UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Detecção e rastreamento de componentes de vagões ferroviários utilizando Redes Neurais Convolucionais e restrições geométricas

DM: 17/2020

Camilo Lélis Assis Gonçalves

BELÉM - PARÁ

2020

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Camilo Lélis Assis Gonçalves

Detecção e rastreamento de componentes de vagões ferroviários utilizando Redes Neurais Convolucionais e restrições geométricas

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Mestre em Engenharia Elétrica, do Instituto de Tecnologia, do Programa de Pós-Graduação em Engenharia Elétrica.

BELÉM - PARÁ

2020

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"DETECÇÃO E RASTREAMENTO DE COMPONENTES DE VAGÕES FERROVIÁRIOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS E RESTRIÇÕES GEOMÉTRICAS"

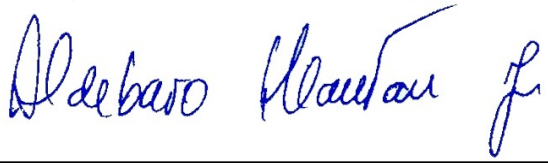
AUTOR: **CAMILO LÉLIS ASSIS GONÇALVES**

DISSERTAÇÃO DE MESTRADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA NA ÁREA DE TELECOMUNICAÇÕES.

APROVADA EM: 27/04/2020

BANCA EXAMINADORA:

Prof. Dr. Fabrício José Brito Barros
(Orientador – PPGEE/UFPA)



Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior
(Avaliador Interno – PPGEE/UFPA)

Prof. Dr. Ronaldo de Freitas Zampolo
(Avaliador Externo ao Programa – FCT/UFPA)

VISTO:

Prof.ª Dr.ª Maria Emília de Lima Tostes
(Coordenadora do PPGEE/ITEC/UFPA)

Aos meus pais, irmãos e companheiros.

Agradecimentos

Agradeço, acima de tudo, à minha família, pelo carinho, compreensão, paciência e sabedoria. Por terem dado a mim a força e a vontade para trilhar o caminho que segui até hoje e me tornar, amanhã mais do que hoje, um homem melhor.

Agradeço ao meu pai Antônio e minha mãe Helena, pela honestidade, razão, determinação e amor que foram pilares de minha criação e são guias para todas as minhas ações. Obrigado pelas noites em claro, pelos abraços apertados, pelos sorrisos esbanjados e pelas lágrimas derramadas. Meus companheiros e amigos não só de todas as horas, mas também por toda a minha vida. Essa vitória é também de vocês e para vocês.

Agradeço aos meus irmãos Yuri e Luan, com os quais troquei o papel de irmão mais velho por tantas vezes. Desde nossas brigas até nossos momentos mais fraternais, amo cada segundo com os quais vivo com vocês. Tenho orgulho de ver em vocês todos os ensinamentos de nossos pais e luto para que possam ver isso também em mim.

Agradeço aos meus tios Delma, José Maria (em memória), Celso, Nazaré, Marizete (em memória), Lizete, Renato e dentre tantos outros. Da maneira de cada um, não chegaria onde cheguei sem vocês.

Agradeço à minha companheira Rafaela, por me impulsionar sempre em direção aos meus sonhos e compartilhar suas forças comigo em momentos difíceis. Obrigado por fazer parte da minha vida.

Agradeço aos amigos de curso da turma 2007: Brenno, Cássio, Claudio, Dael, Edicleiton, Elison, Elton, Felipe Freire, Fernanda, Flávio, Francis, Bruno Guedes, Bruno Haick, Jairo, Janize, João, Joarley, Lauro, Leonardo, Maylson, Natália, Rafael, Rômeo, Werson e tantos outros que conheci durante minha longa jornada pela Graduação.

Agradeço especialmente aos meus amigos Brenno, Cláudio, Leonardo, Natália e Rafael por perdurarem por tanto tempo na difícil tarefa de serem meus amigos. Nem sempre consegui demonstrar isso, mas levo cada um na memória.

Agradeço também os amigos que conheci durante a busca pelo meu título de Mestre. Em particular, agradeço à meus amigos, Charles Ferreira, Leidiane Castro, Paulo Alexandre e Raphael Navegantes. Nossos momentos de avacalhação entre os momentos tensos fizeram essa jornada menos cansativa e mais proveitosa.

Agradeço também à todos os colaboradores do Laboratório de Processamento de Sinais, que se tornou minha segunda casa por muito tempo. Espero que tenhamos muito mais oportunidades de trabalharmos juntos no futuro.

Agradeço ao Alan Maia, meu primeiro chefe e grande amigo, seja entre risos e avacalhações ou entre cabos de rede e servidores.

Agradeço à todos do Hospital Betina Ferro de Souza e LPRAD. Estes foram mais do que lugares pelos quais passei. São lugares os quais deixei uma parte de mim e levo uma parte deles comigo.

Agradeço à todos do Instituto SENAI de Inovação em Tecnologias Minerais por todo o apoio que recebi ao longo dos dois anos de mestrado. Agradeço à Ana Claudia S. Gomes, Bruno Victor M. Ferreira, Eduardo C. de Carvalho e Rafael L. Rocha, integrantes da equipe de pesquise de Visão Computacional do ISI, pelo companheirismo e empenho durante todo o decorrer do que desenvolvemos juntos.

Agradeço especiamente ao Adriano Reis Lucheta, Diretor do Instituto SENAI de Inovação em Tecnologias Minerais, e à Thaís Haber, Assessora de Inovação do Instituto SENAI de Inovação em Tecnologias Minerais. O apoio que me foi oferecido foi imprescindível para a conclusão deste trabalho.

Um obrigado em especial ao professor Evaldo Pelaes, pelo apoio imediato ao meu ingresso no mestrado, ao professor Ronaldo Zampolo, meu orientador durante a Graduação e guia de longa data; e ao professor Fabrício Brito, por me orientar durante o mestrado e me ajudar de imensamente, apesar do pouco tempo em que nos conhecemos. Obrigado pelo tempo, paciência e sabedoria empregados em mim por tanto tempo. Espero de coração corresponder aos seus esforços e expectativas.

“Reivindicações extraordinárias
requerem evidências extraordinárias”
Carl Sagan.

Resumo

A inspeção de componentes de trem que podem causar descarrilamento possui um papel importante na manutenção ferroviária. A fim de aumentar a produtividade e segurança, empresas prestadoras de serviços procuram por soluções de inspeção automáticas e confiáveis. Apesar da inspeção automática baseada em visão computacional ser um conceito consolidado, tais aplicações desafiam a comunidade de desenvolvimento em razão de fatores ambientais e logísticos a serem considerados. Este trabalho propõe um técnica de detecção e estimativa das posições das regiões de dreno presentes em vagões de trem. Nosso detector/rastreador consiste em uma rede neural convolucional e um conjunto de restrições geométricas, que levam em conta a trajetória ideal dos componentes de interesse dos vagões e as distâncias entre eles. Detalhamos os procedimentos de treinamento e validação, juntamente com as métricas utilizadas para aferir a performance do sistema proposto. Os resultados apresentados são comparados com outras duas técnicas, e exibe um bom custo benefício entre confiança e complexidade computacional para a detecção dos componentes de interesse.

Palavras-chave: visão computacional, detecção de objetos, rastreamento de objetos, aprendizagem profunda, aprendizado de máquina, inspeção automática.

Abstract

The inspection of train components that can cause derailment plays a key role in rail maintenance. To improve productivity and safety, service providers look for automatic and reliable inspection solutions. Although automatic inspection based on computer vision is a standard concept, such an application challenges development community due to the environmental and logistic factors to be considered. In this work, we propose a technique to detect and estimate the position of drain regions present in train wagons. Our object detector/tracker consists of convolutional neural network and a set of geometric constraints, which takes in account the ideal trajectory path of the wagon's components of interest and the distances between them. We detail training and validation procedures, together with the metrics used to assess the performance of the system. Presented results compare two other techniques with our approach, which exhibits a fair trade-off between reliability and computational complexity for the application of wagon component detection.

Keywords: computer vision, object detection, object tracking, deep learning, machine learning, automatic inspection.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ANN *Artificial Neural Network.*

CNN *Convolutional Neural Network.*

FC *Fully Conected Layer.*

FN Falso-negativo(a).

FP Falso-positivo(a).

GPU *Graphics Processing Unit*

MLP *Multilayer Perceptron.*

ReLU *Rectified Linear Unit.*

SSD *Single-Shot Multibox Detector.*

VGG *Visual Geometry Group, o acadêmico que criou a aquitetura de redes neural aritificial VGG.*

VN *Verdadeiro-negativo(a).*

VP *Verdadeiro-positivo(a).*

Sumário

Capítulo 1

Introdução

1.1 Motivação

Nos últimos anos, o desenvolvimento das Redes Neurais Convolucionais e o rápido desenvolvimento de técnicas de *Deep Learning* trouxe novos ares para a área da Visão Computacional, atraindo um nível de atenção sem precedentes. Técnicas de Visão Computacional vem sido utilizadas em aplicações do mundo real, com veículos autônomos [1, 2], vídeo vigilância [3], reconhecimento de atividades [4, 5], etc.

Um exemplo onde o uso de técnicas de Visão Computacional é proeminente é o de controle e inspeção industrial. Entre as várias industrias onde o uso de inspeção visual é necessária, muitas dessas aplicações são consideradas como atividades de alta prioridade e consequência em razão dos custos potencialmente altos que erros no processo de inspeção podem gerar (ex: acidentes, fatalidades, perda de equipamentos de alto custo, perda de consumidores, etc.)

No caso da indústria ferroviária, a inspeção de componentes de trem e de ferrovias que podem causar o descarrilhamento de vagões é uma questão de manutenção crucial. Apesar da inspeção automática por Visão Computacional ser uma aplicação já estabelecida na área, a tarefa, neste caso, não é trivial em razão de vários aspectos conflitantes inerentes ao problema. As instalações de inspeção ferroviária, por exemplo, são sujeitas à vibração, poeira e à condições climáticas adversas; enquanto o *pipeline* de inspeção requer estágios confiáveis de estimativa, detecção e classificação de componentes.

1.2 Visão geral do trabalho

Nesta dissertação serão apresentados os conceitos e técnicas de Visão Computacional que foram utilizados no desenvolvimento do projeto "Análise Automática de Componentes de Vagões por Visão Computacional", uma parceria entre o Instituto Tecnológico VALE (ITV), o Instituto SENAI de Inovação em Tecnologias Minerais (ISI-TM) e a Universidade Federal do Pará (UFPA).

Mais especificamente, o objetivo desta pesquisa foi o de executar a detecção e rastreamento das regiões de *dreno* presentes nos vagões de trem ferroviário exibidos em vídeo utilizando técnicas de *visão computacional*. A realização desta tarefa serve de base para a inspeção da *superestrutura*, o componente do vagão o qual armazena os sedimentos de minério a serem transportados. Além disso, devido à disposição dos componentes no vagão, as informação de localização dos *drenos* podem ser utilizadas para auxiliar na detecção e inspeção de outros componentes.

Para o desenvolvimento deste trabalho[1], uma câmera de alta resolução foi instalada no local onde a inspeção é atualmente realizada por funcionários da Estrada de Ferro Carajás (EFC) de modo visual. Um detector de objetos baseado em uma Rede Neural Convolucional foi utilizado para estimar as localizações dos *drenos* nas imagens obtidas pela câmera. Os elementos detectados são rastreados individualmente. Restrições geométricas são responsáveis pela eliminação de possíveis erros de detecção e correção de erros de rastreamento.

São apresentados em detalhes os procedimentos experimentais de avaliação de desempenho da técnica proposta, incluindo uma comparação do desempenho da técnica com os de duas outras metodologias de detecção e rastreamento; sendo uma uma versão simplificada da técnica proposta, porém sem a utilização de restrições geométricas; e a outra utilizando rastreadores do tipo *Kernelized Correlation Filters*; ambas bastante utilizadas em aplicações reais.

Ao final citamos algumas melhorias a serem incorporadas na metodologia atual, e possíveis direcionamentos em trabalhos futuros.

[1] O código desenvolvido durante o desenvolvimento deste trabalho encontra-se em https://github.com/TheCamilovisk/WagonDetection.

1.3 Organização do trabalho

O restante deste trabalho é organizado como se segue.

O Capítulo 2 aborda as tecnologias relacionadas a este estudo. Mais especificamente, são apresentados os conceitos principais de redes neurais artificiais e aprendizagem profunda, além das arquiteturas de redes neurais convolucionais utilizadas pela metologia proposta.

O Capítulo 3 apresenta a metodologia de detecção e rastreamento das regiões de *dreno* proposta, abordando o procedimento de treinamento do detector de objetos utilizado e as restrições geométricas utilizadas para auxiliar o rastreamento e eliminação erros.

O Capítulo 4 detalha os procedimentos experimentais aplicados para a avaliação de desempenho da metodologia de detecção e rastreamento proposta e apresenta os resultados da avaliação.

O Capítulo 5 sumariza os pontos principais deste trabalho, identificando as principais contribuições e citando o direcionamento de possíveis melhorias para a técnica apresentada aqui e trabalhos futuros.

Capítulo 2

Tecnologias relacionadas

2.1 Introdução

Neste capítulo, são apresentados um breve histórico e as principais motivações para o uso de *redes neurais artificiais* no contexto deste trabalho, com ênfase nas *redes neurais convolucionais*, dentre as quais destacamos duas arquiteturas: a *MobileNet* e a *Single-Shot Multibox Detector*.

2.2 Redes neurais artificiais

Uma Rede Neural Artificial é um tipo de sistema de computação inspirado nas redes neurais biológicas que constituem os cérebros de mamíferos. Em sua forma mais geral, uma rede neural é uma máquina projetada para *modelar* a maneira como o cérebro biológico executa uma tarefa particular ou função de interesse.

O neurofisiologista Warren McCulloch e o lógico Walter Pitts, ambos estadunidenses, foram pesquisadores pioneiros na área ao propor o primeiro modelo matemático para redes neurais [6]. Quase quinze anos depois, o estadunidense Frank Rosenblatt, inspirado pela teoria Hebbiana [7] da neuroplasticidade desenvolveu o perceptron [8], considerada a primeira rede neural moderna.

A publicação de Minsky e Papert [9], em 1969, expôs as limitações do modelo de Rosenblatt, provando que tais redes são incapazes de resolver uma ampla classe de problemas. O impacto desta publicação foi devastador, praticamente desaparecendo o interesse em redes neurais artificiais nos anos seguintes.

Em 1970, S. Linnainmaa formulou a metodologia geral de *diferenciação automática* de redes discretas conectadas de *funções aninhadas diferenciáveis* [10]. Em 1974, P. Werbos mencionou a aplicação destes princípios em redes neurais artificiais [10] e, em 1982, ele aplicou esta metodologia para a análise de *funções não-lineares* [11]. O termo *backpropagation* e seu uso geral em redes neurais artificiais foi estabelecido em 1986 por D. Rumelhart, G. Hinton, e R. Williams [12].

Em 2006, R. Salakhutdinov, A. Mnih, e G. Hinton [13] propuseram que com o aumento do número das camadas e dos neurônios por camada, certos tipos de redes neurais poderiam superar o desempenho dos classificadores no estado-da-arte da época.

Em 2012, Ng e Dean criaram uma rede neural que aprendeu a reconhecer conceitos de alto-nível (como gatos, carros, cadeiras, humanos, etc.) utilizando imagens não-rotuladas [14]. Pré-treinamento não supervisionado, aumento do poder computacional das GPUs[1] e popularização da computação distribuída permitiram o uso de redes com grande número de camadas (redes profundas), principalmente em tarefas visuais de alto-nível, levando ao desenvolvimento da área hoje conhecida como *aprendizado profundo* (do inglês *deep learning*).

2.2.1 O neurônio artificial

Uma rede neural é baseada em uma coleção de unidades conectadas chamados *neurônios artificiais*, ou simplesmente *neurônios*, os quais são uma representação matemática simplificada de sua contraparte biológica. Um neurônio atua recebendo um sinal, o processa e envia o sinal processado para outros neurônios aos quais esteja conectado.

