UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Pedro Henrique Candido Lenzi

Aplicação de *Deep Appearance Descriptor* em algoritmo de visão computacional para rastreamento de múltiplos objetos em imagens de VANT

Aalen, Baden-Württemberg, Alemanha
2022

Pedro Henrique Candido Lenzi

Aplicação de *Deep Appearance Descriptor* em algoritmo de visão computacional para rastreamento de múltiplos objetos em imagens de VANT

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Dr. Rodolfo César Costa Flesch
Supervisor: Eng. Dr. Bernardo Cassimiro Fonseca de Oliveira

Aalen, Baden-Württemberg, Alemanha
2022

Ficha de identificação da obra

A ficha de identificação é elaborada pelo próprio autor.

Orientações em:

http://portalbu.ufsc.br/ficha

Pedro Henrique Candido Lenzi

Aplicação de *Deep Appearance Descriptor* em algoritmo de visão computacional para rastreamento de múltiplos objetos em imagens de VANT

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 13 de dezembro de 2022.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Rodolfo Flesch, Dr.
Orientador
UFSC/CTC/DAS

Bernardo de Oliveira ,Dr. Eng.
Supervisor
UFSC

Prof. Marcelo Stemmer, Dr.
Avaliador
UFSC/CTC/DAS

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado aos meus colegas de classe e
aos meus queridos pais.

AGRADECIMENTOS

A minha mãe Miriam e ao meu pai Elon que cada qual, a sua maneira, me ajudaram e apoiaram no ingresso e na permanência nesta Universidade, me guiaram e foram meu porto seguro incontáveis vezes.

A minha melhor amiga e irmã Fernanda, pelo seu companheirismo, suas palavras reconfortantes e pela disposição na revisão deste texto.

Aos meus colegas da Automação Guilherme, Gabriel e Matheus, por muitas vezes terem atitudes que me serviram de exemplo e pelo companheirismo nos melhores e piores momentos na graduação, que definitivamente foi essencial para a conclusão desta etapa da minha vida.

Ao meu professor Aldo, por despertar em mim com maestria a curiosidade sobre tema de visão computacional e me orientar na pesquisa desta ferramenta.

Ao colega de pesquisa Rodrigo, por se dispor tão gentilmente a me guiar na pesquisa sobre o tema de rastreamento de objetos.

Ao meu professor orientador Rodolfo, pela sua disposição de orientação neste trabalho e que incansavelmente auxiliou no desenvolvimento e revisão deste texto.

Ao meu supervisor Bernardo, por ter apostado no meu potencial, por ter me guiado e orientado com um soberbo senso crítico neste último trecho tão importante na minha graduação e por me incentivar a superar meus limites.

Aos colegas de trabalho da instituição parceira deste projeto, por me receber com braços abertos e pela disposição para me auxiliar nas adversidades.

A todas as mãos que sustentam a universidade pública e contribuem para o futuro do Brasil.

A todos que contribuíram direta ou indiretamente com este trabalho meu eterno agradecimento.

“I will be glad to talk with someone who knows this better than I.“
(BOCHKOVSKIY Alexey, 2020)

RESUMO

A solução de adicionar a ferramenta de *Deep Appearance Descriptor* à rotina do algoritmo de rastreamento de múltiplos objetos para melhorar o desempenho de reidentificação de objetos em sequências de quadros capturados por veículos aéreos não tripulados foi trazida pela empresa parceira a esse projeto como um projeto interno. Este trabalho relata como foram feitos o aperfeiçoamento e a integração desta ferramenta no programa de rastreamento da empresa, baseado na abordagem *Simple Online and Realtime Tracking*. Para o aperfeiçoamento desta ferramenta, criou-se um banco de imagens a fim de treinar uma rede neural artificial extratora de vetores de características pertencente à rotina desta ferramenta. Na criação desse banco de imagens, foi usada a distância de Mahalanobis dos vetores de características das mesmas como critério para a limpeza do *dataset*. Neste projeto, foram empregadas diferentes métricas a fim de avaliar o desempenho de rastreamento de reidentificação do código. Por fim, foi obtida uma redução no valor de reidentificações errôneas em 12%, representando uma melhoria nesta função.

Palavras-chave: Visão computacional. Associação de dados. Rastreamento de múltiplos objetos.

ABSTRACT

The solution of adding a Deep Appearance Descriptor tool to the routine of the multiple object tracking algorithm to improve the re-identification performance of objects in sequences of structures captured by unmanned aerial vehicles was brought by the partner company to this work as an internal project. This monography describes how the enhancement and integration of this tool into the tracking program based on the Simple Online and Realtime Tracking approach was done. To improve this tool, a dataset of images was created in order to train an artificial neural network that extracts the characteristic vectors and belong to the routine of this tool. When creating this dataset of images, the Mahalanobis distance of the feature vectors was used as a criterion for cleaning the dataset. In this project, different metrics were used in order to evaluate the code re-identification tracking performance. Finally, a reduction in the value of erroneous re-identifications was obtained representing an improvement of 12% in this function.

Keywords: Computer vision. Data association. Multi object tracking.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BB	*Bounding Box*
DAD	*Deep Appearance Descriptor*
DeepSORT	*Simple Online and Realtime Tracking with Deep Association Metric*
FV	*Features Vector*
FVENN	*Features Vector Extractor Neural Network*
GT	*Ground Truth*
HOTA	*Higher Order Tracking Accuracy*
ID	Identificação
MOT	*Multi Object Tracking*
Object Re-ID	*Object Re-identification*
OS	*Operating System*
RNA	Rede Neural Artificial
SORT	*Simple Online Realtime Tracking*
VANT	Veículo Aéreo não Tripulado
WRN	*Wide Residual Network*

SUMÁRIO

1 INTRODUÇÃO

Este documento relata o projeto de fim de curso do aluno Pedro Lenzi em parceria com uma empresa do ramo de segurança e vigilância. Essa empresa oferece produtos e serviços relacionados a soluções tecnológicas de sensoriamento para missões de monitoramento e vigilância. O setor da empresa em que o aluno desenvolveu seu projeto é encarregado do desenvolvimento de dispositivos ópticos para aplicações em segurança civil e militar (HENSOLDT, 2022).

Devido ao atual cenário de conflitos geopolíticos que a região da Europa vem enfrentando nos últimos anos, a empresa trouxe para o setor de optrônica uma maior atenção aos projetos envolvendo equipamentos de vigilância para ser embarcado em veículo aéreo não tripulado (VANT). Essa categoria de equipamento tem mostrado grande vantagem estratégica ao longo dos anos nas missões de monitoramento, porém, apesar do constante aprimoramento tecnológico embarcado nessas aeronaves, algumas tarefas ainda são desafiadoras para missões de monitoramento terrestre. Um desses desafios é implementar o rastreamento automático de múltiplos objetos de interesse (MOT, do inglês *multiple object tracking*).

O algoritmo de rastreamento automático de múltiplos objetos de interesse tem a função de, além de identificar e classificar objetos de interesse, como veículos terrestres, civis ou militares em solo, conseguir estabelecer a trajetória percorrida por eles usando visão computacional. Logo, essa ferramenta tem grande potencial, pois permite que o operador do VANT tenha acesso a informações sobre objetos de interesse pré-processadas, fazendo com que o mesmo possa tomar decisões de forma mais rápida. Um exemplo desta aplicação é um cenário em que um operador monitora o deslocamento de mais de um veículo militar em uma via pública com veículos civis. Neste ambiente descrito, são importantes a diferenciação entre veículos militares e civis e o rastreamento de mais de um objeto de interesse simultaneamente.

A atividade de MOT aplicada em cenários de monitoramento terrestre com Veículo Aéreo não Tripulado (VANT) é um desafio no setor tecnológico, pois envolve uma cadeia de atividades que devem ser processadas em um curto período de tempo em um hardware limitado. Para esse desafio, os desenvolvedores contam com técnicas clássicas de visão computacional e inteligência artificial para obter a melhor combinação de qualidade de rastreamento e tempo de processamento.

A empresa hoje desenvolve um algoritmo MOT para ser aplicado no cenário descrito nos parágrafos acima. Durante o desenvolvimento, foram mapeadas algumas hipóteses e estratégias que poderiam ser implementadas no algoritmo a fim de deixá-lo mais eficiente para a aplicação. Uma dessas hipóteses de melhoria é a adição da estratégia de descritor profundo de aparência (DAD, do inglês deep appearance descriptor) no *pipeline* do MOT.

Este trabalho relata as pesquisas, a utilização e a personalização de função de DAD para ser integrada no algoritmo de MOT usando imagens de VANTs em monitoramento terrestre.

1.1 MOTIVAÇÃO - APRESENTANDO A DEMANDA DA EMPRESA

O setor em que o aluno atuou durante o período de desenvolvimento deste trabalho tem como foco desenvolver soluções para processamento de imagens usando ferramentas de visão computacional. Uma das demandas do setor é desenvolver sistema para ser embarcado em VANT a fim de rastrear a posição de múltiplos objetos de interesse em uma sequência de imagens obtidas pelo veículo aéreo. Esse sistema tem a função de entregar ao usuário informações pré-processadas de trajetórias de múltiplos objetos terrestres em tempo real usando um hardware limitado embarcado na aeronave até então não definido. Os responsáveis pelo projeto determinaram os pré-requisitos necessários para o sistema: o sistema deve ser capaz de efetuar inferências com desempenho semelhante aos desempenhos de códigos de *tracking* considerados estado da arte, em uma frequência superior a 10 quadros por segundo e ser capaz de ser executado em um hardware limitado, até então não definido, a bordo da aeronave não tripulada.

