do desafio e análogo ao terceiro quanto a maneira de escolha de imagem, isto é, um

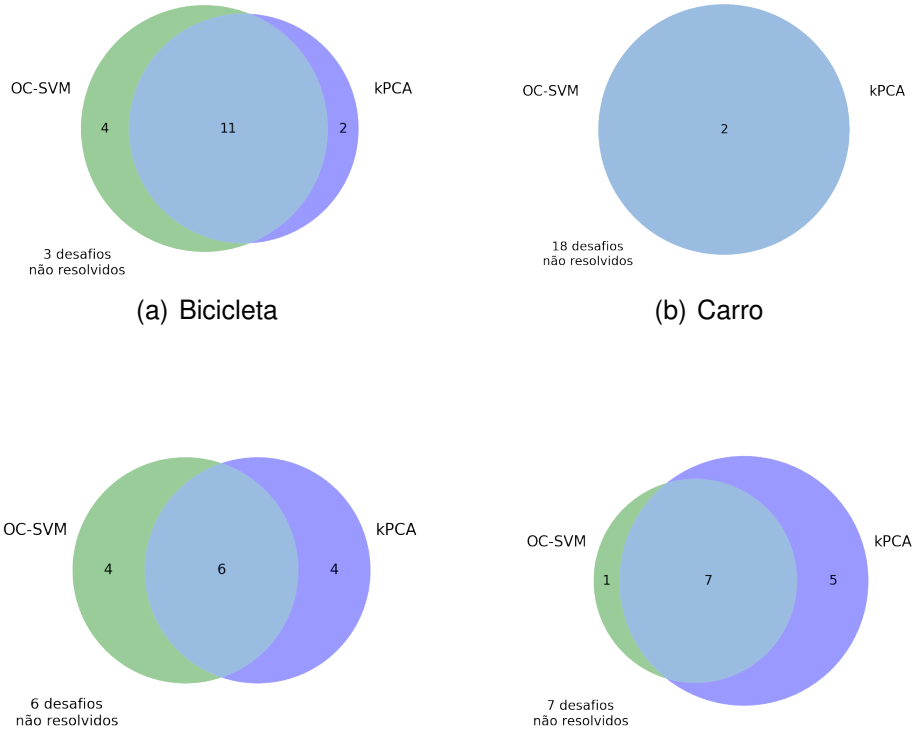
(a) (b)

Figura 31: *Exemplo de um desafio que foi resolvido no segundo cenário, mas não foi resolvido no terceiro. A figura 31(a) mostra a avaliação do OC-SVM com o critério do segundo cenário. Já a figura 31(b) mostra o mesmo desafio no terceiro cenário com a pontuação de cada imagem, quando avaliado pelos modelos OC-SVM, que seleciona as três maiores, e kPCA, que seleciona as três menores.*



(a) (b)

Figura 32: *Exemplo de um desafio que foi resolvido no terceiro cenário, mas não foi resolvido no segundo. A figura 32(a) mostra o resultado da avaliação no segundo cenário. O OC-SVM deixa de selecionar uma imagem (falso-negativo) e o kPCA seleciona duas (falso positivo) que não tem o objeto Semáforo. A figura 32(b) mostra o mesmo desafio, juntamente com a pontuação de cada imagem, avaliado pelo modelo kPCA no terceiro cenário, onde foi solucionado.*



(a) Bicicleta (b) Carro

(c) Ônibus (d) Semáforo

Figura 33: *Quantidade de desafios resolvidos por cada modelo no cenário 3 separados por objeto requerido. Em 33(a), 33(b), 33(c) e 33(d) estão os desafios referentes ao objeto Bicicleta, Carro, Ônibus e Semáforo, respectivamente.*

desafio será considerado resolvido com a escolha das três imagens corretas mais um falso positivo e a seleção das imagens será feita por meio da pontuação obtida na avaliação pelo modelo. Na prática, são selecionadas as quatro imagens com maior pontuação para o OC-SVM e as quatro imagens com menor pontuação para o kPCA.