Um neurônio possui três elementos básicos (Figura 2.1):

1. Um conjunto de *conexões* ou *sinapses*, cada uma caracterizada por um *peso*. Um sinal x_j na entrada de uma sinapse j conectado a um neurônio k é multiplicado pelo peso sináptico w_{kj}. Diferente do peso de uma sinapse cerebral, o peso sináptico de um neurônio artificial pode estar em uma faixa de valores que incluem tanto valores negativos quanto positivos.

2. Um *somador* para somar os sinais de entrada, ponderados pelos respectivos pesos sinápticos do neurônio.

[1] Do inglês *Graphics Processing Units*, ou Unidades de Processamento Gráfico

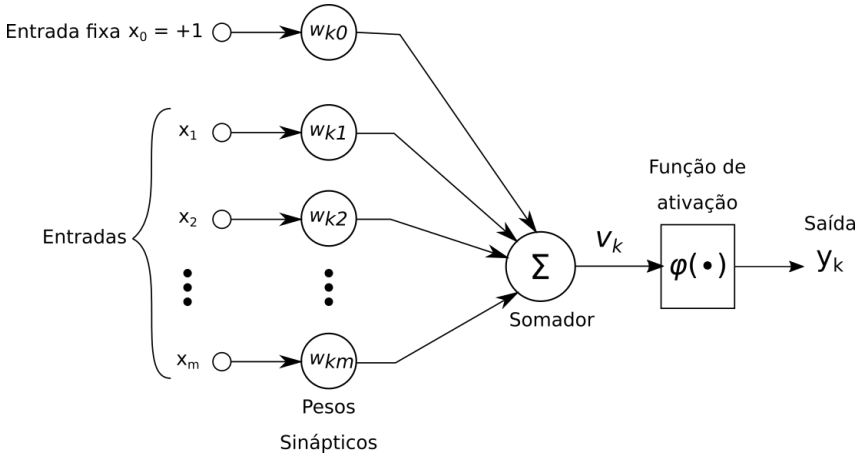


Figura 2.1: Modelo de um neurônio não-linear, rotulado k. Fonte: adaptado de [15]

3. Uma *função de ativação* que limita a amplitude da saída de um neurônio. Pode ser entendida como uma abstração representando a taxa de ativação do *potencial de ação* de uma célula.

   Apesar da possibilidade de utilização de funções lineares, o emprego de funções não-lineares como função de ativação é mais comum por permitir a utilização do algoritmo de *retro-propagação* como método de aprendizagem da rede. Falaremos mais desse método na Seção 2.2.3.

   A Figura 2.2 mostra algumas das funções de ativação mais utilizadas.

   Em termos matemáticos, podemos descrever o neurônio k na Figura 2.1 pelas equações:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2.1}$$

e

$$y_k = \varphi(v_k) \tag{2.2}$$

onde $x_1, x_2, ..., x_m$ são os sinais de entrada; $w_{k0}, w_{k2}, ..., w_{km}$ são os respectivos pesos sinápticos de neurônio k; v_k é o *campo local induzido*, ou *potencial de ativação*, do neurônio k; $\varphi(\cdot)$ é a função de ativação; e y_k é o sinal de saída do neurônio. Podemos também notar a existência do peso sináptico w_0 que, por possuir como entrada fixa o valor $x_0 = +1$, não depende de quaisquer dos sinais de entrada do neurônio. Este termo é denominado de *viés*, ou *bias*, cuja função é, dependendo do valor do seu sinal, aumentar ou diminuir o valor de entrada da função de ativação do neurônio.

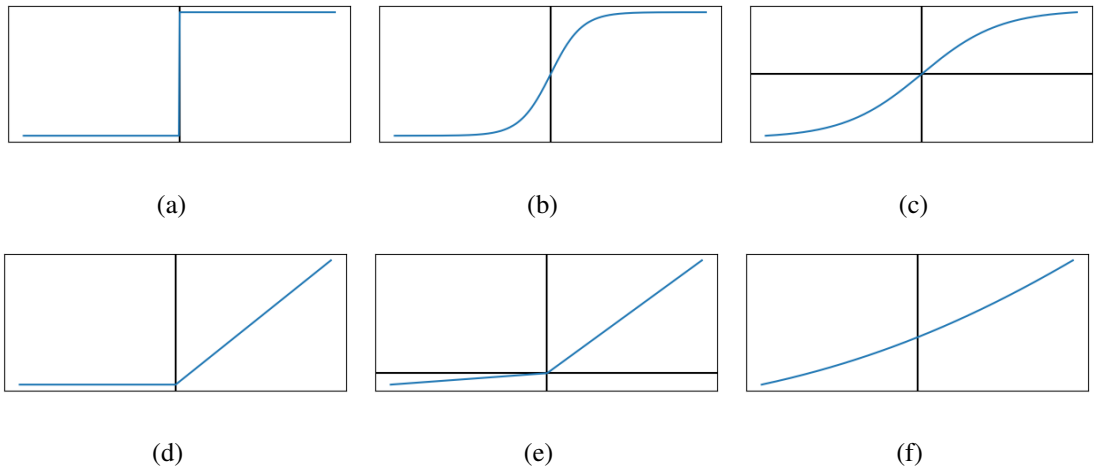


Figura 2.2: Algumas das funções de ativação mais utilizadas: (a) limiar, também conhecida como função de Heaviside; (b) sigmóide; (c) tangente hiperbólica; (d) linear retificada, ou ReLU (do inglês *rectified linear unit*); (e) leaky ReLU; e (f) softplus. Fonte: o autor.

2.2.2 O perceptron multi-camada.

Tipicamente, uma rede neural consiste em um conjunto de unidades sensoriais (nós-fonte) que constituem a *camada de entrada*, uma ou mais *camadas ocultas*, ou *escondidas*, de nós de computação, e uma *camada de saída*. O sinal de entrada é propagado em direção à saída da rede, camada a camada. Esse tipo de rede neural é comumente chamada de *perceptron multi-camada*[2], ou *MLP* (do inglês *multilayer perceptron*). A Figura 2.3 exibe uma representação de uma MLP com duas camada ocultas.

Um perceptron multi-camada possui três características distintivas:

1. O modelo de cada neurônio na rede inclui uma função de ativação não-linear que seja *diferenciável*.

2. A rede contém uma ou mais camada *escondidas* além dos nós de entrada e de saída.

3. A rede exibe um alto grau de conectividade, determinado pelas conexões sinápticas da rede.

[2]Apesar do nome, perceptrons multi-camada podem empregar neurônios com qualquer tipo de função de ativação não-linear, ao contrário do modelo original do perceptron de Rosenblatt [8], o qual utiliza a função limiar como função de ativação. A utilização do termo para referir a redes genéricas de alimentação direta é considerado um erro histórico.

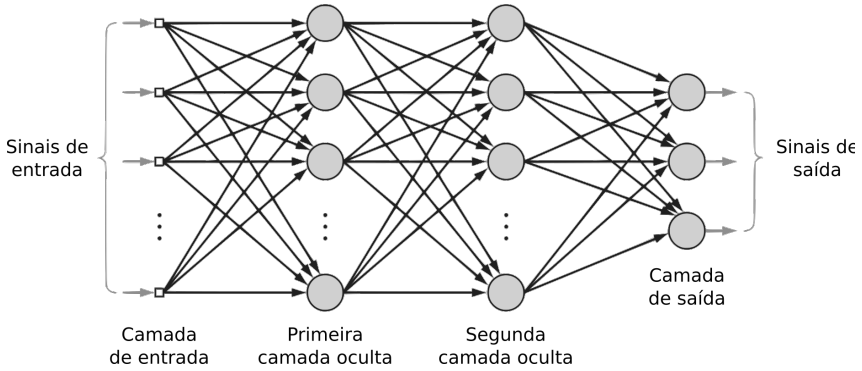


Figura 2.3: Modelo de um perceptron multi-camada com duas camadas ocultas. Fonte: adaptado de [15]

Perceptrons multi-camada vêm sendo aplicados com sucesso para resolver diversos problemas difíceis ao treiná-las de maneira supervisionada utilizando o popular *algoritmo de retro-propagação do erro* (do inglês *back-propagation algorithm*). Esse algoritmo é baseado na *regra de aprendizagem por correção de erro*, e na seção seguinte será dada uma visão dele.

2.2.3 O algoritmo de retro-propagação do erro.

Chamamos de *aprendizagem* o processo de ajustar os parâmetros de uma rede de modo que sua saída se aproxime cada vez mais de seu resultado esperado. Isto ocorre durante o processo de treinamento, no qual um grande número de vetores de entrada é submetido à rede. Esses exemplos pertencem a um conjunto de vetores de entrada para os quais os sinais de saída da rede são conhecidos, geralmente obtidos através da medição dos parâmetros de um problema real cujo comportamento deseja-se reproduzir artificialmente. Tal conjunto de valores de entrada é comumente chamado de *conjunto de treinamento*, e suas respectivas saídas desejadas da rede são conhecidas pelo termo em inglês *ground truth*.

Basicamente, a aprendizagem por retro-propagação do erro consiste em duas fases. Durante a fase de *propagação*, um vetor de entrada é aplicado aos nós da camada de entrada da rede, e seu efeito é propagado de camada em camada, com todos os pesos sinápticos *fixos*. Ao final, um conjunto de saídas é produzido como resposta atual da rede e o *erro de predição* é computado levando em consideração a resposta desejada. Durante a fase de *retro-propagação*, todos os pesos sinápticos são ajustados de acordo com uma regra de correção de erro.

Antes de iniciarmos o treinamento, é preciso definir uma *função de custo*, responsável por informar o quão longe está o resultado encontrado pela rede do resultado esperado. Con-

siderando $\boldsymbol{x}_T = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_N]$ como o conjunto de N vetores de treinamento, $\boldsymbol{d}_T = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, ..., \mathbf{d}_N]$ como sendo o conjunto de vetores de saídas desejadas para cada vetor de treinamento, $\boldsymbol{y}_T = [\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, ..., \mathbf{y}_N]$ como sendo o conjunto de vetores de saídas da rede para cada vetor de treinamento, e $L(\mathbf{y}_n, \mathbf{d}_n, \mathbf{w})$ como a função custo entre a saída da rede e a saída desejada para a entrada $\mathbf{x}_n$ dado o conjunto de pesos sinápticos $\mathbf{w}$. O processo de aprendizagem é definido como o problema de minimização

$$\min_{\mathbf{w}} E[L(\boldsymbol{y}_T, \boldsymbol{d}_T, \mathbf{w})] \approx \min_{\mathbf{w}} \frac{1}{|N|} \sum_{n \in N} L(\mathbf{y}_n, \mathbf{d}_n, \mathbf{w}) \tag{2.3}$$

O treinamento consiste em alimentar a rede com os vetores $\boldsymbol{x}_T$ do conjunto de treinamento, atualizando os pesos sinápticos da rede para cada vetor de entrada. Completa-se uma *época* quando todos os vetores de treinamento são fornecidos de entrada para a rede. O treinamento termina quando um certo número de *épocas* é alcançado ou quando o erro médio de predição da rede estiver abaixo de um valor de tolerância.

O algoritmo de retro-propagação aplica um fator de correção Δw_{ji} ao peso sináptico w_{ji} segundo a *regra delta*:

$$\Delta w_{ji} = -\eta \frac{\partial L(\mathbf{y}_n, \mathbf{d}_n, \mathbf{w})}{\partial w_{ji}} \tag{2.4}$$

onde η é a *taxa de aprendizado* do algoritmo. O uso do sinal negativo na equação 2.4 é devido ao método do *gradiente descendente* (do inglês *gradient descent*) [16] no espaço de pesos sinápticos.

Portanto, o valor atualizado $\overline{w_{i,j}}$ do peso sináptico é computado por

$$\overline{w_{ji}} = w_{ji} + \Delta w_{ji} \tag{2.5}$$

A atualização dos parâmetros ocorre a cada novo exemplo de treinamento inserido na rede. Porém, ao lidarmos com bases de dados que possuem milhares de exemplos, como a maioria das utilizadas atualmente, este processo se torna longo e desnecessário. Uma alternativa é realizar o cálculo do gradiente em pequenos *lotes* (*mini-batchs* em inglês) de dados. Dessa forma. o treinamento ocorre mais rapidamente.

2.2.4 Redes neurais convolucionais

Uma *rede neural convolucional* [17, 18], ou *CNN*[3], consiste em uma rede neural na qual a principal operação realizada por suas camadas é a convolução. Foi projetada especificamente

[3]Do inglês *Convolutional Neural Network*.

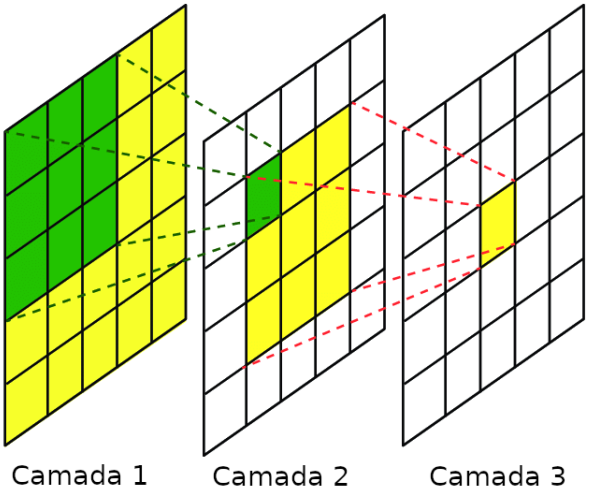


Figura 2.4: Campo receptivo de cada camada convolucional com kernel 3×3. A área verde denota o campo receptivo de um pixel na camada 2, enquanto a área amarela denota o campo receptivo de um pixel na camada 3. Fonte: adaptado de [23].

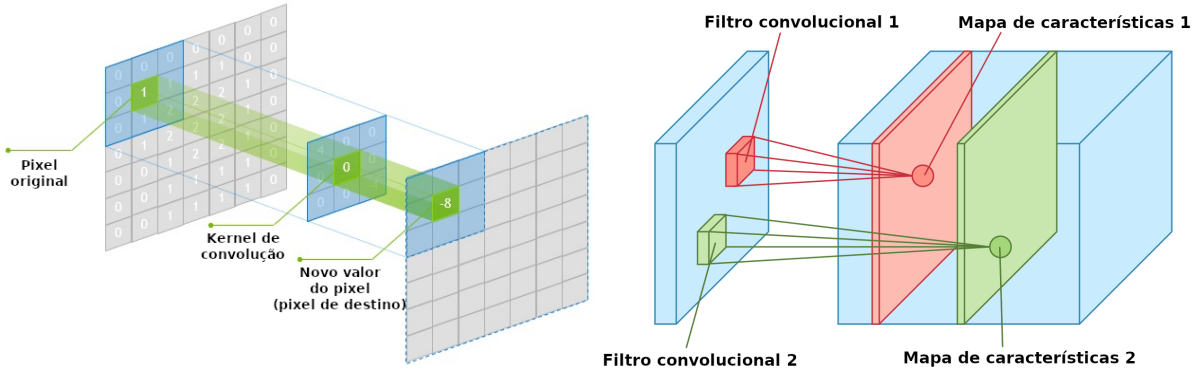
para tarefas de *processamento de imagens* e *visão computacional*[4].

Sua arquitetura é inspirada no córtex visual, no qual cada neurônio é responsável por apenas uma parte do campo de visão; sendo que, ao final, estas partes se complementam e todo o campo é tratado. Da mesma forma, a rede convolucional utiliza filtros para analisar pequenos grupos de dados vizinhos, extraindo suas características mais relevantes. Apenas estas características são passadas adiante para as próximas camadas. Por isso, é dito que as rede convolucionais atuam como extratores de características, ou *feature extractors*.