A fim de atender esses pré-requisitos, foi iniciado o mapeamento de algoritmos disponíveis para tal aplicação. Após um estudo entre os algoritmos disponíveis, foi avaliado qual desses poderiam atender os pré-requisitos descritos acima e foi selecionado, como algoritmo base para o sistema proposto, o *Simple Online Realtime Tracking* (SORT) descrito em Bewley *et al.* (2016). Esse algoritmo tem as vantagens de efetuar as inferências mais rápido do que outros algoritmos estudados e apresenta modularidade da ferramenta de detecção de objetos.

Durante o desenvolvimento do sistema baseado no algoritmo SORT, notou-se que a função de re-identificação de objetos, etapa fundamental no rastreamento de objetos, poderia ter um ganho de desempenho adicionando a função de *Deep Appearance Descriptor* (WOJKE; BEWLEY; PAULUS, 2017) ao algoritmo. Detalhes sobre essa ferramenta são apresentados na Seção 2.4 deste trabalho.

Durante o estudo dessa ferramenta de DAD, levantou-se a hipótese de que se a rede neural extratora de vetores de características (FVENN, do inglês *features vector extractor neural network*) usada pela ferramenta fosse treinada com um *dataset* de imagens com as mesmas características das imagens coletadas por uma câmera embarcada em um VANT, o desempenho do algoritmo de MOT melhoraria.

Logo, a demanda da empresa envolve: desenvolver um *dataset* de imagens com as características das imagens coletadas por um VANT; fazer o treinamento de uma FVENN com esse *dataset*; integrar a ferramenta de DAD no algoritmo de MOT baseado na aplicação SORT e comparar resultados a fim de analisar mudanças de

desempenho.

1.2 OBJETIVOS

Objetivo geral

O objetivo geral deste trabalho é relatar como, durante a experiência de estágio, buscou-se melhorar o desempenho do algoritmo de rastreamento de múltiplos objetos da empresa por meio da melhora do processo de reassociação de Identificação (ID) de objetos de interesse.

Objetivos específicos

Os objetivos específicos do trabalho são relatar como buscou-se o aperfeiçoamento da ferramenta de Deep Appearance Descriptor para algoritmo de DeepSORT cuja Rede Neural Artificial (RNA) deve ser treinada com banco de imagens com propriedades similares às encontradas no cenário de aplicação do algoritmo de rastreamento e aplicar esta ferramenta no algoritmo de rastreamento de múltiplos objetos da empresa. Esses objetivos específicos são divididos nas seguintes etapas:

- obter uma função de DAD;
- obter um banco de imagens com as propriedades similares ao cenário ao qual o algoritmo de *tracking* da empresa será submetido;
- fazer o treinamento da *Features Vector Extractor Neural Network* (FVENN) usada na ferramenta de DAD com as imagens do banco de dados obtido; e
- aplicar a ferramenta DAD com a nova versão da FVENN no algoritmo de rastreamento de múltiplos objetos da empresa.

1.3 ESTRUTURA DO DOCUMENTO

O presente trabalho é estruturado da forma descrita nos parágrafos seguintes.

No capítulo 2 são apresentados os trabalhos relacionados que serviram de base para o desenvolvimento deste projeto. Inicialmente, é levantado o trabalho *Object Re-Identification Based on Deep Learning*, que teve como maior contribuição ser uma introdução aos temas abordados neste projeto. Na sequência, são introduzidas duas abordagens para algoritmos de *tracking*, SORT e DeepSORT, que foram estratégias bases para este trabalho. Depois dessas introduções, são apresentados trabalhos bases nos temas de treinamento de RNA para extração de vetores de características e métricas de desempenho de algoritmos de *tracking* utilizadas nas análises durante o projeto.

No capítulo 3, desenvolvimento, são relatadas as atividades desenvolvidas pelo aluno. Inicialmente, são retratados os mapeamentos e escolhas de algoritmos e banco de imagens base para rastreamento de múltiplos objetos. Em seguida, é apresentado o levantamento de hipóteses para melhoria do desempenho de *tracking* do código Deep-SORT e hipóteses sobre construção e refinamento de banco de imagens próprio para a aplicação. Na sequência, é descrito o treinamento da RNA da função de descritor de aparência com o *dataset* de imagens criado e limpo. Por fim, é explicado como foi feita a integração da ferramenta de descritor de aparência aperfeiçoada pelo aluno no algoritmo de MOT até então desenvolvido pela empresa.

No capítulo 4, resultados, são apresentados os resultados mais relevantes das atividades do capítulo de desenvolvimento. Inicialmente nessa seção, são exibidos os resultados do primeiro e do segundo treinamentos da RNA da ferramenta *Deep Appearance Descriptor* e também são expostos os resultados das análises de desempenho do código de *tracking* com estas duas versões. Na sequência, são apresentados os efeitos da adição da ferramenta de DAD no código de *tracking* da empresa, bem como a análise de desempenho da nova versão desse código.

No capítulo 5 são expostas as conclusões de maiores relevâncias no projeto também é apresentado quais foram os fatos que levantaram essa conclusões. Por último, são apresentadas as sugestões de trabalhos futuros para continuação deste projeto.

2 TRABALHOS RELACIONADOS

Neste capítulo serão introduzidos os principais trabalhos que serviram como base para este projeto. Inicialmente, é descrito o trabalho de *Object Re-Identification Based on Deep Learning* que teve como maior contribuição ser uma introdução ao tema de códigos de rastreamento de múltiplos objetos, reidentificação de objetos e treinamento de RNA baseado em *metric learning*. Na sequência, são introduzidas duas abordagens para algoritmos de *tracking*, SORT e DeepSORT que foram estratégias bases para este trabalho. Depois dessas introduções, é apresentada a abordagem de treinamento de RNA para extração de vetores de características com o método *Cosine Cetric Learning* e apresentados os bancos de imagens de maior importância no desenvolvimento deste projeto. Por fim, é exposto o trabalho *Higher Order Tracking Accuracy* (HOTA), que explica as métricas de desempenho de algoritmos de *tracking* mais importantes utilizadas nas análises durante o projeto.

2.1 OBJECT RE-IDENTIFICATION BASED ON DEEP LEARNING

O trabalho de (LI; ZHOU, 2019) introduz as principais técnicas de aprendizado de máquina utilizadas na reidentificação de objetos (*Object Re-ID*, do inglês *object re-identification*) para a aplicação em ambientes com múltiplas câmeras. As principais aplicações desse trabalho são *Object Re-identification (Object Re-ID)* de pedestres e de veículos. Os autores introduzem o tema levantando a atual relevância destas duas aplicações em temas de controle inteligente de tráfego e segurança pública urbana. Em seguida, os autores citam as estratégias dos trabalhos mais relevantes de *Object Re-ID* de pedestre e de veículos e como são usadas ferramentas de visão computacional clássica e ferramentas de redes neurais para tais estratégias. Na sequência, os autores apresentam alguns bancos de dados públicos para *Object Re-ID*, como *Market1501*, *Ve-Ri-776* e *VRID-1*. Em seguida, é apresentada a sequência lógica do processo de *Object Re-ID* que se constitui de, primeiro, uma entrada de dados em uma rede neural artificial, seguida de uma inferência de rede neural artificial e, por fim, como é medida a distância entre os vetores de características (*Features Vector* (FV), do inglês *features vector*) resultantes da inferência da rede. Ainda nesse trabalho, é apresentado o modelo de RNA como o Histograma de Gradiente Orientado (HOG, do inglês *histogram of oriented gradient*), o modelo Rede Neural Convolucional (CNN, do inglês *convolutional neural network*) e como é realizada a medição de distância entre FV resultantes .

Esse estudo teve fundamental papel de introdução do aluno ao tema de algoritmos de rastreamento de múltiplos objetos, treinamento de RNA extratoras de vetores de características, introdução dos algoritmos de estado da arte de *tracking* e principais banco de imagens utilizados nesses códigos. Por fim, esse artigo foi chave para as

identificações dos requisitos necessários para este projeto.

2.2 SIMPLE ONLINE REALTIME TRACKING, SORT

O trabalho de (BEWLEY *et al.*, 2016) tem o objetivo de apresentar uma estratégia de algoritmo de rastreamento de múltiplos objetos MOT, simples e eficiente que combina ferramentas clássicas de visão computacional, como filtro de Kalman e *Hungarian algorithm*, e atinge uma taxa de inferência que pode chegar a ser vinte vezes mais rápida que outros rastreadores considerados estado da arte. É apresentado que o método empregado utiliza somente o quadro anterior e o atual para rastrear os objetos de interesse e utiliza somente a posição e as dimensões das caixas delimitadoras (BB, do inglês *bounding boxes*) para a associação de dados e estimativa do movimento. Outra limitação é que este método não lida com oclusão dos objetos de interesse. Nesse trabalho, os autores comentam que usou o modelo da rede neural de detecção de objetos *Faster Region CNN (FrRCNN)* e em seguida é apresentado o modelo de estimativa de estados da posição do objeto de interesse para o quadro seguinte. Na sequência, é apresentado como a intersecção sobre união (IOU, do inglês *intersection over union*) e *Hungarian algorithm* para são usados a etapa de associação de dados. Por fim, os autores apresentam os resultados desta abordagem de SORT e compara o desempenho dela com outros algoritmos de *tracking* do estado da arte e conclui afirmando que a qualidade de rastreamento é altamente vinculada ao algoritmo de detecção.