Conforme demonstrado na tabela 12, foram resolvidos 52 desafios pelo OC-SVM e 57 pelo kPCA. Ambos os modelos tiveram um aumento significativo na quantidade de desafios resolvidos em relação terceiro cenário. Foram 17 desafios a mais no OC-SVM e 20 no kPCA. Esse aumento era esperado uma vez que os desafios resolvidos no cenário anterior também seriam resolvidos aqui. Destaca-se que apesar da melhora no resultado do objeto Carro, ele ainda continua sendo o desafio menos resolvido.

A figura 34 mostra que 63 desafios (78.75 %) foram resolvidos considerando o resultados dos dois modelos e, de forma geral, eles não são equivalentes.

Observando o resultado separadamente por objeto requerido na figura 35, com exceção do Carro, todos os outros foram quase que totalmente resolvidos, com destaque para a generalização do OC-SVM sobre o kPCA quando resolvendo desafios de bicicletas.

Tabela 12: *Quantidade de desafios resolvidos por cada modelo no quarto cenário de avaliação.*

	OC-SVM		kPCA	
	Resolvidos	%	Resolvidos	%
Bicicleta	19	95	18	90
Carro	6	30	9	45
Ônibus	13	65	15	45
Semáforo	14	70	15	45
Total	52	65	57	71

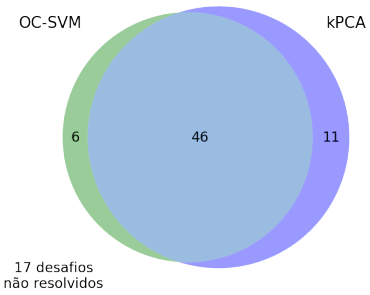
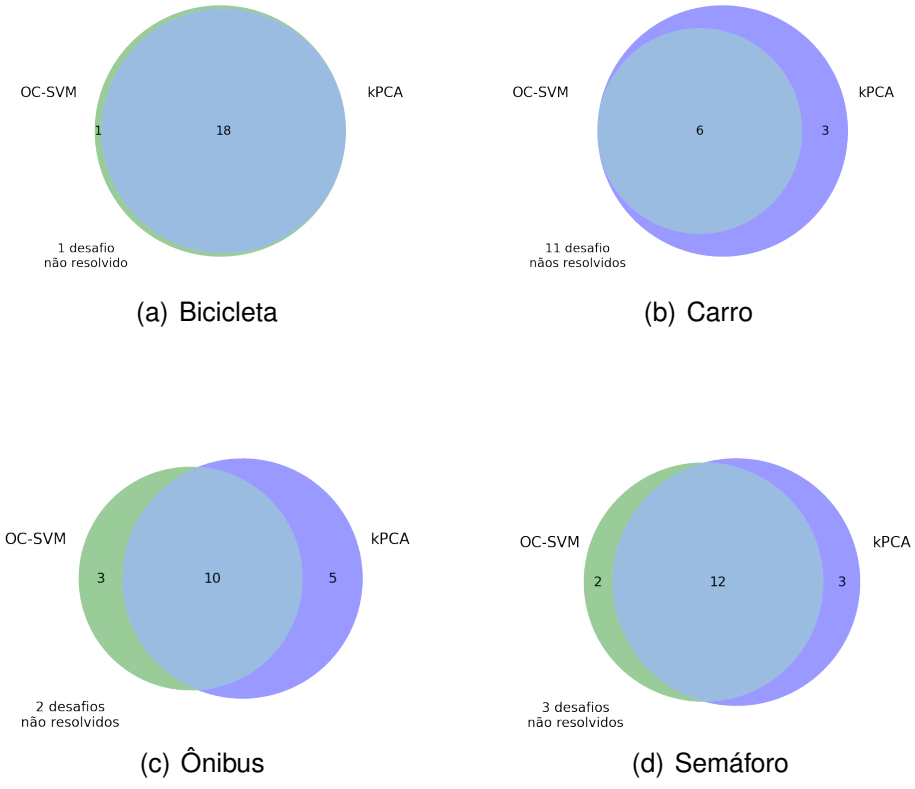


Figura 34: *Quantidade de desafios resolvidos por cada modelo e o total de não resolvidos no cenário 4. As cores verde e azul representam os modelos OC-SVM e kPCA, respectivamente.*