Como também visto na Seção 2.2.4.1, os filtros de uma camada convolucional podem identificar padrões específicos na imagem, como, por exemplo, bordas ou cores. Portanto, em uma rede convolucional com múltiplas camadas existirão diversos filtros, cada um especializado na extração de um padrão específico. Já a utilização sequencial de camadas convolucionais permite a identificação de padrões mais complexos ao combinar padrões identificados anteriormente, pois os filtros de camadas seguintes serão aplicados a todos os mapas da camada anterior. Dessa forma, o *campo de receptivo* de um neurônio aumenta de acordo com sua profundidade na rede, como ilustrado na Figura 2.4.

Além da camada convolucional, existem diversos tipos de camadas que podem ser utiliza-

[4]Apesar da utilização de CNNs ser predominante em problemas de *visão computacional*, esse tipo de rede vêm sido aplicado com sucesso em experimentos envolvendo processamento de sinais de voz [19, 20] e linguagem natural [21, 22].



(a) (b)

Figura 2.5: (a) Operação de um filtro convolucional sobre um tensor bi-dimensional e (b) a operação de uma camada convolucional sobre um tensor multi-dimensional e os mapas de características resultantes. Fonte: adaptado de [24]

dos em uma rede neural. Nesta seção, serão abordadas aqueles mais relevantes para a aplicação apresentada neste trabalho.

2.2.4.1 A camada convolucional

A camada convolucional [17] é a unidade básica principal de uma *rede neural convolucional*. Esta abordagem reduz consideravelmente a quantidade de parâmetros necessários a sua execução quando comparada com as camadas totalmente conectadas de uma MLP comum, além de fazer com que a camada seja invariante à translação.

A camada atua na extração de características dos dados de entrada, explorando as correlações locais que possam estar presentes esntre os pixels de imagens naturais. Conforme indicado pelo nome, a camada convolucional realiza sobre os dados de entrada a operação de convolução (Figura 2.5(a)). Seus neurônios são arranjados em *filtros*, também conhecidos por *kernels*, de tamanhos $K \in \mathbb{R}^{(D_K \times D_K \times C)}$, onde D_K é a dimensão espacial do *kernel* e C é o número de canais de entrada. Além disso, a camada pode dispor de um termo de viés e possui uma função de ativação definida para a camada.

A convolução pode ser definida matematicamente, para um tensor de entrada tri-dimensional $I \in \mathbb{R}^{(H_I \times W_I \times C)}$ e um filtro convolucional K como

$$Y[i,j] = (I * K)[i,j] = \sum_{m,n,c} K[m,n,c]I[i-m, j-n, c], \tag{2.6}$$

ou seja, esta operação consiste, basicamente, em deslizar o filtro sobre os dados de entrada,

calculando o produto interno a cada nova posição. A este valor pode ser adicionando um termo de viés. O resultado dessa operação, após passar por uma função de ativação não linear, é chamado de valor de ativação. O resultado gerado é um mapa de características $Y \in \mathbb{R}^{H_y \times W_y \times F}$ (Figura 2.5(b)).

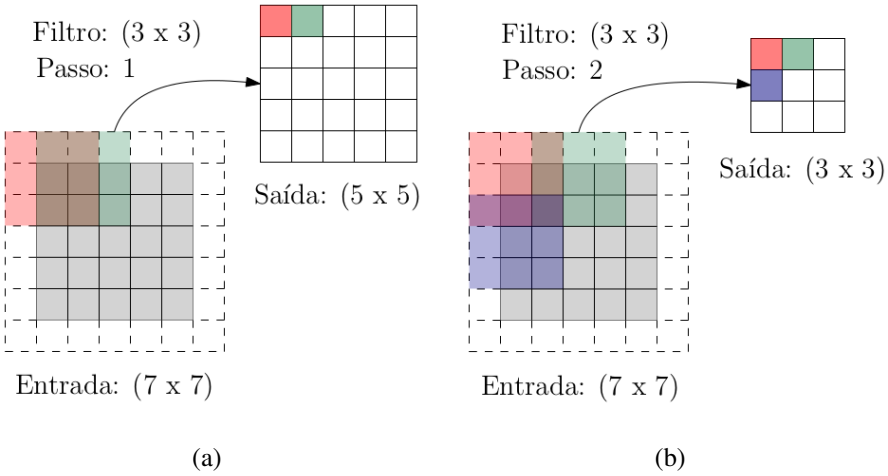


Figura 2.6: Funcionamento dos parâmetros de passo e preenchimento para um filtro de tamanho (3x3). Em (a) temos uma entrada (7x7) resultando em uma saída (5x5) dado o valor de *passo* 1, enquanto em (b) a entrada (7x7) resulta em uma saída (3x3) pois o valor do *passo* é 2. A área pontilhada corresponde ao *preenchimento* com zeros. Fonte: o autor.

O mapa de ativação gerado pela convolução manterá a informação espacial sobre padrões específicos. Ou seja, um padrão que ativa determinado mapa pode ser encontrado em diferentes posições da entrada, tornando a camada invariante a translações. Além disso, muitos de seus parâmetros são compartilhados entre os neurônios, uma vez que todos aqueles referentes a um mesmo mapa de ativação utilizam o mesmo filtro. Já o número de parâmetros total da rede dependerá apenas do tamanho e quantidade dos filtros, independente do tamanho da entrada. Isso permite que a camada atue sobre os dados de entrada grandes com um quantidade razoável de parâmetros.

Na prática, a convolução pode ser realizada de diversas formas. É possível definir o padrão de deslocamento do filtro sobre os dados, a quantidade de valores considerados em cada operação e até inserir novos valores ao conjunto a ser analisado. Tais definições são muito importantes pois determinam a formação do mapa de ativação gerado pela camada. Ou seja, são responsáveis por influenciar o tipo de informação que será gerado pela camada. Os parâmetros de *passo* (do inglês *stride*) e *preenchimento* (do inglês *padding*) são os mais frequentemente

utilizados para realizar tais definições.

O passo dita o movimento do filtro, ou seja, com um passo de tamanho unitário o filtro caminha uma amostra por vez. Quanto maior o valor do passo, menor será a dimensão da saída. Já o preenchimento consiste em adicionar valores ao redor da entrada para garantir que seu tamanho seja preservado na saída. Podemos calcular o tamanho W_y da saída a partir do tamanho da entrada w, filtro K e o valor do passo S como:

$$W_y = \left\lceil \frac{W - K}{S} \right\rceil + 1. \tag{2.7}$$

Ao adicionarmos um preenchimento P, temos na entrada o novo tamanho $\overline{W} = W + 2P$. Logo, para S = 1 utiliza-se $P = \frac{K-1}{2}$ para que a saída e a entrada tenham o mesmo tamanho. Na Figura 2.6, estão ilustrados os conceitos de passo e preenchimento para a convolução.

2.2.4.2 Camada totalmente conectada

Uma *camada totalmente conectada* (do inglês *fully conected layer*) segue o mesmo princípio de uma rede MLP tradicional, onde cada neurônio que a constitui é conectado a todos os neurônios da camada imediatamente anterior, como é exibido na Figura 2.7. Portanto, a saída de cada neurônio pode ser obtida utilizando as equações 2.1 e 2.2.

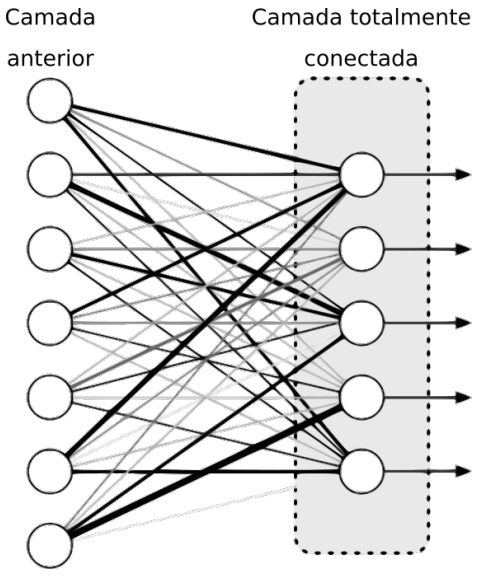


Figura 2.7: Representação das conexões sinápticas de uma camada totalmente conectada com sua camada imediatamente anterior. Fonte: adaptado de [25]

Como cada neurônio deve se relacionar com todos os dados de entrada, o número de conexões escala de acordo com as dimensões desses dados. Esse problema é conhecido como

maldição da dimensionalidade[5]. Por exemplo, se considerarmos uma camada composta por 100 neurônios, que recebe dados de entrada de tamanho 1.000, serão necessários 100.000 parâmetros para sua operação. Se o dado de entrada tiver seu tamanho alterado para 1.100, serão necessários 110.000 parâmetros. Se considerarmos como entrada, por exemplo, uma imagem, que usualmente é composta de milhares de pixels, a quantidade de parâmetros seria tamanha que sua utilização ficaria inviável. Portanto, tal camada é geralmente utilizada ao final de uma rede neural, com o intuito de mapear os valores intermediários para cada uma das saídas finais possíveis. Nos casos em que a rede é utilizada para resolver um problema de classificação, é desejado que apenas uma classe seja ativada. Para obter tal resultado, uma função é normalmente acoplada à última camada totalmente conectada da rede, a fim de interpretar cada saída dos neurônios como probabilidades de classe. Uma função geralmente utilizada para esta finalidade é a *softmax*, na qual a probabilidade para uma classe k, dado uma entrada x e um vetor de saída da rede y é obtida como

$$p(classe = i|x) = \frac{e^{y_i}}{\sum_{c=1}^{K} e^{y_c}} \tag{2.8}$$

sendo K o número total de classes. Assim, os valores finais para cada classe se encontrarão dentro do intervalo $[0, 1]$ e sua soma será igual à 1.

2.2.4.3 A camada de agrupamento

Uma limitação dos mapas de características resultantes das camadas convolucionais é a de que eles são sensíveis quanto a localização das características no tensor de entrada. Isso significa que pequenas mudanças nas posições das características da imagem irão resultar em um mapa de atributos diferente. Isto pode acontecer quando aplicamos alguma transformação na imagem de entrada como rotação, deslocamento, recorte e outros.

A camada de agrupamento [27], ou *pooling* em inglês, aplica independentemente à cada mapa de caraterísticas uma operação de *downsampling*, que reduz as dimensões dos dados de entrada da camada. Essa operação resulta na redução da redundância de informação para dados espacialmente próximos; pois para uma área pequena, pixels adjacentes costumam possuir valores muito parecidos. Além disso, a rede torna-se menos suscetível a pequenas alterações locais e distorções; uma vez que o resultado dos agrupamento de um certo conjunto de dados será

[5]O termo en inglês *curse of dimensionality* foi introduzido pela primeira vez por Bellman [26], e indica que o número de amostras necessárias para estimar uma função arbitrária com um determinado nível de precisão cresce exponencialmente em relação ao número de variáveis de entrada (isto é, dimensionalidade) da função.

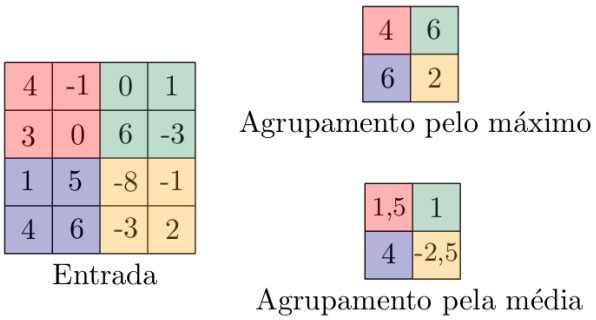


Figura 2.8: Funcionamento de uma camada de agrupamento pelo máximo e pela média para um filtro de tamanho (2×2) e passo 2. Fonte: o autor

sempre o mesmo, independentemente de como estão organizados. Assim, uma modificação na posição dos valores causada, por exemplo, ao se descomprimir uma imagem, não causará varições muito grandes na saída da camada.

A camada atua como um filtro $P \in \mathbb{R}^{H_p, W_p}$ sobre cada canal do tensor de entrada. A cada passo, um dos valores da região selecionada pela janela do filtro será escolhido, de acordo com um método de seleção pré-estabelecido, que irá representar a região no mapa de caraterísticas resultante. Sendo assim, a saída da camada de agrupamento terá o mesmo número de canais da entrada; porém, cada canal terá seus tamanhos reduzidos.

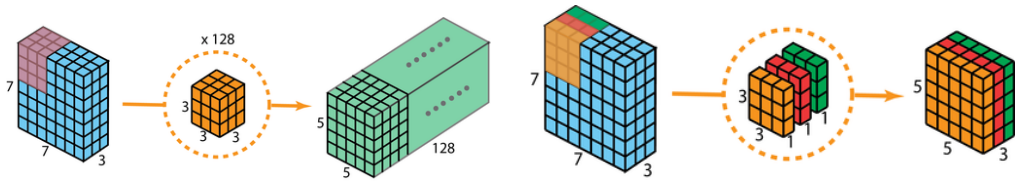
As formas mais comuns de realizar o agrupamento é aplicando a função de máximo (*max-pooling*) ou de média (*average-pooling*) sobre os dados presentes em cada janela. Um caso muito comum é a utilização de um filtro e passo de tamanho 2, como ilustra a Figura 2.8.

2.3 Arquiteturas de redes neurais convolucionais utilizadas

Nesta seção, iremos abordar resumidamente as arquiteturas de *redes neurais convolucionais* utilizadas neste trabalho. Tais arquiteturas são: *MobileNet* [28] versão 1, e *Single-Shot Multibox Detector* [29], ou *SSD*.

2.3.1 MobileNet versão 1

Classificação de imagens é a tarefa de *visão computacional* de classificar uma imagem de acordo com o seu conteúdo visual. Por exemplo, um algoritmo de classificação de imagens pode determinar se uma imagem contém uma pessoa ou não. Enquanto esta tarefa é trivial



(a) (b)

Figura 2.9: Diferença entre (a) o *filtro convolucional padrão*, onde *cada filtro* convolucional age independentemente em *todos* do tensor de entrada, gerando um único mapa de atributos para cada filtro; e (b) o *filtro de convolução separável em profundidade*, onde *cada kernel* pertencente à um filtro convolucional age independentemente em *cada* canal do tensor de entrada, gerando um mapa de atributos para cada *kernel* do filtro. Fonte: adaptado de [30]

para um ser humano, a classificação robusta de imagens ainda é um tópico desafiador de visão computacional.

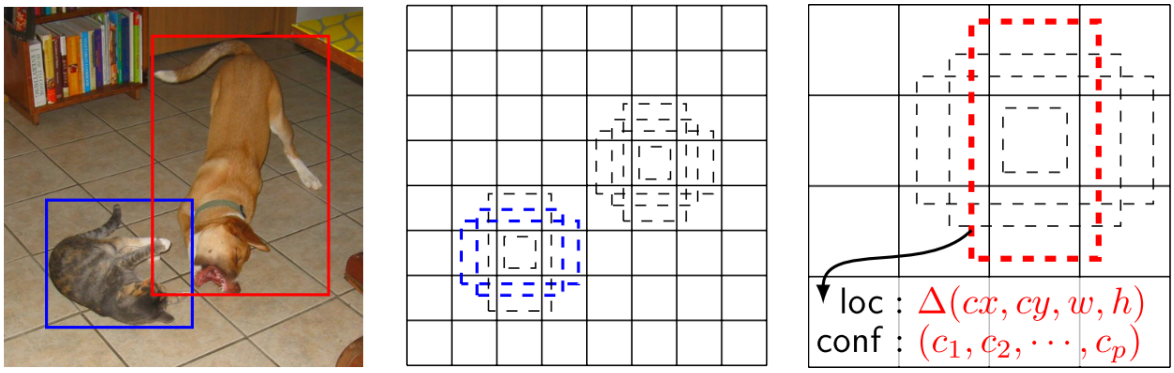
Modelos baseados em redes neurais convolucionais são particularmente úteis para este tipo de tarefa. Como foi dito na Seção 2.2, suas camadas convolucionais extraem representações de alto nível do conteúdo de uma imagem. Tais representações, as quais chamamos de *atributos* ou *features*, tornam-se progressivamente de alto nível de acordo com a profundidade da camada convolucional na rede. Uma rede neural convolucional "aprende" a extrair tais atributos de imagens durante o treinamento e os utiliza para inferir a classe dos objetos contidos em uma determinada imagem.