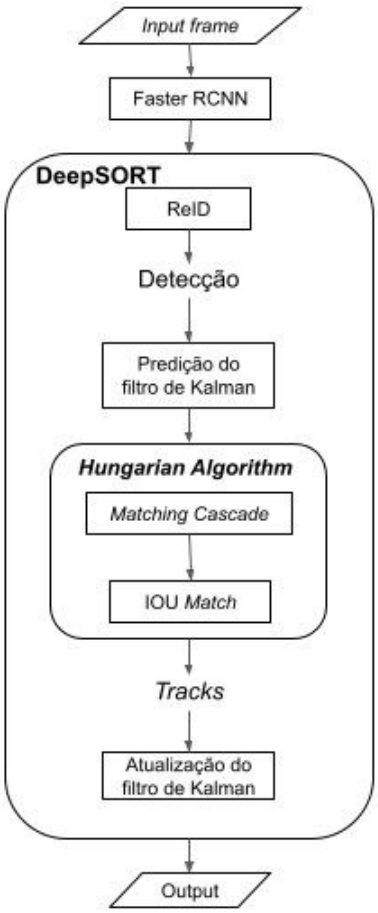
Esse trabalho foi fundamental para esta monografia, pois ele foi o algoritmo base para o desenvolvimento do código que a empresa desenvolveu para rastreamento de múltiplos objetos.

2.3 SIMPLE ONLINE AND REALTIME TRACKING WITH A DEEP ASSOCIATION METRIC, DEEPSORT

O trabalho apresenta um aperfeiçoamento do trabalho de SORT por meio da adição de um descritor de aparência baseado em rede neural. Desta forma, os autores conseguiram prolongar o tempo de rastreamento de objetos oclusos e, assim, melhorar o desempenho da tarefa de rastreamento. A ferramenta de descrição de aparência usada foi treinada baseada no trabalho (WOJKE; BEWLEY, 2018). Nesse trabalho, os autores usaram a RNA de detecção de objetos Faster RCNN. Já a estrutura da RNA usada para a ferramenta descritora de aparência foi a *Wide Residual Networks*, que conta com 10 camadas. Essa estrutura foi considerada, pelos autores, bem adequada para os requisitos temporais do algoritmo (WOJKE; BEWLEY, 2018). O resultado da implementação desta ferramenta foi a redução de 45% na criação de novos IDs nas

sequências de imagens testadas. A Figura 1 apresenta um diagrama com a arquitetura do algoritmo DeepSORT proposta.

Figura 1 – Arquitetura algoritmo DeepSORT com rede Faster RCNN



Fonte: Original do autor

Esse trabalho foi importante no desenvolvimento desta monografia pois ele foi a fundamentação teórica base para a hipótese levantada pela empresa de que com a adição de um DAD no seu algoritmo de MOT, desenvolvido sem a participação do aluno, poderia acarretar uma melhora de desempenho na função de reidentificação de objetos.

2.4 DEEP COSINE METRIC LEARNING FOR PERSON RE-IDENTIFICATION

Nesse trabalho, os autores utilizaram do treinamento convencional de RNA de classificação *softmax* para fazer com que dois vetores de características correspondentes às figuras de mesmas classes tenham menor distância entre si do que dois vetores de figuras de diferentes classes através da amostragem tripla. Os autores usaram dois bancos de imagens de reidentificação para avaliar o método abordado. Após o trei-

namento com esses bancos de imagens, a camada *softmax* de classificação da rede neural foi retirada, fazendo com que a rede neural convolucional passasse a ter uma saída de 128 valores (*float*). Os autores concluíram que apesar de ter somente testado uma arquitetura de rede neural para esse trabalho, foi obtido um ganho modesto no desempenho do teste e que em futuros trabalhos deveriam ser usadas redes com arquiteturas com maior capacidade de extração de características e usados diferentes bancos de imagens (WOJKE; BEWLEY, 2018).

Esse trabalho foi fundamental para esta monografia pois nele estão contidas as fundamentações teóricas dos algoritmos usados para o treinamento da rede neural utilizada no descritor de aparência que o aluno utilizou durante o período do Projeto de Fim de Curso.

2.5 MOTION ANALYSIS AND RE-IDENTIFICATION SET *DATASET*, MARS

Os bancos de imagens utilizados no processo de treinamento de RNA através do método de *cosine metric learning*, têm fundamental importância na desempenho das funções de reidentificação de objetos, como pessoas. Os autores Zheng *et al.* (2016) relatam que esse *dataset* é uma extensão do Market-1501, que também é um banco de imagens construído para o propósito de treinamento de FVENN para reidentificação de pessoas. Os autores apresentam que para a construção desse banco de imagens foram usadas seis câmeras sincronizadas de resoluções diferentes que cobriam uma mesma área da universidade de Tsinghua e 1261 pedestres. Dentro desse *dataset*, cada pedestre conta com uma sequência de imagens coletadas em um trecho por onde essa pessoa efetua um deslocamento. As sequências de imagens de cada pessoa foram coletadas com pelo menos duas câmeras síncronas das seis utilizadas nesse projeto e no total foram produzidas 1.191.003 marcações de pedestres.

Esse trabalho foi fundamental para o entendimento de como uma rede neural extratora de características de imagens é treinada. Também foi usada a estrutura de informações desse *dataset* na criação do *dataset* para este trabalho de fim de curso.

2.6 VISDRONE-MULTI OBJECT *CHALLENGE* E *DATASET*

O trabalho (ZHU *et al.*, 2021) descreve uma competição que busca estimular o estudo de redes neurais detectoras de objetos e algoritmos de *tracking* para imagens coletadas com VANT. Nessa competição, são disponibilizadas sequências de quadros e anotações (GT, do inglês *ground truth*) dos objetos de interesse nesses quadros. Entre as tarefas desse desafio proposto estão: detecção de objetos em quadros estáticos e vídeos, rastreamento de objetos únicos, rastreamento de múltiplos objetos e contagem em multidão (ZHU *et al.*, 2021).

As imagens desse banco foram coletadas com diferentes equipamentos, em diferentes condições climáticas e em múltiplas cidades e regiões rurais a fim de prover um universo de propriedades diferentes e tornar as RNAs treinadas com este banco mais generalistas.

Os bancos de imagens disponibilizados pelos responsáveis do VisDrone *challenge* são divididos em duas categorias principais: *single object tracking* ou *multi object tracking*. Nesse projeto, foram utilizados 56 vídeos coletados por VANTs em centros urbanos com marcações divididas em 10 classes, sendo elas: pedestre, pessoa, carro, van, ônibus, caminhão, motocicleta, bicicleta, triciclo com toldo e triciclo. Em média, cada sequência de quadros usada nesse projeto conta com 400 quadros.

As imagens desse *dataset* apresentaram as propriedades do cenário de aplicação do código de *tracking* em desenvolvimento. Logo, essas imagens serviram como base para o treinamento da RNA da ferramenta DAD responsável por extrair os vetores de características.

2.7 HIGHER ORDER METRIC FOR EVALUATING MULTI-OBJECT TRACKING, HOTA

Esse trabalho apresenta o desenvolvimento de uma métrica para avaliar performances de algoritmos MOT. Os autores relatam que em outras métrica utilizadas para avaliar esses tipos de códigos usadas anteriormente à criação da métrica Acurácia de Rastreamento de Ordem Superior (HOTA, do inglês *Higher Order Tracking Accuracy*), como MOTA, IDF1 e Track mAP, apresentavam limitações e o HOTA foi projetado para suprir essas deficiências.

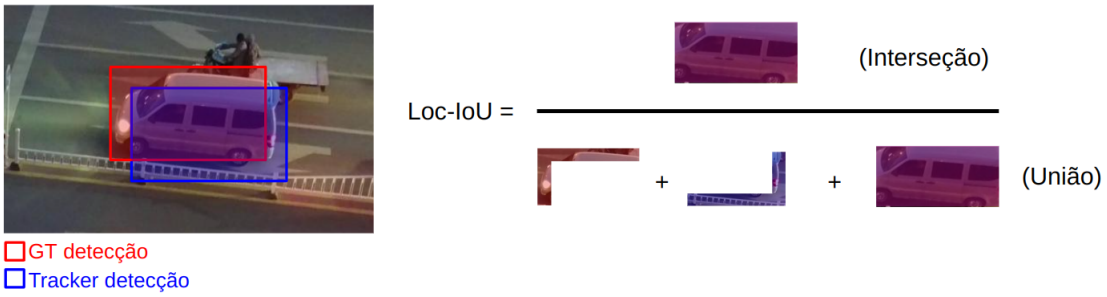
Na sequência, os autores apresentam as definições de *Multi Objetc Tracking* e como as marcações de *ground truth* são tratadas pelos algoritmos de avaliação de desempenho. Após essa apresentação, ele comenta como essas outras métricas efetuam a avaliação dos algoritmos de rastreamento.

Depois, os autores explicam como o HOTA faz uso de três submétricas para gerar uma média geométrica e avaliar o algoritmo de MOT. Essas submétricas são: de detecção, de associação e de localização.