(a) Bicicleta (b) Carro

(c) Ônibus (d) Semáforo

Figura 35: *Quantidade de desafios resolvidos por cada modelo no cenário 4 separados por objeto requerido. Em 35(a), 35(b), 35(c) e 35(d) estão os desafios referentes ao objeto Bicicleta, Carro, Ônibus e Semáforo, respectivamente.*

Os dois modelos juntos resolveram no segundo e terceiro cenário 50 desafios, enquanto que neste foram resolvidos 13 a mais, conforme figura 36. Dos 17 desafios não resolvidos pelos dois modelos, claramente todos os desafios apresentaram mais de dois falsos positivos.

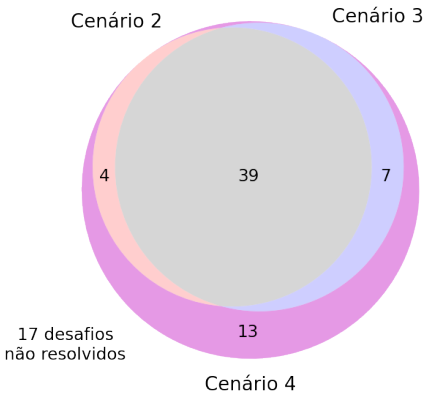


Figura 36: *Desafios resolvidos pelos modelos OC-SVM e kPCA no segundo, terceiro e quarto cenários.*

6 CONCLUSÃO

Todo o estudo realizado sobre aprendizado de máquina e o esforço descrito neste trabalho, desde a etapa de coleta e tratamento das imagens até o treinamento e obtenção dos modelos, foi para dotar à máquina a resolver o desafio proposto pelo reCAPTCHA.

Foram estudados os conceitos teóricos das redes neurais convolucionais, que tem obtido grande sucesso em várias aplicações que necessitam reconhecer objetos, e modelos de classificação de classe única, que são utilizados na detecção de novidades.

Três modelos foram desenvolvidos e avaliados para atingir o objetivo do trabalho: o primeiro utilizando um classificador multiclasse, que neste caso nada mais é que uma rede neural formada por um conjunto de camadas totalmente conectadas, os outros dois modelos utilizam um classificador de classe única OC-SVM e kPCA. Todos precedidos por uma etapa de extração de características feita pela parte convolucional da rede neural MobileNet.

A avaliação de cada modelo foi realizada em quatro cenários que se distinguem na escolha de quais imagens selecionar e no critério para determinar se o desafio foi resolvido. Observa-se na tabela 13 que nos cenários 1 e 2 os modelos de classe única (OC-SVM e kPCA) apresentaram desempenho inferior ao modelo multiclasse (CNN). Já nos cenários 3 e 4, onde o modelo CNN não foi avaliado, o OC-SVM e kPCA tiveram desempenho semelhante, com ligeira vantagem para o kPCA.

Tabela 13: *Quantidade de desafios resolvidos por cada modelo em cada cenário de avaliação.*

	CNN		OC-SVM		kPCA	
Cenário	Resolvidos	%	Resolvidos	%	Resolvidos	%
1	23	29	19	24	17	21
2	44	55	37	46	31	39
3	-	-	35	44	37	46
4	-	-	52	65	57	71

O modelo com melhor desempenho foi o kPCA com 71% dos desafios resolvidos no quarto cenário, onde, novamente, são selecionadas as quatro imagens melhor classificadas e é permitido um falso-positivo.

Na hipótese do reCAPTCHA não admitir a seleção de um falso-positivo, o cenário 3 demonstra que quase 50% dos desafios foram resolvidos por ambos os modelos de classe única, com uma pequena vantagem para o kPCA.

O percentual de desafios resolvidos foi bastante prejudicado pela presença de ruído na maioria das imagens da classe Carro uma vez que as imagens do conjunto de

treinamento e teste não têm ruído, ou seja, os classificadores não foram treinados para reconhecer uma imagem ruidosa.

Importante destacar que a acurácia obtida por cada modelo sobre o conjunto de teste na etapa de treinamento não é uma cota inferior para o percentual de desafios resolvidos, uma vez que a “métrica” para determinar a quantidade de desafios resolvidos não é equivalente à acurácia, então não há nenhuma incoerência entre o resultado obtido no treinamento com o resultado obtido na solução dos desafios.