A *MobileNet* é uma classe de arquitetura para redes neurais convolucionais projetada para ter um bom custo-benefício entre a utilização de *recursos computacionais* e *velocidade de inferência*. A principal diferença entre a arquitetura *MobileNet* e as CNNs "tradicionais" é que, ao invés de filtros convolucionais 3×3 padrão da Equação 2.6 seguida de *ReLU*, suas camadas convolucionais utilizam filtros 3×3 de convolução *separável em profundidade*, definida por

$$Y[i,j,c] = (I * K)[i,j,c] = \sum_{m,n} K[m,n,c] \cdot I[i-m, j-n, c] \tag{2.9}$$

seguidos de um filtro 1×1 de convolução padrão. A Figura 2.9 ilustra a diferença entre as convoluções.

Além disso, a arquitetura expõe dois parâmetros que controlam a proporção entre velocidade de inferência e precisão da rede. O *Width Multiplier* permite a criação de modelos menores ao controlar a quantidade de mapas de atributos resultantes de cada camada. Já o *Resolution*



(a) (b) (c)

Figura 2.10: (a) Uma imagem de exemplo da base de dados de treinamento com as *caixas delimitadoras* de seus objetos de interesse anotados; e exemplos de *caixas padrão* de diferentes aspectos e suas rescpectivas grades, sendo (b) um grade de resolução 8×8 e em (c) uma de resolução 4×4. Durante o treinamento, é realizada a associação entre as *caixas padrão* e as *caixas delimitadoras* dos objetos das imagens de treinamento. Fonte: adaptado de [29]

Multiplier controla a resolução da imagem de entrada e dos mapas de atributos resultantes de cada camada convolucional, controlando assim o custo computacional da rede.

Neste estudo, optou-se por utilizar a *MobileNet* em detrimento a outras arquiteturas de rede mais robustas, como a *VGG* [31], em razão do baixo custo computacional necessário para sua operação, possibilitando que o sistema proposto aqui seja executado a uma boa taxa de *frames* por segundo. Na Tabela 2.1 está descrita a arquitetura da *MobileNet*, onde cada linha descreve uma camada da rede.

2.3.2 Single-Shot Multibox Detector

A *detecção de objetos* é uma importante tarefa de *visão computacional*, a qual consiste em detectar instâncias de objetos visuais de classes pré-definidas (como humanos, animais, carros, etc.) em imagens digitais, determinando as regiões ocupadas por tais objetos as quais são representadas por *caixas delimitadoras* (Figura 2.10(a)). Como um dos problemas fundamentais de visão computacional, *detecção de objetos* serve de base para uma variedade de tarefas, como segmentação de instância [32,33], rastreamento de objetos [34–37], legenda de imagem [38,39], reconhecimento de atividades [4,5], etc.

A técnica *Single-Shot Multibox Detector*, ou *SSD*, é um método de detecção de objetos utilizando uma única *rede neural profunda*. A abordagem discretiza o espaço de saída de caixas

Tabela 2.1: Arquitetura de uma MobileNet. A letra “s” seguida de um número indica o *passo* utilizado para a convolução na camada. A presença da sigla “dw” indica quando a a convolução da camada é *separável em profundidade*. A sigla “FC” indica uma camada do tipo *totalmente conectada*. Fonte: adaptado de [28]

Tipo / Passo	**Tamanhos do Filtro**	**Tamanho da Entrada**
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5 x Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5 x Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pooling 7×7	$7 \times 7 \times 1024$
FC	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classificador	$1 \times 1 \times 1000$

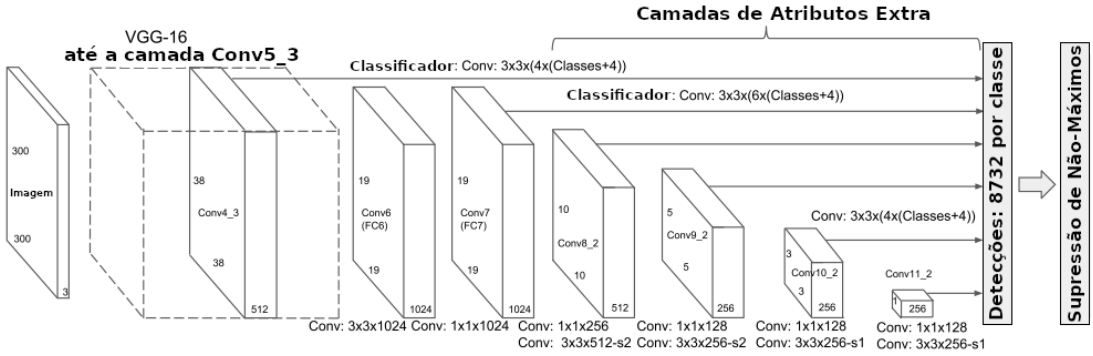


Figura 2.11: Rede SSD utilizando um VGG como rede base. Fonte: adaptado de [29]

delimitadoras em um conjunto de *caixas padrão* de diferentes aspectos e escalas por localidade nos *mapas de atributos*. Durante a inferência da rede são geradas as probabilidades de presença de cada categoria de objetos em cada *caixa padrão*, em conjunto com as correções de cada caixa para que as mesmas possam estar melhor ajustadas à forma dos objetos.

As camadas iniciais da rede neural são baseadas em uma arquitetura de *rede convolucional* padrão para a classificação de alta qualidade de imagens (truncada antes de qualquer camada de classificação) a qual é chamada de *rede base*. À *rede base* são adicionadas uma série de *estruturas convolucionais adicionais*, que são responsáveis pela predição de detecções em múltiplas escalas.

Cada estrutura adicional tem seus mapas de atributos resultantes associados com um conjunto de *caixas padrão*. Cada conjunto de *caixas padrão* divide o mapa de atributos associado em uma grade, de modo que as posições das caixas relativas às suas células correspondentes sejam fixas (Figura 2.10). Para cada célula do mapa de atributos são gerados os *offsets* relativos para cada caixa delimitadora e as probabilidade de cada classe de objetos, que indicam a presença de cada instância de classe na célula.

O conjunto de dados de treinamento da rede é formado por imagens que contenham as classes de objetos de interesse, juntamente com as definições das caixas delimitadoras de cada objeto. O treinamento da rede consiste em minimizar, simultaneamente, a função de custo de predição das classes presentes nas imagens, a qual é referida pelos autores da técnica como *classification loss*; e a função de custo da predição das *caixas delimitadoras* de cada objeto, a qual é referida pelos autores da técnica como *regression loss*. A Figura 2.11 exibe uma visão geral da arquitetura SSD, e na Figura 2.12 são exibidas em detalhes as estruturas adicionais da rede.

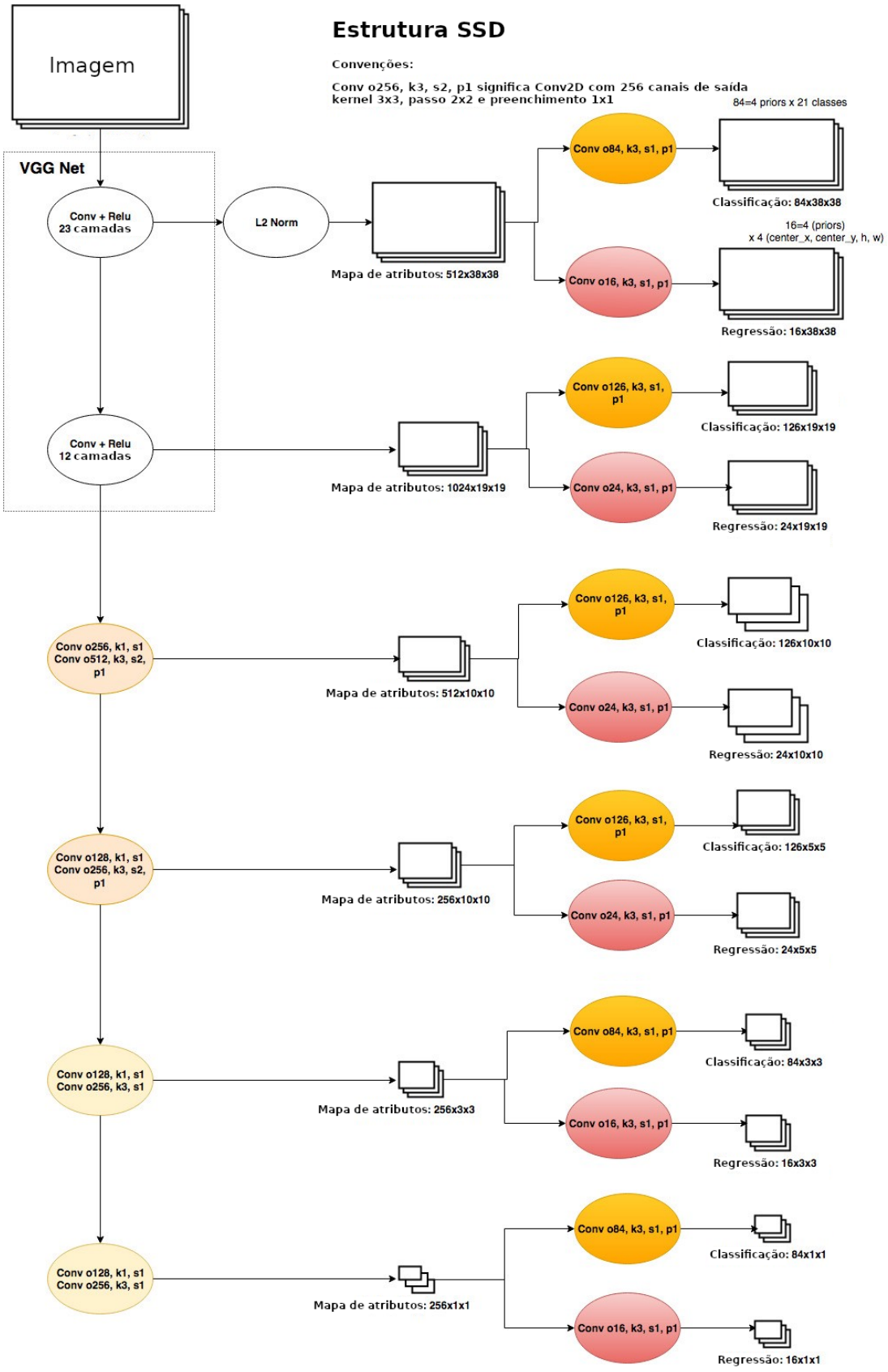


Figura 2.12: Estruturas adicionais da arquitetura SSD apresentada na Figura 2.11. Fonte: adaptado de [40]

Capítulo 3

Metodologia proposta

3.1 Introdução

Para a realização dos experimentos que são descritos neste trabalho, uma câmera de alta resolução foi colocada ao lado da estrada de ferro, em uma posição perpendicular à linha ferroviária. A Figura 3.1 exibe uma imagem tomada do ponto de vista escolhido para a câmera. Nela podemos identificar a lateral da superestrutura de um vagão, que consiste em uma caixa metálica cujo objetivo é armazenar os sedimentos de minério para transporte. A abordagem apresentada aqui visa determinar as localizações das extremidades das superestruturas (delineadas em vermelho), as quais chamaremos de região dos *drenos*, e rastrear tais elementos no decorrer do dos vídeos coletados pela câmera de testes. Tal metodologia consiste na ação conjunta de dois componentes principais: um detector de objetos e um conjunto de restrições geométricas.

3.2 Estratégia de detecção e elemento de interesse.

O detector de objetos utilizado neste trabalho é baseado na técnica SSD, empregando como rede base a arquitetura *MobileNetV1*. Sua função é identificar quaisquer novos elementos de interesse que possam vir a aparecer nos frames do vídeo em análise. Tais elementos são adicionados à lista de elementos detectados, os quais são rastreados nos frames subsequentes, enquanto estiverem à vista, ou suas localizações são estimadas, em caso de oclusão.

Como dito anteriormente, o elemento de interesse deste estudo foi a região dos *drenos* das superestruturas. Dentre as razões para a escolha desse elemento, podemos citar três principais:

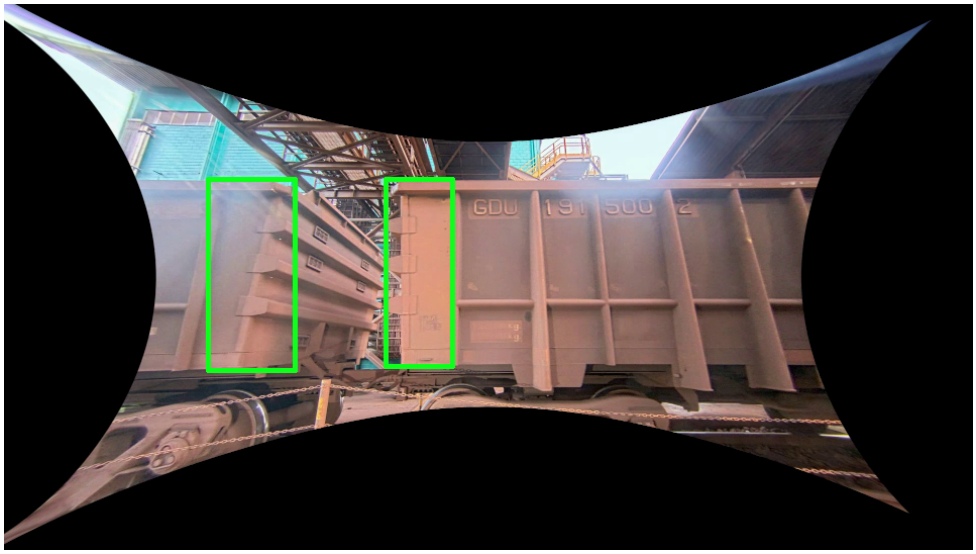


Figura 3.1: Imagem de exemplo do ponto de vista da câmera instalada para testes. A imagem original foi pré-processada para corrigir a distorção introduzida pela lente tipo *olhe-de-peixe* da câmera, o que ocasiona a presença das regiões de cor preta próximas às bordas da imagem. As regiões dos *drenos* da superestrutura estão delineadas em verde. Fonte: o autor.

- Ao analisar as Figuras 3.2(a) e 3.2(b), pode constatar que cada superestrutura possui um par de regiões de *dreno*, sendo que uma região pode ser considerada um versão espelhada da outra. Tal informação pode ser utilizada para a estimação da área total ocupada por uma superestrutura em um frame.

- As regiões de *dreno* possuem características singulares na superestrutura. Tais características são, em maioria, comuns entre as regiões de *dreno* de vagões de modelos diferentes (Figuras 3.2(c) e 3.2(d)).

- As informações de detecção das regiões de *dreno* podem ser utilizadas para auxiliar na detecção de outros elementos pertencentes ao vagão e no monitoramento dos mesmos.

3.3 Treinamento da rede convolucional

O treinamento do detector de objetos consiste em alimentar a rede convolucional com as imagens e anotações da base de dados de treinamento. A rede irá ajustar seus pesos sinápticos segundo uma função de otimização, minimizando a função custo definida para a rede (Seção 2.2.3).

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

Figura 3.2: Exemplos de regiões de *drenos* de superestruturas. Em (a) e (b) são exibidas as duas regiões de uma mesma superestrutura, enquanto em (c), (d) e (e) são exibidas as regiões de superestruturas de modelos diferentes. Fonte: o autor.

A rede base do detector foi inicializada com os pesos sinápticos resultantes do treinamento de uma rede de arquitetura MobileNetV1 para a tarefa de classificação utilizando a base de dados *ImageNet* [41]. O valor utilizado do parâmetro de aprendizagem para a rede base foi de 0.001, enquanto o valor de aprendizagem das estruturas adicionais de detecção foi de 0.0001. Ambos os parâmetros de aprendizagem possuem esquema de decaimento do tipo cosseno [42] (Figura 3.3).