HOTA: Submétrica de localização

O valor da submétrica de localização vem da área de interseção entre a área coberta pela *Bounding Box* (BB) das marcações do *Ground Truth* (GT) e pela área das detecções, BB, resultante do código de *tracking*. Em seguida, para obter o valor dessa submétrica, essa interseção é dividida pela soma da área desta interseção e as áreas das duas regiões que não são comuns. Este resultado é chamado de Localização de Interseção sobre União (Loc-IoU, do inglês *localization intersection over union*) e a Figura 2 apresenta como é feito o cálculo para obter esse valor.

Figura 2 – Cálculo do valor de Loc-IoU



Fonte: Original do autor

Quanto mais próximo de 1 o valor de Loc-IoU ficar, melhor é o desempenho de localização do código de tracking avaliado. É usada uma submétrica de precisão geral de localização, LocA, que calcula a média dos Loc-IoU em todos os pares de detecção e *ground truth*, que corresponde ao conjunto Verdadeiros Positivos (TP, do inglês *true positives*). Esse valor de LocA é representado na equação (1). Nesta equação o valor de $|TP|$ representa a quantidade de elementos neste conjunto.

$$LocA = \frac{1}{|TP|} \sum_{c \in TP} Loc\_IoU(c) \tag{1}$$

Conclui-se que essa métrica fica fortemente vinculada com o desempenho da rede de detecção usada no algoritmo de MOT.

HOTA: Submétrica de detecção

A Interseção sobre União da Detecção (Det-IoU, do inglês *detection intersection over union*) apresenta a precisão entre o cruzamento do conjunto de todas as detecção do *ground truth* e detecções do *tracker* observadas. Essa análise é feita com valores diferentes de Loc-IoU, que pode variar entre 0 e 1, e que tem a função de um *threshold*. Por exemplo, é definido o valor de Loc-IoU $> 0,6$ entre detecções e GT e, em seguida, é contabilizado quantas detecções atendem esse requisito. Porém, nessa análise é notado que uma área de detecção do *tracking* pode se sobrepor a mais de uma área de detecção GT. Logo, é usado o *Hungarian algorithm* para estabelecer correlação de um-para-um entre detecções do *tracking* e do GT. Nesta submétrica, cada par entre esses dois conjuntos resultante desse algoritmo, é incluído no conjunto TP. As combinações entre detecções e GTs que atendem o valor de Loc-IoU e não fazem parte do conjunto de TP são chamadas de Falsos Positivos (FP, do inglês *false positive*), e as detecções de GT sem correspondência com uma detecção do *tracker* são consideradas Falsos Negativos (FN, do inglês *false negative*). O cálculo da Det-IoU é descrito na equação

(2).

$$Det\_IoU = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{2}$$

Nota-se uma semelhança entre as equações de Det_IoU e Loc-IoU, nas quais um valor de interesse é dividido pela soma de todos os valores encontrados. Nesse trabalho estudado sobre o HOTA é substituída a nomenclatura IoU por A, desta forma a submétrica passa a ser chamada de *DetA*.

$$DetA = Det\_IoU \tag{3}$$

HOTA: Submétrica de associação

A associação mede o quão bem um algoritmo de *tracking* atribui um ID a um objeto que se movimenta na sequência de quadros. Assim como as submétricas descritas acima, a associação também usa valores de IoU como *threshold* e *Hungarian algorithm* para determinar pares de associações verdadeiros positivos (TPA, do inglês *true positive associations*), associações falsos positivos (FPA, do inglês *false positive associations*) e associações falsos negativos (FNA, do inglês *false negative associations*). A Interseção sobre União da Associação (Ass-IoU, do inglês *association intersection over union*) apresentada na equação (4), pode ser calculada de maneira semelhante a vista anteriormente quando os valor de casos TPA é dividido pela soma dos casos todos conjuntos FPA, FNA e TPA.

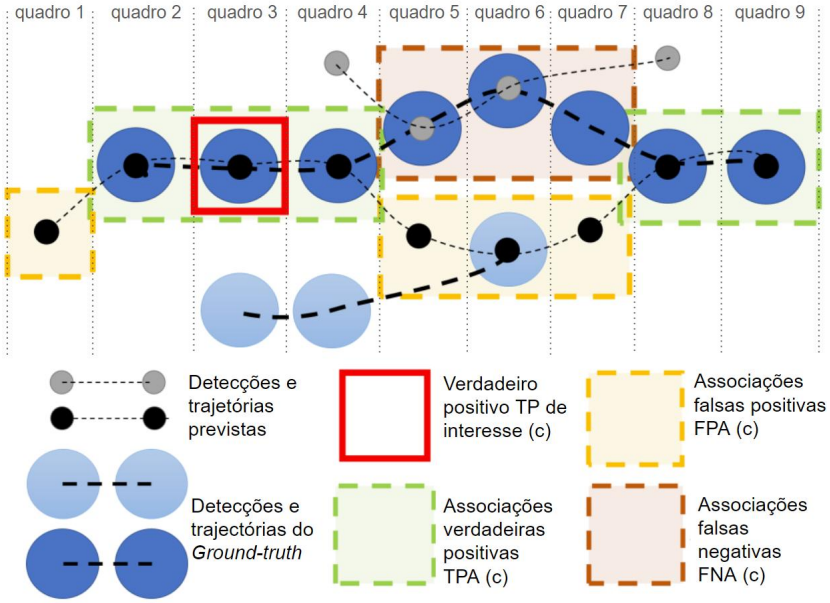
$$Ass - IoU = AssA = \frac{|TPA|}{|TPA| + |FNA| + |FPA|} \tag{4}$$

A Figura 3 ilustra casos de FPA, TPA e FNA em um rastreamento de objetos em uma sequência de quadros. Nesta Figura nota-se no quadro 1 que o GT apresenta uma presença de um objeto anotado porém neste exemplo este objeto não é detectado representando um caso de FPA. No quadro 2 da Figura 3 pode-se ver um exemplo de caso que o *tracker* detectou um objeto que também foi anotado no GT. Já no quadro 5, é apresentado um caso de falha de rastreamento no qual o *tracker* atribui o mesmo ID para objetos diferentes de acordo com a marcação do GT, representando um caso de FNA.

A equação (5) define o quão bem os rastreamentos de objetos estão sendo feitos.

$$\begin{aligned} AssA &= \frac{1}{|TP|} \cdot \sum_{c \in TP} Ass\_IoU(c) \\ &= \frac{1}{|TP|} \cdot \sum_{c \in TP} \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \end{aligned} \tag{5}$$

Figura 3 – Como a submétrica de associação identifica os casos de TPA, FPA e FNA



Fonte: Luiten *et al.* (2020)

HOTA: Cálculo da métrica HOTA a partir das pontuações das submétricas

Os autores argumentam que as submétricas apresentadas são muito importantes para a análise de desempenho de um algoritmo de rastreamento de objetos, porém, é conveniente definir uma métrica que resume as características dessas submétricas de localização, detecção e associação. Essa métrica é o HOTA, que combina estas três sub métricas da seguinte forma:

$$\begin{aligned} HOTA_\alpha &= \sqrt{DetA_\alpha \cdot AssA_\alpha} \\ &= \sqrt{\frac{\sum_{c \in TP_\alpha} AssA\_IoU_\alpha(c)}{|TPA_\alpha| + |FNA_\alpha| + |FPA_\alpha|}} \end{aligned} \tag{6}$$

$$\begin{aligned} HOTA &= \int_{0<\alpha\leq 1} HOTA_\alpha \\ &\approx \frac{1}{19} \cdot \sum_{\alpha=0,05;\alpha+=0,05}^{0,95} HOTA_\alpha \end{aligned} \tag{7}$$

Nas equações (6) e (7), o valores de α substitui o valor de *Loc_IoU* que representam a área de interseção entre detecções e GT e no final é calculada a média geométrica das pontuações das detecções *DetA* e *AssA*. A razão de usar a média geométrica é a ponderação adequada entre detecções e associações e garantir que se o valor de alguma dessas submétricas vá para zero, o valor de HOTA também irá para zero.

Os autores ainda finalizam o texto apresentando: como comparar algoritmos de *tracking* de múltiplos objetos usando as submétricas HOTA; uma comparação mais detalhadas com da métrica HOTA e outras métricas e; por fim, ele compara alguns algoritmos de *tracking* usando a métrica apresentada.

O trabalho (LUITEN *et al.*, 2020) teve grande relevância no trabalho de fim de curso do aluno, pois a partir do conhecimento e interpretação dos valores da métrica HOTA e suas submétricas, o aluno pôde julgar e identificar melhorias entre as versões do código de MOT e avaliar o quanto a adição de uma *feature* nova trouxe melhorias para o desempenho do MOT.

3 DESENVOLVIMENTO

Neste capítulo são apresentadas as atividades que o aluno realizou durante o período deste trabalho e como a fundamentação teórica foi aplicada. Inicialmente, é exposto como foram feitos o mapeamento e a seleção dos algoritmos de rastreamento de múltiplos objetos. Em seguida, é feito um relato de quais foram as hipóteses e os fatores que as sustentam em relação a o que poderia ser feito para obter um melhor desempenho na função de reidentificação de objetos do algoritmo. Na sequência, há uma descrição de como o aluno efetuou o treinamento da FVENN e quais foram os fatores que levaram-no a optar por criar um banco de imagens que seria usado para o treinamento da rede. Após essas descrições, ainda nesta seção de desenvolvimento, é apresento como foi feita a limpeza do banco de imagens criado e como foi desenvolvida a parte de integração da ferramenta descritora de aparência aperfeiçoada pelo aluno no algoritmo de MOT que a empresa desenvolveu. E, por último, é feita uma breve descrição de como este código de MOT funciona e em que ponto do código a ferramenta aperfeiçoada pelo aluno está localizada.