Apesar de trabalhos relacionados terem resolvido uma quantidade maior de desafios, é importante destacar que este trabalho utilizou apenas a visão computacional para resolvê-los sem explorar características do reCAPTCHA, por exemplo, simular o comportamento humano durante a interação com a aplicação que usa o reCAPTCHA para evitá-lo. Como o uso exclusivo da visão computacional, ferramentas de automação podem utilizar os modelos criados aqui sem se preocuparem com barreiras técnicas adicionais impostas pelo reCAPTCHA. Além disso, o conjunto de treinamento utilizado foi formado apenas por imagens não usadas pelo desafio, dando a possibilidade de se fazer uso dos modelos para resolver desafios baseados em imagens que não sejam da Google.

Tomando como ponto de partida este trabalho, fica a oportunidade de experimentar redes neurais mais recentes e com melhor desempenho que a MobileNet para a extração de características. A utilização de um banco de imagens com mais classes e que melhor se aproxime das imagens utilizadas pelo reCAPTCHA, tais como as imagens com ruído, pode ser uma estratégia interessante para aumentar a eficiência dos modelos.

REFERÊNCIAS

[Alp20] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020. 8, 11, 21, 24, 38

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 57

[CLRS01] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction to algorithms second edition. *The Knuth-Morris-Pratt Algorithm*, 2001. 19

[DV18] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2018. 44

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016. 8, 20, 25, 29, 37, 39, 40, 42, 43, 44

[goo] recaptcha v3 | google developers. 15

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 15

[Hay07] Simon Haykin. *Redes neurais: princípios e prática*. Bookman Editora, 2007. 33, 37, 38

[Hof07] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern recognition*, 40(3):863–874, 2007. 57

[HW68] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968. 40

[HZC+17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 50

[JDM00] Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000. 50

[KNH17] Martin Kopp, Matej Nikl, and Martin Holena. Breaking captchas with convolutional neural networks. In *ITAT*, pages 93–99, 2017. 17

[Kov02] Zsolt László Kovács. *Redes neurais artificiais*. Editora Livraria da Fisica, 2002. 28

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 8, 43

[M+97] Tom M Mitchell et al. *Machine learning*. McGraw-hill New York, 1997. 19

[MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 52

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 27

[MP69] Marvin Minsky and Seymour Papert. Perceptron: An introduction to computational geometry. *Cambridge tiass., HIT*, 1969. 28

[Neg05] Michael Negnevitsky. *Artificial intelligence: a guide to intelligent systems*. Pearson education, 2005. 29

[PCCT14] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014. 23

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 33

[Ros57] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957. 28, 37

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 21

[SPK16] Suphannee Sivakorn, Jason Polakis, and Angelos D Keromytis. I'm not a human: Breaking the google recaptcha. *Black Hat*, pages 1–12, 2016. 17

[SS17] A de L SOUSA and Marcos Filipe Alves Salame. Uma abordagem comparativa de algoritmos de aprendizado supervisionado para classificação dos cultivares da planta paullinia cupana. In *Embrapa Amazônia Ocidental-Artigo em anais de congresso (ALICE)*. In: ENCONTRO REGIONAL DE COMPUTAÇÃO E SISTEMAS DE INFORMAÇÃO, 6., 2017. 8, 37

[VABHL03] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *International conference on the theory and applications of cryptographic techniques*, pages 294–311. Springer, 2003. 14

[Yan20] Haolin Yang. Captcha recognition using convolutional neural networks with low structural complexity. In *Journal of Physics: Conference Series*, volume 1693, page 012040. IOP Publishing, 2020. 17

[ZC88] Yi-Tong Zhou and Rama Chellappa. Computation of optical flow using a neural network. In *ICNN*, pages 71–78, 1988. 44

[ZYWB18] Yuan Zhou, Zesun Yang, Chenxu Wang, and Matthew Boutell. Breaking google recaptcha v2. *Journal of Computing Sciences in Colleges*, 34(1):126–136, 2018. 17