A fim de tornar a rede convolucional mais genérica, evitando *overfitting* [43], cada imagem anotada de treinamento foi pré-processada utilizando técnicas de *data augmentation* [44]. *Data augmentation* são técnicas que visam gerar novos e variados dados de treinamento a partir dos dados já existentes. Cada imagem poderá sofrer algum tipo de transformação, e cada transformação possui uma certa probabilidade de ser aplicada.

As transformações utilizadas neste trabalho são:

- distorção fotométrica;
- preenchimento aleatório;
- recorte aleatório;
- espelhamento horizontal aleatório.

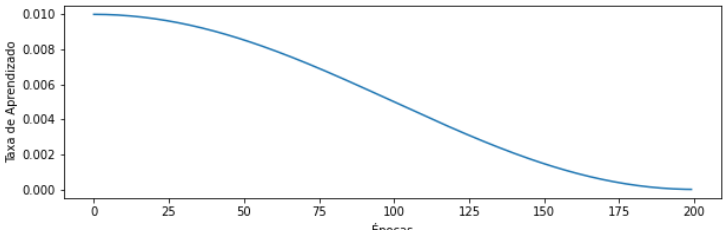


Figura 3.3: Curva de decaimento *cosseno* da taxa de aprendizagem. O valor da aprendizagem mantém-se próximo ao valor máximo durante as primeiras épocas, permitindo um convergência mais rápida do algoritmo. Os valores decaem gradualmente nas épocas intermediárias e estabilizam próximos ao valor de aprendizagem mínima nas épocas finais. Fonte: o autor.

A Figura 3.4 ilustra alguns exemplos de imagens transformadas. Além das transformações de *data augmentation*, todas as imagens passaram por duas transformações adicionais:

- redimensionamento para o tamanho de 312×312 pixels;
- seus canais foram subtraídos das médias de cada canal de todas as imagens da base de dados.

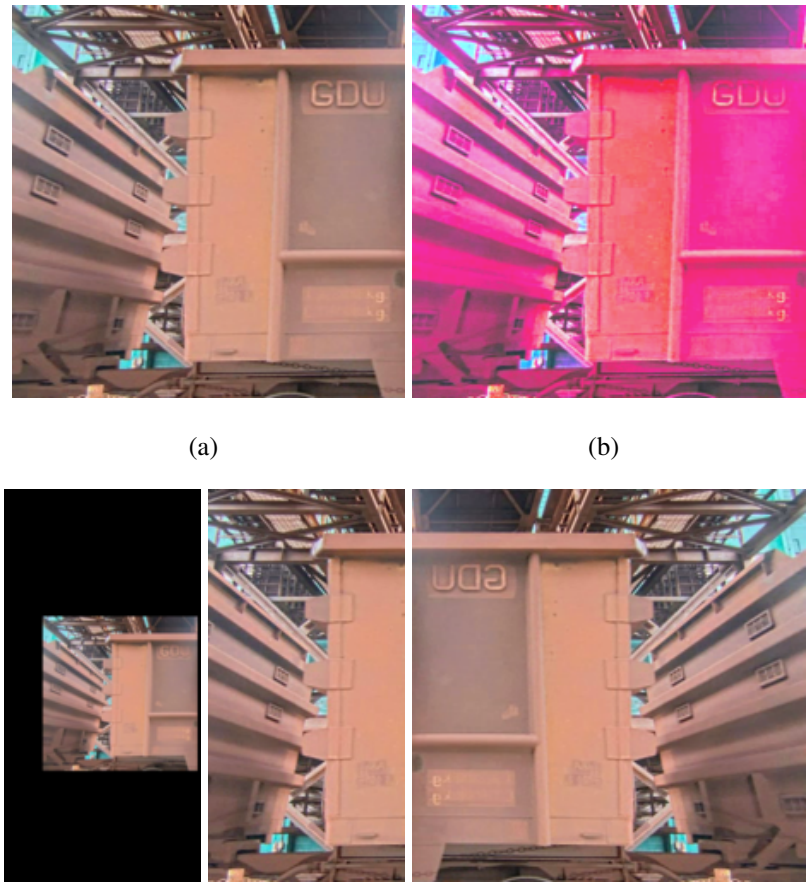
3.4 Criação da base de dados

Para a criação da base de dados para treinamento e teste do detector, foram coletados exemplos de imagens exibindo a visão lateral dos vagões. As regiões dos *drenos* nas imagens foram manualmente anotadas[1] no formato da base de dados *PASCAL VOC* [45], um padrão de descrição de posições e dimensões de objetos em imagens, popular na comunidade de visão computacional [46].

A fim de tornar o detector o mais confiável possível, as imagens coletadas incluem uma grande variedade e modelos de vagões. Ao total, foram coletadas 543 imagens anotadas, sendo 434 exemplos (aproximadamente 80%) utilizados para treinamento do detector e 108 exemplos restantes utilizados para teste.

Um exemplo de imagem anotada é exibido na Figura 3.5. Pode-se averiguar que as regiões anotadas abrangem completamente as extremidades das regiões dos *drenos* na maioria dos ca-

[1] As anotações foram realizadas utilizando a ferramenta de código aberto OpenLabeling, disponível em https://github.com/Cartucho/OpenLabeling



(a) (b)

(c) (d) (e)

Figura 3.4: Representação visual de algumas das transformações de imagens. (a) imagem original. (b) distorção fotométrica. (c) preenchimento aleatório. (d) recorte aleatório. (e) espelhamento horizontal aleatório. Fonte: o autor.

sos. Porém, a fim de tornar a rede convolucional utilizada como detector mais robusta a certos níveis de oclusão, alguns exemplos de drenos parcialmente visíveis (elemento mais à esquerda na Figura 3.5) também foram coletados.

3.5 Estratégia de rastreamento dos elementos de interesse.

A estratégia de rastreamento desenvolvida neste trabalho é baseada na aplicação de uma abordagem de *rastreamento-por-detecção* associada a um conjunto de restrições, cuja função é auxiliar na eliminação de erros de detecção, tornando o rastreamento mais resiliente a falhas. Nas seções posteriores, será discutida em detalhes a metodologia escolhida.

Figura 3.5: Imagem de exemplo pertencente à base de dados, com as anotações manuais das regiões de *dreno* presentes delineadas em vermelho. Fonte: o autor.

3.5.1 Rastreamento-por-detecção

Para representar a *caixa delimitadora* $\mathbf{b}_i$ de um elemento i qualquer, utilizamos o vetor

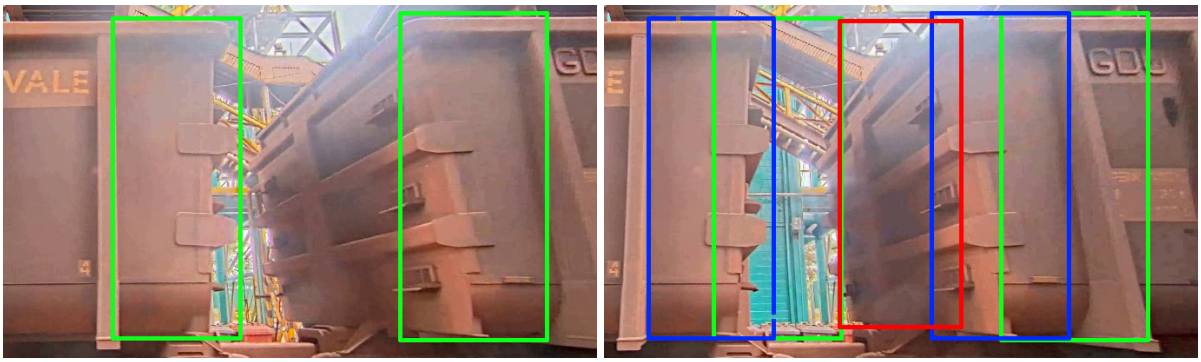
$$\mathbf{b}_i = (b_{ix}, b_{iy}, b_{iw}, b_{ih}), \tag{3.1}$$

onde b_x e b_y são as coordenadas horizontal e vertical, respectivamente, do centro do elemento; e b_w e b_h denotam a largura e altura do elemento, respectivamente.

Consideramos o vetor $\boldsymbol{b}_F = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, ..., \mathbf{b}_M)$ das M *caixas delimitadoras* dos elementos de interesse sendo atualmente rastreados pelo sistema no frame F, e o vetor $\boldsymbol{d}_{F+1} = (\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, ..., \mathbf{d}_N)$ das N *caixas delimitadoras* de novas detecções encontradas pelo detector de objetos no frame $F + 1$. A correspondência entre as caixas em $\boldsymbol{b}_F$ e $\boldsymbol{d}_{F+1}$ é determinada pelos pares de caixas com maior índice de *interseção pela união* [46], ou *IoU* [2]. No Apêndice A encontra-se o algortimo de correspondência entre os elementos de $\boldsymbol{b}_F$ e $\boldsymbol{d}_{F+1}$.

As novas detecções de $\boldsymbol{d}_{F+1}$ que não possuírem correspondência com nenhum elemento de $\boldsymbol{b}_F$, mas que estão de acordo com as restrições geométricas descritas na Seção 3.6, serão adicionadas à lista atualizada de elementos rastreados $\boldsymbol{b}_{F+1}$. As detecções em $\boldsymbol{d}_{F+1}$ que não estiverem de acordo com as restrições geométricas ou não possuírem correspondência com algum elemento de $\boldsymbol{b}_F$ serão ignorados como *erros de detecção*. Os elementos em $\boldsymbol{b}_F$ que não possuem correspondência com alguma detecção em $\boldsymbol{d}_{F+1}$ serão tratados pelo método descrito na próxima seção. A Figura 3.6 ilustra a representação desses conceitos.

[2] Do inglês *intersection over union*



(a) (b)

Figura 3.6: Representação do método de *rastreamento-por-detecção*. Considerando que o sentido do movimento dos vagões é da direita para a esquerda, em (a) são exibidos os elementos já rastreados em um determinado frame F, delineados em verde; (b) no frame $F + 1$ novas estimativas de localizações de elementos são exibidas, sendo, delineadas em azul, as detecções que serão utilizadas para atualizar as posições dos objetos rastreados, enquanto a que será ignorada está delineada em vermelho. Fonte: o autor.

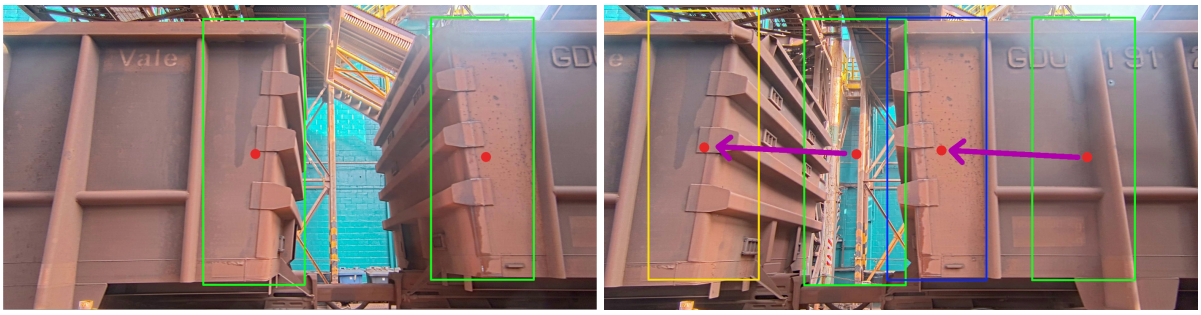
3.5.2 Tratamento de oclusão

O desempenho do detector de objetos pode deteriorar em razão de fatores inerentes do ambiente o qual a câmera de testes foi posicionada, como mudanças de exposição luminosa, vibração da câmera, oclusão dos elementos de interesse (como consequência de poeira, trabalhadores passando na frente da câmera, etc.) e presença de artefatos de movimento entre composição férrea e câmera, dentre outros. Tal deterioração pode levar a falhas de rastreamento de elementos que já estão sendo rastreados.

Tomando como premissas a pouca variação na geometria do vagão e a direção fixa da movimentação dos elementos de interesse, pode-se aplicar duas metodologias simples para correção de posição e tratamento de oclusão dos elementos já rastreados pelo sistema, vistos a seguir.

3.5.2.1 Oclusão com outros elementos visíveis no frame.

Esta metodologia é utilizada apenas quando, após a atualização de todos os elementos possíveis já rastreados, os elementos oclusos estiverem acompanhados de outros elementos visíveis no frame. Neste caso, a provável localização do elemento ocluso será determinada



(a) (b)

Figura 3.7: Tratamento de oclusão pelo vetor de deslocamento de *elementos visíveis*. Considerando o sentido do movimento dos vagões como sendo da direita para a esquerda, em (a) estão delineados em verde os elementos já rastreados no frame F. Em (b), no frame $F+1$, o elemento delineado em amarelo não pôde ser encontrado pelo detector, sendo estimado pelo vetor de deslocamento do elemento que foi detectado com sucesso, delineado em azul. Fonte: o autor.

através do vetor de *deslocamento médio* dos elementos que puderam ser atualizados. A Figura 3.7 exibe este conceito.

Consideremos $\mathbf{k}_F$ como o vetor das coordenadas centrais dos N *drenos* detectados no frame atual F (portanto, um subconjunto de $\mathbf{b}_F$) que possuem correspondência com os elementos $\mathbf{k}_{F+1}$ do conjunto $\mathbf{d}_{F+1}$.

O vetor de deslocamento médio $\boldsymbol{\Delta k}_{F+1}$ dos *drenos* detectados com sucesso no frame $F+1$ é dados por:

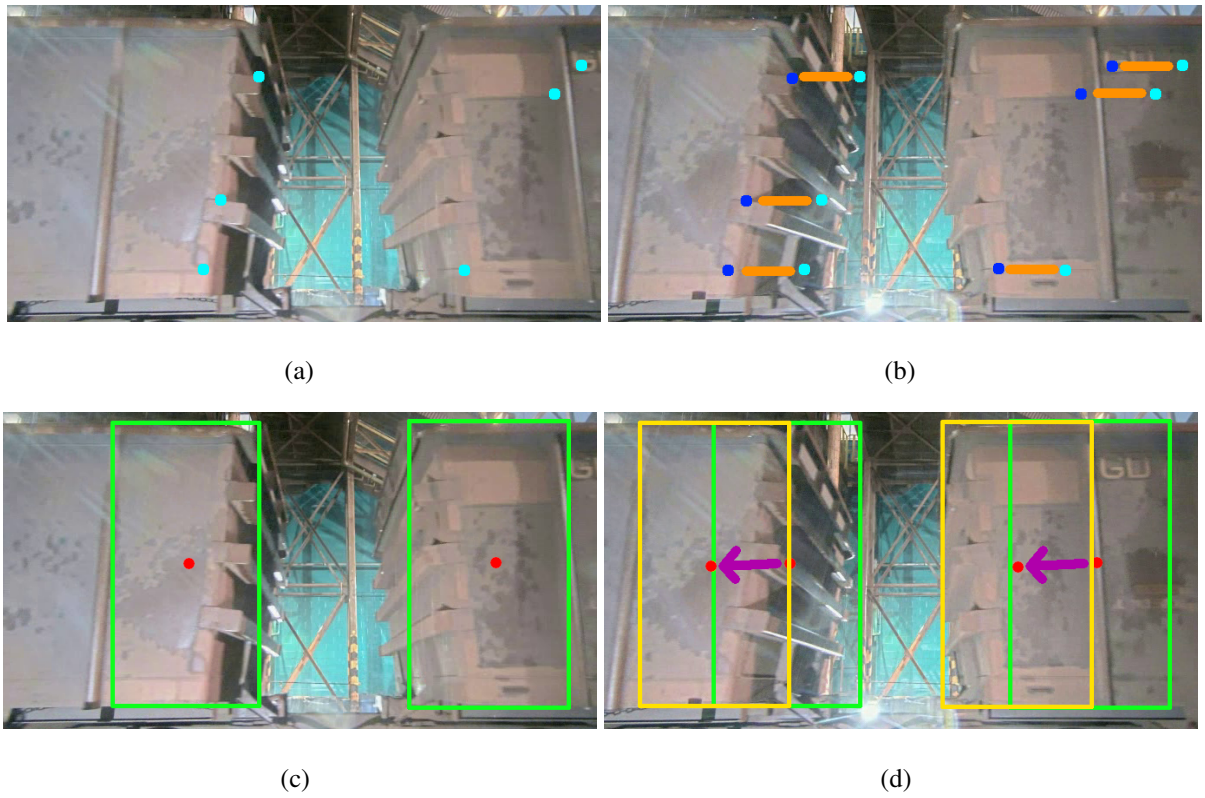
$$\boldsymbol{\Delta k}_{F+1} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{k}^i_{F+1} - \boldsymbol{k}^i_F) \tag{3.2}$$

Portanto, as posições atualizadas da lista $\mathbf{u}_{F-1}$ dos $M-N$ elementos rastreados no frame $F+1$ que não possuem uma detecção correspondente no frame F é computada por:

$$\mathbf{u}^i_F = \boldsymbol{\Delta k}_F + \mathbf{u}^i_{F-1} \tag{3.3}$$

3.5.2.2 Oclusão sem elementos visíveis no frame.

Caso os elementos oclusos não estejam acompanhados de elementos atualizados com sucesso no frame sendo analisado, a estimativa das localizações de tais elementos será realizada utilizando o vetor de deslocamento médio de *pontos de interesse* presentes no frame anterior e no atual, como está exibido na Figura 3.8.