3.1 ESCOLHA DE ALGORITMO PARA TESTE

Após a ambientação do aluno no cenário de estratégias e métricas de algoritmos de MOT, foi feita a escolha da abordagem DeepSORT. Esta escolha foi feita baseada nos requisitos e limitações da aplicação abordada na Seção 1.1. Alguns dos requisitos foram: documentação disponível; resultados de métricas de desempenho parecidos com resultados de algoritmos estado da arte; *pipeline* modular; possibilidade de implementação em linguagem C++ para programação embarcada; e atender requisitos de tempo real de inferências de rastreamento em um hardware limitado.

Vale ressaltar que o aluno, durante as atividades relacionadas a este trabalho, não fez o processo de embarcar nenhum código de MOT em um hardware de aplicação. Porém, todo o desenvolvimento deste trabalho foi feito usando um hardware superior, *desktop*, porém todas as atividades focaram no objetivo de embarcar o algoritmo de MOT em um hardware de aplicação.

Dentro dos exemplos mapeados de códigos baseados na estratégia DeepSORT, foi eleito o código *yolov4-deepsort*, detalhado em (GUY, 2021). Esse código foi escolhido pela sua disponibilidade na plataforma *Google Colab*, pelo seu número de usuários, representado pela pontuação de *Stars* superior a mil pontos na sua página no *GitHub*, pela sua ampla documentação de uso e possibilidade de customização para demandas específicas.

Seguindo a abordagem DeepSORT, esse código também já continha uma rede de detecção de objetos *Yolo V4* (BOCHKOVSKIY; WANG; LIAO, 2020) com as classes de interesse para este projeto. Apesar de já estarem disponíveis versões desta rede

neural de detecção mais atuais durante o desenvolvimento deste trabalho, optou-se manter a versão anterior já integrada no código, visto que a mudança desta rede não necessariamente alteraria a funcionalidade alvo do aperfeiçoamento deste trabalho, que era a reidentificação de objetos.

Apesar de a rede inicial de detecção de objetos ter um papel fundamental no desempenho de um algoritmo de *tracking* (WOJKE; BEWLEY; PAULUS, 2017), A versão desta rede teve pouca importância para no processo de melhoria da função de reidentificação, pois essa função, dentro da rotina do DeepSORT, ocorre após o processo de detecção.

O *dataset* de imagens usado para treinar a rede neural de detecção de objetos foi a COCO (LIN *et al.*, 2014) e durante as inferências essa rede foi parametrizada para detectar somente pedestres e carros.

Inicialmente, foi usada a plataforma *Google Colab* para executar o algoritmo, mas posteriormente foi necessário executar esse código localmente a fim de obter valores de frequência de inferência em um hardware conhecido e de explorar ferramentas que auxiliariam no processo como GIT e extensões do ambiente de desenvolvimento integrado usado.

3.2 LEVANTAMENTO DE HIPÓTESES

A rede neural usada no DAD do algoritmo escolhido como modelo de DeepSORT foi treinada com o *dataset* MARS disponibilizado em (ZHENG *et al.*, 2016). Dessa forma, ela foi treinada para receber imagens de objetos somente da classe pedestre, em uma perspectiva horizontal, e as imagens desses objetos contêm mais detalhes dos mesmos, pois foram coletadas a uma distância menor. A Figura 4 ilustra três imagens do *dataset* MARS que contêm as características citadas. Porém, essas características das imagens não são observadas na aplicação alvo do algoritmo de MOT deste trabalho.

O objetivo deste algoritmo é fazer *tracking* de múltiplos objetos de múltiplas classes em sequências de quadros capturados por VANTs. Nesse cenário, as imagens dos objetos de interesse tendem a ser capturadas de uma perspectiva superior, ter menor resolução de detalhes e conter mais classes além da classe pedestre. A Figura 5 ilustra três imagens dos objetos com as características encontradas no cenário de aplicação.

Baseado na diferença entre as características das imagens do *dataset* usado para o treinamento da rede neural do DAD e do cenário de aplicação deste algoritmo, foi levantada a hipótese de que se essa rede neural for treinada com um banco de imagem com características mais próximas às esperadas, o processo de reidentificação de objetos de interesse tende a ter melhor desempenho.

Figura 4 – Imagens do *dataset* MARS

Fonte: Zheng *et al.* (2016)

Figura 5 – Imagens do *dataset* de imagens criado

Fonte: Original do autor

3.3 CÓDIGO USADO PARA TREINAMENTO DA REDE NEURAL EXTRATORA DE VETORES DE CARACTERÍSTICA

Após o levantamento da hipótese citado na Seção 3.2, deu-se o início ao mapeamento de códigos disponíveis de *metric learning* para o treinamento da FVENN.

Para este treinamento foi selecionado o código (WOJKE, 2018), que aplica a abordagem do trabalho (WOJKE; BEWLEY; PAULUS, 2017) (WOJKE; BEWLEY; PAULUS, 2017). Esse código inicia o modelo de uma RNA baseado em Redes Residuais Amplas (*WRN*, do inglês *wide residual networks*) e, em seguida, é feito o treinamento usando o *dataset* disponibilizado. Esse código foi implementado na linguagem *Python*

e usa a biblioteca *TensorFlow* para iniciar o modelo da RNA e fazer o treinamento da mesma. Durante esse treinamento, é usada a ferramenta *TensorBoard* para o usuário acompanhar os valores de acurácia.

Durante o desenvolvimento deste trabalho, esse código foi usado com diferentes *datasets* e foram geradas diferentes versões de redes neurais extratoras de vetores de características.

Pesquisa sobre *datasets*

Visando o objetivo de fazer o treinamento de uma RNA usando a abordagem de consultas simples na vizinhança próxima (do ingles, *simple nearest neighbor queries*) (WOJKE; BEWLEY, 2018) e a utilização de um universo de imagens com as características que seriam encontradas no cenário de aplicação, foi feita uma pesquisa de *datasets* públicos que atendessem essa demanda. As características do cenário de aplicação são mencionadas na Seção 3.2 e são resumidas nos seguintes tópicos:

- perspetivas superiores;
- nível do detalhamento dos objetos de interesse reduzido; e
- carros e pedestres como objetos de interesse.

Além desses atributos, o *dataset* usado para o treinamento deveria estar em uma estrutura de organização compatível com o *cosine_metric_learning*, algoritmo usado para o treinamento da rede neural da ferramenta DAD.

Os principais *datasets* usados para o treinamento da rede neural na ferramenta de reidentificação de pedestres são: CUHK03, Market1501 e MARS. Já para reidentificação de veículos são: VRID-1, VeRi-776 e VehicleID (LI; ZHOU, 2019). Neste trabalho, foram mapeados os seguintes *datasets*: MARS, Market1501, VeRi-776 e VisDrone, porém, apesar de eles serem avaliados por conterem classes de objetos de interesse, pedestres e veículos, nenhum deles continha todos os pré-requisitos mencionados anteriormente. A coleção VisDrone apresentou maior quantidade de propriedades desejadas, como resoluções e perspectiva dos objetos de interesse e diferentes classes de objetos, porém a estrutura de dados que esta coleção apresenta não é compatível com o *script* usado para o treinamento da RNA.

3.4 CRIAÇÃO DE *DATASET*

Devido à escassez de *dataset* disponível com as características de interesse do projeto, comentada na Seção 3.3, foi decidido criar um *dataset* de imagens novo. A intenção era usar esse novo *dataset* no algoritmo de *cosine_metric_learning* (WOJKE, 2018), portanto este conjunto de imagens deveria conter a estrutura de dados compatível com esse algoritmo de treinamento escolhido. A Figura 6 ilustra dois conjuntos de

imagens representando duas classes diferentes de pessoas. Nota-se que essas duas classes são consideradas distintas, apesar de ambas serem de imagens de pessoas, pois contêm atributos diferentes, como cores das roupas e silhuetas.

Figura 6 – Exemplo de duas classes distintas do *dataset* criado.

Fonte: Original do autor

Derivação das imagens para a formação do *dataset* criado

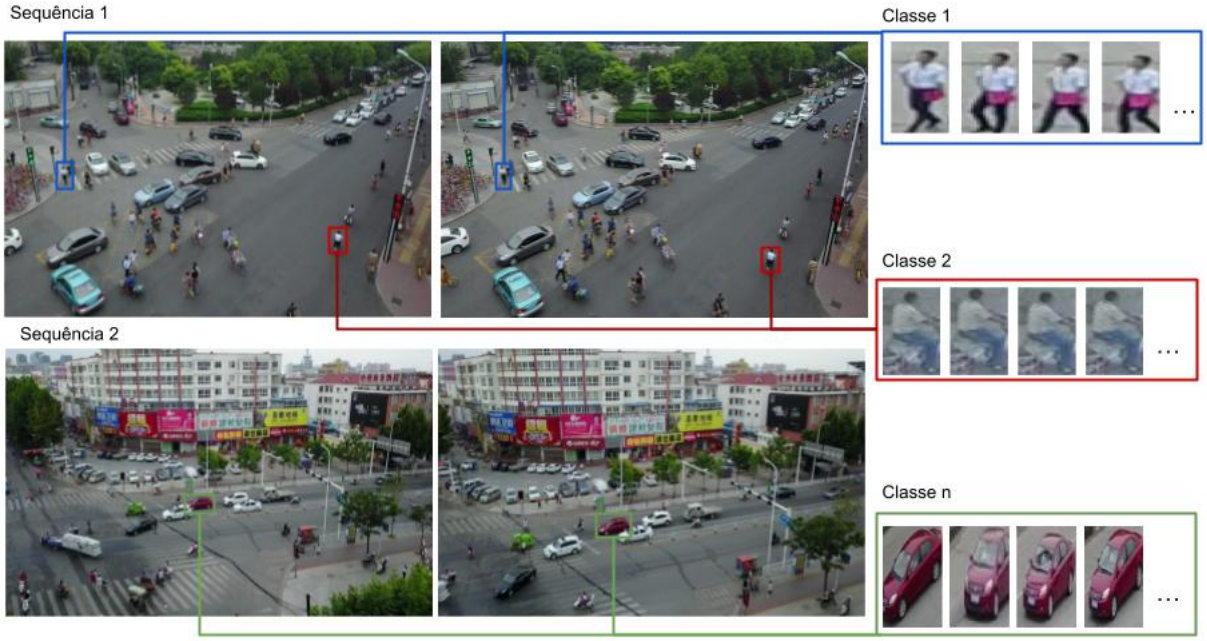
Para se obter uma sequência de imagens de um mesmo objeto, foi usada a sequência de quadros do banco de imagens VisDrone (ZHU *et al.*, 2021). Esse banco de imagens, como visto na Seção 2.6, contém uma sequência de quadros e anotações de BB de cada objeto no quadro. Foi utilizado um algoritmo *parser* (A) para fazer a leitura dessas BBs e fazer o recorte das imagens dos objetos no quadro. A Figura 7 apresenta o como as imagens dos objetos da sequência de quadros do VisDrone foram extraídas para compor as imagens do *dataset* criado.

Devido à compatibilidade com as dimensões da camada de entrada da rede neural treinada com esse conjunto de imagens, as imagens após o recorte foram redimensionadas para 256 *pixels* $\times$128 *pixels* (altura e largura).

É importante ressaltar que as sequências de quadros usadas para esse processo estavam na seção *train* dentro do *dataset* VisDrone. Já a sequência de quadros usados para avaliação de desempenho dos algoritmos com a DAD estava na seção *test* do *dataset*, assim assegurando que os dados de treinamento da rede neural extratora de FV são diferentes dos dados de teste do desempenho do algoritmo com essa rede neural.

Baseado na quantidade mediana de imagens por classe do banco de imagens MARS, de 586 imagens por classe, foi decidido que classes com menos de 100 ima-

Figura 7 – Processo de extração das imagens dos objetos no banco de imagens VisDrone.



Fonte: Original do autor

gens dentro do *dataset* criado seriam excluídas, a fim de torná-lo mais coerente com a estrutura do banco de imagens original para o qual o algoritmo de treinamento *cosine_metric_learning* foi projetado.

Devido ao fato de a rede neural de detecção de objetos estar configurada para detectar apenas objetos das classes pedestres e carros, foram excluídas classes que não eram carros ou pedestres do banco de dados criado. Ao término desse processo, foi gerada a primeira versão do *dataset* criado, que contou com uma coleção de 3970 classes de imagens com mediana de 213 imagens por classe.

3.5 ANÁLISE E LIMPEZA DO *DATASET* CRIADO

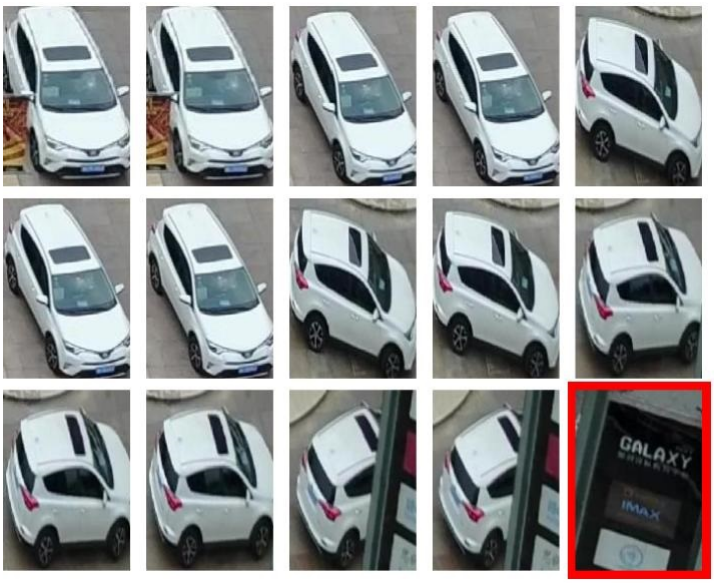
Após a criação da primeira versão do *dataset*, treinamento de uma FVENN com este *dataset* e implementação desta rede neural na ferramenta de DAD no algoritmo escolhido de DeepSORT, foi feita a avaliação de desempenho desse algoritmo. Essas atividades e seus resultados são apresentados na seções 4.1 e 4.2, respectivamente.

Devido ao resultado de desempenho de rastreamento do código inferior ao desempenho da versão padrão desse código, foram levantadas hipóteses de fatores que poderiam estar prejudicando-o e o que poderia ser feito para melhorá-lo. Essas hipóteses são apresentadas na Seção 3.6.

3.6 HIPÓTESE SOBRE IMAGENS QUE PREJUDICAM O TREINAMENTO DA REDE NEURAL DA FUNÇÃO DAD

Foi levantada a hipótese de que havia algumas imagens dentro do *dataset* que não apresentavam características gerais de suas respectivas classes e por conta disso, prejudicavam o treinamento das RNAs extratoras de vetores de características. Uma provável origem desse tipo de imagem é o fato de o *groundtruth* do *dataset* VisDrone ter anotações de objetos parcialmente ou completamente obstruídos. Assim, há imagens dentro de classes que não contêm as características do objeto de interesse, dado que a visualização apresentada é do objeto que está à frente do objeto de interesse. A Figura 8 apresenta algumas imagens de uma classe da primeira versão do *dataset* criado e é destacada uma imagem na qual o objeto de interesse da classe é totalmente obstruído.

Figura 8 – Exemplo de classe do *dataset* criado com objeto de interesse obstruído.



Fonte: Original do autor

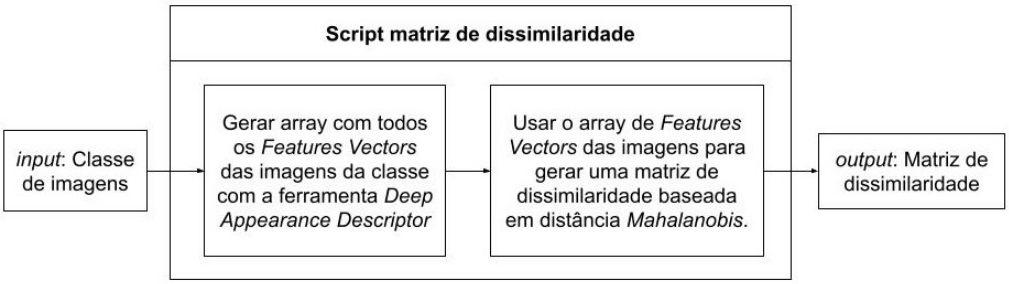
A fim de identificar essas imagens dentro da primeira versão do *dataset* criado, foi feito um estudo das distâncias de Mahalanobis entre os vetores de características, FV, das imagens dentro de uma mesma classe. Esses vetores foram gerados pela rede neural da ferramenta *Deep Appearance Descriptor*, DAD.

Análise distâncias entre vetores de características de imagens dentro da mesma classe

Baseado na hipótese apresentada na Seção 3.6, foi desenvolvido um *script* que tinha como entrada uma classe de imagens do *dataset* criado e tinha como saída uma

matriz quadrada de dissimilaridade baseada em distâncias de Mahalanobis entre os FV de todas as imagens da classe de entrada. A Figura 9 apresenta um diagrama com as sequências das atividades desse *script*.

Figura 9 – Etapas do funcionamento do *script* que constrói matriz de dissimilaridade