(a) (b)

(c) (d)

Figura 3.8: Tratamento de oclusão pelo vetor de deslocamento de *pontos de interesse*. A atualização dos *pontos de interesse* do frame F, ilustrados em (c), para os pontos resultantes em (d) do frame $F+1$ é calculada pela média dos deslocamentos dos pontos de interesse do frame F em (a) para o frame $F+1$ em (b). Fonte: o autor.

Primeiramente, computa-se uma estimativa do movimento do vagão utilizando *fluxo óptico* [47]. Considerando $\mathbf{p} = (x, y)$ como sendo as coordenadas de um ponto em uma imagem, dado um frame $F-1$, as posições $\boldsymbol{p}_{F-1} = (\mathbf{p}^1_{F-1}, \mathbf{p}^2_{F-1}, \mathbf{p}^3_{F-1}, ..., \mathbf{p}^H_{F-1})$ de H pontos de interesse pertencentes à imagem são determinadas (Figura 3.8(a)). No frame de vídeo imediatamente posterior F, as posições atuais $\boldsymbol{p}_F = (\mathbf{p}^1_F, \mathbf{p}^2_F, \mathbf{p}^3_F, ..., \mathbf{p}^H_F)$ dos H pontos são encontradas (Figura 3.8(b). Calcula-se, então, o *vetor de deslocamento* $\Delta\mathbf{v}_F$ do vagão no frame F por

$$\Delta\mathbf{v}_F = \frac{1}{H}\sum_{i=1}^{H}(\mathbf{p}^i_F - \mathbf{p}^i_{F-1}). \tag{3.4}$$

Portanto, as posições atualizadas da lista $\mathbf{u}_{F-1}$ dos $M-N$ elementos rastreados no frame $F+1$ que não possuem uma detecção correspondente no frame F é computada por:

$$\mathbf{u}^i_F = \Delta\mathbf{v}_F + \mathbf{u}^i_{F-1}. \tag{3.5}$$

3.6 Restrições geométricas

3.6.1 Perfil de trajetória dos elementos

Em nossa configuração experimental, como a câmera de vídeo permanece fixa e o movimento do vagão de trem ocorre em paralelo ao plano da imagem, pode-se inferir que cada elemento rastreado irá ocupar regiões pré-definidas nos frames de vídeo, descrevendo uma trajetória retilínea fixa, a qual iremos nos referir como *perfil de trajetória*.

Portanto, o *perfil de trajetória* é definido como a equação da reta que se aproxima do caminho da trajetória real descrito pelos elementos de interesse:

$$\alpha b_{ix} + \beta b_{iy} + \theta = 0 \tag{3.6}$$

onde α, β e θ são os parâmetros da reta; e b_{ix} e b_{iy} são as coordenadas centrais da *caixa delimitadora* do i-ésimo elemento sendo atualmente rastreado.

O emprego do *perfil de trajetória* possibilita uma detecção e rastreamento de novos elementos mais resiliente a falhas de detecção. Quaisquer novos elementos detectados cuja a distância entre as coordenadas de seus centros e a linha de trajetória estiver acima de um limiar pré-definido será ignorada e, consequentemente, não serão rastreados, diminuindo o risco de erros de rastreamento nos frames subsequentes.

A Figura 3.9(a) exibe a representação visual do *perfil de trajetória*.

3.6.2 Distância entre elementos detectados

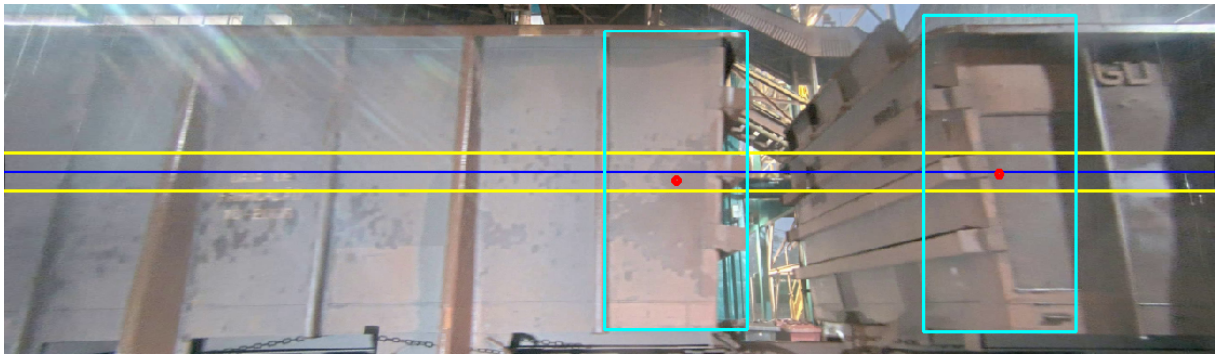
Analisando a Figura 3.1, pode-se notar um padrão na distância entre *drenos* detectados. Esse padrão pode ser utilizado para determinar se um par de drenos pertence à mesma superestrutura de um vagão.

Considerando $\mathbf{b}_m$ e $\mathbf{b}_n$ como sendo as caixas delimitadoras (Equação 3.1) de duas regiões de dreno, com o objetivo de utilizar medições invariantes à escala para um dado frame, definimos a função R como:

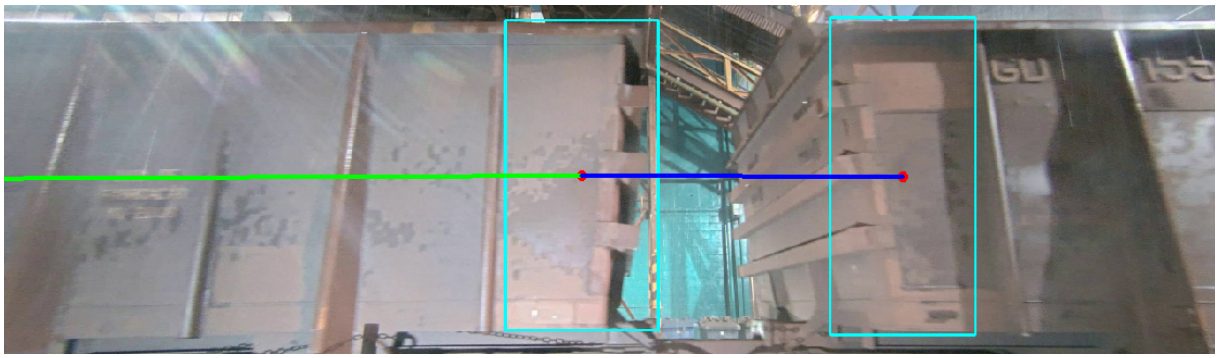
$$R(\mathbf{b}_m, \mathbf{b}_n) = \frac{\|(b_{mx} - b_{nx}, b_{my} - b_{ny})\|_2}{(b_{mh} + b_{nh})/2} \tag{3.7}$$

Utilizando R, a distância entre dois drenos pode ser classificada em:

- *intra-vagão* – entre dois drenos consecutivos de um mesmo vagão.



(a) Representação visual do *perfil de trajetória*. A linha azul representa a trajetória ideal do ponto central de cada *dreno* no decorrer do vídeo, e as linhas amarelas delimitam a região de tolerância.



(b) Representação visual da *distância entre elementos detectados*. A linha azul representa a distância entre drenos pertencentes a vagões diferentes, e a linha verde representa a distância entre drenos pertencentes ao mesmo vagão. Vale notar que, devido ao comprimento dos vagões, apenas um dos drenos de um mesmo vagão pode estar presente em um frame.

Figura 3.9: Representação visual das restrições geométricas. Fonte: o autor.

- *inter-vagões* – entre dois drenos de vagões diferentes.

A Figura 3.9(b) exibe exemplos de uma distância *intra-vagão* (linha verde) e de uma distância *inter-vagões* (linha azul). Em razão da posição da câmera no local de testes e do comprimento dos vagões, apenas a distância *inter-vagões* pode ser facilmente visualizada. A distância *intra-vagões* pode ser apenas estimada.

Considerando $\mathbf{d}$ como sendo a caixa delimitadora de uma nova possível detecção de um dreno; e $\mathbf{t}$ como a caixa delimitadora do último dreno rastreado pelo sistema; uma detecção do tipo *falso positivo* pode ser evitada analisando se $R(\mathbf{d}, \mathbf{t})$ pode ser classificado em uma das categorias mencionadas acima. Caso não possa, o novo elemento será ignorado. O procedimento de classificação também leva em consideração as classes previamente atribuídas à elementos já rastreados, com a finalidade de reduzir a ocorrência de erros de detecção. Por exemplo, na Fi-

gura 3.9(b), está claro que o próximo *dreno* a ser detectado (ainda não visível) deverá pertencer à classe *intra-vagões*; e, em seguida, o próximo *dreno* deverá ser da classe *inter-vagões*.

3.7 Ajuste das restrições geométricas

Para o ajuste das restrições geométricas, um vídeo de 30 segundos de duração, contendo o movimento típico de alguns vagões, foi selecionado. As posições e dimensões de cada dreno existente nos frames foram utilizadas como dados de referência para as rotinas que serão apresentadas a seguir.

3.7.1 Trajetória ideal de cada dreno

Considerando que os dados de referência contêm informação suficiente para a determinação do *perfil de trajetória* descrito na seção 3.6.1, os objetos encontrados pelo detector são utilizados para computar a variância média dos centros de suas coordenadas em relação à trajetória ideal. A variância média é, então, utilizada para determinar a distância de limiar.

A Figura 3.9(a) exibe a *região de interesse* deste estudo sobreposta pela linha do *perfil de trajetória* (linha azul), a região de *limiar de distância* (delineada em amarelo) e os drenos detectados (retângulos azul-claro), juntamente com seus centros (pontos vermelhos). Idealmente, o limiar de distância deve ser pequeno os suficiente para excluir quaisquer detecções *falso-positivas* que possam vir de ocorrer, como a detecção mais acima na imagem, mas alta o suficiente para acomodar a variação de posição dos elementos corretamente detectados.

3.7.2 Distâncias entre drenos

A segunda rotina é responsável por determinar as faixas de classificação das distâncias entre drenos. A Figura 3.9(b) exibe a saída visual da estratégia visual do script. As distâncias entre drenos do vídeo de exemplo são coletadas e utilizadas para determinar as faixas de classificação da razão R, como descrito na Seção 3.6.2.

Capítulo 4

Procedimentos experimentais e resultados

4.1 Introdução

Esta seção detalha a avaliação de desempenho do sistema proposto, que consistiu em dois estágios: avaliação do detector de objetos e avaliação das restrições geométricas aplicadas ao rastreamento.

4.2 Avaliação do detector SSD

A metodologia de *validação cruzada* [48] foi utilizada para avaliar a desempenho do detector SSD. O procedimento possui um único parâmetro, referido como k, que denota o número de grupos, também chamados de *folds*, nos quais os dados de treinamento serão divididos.

Utilizando $k = 5$, o procedimento consistiu em:

1. embaralhar a base de dados aleatoriamente e dividir os dados em k grupos;
2. para cada *fold* l:
   (a) assumir o grupo l como conjunto de teste e os grupos restantes como conjuntos de treinamento;
   (b) realizar o treinamento de 10 detectores SSD (com pesos sinápticos inicializados independentemente) utilizando o conjunto de treinamento e seguindo a metodologia descrita na Seção 3.3;
   (c) avaliar o desempenho de cada detector utilizando o conjunto de teste e armazenar os valores das métricas de avaliação;

Tabela 4.1: Médias das métricas computadas para o conjunto de teste cada *fold* da validação cruzada.

Fold	**VP**	**FP**	**FN**	**Precisão**	**Recall**	**F1 Score**	**AP**
Teste							
1	139.000	1.167	1.000	0.992	0.993	0.992	0.939
2	136.500	2.833	5.500	0.980	0.961	0.970	0.909
3	138.667	2.167	1.333	0.985	0.990	0.988	0.922
4	145.500	2.667	3.500	0.982	0.977	0.979	0.908
5	133.667	2.167	5.333	0.984	0.962	0.973	0.909

(d) calcular as médias das métricas de avaliação;

3. Sumarizar o desempenho do modelo utilizando as médias das métricas obtidos em cada *fold*.

A Tabela 4.1 exibe as médias das métrica de desempenho do conjunto de teste de cada *fold*. As métricas consideradas são:

- Número de detecções *verdadeiro-positivas* (**VP**), ou *acertos de detecção* – As caixas delimitadoras de uma anotação e de uma detecção estimada são correspondentes quando possuem um IoU de, pelo menos, 50%. Uma correspondência é considerada uma detecção positiva-verdadeira (Figura 4.1(a)).

- Número de detecções *falso-positivas* (**FP**), ou *alarmes falsos* – Caixas delimitadoras de detecções que não possuem anotações correspondentes (Figura 4.1(c)).

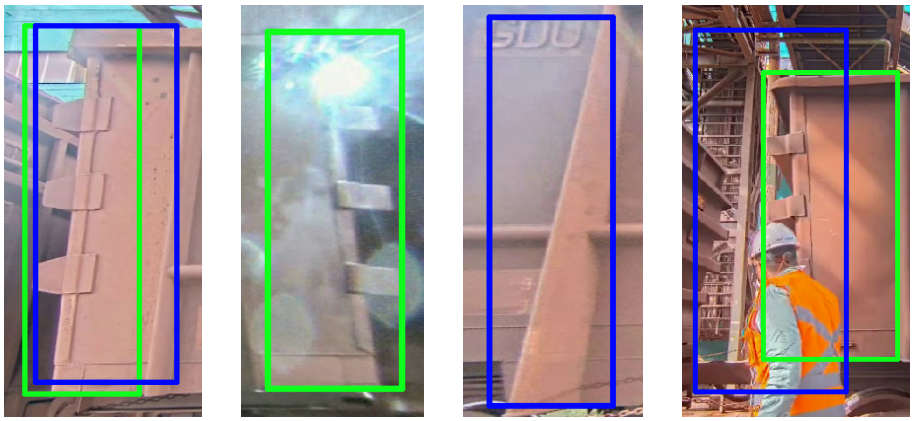
- Número de detecções *falso-negativas* (**FN**), ou *erros de detecção* – Caixas delimitadoras de anotações que não possuem detecções correspondentes (Figuras 4.1(d) e 4.1(b)).

- **Precisão** – Fração de acertos de detecção entre os casos positivos.

$$\text{precisão} = \frac{\text{número de acertos de detecção}}{\text{número total de objetos}}$$

- **Recall**, ou *sensitividade* – Fração de acertos de detecção entre as detecções.

$$\text{recall} = \frac{\text{número de acertos de detecção}}{\text{número total de detecções}}$$



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

Figura 4.1: Tipos de detecção considerados. As detecções estimadas pelo detector estão delineadas em azul, enquanto as anotações manuais estão delineadas em verde. Em (a) é exibida uma detecção verdadeiro-positiva; em (b) é exibido um exemplo de detecção falso-negativa e em (c) é exibida uma detecção falso-positiva. Em (d), devido a detecção estimada não possuir o *IoU* mínimo com a anotação manual, são exibidos, ao mesmo tempo, uma detecção falso-positiva e uma falso-negativa. Fonte: o autor.

- **F Score** – Média harmônica entre precisão e sensitividade.

$$\text{f score} = 2 \times \frac{\text{precisão} \times \text{recall}}{\text{precisão} + \text{recall}}$$

- *Average precision* (**AP**), ou *precisão média* [49] – Combina sensitividade e precisão para resultados de detecção *ordenados*.

  As detecções são ordenadas em valores de 0 a 1, e N intervalos igualmente espaçados são definidos. A *precisão média* é computada como:

$$AP = \frac{1}{N} \sum_{n=1}^{N} (\text{recall}_n - \text{recall}_{n-1}) \text{precisão}_n$$

  onde precisão_n e recall_n são, respectivamente, a precisão e recall para detecções acima do *n-ésimo* limiar.

  Neste trabalho, o valor de N utilizado é 11.

Analisando tais resultados, pode-se constatar a ação consistente do detector. Apesar das restrições geométricas serem os conceitos principais para o sistema proposto nesse estudo, é importante salientar a importância da atuação do detector de objetos para a metodologia em geral, e a qualidade de suas predições influenciam diretamente no desempenho geral do sistema.

4.3 Avaliação das restrições geométricas

Para aferir a efetividade das restrições geométricas impostas, o *perfil de trajetória* (seção 3.6.1) e as faixas de *classificação de distâncias* (Seção 3.6.2) foram ajustadas utilizando a amostra de vídeo e as rotinas descritas na seção 3.7. Feito isso, as faixas de classificação da razão R estabelecidas foram:

- Intra-vagão – de 2,5 a 5,0;
- Inter-vagões – de 0,5 a 1,5;

Comparamos a técnica de rastreamento proposta com outros dois sistema de detecção e rastreamento similares:

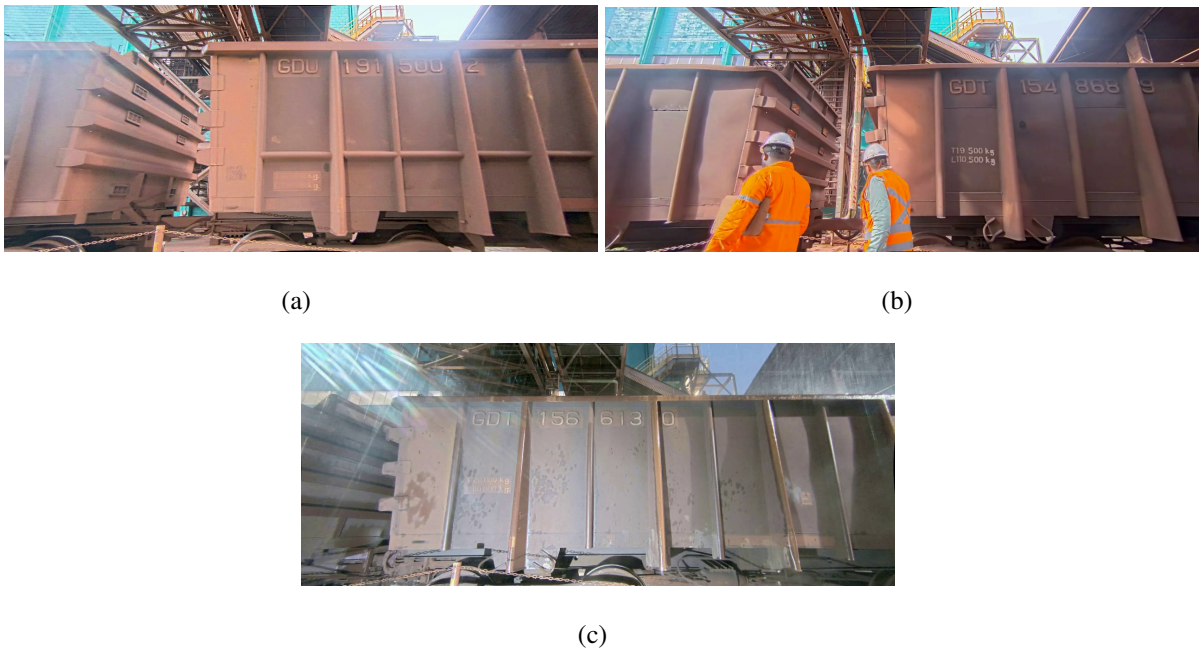
- Sistema 1 – utiliza a capacidade de detecção do algoritmo SSD puro tanto para detecção quanto para o rastreamento de elementos, como descrito na seção 3, sem a aplicação de quaisquer restrições geométricas.
- Sistema 2 – aplica a detecção de novos elementos a cada 10 frames. Tais detecções são utilizadas para a inicializar instâncias de rastreadores de objetos baseados na técnica *filtro de correlação "kernelizados"* (ou *KCF*[1] [35]). Tais rastreadores serão responsáveis pelo rastreamento dos objetos durante os próximos 10 frames. Após isso, o detector de objetor gerará novas detecções e novos rastreadores serão instanciados.

A escolha por tais sistemas é justificada pelo seus usos consolidados na solução de problemas comuns de visão computacional [36, 50–52]. A mesma base de dados foi utilizada para computar a desempenho de todos os sistemas.

Para esta avaliação, com a finalidade de obter resultados mais confiáveis, foram coletadas, além da amostra descrita na Seção 3.7, mais duas amostras de vídeo (30 segundos cada) manualmente anotados. Tais amostras apresentam condições e modelos de vagões diferentes entre si. A Figura 4.2 apresenta frames de exemplo de cada vídeo anotado.

As métricas de desempenho foram computadas para a técnica proposta e os dois sistemas mencionados utilizando cada um dos vídeo anotados. Os resultados são exibidos na Tabela 4.2 para o vídeo da Figura 4.2(a), na Tabela 4.3 para o da Figura 4.2(b) e na Tabela 4.4 para o da

[1] Do inglês *kernelized correlation filter*.



(a) (b)

(c)

Figura 4.2: Frames de exemplos dos videos utilizados para a avaliação das restrições. (a) Frame de exemplo do vídeo de amostra original e dos outros dois vídeos adicionais. Em (b) está presente um caso de oclusão das regiões de *dreno*; e (c) ilustra a amostra coletada à noite e na presença de chuva. Fonte: o autor.

Figura 4.2(c). Analisando os resultados, pode-se inferir que a técnica proposta possui um desempenho superior em detectar e rastrear corretamente regiões de *dreno* que estejam presentes nos frames dos vídeos, fato evidenciado pelos altos índices de detecções *verdadeiro-positivas* (métrica **TP**) e pela baixa quantidade de detecções *falso-negativas* (métrica **FN**) apresentada pela técnica em relação às outras técnicas consideradas. Essa afirmação é sustentada pela métrica **miss rate**, que indica a proporção de elementos não detectados para o total de elementos presentes no frame. Nas três tabelas, os valores dessa métrica para o sistema proposto se mantém como os menores.

Porém, pode-se notar também que nossa técnica possui um número de detecções *falso-positivas* particularmente alto ao analisar os resultados para o vídeo 1, e equiparável ao dos outros sistemas nos outros dois vídeos. Para explicar a ocorrência dessas detecções, a Figura 4.3 exibe o exemplo de uma dessas detecções *positivas-falsas*. Nela pode-se notar que a detecção ocorre devido à estimativa do elemento de interesse ter sua posição fora da área de interesse do frame (região à esquerda da linha verde)[2]. Esta é uma característica desejada do sistema, já que,

[2]Para fins ilustrativos, a imagem exibe apenas a região onde a detecção ocorreu.

devido aos pequenos ajustes de posição dos vagões no frame, um elemento que tenha saído da área de interesse do frame pode, eventualmente, voltar à área de visibilidade. Este padrão de estimativa ocorre na maioria dos frames de ocorrência de detecções *positivas-falsas*.

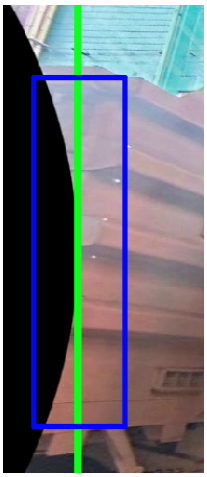


Figura 4.3: Exemplo de uma das detecções *falso-positivas* realizados pela metodologia proposta. Pode-se notar que a detecção (delineada em azul) indica a possível localização do elemento de interesse fora da área de interesse do frame. Fonte: o autor.

Tabela 4.2: Comparação entre as métricas de desempenho dos sistema de referência e de técnica proposta para o vídeo de referência descrito na seção 3.7.

	Sistema 1	**Sistema 2**	**Técnica Proposta**
TP	329	319	333
FP	3	7	14
FN	9	19	5
Precision	0.9910	0.9785	0.9597
Recall	0.9734	0.9438	0.9852
Miss Rate	0.0266	0.0562	0.0148
F1 score	0.9821	0.9608	0.9723
Ground Truths	338		
Quantidade de frames	300		

Tabela 4.3: Comparação entre as métricas de desempenho dos sistema de referência e de técnica proposta para o vídeo com um caso de oclusão dos elementos de interesse.

	Sistema 1	**Sistema 2**	**Técnica Proposta**
TP	344	339	360
FP	1	6	6
FN	20	25	4
Precision	0.9971	0.9826	0.9836
Recall	0.9451	0.9313	0.9890
Miss Rate	0.0549	0.0687	0.0110
F1 score	0.9704	0.9563	0.9863
Ground Truths	364		
Quantidade de frames	300		

Tabela 4.4: Comparação entre as métricas de desempenho dos sistema de referência e de técnica proposta para o vídeo coletado à noite e na presença de chuva.

	Sistema 1	**Sistema 2**	**Técnica Proposta**
TP	330	324	336
FP	2	6	8
FN	12	18	6
Precision	0.9940	0.9818	0.9767
Recall	0.9649	0.9474	0.9825
Miss Rate	0.0351	0.0526	0.0175
F1 score	0.9792	0.9643	0.9796
Ground Truths	342		
Quantidade de frames	300		

Capítulo 5

Considerações Finais

5.1 Conclusão

Neste trabalho, foi proposta uma simples, porém efetiva, abordagem de detecção e rastreamento de vagões de trem de mineração. O foco foi o componente do vagão chamado *superestrutura*; mais especificamente, as regiões das extremidades desse componente, as quais chamamos de *dreno*, em razão da sua estrutura característica e padrão de posicionamento. Nossa metodologia considera a estrutura fixa do vagão, seu padrão de movimento e a disposição usual que a seus componentes assumem em cada frame de vídeo.

A detecção da superestrutura é realizada por uma *rede neural convolucional*, implementada com a arquitetura *Single-Shot Multibox Detector*, cuja principal característica é a de estimar as posições, dimensões e classe de cada objeto presente em uma imagem de modo eficiente e preciso, dispensando abordagens de técnicas clássicas, como a de *janela deslizante*. Para o treinamento da rede, uma base de dados foi construída utilizando imagens capturadas do ponto de vista escolhido para a câmera de captura utilizada em nossos testes. Tais imagens exibem a visão lateral dos vagões de trem em movimento pela linha ferroviária. Os componentes de interesse presentes em cada frame foram manualmente anotados, onde cada anotação serviu de dado de referência para o processo de treinamento. Técnicas de *data augmentation* foram utilizadas, a fim de prover uma maior variabilidade dos dados de treinamento.

O rastreamento das superestruturas adotou uma abordagem de rastreamento-por-detecção. Consideramos como a atualização da posição de um objeto quando uma detecção positiva no frame atual possui um índice de *interseção sobre união* acima de 50% com um objeto de um frame imediatamente anterior.

Com o objetivo de reduzir a ocorrência de detecções *positivas falsas*, além de lidar com detecções *negativas falsas* e oclusão dos elementos de interesse, duas *restrições geométricas* foram propostas.

A primeira restrição foi baseada no fato de que o ponto de visão da câmera é fixo e o movimento dos componentes descrevem uma trajetória retilínea. Definimos o *perfil de trajetória* dos drenos e uma distância de tolerância correspondente. Quaisquer detecções as quais as distâncias de suas coordenadas centrais até o perfil de trajetória estiverem acima do valor de tolerância serão consideradas erros de detecção.

A segunda restrição considera que a distância normalizada entre drenos são pré-definidas de acordo com suas posições no vagão. Estas distâncias podem ser classificadas em *intra-vagões* e *inter-vagões*. Se a distância de uma nova detecção até o objeto vizinho mais próximo não puder ser classificada em uma dessas classes, a detecção é considerada positiva falsa e é, portanto, ignorada.

Para a avaliação de performance de nosso detector SSD, aplicou-se a metodologia de *validação cruzada*, utilizando a base de dados de treinamento da rede e 5 *folds*. Nossos resultados evidenciam a qualidade das estimativas do detector, um importante fator para a performance geral da metologia de rastreamento.

A avaliação de performance de nossas restrições geométricas foi realizada através da computação de métricas de desempenho para o rastreamento de drenos em um vídeo de exemplo. Tais métricas foram comparadas com as métricas de duas metodologias de rastreamento amplamente utilizadas na literatura. Os resultados obtidos mostram que nossa estratégia possui uma boa eficiência na detecção e rastreamento dos elementos de interesse, exibindo uma boa performance em diferentes condições de iluminação e com presença de diversos fatores que podem influenciar na detecção de elementos.

5.2 Trabalhos futuros

Apesar dos resultados satisfatórios obtidos pela metodologia proposta, algumas questões de melhorias foram levantadas durante o desenvolvimento deste trabalho. Tais questões são pertinentes tanto para fins de otimização de utilização de recursos computacionais quanto para melhoria da performance da técnica.

A arquitetura da rede convolucional utilizada do detector foi escolhida devido a seu bom

desempenho e baixo consumo de recursos computacionais. Mas, segundo os próprios autores da técnica SSD [29], a utilização de outras arquiteturas para a rede base pode resultar em um detector de com desempenho equivalente ao do utilizado. Em trabalhos futuros serão realizados experimentos utilizando outras arquiteturas de rede de alto desempenho, como a VGG [31] e ResNet [53], além de testes com outros detectores, como o RetinaNet [54].

A metodologia de rastreamento utilizada também possui aspectos que podem ser melhorados. As restrições geométricas apresentadas aqui foram projetadas considerando a posição da câmera utilizada em nossos experimentos em relação à estrada de ferro e o movimento dos vagões de trem. Em trabalhos futuros experimentos serão realizados com a finalidade de determinar como outras técnicas de rastreamento (por exemplo, *Filtros de Kalman* [55], *SORT* [51], *DaSiamRPN* [56], dentre outras [37, 51, 55, 56]) podem ser utilizadas para auxiliar no rastreamento dos elementos de interesse e, consequentemente, na eliminação de erros.

Por fim, devido à grande quantidade de componentes existentes nos vagões de trem e à necessidade de inspeção dos mesmo, este trabalho pode servir de base para a inspeção de outros desses componentes. Em trabalhos futuros planejamos realizar estudos de viabilidade da utilização da técnica proposta para a detecção de outros componentes.

Referências Bibliográficas

[1] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles," *Control Engineering Practice*, vol. 61, pp. 307–316, apr 2017.