Fonte: Original do autor

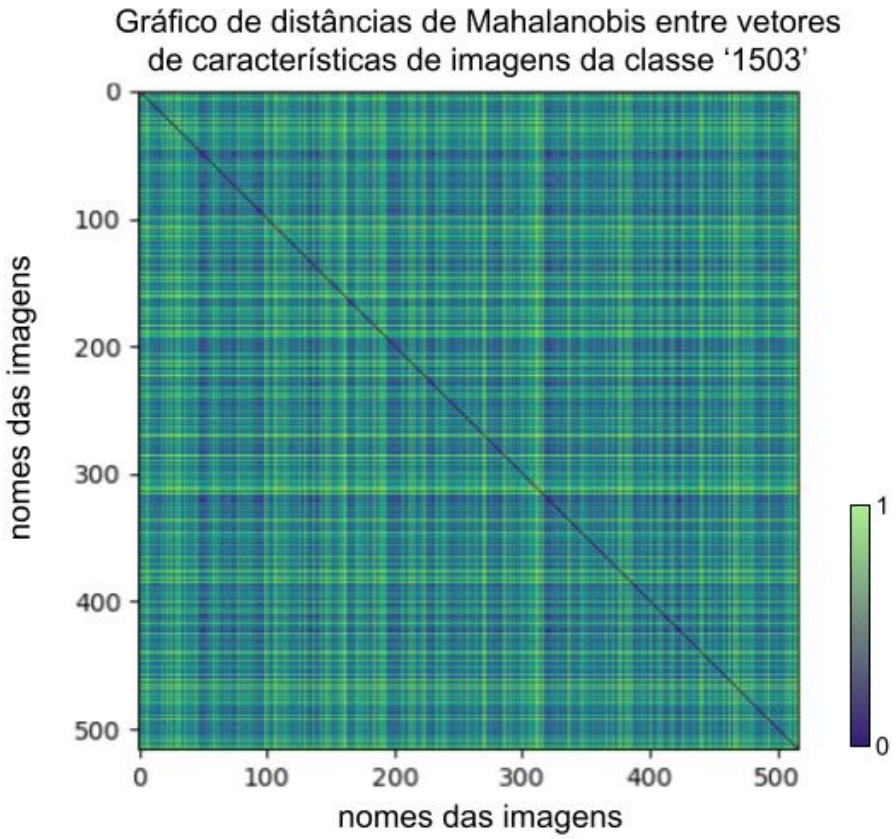
Durante a execução do *script* de dissimilaridade, foi encontrado um problema de execução. Em algumas classes, a quantidade de imagens era consideravelmente maior que a média. Nessas classes, o algoritmo de dissimilaridade, ao gerar a matriz com todas os FV, alocava uma quantidade de memória temporária que era proporcional ao número de imagens multiplicado por 128 (tamanho do FV) multiplicado por 4 *bytes* (quantidade de memória de uma variável *float*). Essa quantidade de memória somada à memória alocada aos processos paralelos sendo executados na máquina acarretou instabilidade durante a atuação do *script*.

A fim de solucionar essa dificuldade, foi proposta uma transformação nos valores da FV das imagens para representações numéricas que necessitam menos memória. Foi observado que todos os valores dos vetores estavam dentro do intervalo de $-0,151435$ a $+0,4569$. Foi proposto que se limitasse a representação desses valores a 4 casas decimais e se multiplicasse por um valor escalar de 10.000, levando a uma representação em ponto fixo. Dessa forma, os valores dos vetores poderiam ser representados pelo tipo de variável *int16* (2 *bytes*), ocupando metade da memória previamente ocupada, e continuariam apresentando boa resolução da informação contida nos vetores de características.

A Figura 10 apresenta um exemplo de matriz de dissimilaridade normalizada de distâncias entre os FVs das imagens de uma classe da primeira versão do *dataset* criado. Essa matriz foi utilizada como tentativa de detectar visualmente imagens com elevado valor de distância de Mahalanobis em uma classe. Na Figura 10, é possível observar fileiras, horizontais e verticais, com tons mais claros representando figuras com elevado valor de distância. Porém, com essa matriz só foi possível fazer uma inspeção qualitativa do padrão de distâncias das imagens de uma classe. Neste exemplo, foi usada a coleção ‘1503‘ pois nela foi identificada uma elevada quantidade de imagens

que distorciam as propriedades gerais da classe.

Figura 10 – Gráfico dissimilaridade



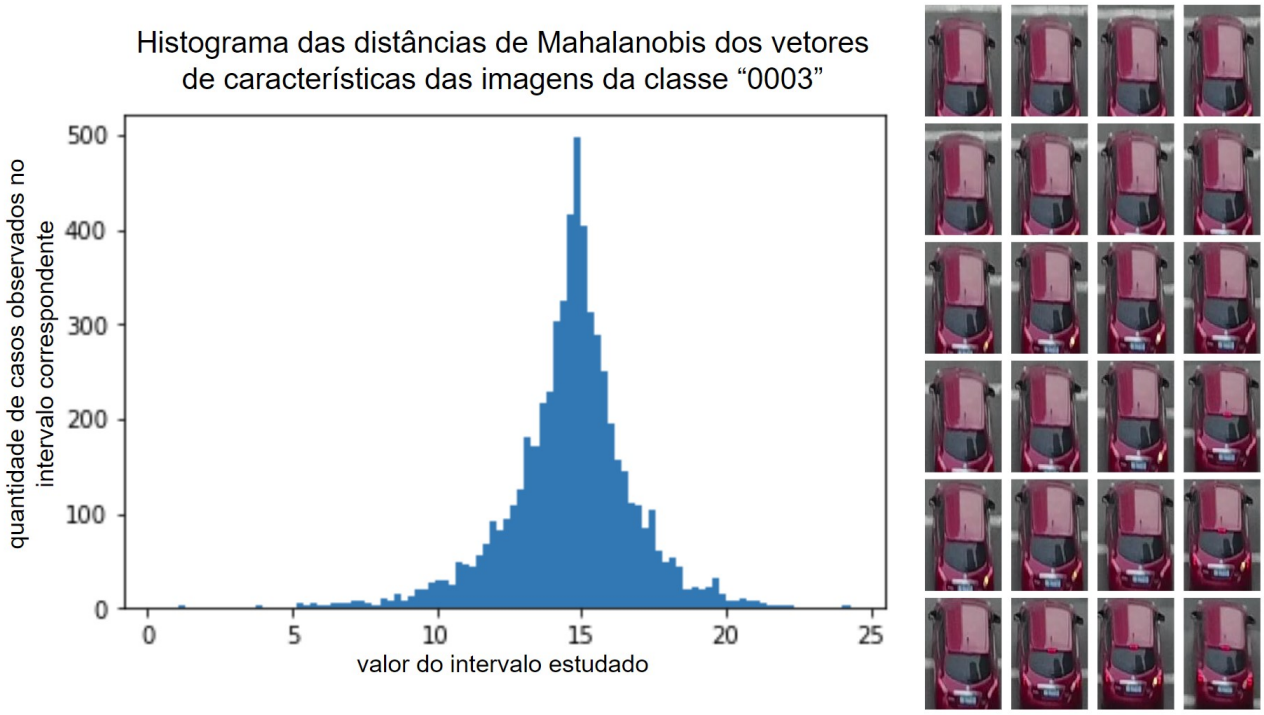
Fonte: Original do autor

Posteriormente a esse processo, foi possível obter todas as matrizes de distâncias de Mahalanobis de todas as classes do banco de imagens criado. Após adquirir estas distribuições de distâncias de cada classe, foi possível fazer a comparação entre coleções com atributos bem definidos, chamadas de classes bem comportadas, com coleções com atributos não tão bem definidos, nomeadas classes mal comportadas. Na comparação entre essas duas categorias de arranjos, foi notado que classes mal comportadas tendem a ter um valor de moda e de desvio-padrão no histograma de distâncias maior que as classes bem comportadas.

A Figura 11 apresenta o histograma das distâncias de Mahalanobis da classe '0003' que tem como valor de moda $14,87$ e uma variância de $5,34$. Já a Figura 12 apresenta o histograma da classe '1873' como moda de $15,74$ e uma variância de $9,86$. A comparação entre estas duas figuras exemplifica essa classificação apresentada no parágrafo anterior. Visualmente as imagens da coleção com maiores valores de moda e variância da distância, Figura 12, apresenta maior discordância das propriedades

de suas imagens, portanto, na comparação entre estas duas coleções, a ‘1873‘ seria classificada como mal comportada e ‘0003‘ bem comportada.