[2] V. A. Laurense, J. Y. Goh, and J. C. Gerdes, "Path-tracking for autonomous vehicles at the limit of friction," in *Proceedings of the American Control Conference*. Institute of Electrical and Electronics Engineers Inc., jun 2017, pp. 5586–5591.

[3] R. Xu, S. Y. Nikouei, Y. Chen, A. Polunchenko, S. Song, C. Deng, and T. R. Faughnan, "Real-Time Human Objects Tracking for Smart Surveillance at the Edge," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[4] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View Adaptive Neural Networks for High Performance Skeleton-based Human Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1963–1978, apr 2018. [Online]. Available: http://arxiv.org/abs/1804.07453

[5] Y. Du, Y. Fu, and L. Wang, "Representation Learning of Temporal Dynamics for Skeleton-Based Action Recognition," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3010–3022, jul 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7450165/

[6] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, dec 1943. [Online]. Available: https://doi.org/10.1007/BF02478259http://link.springer.com/10.1007/BF02478259

[7] G. L. Shaw, "Donald Hebb: The Organization of Behavior," in *Brain Theory*, G. Palm and A. Aertsen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 231–233. [Online]. Available: http://link.springer.com/10.1007/978-3-642-70911-1_15

[8] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.” *Psychological Review*, vol. 65, no. 6, pp. 386–408, nov 1958. [Online]. Available: http://doi.apa.org/getdoi.cfm?doi=10.1037/h0042519http://www.ncbi.nlm.nih.gov/pubmed/13602029

[9] M. Minsky and S. Papert, *Perceptrons: expanded edition*. Cambridge, MA, USA: MIT Press, 1969, vol. 522. [Online]. Available: http://mitpress.mit.edu/book-home.tcl?isbn=0262631113

[10] S. Linnainmaa, “Taylor expansion of the accumulated rounding error,” *Bit*, vol. 16, no. 2, pp. 146–160, jun 1976. [Online]. Available: https://doi.org/10.1007/BF01931367http://link.springer.com/10.1007/BF01931367

[11] P. J. Werbos, *Applications of advances in nonlinear sensitivity analysis*. Berlin/Heidelberg: Springer-Verlag, 2005, pp. 762–770. [Online]. Available: http://dx.doi.org/10.1007/BFb0006203http://link.springer.com/10.1007/BFb0006203

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, oct 1986. [Online]. Available: https://doi.org/10.1038/323533a0http://www.nature.com/articles/323533a0

[13] R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted Boltzmann machines for collaborative filtering,” in *ACM International Conference Proceeding Series*, vol. 227. New York, New York, USA: ACM Press, jan 2007, pp. 791–798. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1273496.1273596

[14] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng, “Building high-level features using large scale unsupervised learning,” *CoRR*, vol. abs/1112.6, dec 2011. [Online]. Available: http://arxiv.org/abs/1112.6209

[15] S. Haykin, *Neural Networks and Learning Machines*, ser. Neural networks and learning machines. Prentice Hall, 2008, vol. 3, no. v. 10. [Online]. Available: https://books.google.com.br/books?id=K7P36lKzI%5C_QC

[16] S. Ruder, “An overview of gradient descent optimization algorithms,” *CoRR*, vol. abs/1609.0, sep 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[17] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, apr 1980. [Online]. Available: http://link.springer.com/10.1007/BF00344251http://www.ncbi.nlm.nih.gov/pubmed/7370364

[18] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” *2012 IEEE Conference on Computer Vision and Pattern Recognition*, vol. abs/1202.2, pp. 3642–3649, jun 2012. [Online]. Available: http://arxiv.org/abs/1202.2745http://ieeexplore.ieee.org/document/6248110/

[19] W. Lim, D. Jang, and T. Lee, “Speech emotion recognition using convolutional and Recurrent Neural Networks,” in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2016*. IEEE, dec 2017, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/7820699/

[20] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional Neural Networks for Speech Recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, oct 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6857341/

[21] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative Study of CNN and RNN for Natural Language Processing,” *CoRR*, vol. abs/1702.0, 2017. [Online]. Available: http://arxiv.org/abs/1702.01923

[22] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language modeling with gated convolutional networks,” *34th International Conference on Machine Learning, ICML 2017*, vol. 2, pp. 1551–1559, 2017. [Online]. Available: http://arxiv.org/abs/1612.08083

[23] S. Martin, “What’s the Difference Between a CNN and an RNN?” 2016. [Online]. Available: https://blogs.nvidia.com/blog/2018/09/05/whats-the-difference-between-a-cnn-and-an-rnn/

[24] D. Paper, “Thoroughly understand the meaning and calculation of receptive field — Develop Paper.” [Online]. Available: https://developpaper.com/thoroughly-understand-the-meaning-and-calculation-of-receptive-field/

[25] Teco KIDS, “Fully-Connected Layer with dynamic input shape - Teco KIDS - Medium.” [Online]. Available: https://medium.com/@tecokids.monastir/fully-connected-layer-with-dynamic-input-shape-70c869ae71af

[26] E. M. Wright and R. Bellman, “Adaptive Control Processes: A Guided Tour,” Princeton, N. J.: Princeton University Press, XVI, 255 p. (1961)., p. 160, may 1962. [Online]. Available: https://www.jstor.org/stable/3611672?origin=crossref

[27] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, “Flexible, high performance convolutional neural networks for image classification,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2011, pp. 1237–1242.

[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” *CoRR*, vol. abs/1704.0, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, dec 2016. [Online]. Available: http://arxiv.org/abs/1512.02325http://dx.doi.org/10.1007/978-3-319-46448-0_2

[30] K. Bai, “A Comprehensive Introduction to Different Types of Convolutions in Deep Learning.” [Online]. Available: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

[31] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[32] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, “Simultaneous detection and segmentation,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8695 LNCS, no. PART 7, pp. 297–312, jul 2014. [Online].

Available: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/sds.http://arxiv.org/abs/1407.1808http://link.springer.com/10.1007/978-3-319-10584-0_20

[33] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, mar 2020. [Online]. Available: https://github.com/http://arxiv.org/abs/1703.06870https://ieeexplore.ieee.org/document/8372616/

[34] G. Christie, A. Laddha, A. Agrawal, S. Antol, Y. Goyal, K. Kochersberger, and D. Batra, "Resolving Language and Vision Ambiguities Together: Joint Segmentation and Prepositional Attachment Resolution in Captioned Scenes," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1090–1097, apr 2016. [Online]. Available: http://arxiv.org/abs/1604.02125

[35] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *CoRR*, vol. abs/1404.7, apr 2014. [Online]. Available: http://arxiv.org/abs/1404.7584http://dx.doi.org/10.1109/TPAMI.2014.2345390

[36] M. Fiaz, A. Mahmood, and S. K. Jung, "Tracking Noisy Targets: A Review of Recent Object Tracking Approaches," *CoRR*, vol. abs/1802.0, feb 2018. [Online]. Available: http://arxiv.org/abs/1802.03098

[37] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaei, "Deep Learning for Visual Tracking: A Comprehensive Survey," *ArXiv*, dec 2019. [Online]. Available: http://arxiv.org/abs/1912.00535

[38] Q. Wu, C. Shen, P. Wang, A. Dick, and A. Van Den Hengel, "Image Captioning and Visual Question Answering Based on Attributes and External Knowledge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1367–1381, mar 2018. [Online]. Available: http://arxiv.org/abs/1603.02814

[39] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *32nd International Conference on Machine Learning, ICML 2015*, vol. 3, pp. 2048–2057, feb 2015. [Online]. Available: http://arxiv.org/abs/1502.03044

[40] H. Gao, “Understand Single Shot MultiBox Detector (SSD) and Implement It in Pytorch,” pp. 4–11, 2018. [Online]. Available: https://medium.com/@smallfishbigsea/understand-ssd-and-implement-your-own-caa3232cd6ad

[41] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2009, pp. 248–255. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206848

[42] I. Loshchilov and F. Hutter, “SGDR: Stochastic gradient descent with warm restarts,” *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, vol. abs/1608.0, 2019. [Online]. Available: http://arxiv.org/abs/1608.03983

[43] I. V. Tetko, D. J. Livingstone, and A. I. Luik, “Neural Network Studies. 1. Comparison of Overfitting and Overtraining,” *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 5, pp. 826–833, sep 1995. [Online]. Available: https://pubs.acs.org/doi/abs/10.1021/ci00027a006

[44] L. Perez and J. Wang, “The Effectiveness of Data Augmentation in Image Classification using Deep Learning,” *CoRR*, vol. abs/1712.0, dec 2017. [Online]. Available: http://arxiv.org/abs/1712.04621

[45] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Part 2 – Detection Task,” http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

[46] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, jul 2019. [Online]. Available: http://arxiv.org/abs/1807.05511

[47] B. D. Lucas and T. Kanade, “Iterative Image Registration Technique With an Application To Stereo Vision.” in *In IJCAI81*, vol. 2, 1981, pp. 674–679.

[48] K. A. Ross, C. S. Jensen, R. Snodgrass, C. E. Dyreson, C. S. Jensen, R. Snodgrass, S. Skiadopoulos, C. Sirangelo, M. L. Larsgaard, G. Grahne, D. Kifer, H.-A. Jacobsen, H. Hinterberger, A. Deutsch, A. Nash, K. Wada, W. M. P. Aalst, C. Dyreson, P. Mitra, I. H. Witten, B. Liu, C. C. Aggarwal, M. T. Özsu, C. Ogbuji, C. Patel, C. Weng,

C. Patel, C. Weng, A. Wright, A. Shabo (Shvo), D. Russler, R. A. Rocha, D. Russler, Y. A. Lussier, J. L. Chen, D. Russler, M. J. Zaki, A. Corral, M. Vassilakopoulos, D. Gunopulos, D. Wolfram, S. Venkatasubramanian, D. Gunopulos, M. Vazirgiannis, I. Davidson, S. Sarawagi, L. Peyton, H. Hinterberger, G. Speegle, V. Vianu, D. V. Gucht, O. Etzion, O. Etzion, F. Curbera, A. Ericsson, M. Berndtsson, J. Mellin, W. M. P. Aalst, P. M. D. Gray, G. Trajcevski, O. Wolfson, P. Scheuermann, C. Dorai, M. Weiner, A. Borgida, J. Mylopoulos, G. Vossen, A. Reuter, G. Grahne, V. Tannen, S. Elnikety, A. Fekete, L. Bertossi, F. Geerts, F. Geerts, W. Fan, T. Westerveld, H.-A. Jacobsen, C. Gurrin, T. Westerveld, O. Etzion, J. Kekäläinen, P. Arvola, M. Junkkari, K. Wada, K. Mouratidis, J. X. Yu, Y. Yao, J. Gehrke, S. Babu, A. Reuter, N. Palmer, C. K.-S. Leung, W. M. P. Aalst, M. W. Carroll, A. Gokhale, M. Ouzzani, B. Medjahed, A. K. Elmagarmid, S. Manegold, G. Cormode, S. Mankovskii, D. Zhang, T. Härder, W. Gao, C. Niu, Q. Li, Y. Yang, P. Refaeilzadeh, L. Tang, H. Liu, T. B. Pedersen, K. Morfonios, Y. Ioannidis, M. H. Böhlen, C. S. Jensen, R. T. Snodgrass, and L. Chen, *Cross-Validation*. Boston, MA: Springer US, 2009, pp. 532–538. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_565

[49] C. S. Jensen, R. T. Snodgrass, J. Chomicki, D. Toman, B. Thalheim, E. Ferrari, E. Ferrari, A. Vakali, E. Pitoura, G. Vossen, M. Mohania, U. Nambiar, M. Schrefl, M. Vincent, M. Berndtsson, J. Mellin, J. Mellin, M. Berndtsson, M. Berndtsson, J. Mellin, M. Berndtsson, J. Mellin, M. Berndtsson, J. Mellin, A. Ericsson, M. Berndtsson, J. Mellin, K. Goda, S. Abiteboul, O. Benjelloun, T. Milo, N. Palmer, L. Baresi, N. Palmer, M. Matera, C. Taton, N. D. Palma, S. Bouchenak, E. Pitoura, Z. Ives, Y. Zhang, J. B. D. Joshi, P. Bonnet, D. Shasha, S. Cohen, T. Tsikrika, B. Zheng, A. Wun, P. Revesz, P. Terenziani, P. M. D. Gray, A. Dobra, V. Plachouras, A. Gupta, X.-J. Wang, L. Zhang, S. Papadimitriou, S. Fischer-Hübner, Y. Zhang, J. B. D. Joshi, V. Vassalos, C. K.-S. Leung, C. S. Jensen, R. T. Snodgrass, D. Barbosa, I. Manolescu, J. Xu Yu, D. Lomet, H. Schuldt, P. Bonnet, D. Shasha, G. Dong, J. Li, F. Banaei-Kashani, C. Shahabi, W. Siberski, W. Nejdl, Q. Liu, V. Novák, J. Liu, B. Goethals, J. Cieslewicz, K. A. Ross, R. W. Moore, S. Kolahi, P. S. Yu, Y. Chi, J. Pei, N. Li, V. Khatri, R. T. Snodgrass, P. Terenziani, J. Mellin, M. Berndtsson, G. Weikum, L. Lu, A. Hanjalic, L. Lu, A. Hanjalic, L. Lu, A. Hanjalic, W. Kriechbaum, L. Lu, A. Hanjalic, L. Lu, A. Hanjalic, B. Levine, G. Miklau, M. Blanton, N. Hervé, N. Boujemaa, C. Amza, E. Zhang, Y. Zhang,

N. Craswell, S. Robertson, S. M. Beitzel, E. C. Jensen, O. Frieder, S. M. Beitzel, E. C. Jensen, and O. Frieder, *Average Precision*. Boston, MA: Springer US, 2009, pp. 192–193. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_482http://link.springer.com/10.1007/978-0-387-39940-9_482

[50] H. Zhang, A. Geiger, and R. Urtasun, "Understanding high-level semantics by modeling traffic patterns," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, dec 2013, pp. 3056–3063. [Online]. Available: http://ieeexplore.ieee.org/document/6751491/

[51] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2016-Augus, pp. 3464–3468, sep 2016. [Online]. Available: http://arxiv.org/abs/1602.00763http://dx.doi.org/10.1109/ICIP.2016.7533003http://ieeexplore.ieee.org/document/7533003/

[52] M. George, B. R. Jose, and J. Mathew, "Performance Evaluation of KCF based Trackers using VOT Dataset," *Procedia Computer Science*, vol. 125, pp. 560–567, 2018. [Online]. Available: www.sciencedirect.comwww.sciencedirect.comwww.elsevier.com/locate/https://linkinghub.elsevier.com/retrieve/pii/S1877050917328363

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2016-Decem, pp. 770–778, jun 2016. [Online]. Available: http://arxiv.org/abs/1512.03385http://ieeexplore.ieee.org/document/7780459/

[54] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, feb 2020. [Online]. Available: http://arxiv.org/abs/1708.02002https://ieeexplore.ieee.org/document/8417976/

[55] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, pp. 35–45, mar 1960.

[56] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-Aware Siamese Networks for Visual Object Tracking," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, aug 2018, vol. 11213 LNCS,

pp. 103–119. [Online]. Available: https://github.com/foolwood/DaSiamRPN.http://arxiv.org/abs/1808.06048http://link.springer.com/10.1007/978-3-030-01240-3_7

Apêndice A

Apêndice

Algorithm 1 Algoritmo para a correspondência entre as caixa delimitadoras de duas listas.

```
function CORRESPONDENCE(b_F, d_{F+1})
    matches ← [ ]
    for all b_m in b_F do
        candidates ← [ ]
        for all d_n in d_{F+1} do
            if d_n intersects b_m then
                append d_n to candidates
            end if
        end for
        best_match ← null
        best_value ← 0
        for all c in candidates do
            overlap ← jaccard(c, b_m)
            if overlap > best_value then
                best_value ← overlap
                best_match ← c
            end if
        end for
        append (b_m, best_match) to matches
    end for
    return matches
end function
```