Figura 11 – Histograma de distâncias e imagens da classe bem comportada ‘0003‘



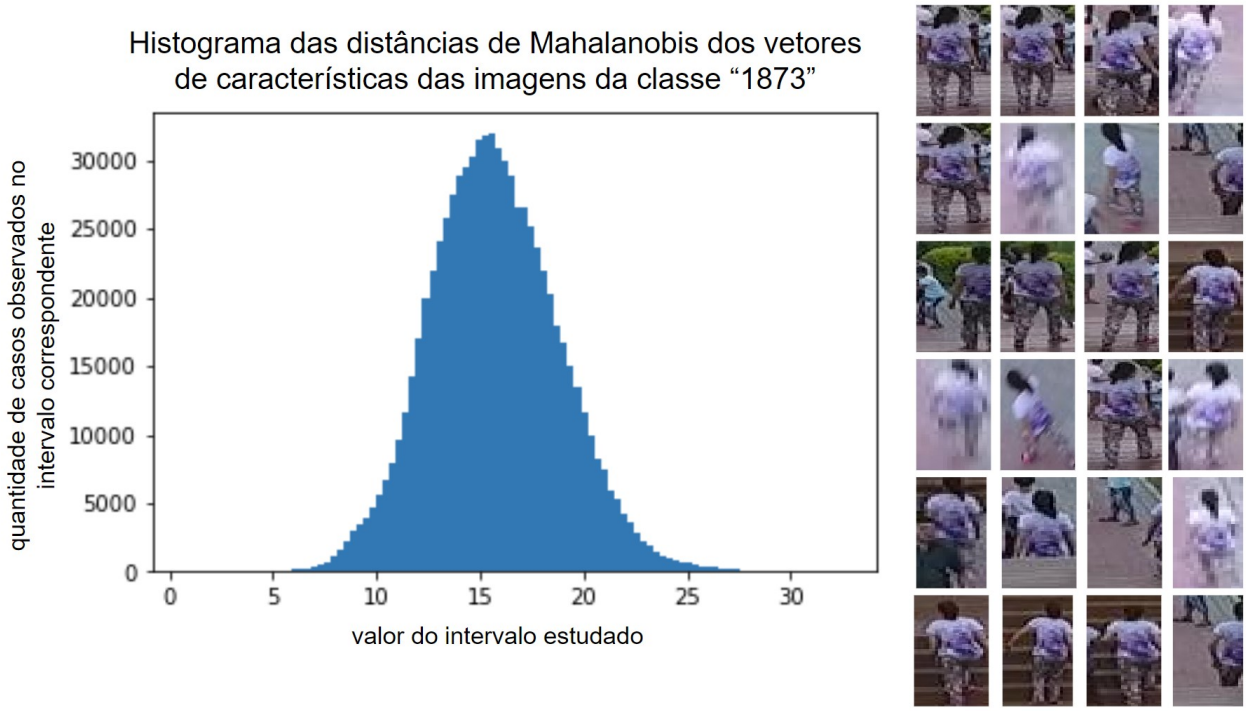
Fonte: Original do autor

3.7 PROCESSO DE LIMPEZA DE BANCO DE DADOS DE IMAGENS CRIADO

Baseado na afirmação que classes mal comportadas tendem a ter valores de moda e desvio-padrão das distâncias de Mahalanobis entre os FV de suas imagens altos, foi feita uma análise para identificar quais seriam essas classes. A Figura 13 ilustra o gráfico cartesiano que correlaciona valores de moda e valores de desvio-padrão das classes de imagens do *dataset* criado e apresenta exemplos de imagens de três classes selecionadas. Dois conjuntos de exemplos são de classes consideradas bem comportadas, classe ‘1555‘ e ‘2539‘, nas quais os valores de desvio-padrão e moda da distância são relativamente baixos. Ao explorar as imagens dessas duas classes selecionadas, nota-se que existe um padrão bem definido das propriedades das imagens. Por outro lado, quando explorado uma coleção com valores de moda da distância e desvio-padrão altos, grupo ‘1503‘, nota-se uma variação maior de propriedades nas imagens desta classe.

O processo de limpeza foi baseado nos valores de desvio-padrão e moda de distância. A Figura 14 apresenta um histograma com os valores dos desvios-padrões

Figura 12 – Histograma de distâncias e imagens da classe mal comportada ‘1873‘



Fonte: Original do autor

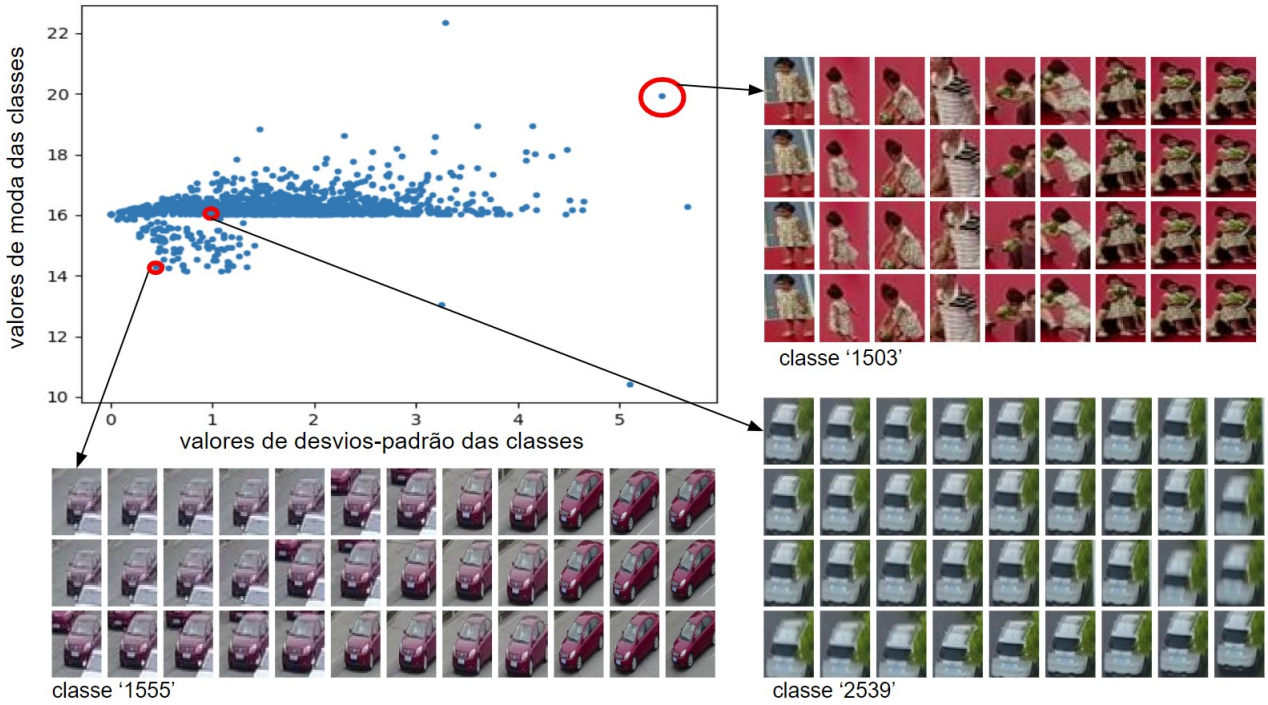
das classes de imagens do banco de imagem. Foi adotado que as classes com valores maiores que duas vezes o desvio-padrão deste histograma teriam que passar pelo processo de limpeza, ou seja, classe com valores de desvio-padrão maior que 3. Após essa consideração, foram selecionadas 100 classes para passarem por um processo de limpeza. O objetivo desse processo foi identificar quais imagens dentro dessas 100 classes consideradas mal comportadas teriam atributos que estariam distorcendo as propriedades da respectiva classe.

Como primeiro passo da limpeza, foi usado um algoritmo para identificar quais imagens dentro das classes selecionadas que tinham a distância média entre as outras imagens da mesma classe maior que uma vez o desvio-padrão da distância média da classe. Após identificar as imagens com essa propriedade, o algoritmo excluía essas imagens do banco de imagens.

O segundo passo de limpeza foi um processo manual de limpeza. Esse segundo passo foi necessário pois após o primeiro passo da limpeza, julgou-se que ainda existia um número relevante de imagens com ausência de propriedades características de suas classes. Essa etapa de limpeza se constituiu em processo manual no qual o autor examinou as imagens restantes das classes mal comportadas, selecionou aquelas com propriedades relevantes para suas classes e removeu as demais do *dataset*.

Uma terceira etapa foi necessária, visto que no final dos dois processos ante-

Figura 13 – Gráfico de moda e desvio-padrão das distâncias de Mahalanobis dos vetores de características de cada classes do banco de imagens criado



Fonte: Original do autor

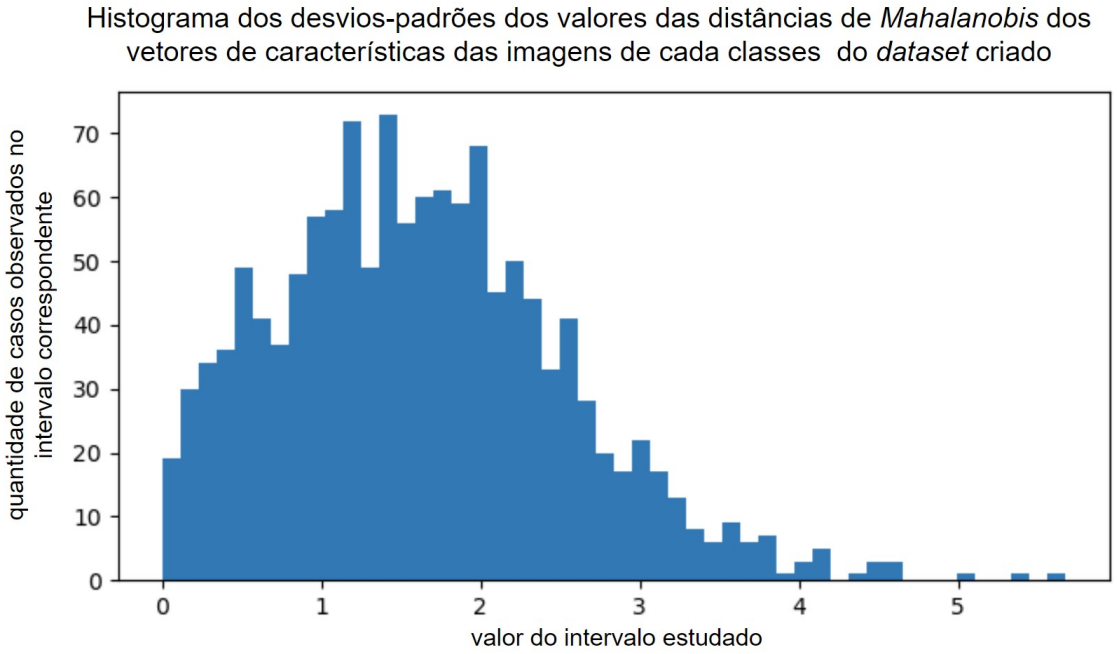
riores algumas classes ficaram com um número de imagens considerado insuficiente. Nessa etapa, todas as classes com menos de 100 imagens foram removidas do *dataset*.

3.8 INTEGRAÇÃO DAS FERRAMENTAS DE *DEEP APPEARANCE DESCRIPTOR* NO CÓDIGO DE *MULTI OBJECT TRACKING* DA EMPRESA

Realizou-se mais uma vez o treinamento da RNA da ferramenta de DAD usando o algoritmo de Wojke (2018). Nesta segunda vez foi utilizada a versão obtida após a limpeza do *dataset* criado. Em seguida, essa RNA foi integrada no código (GUY, 2021) e, mais uma vez, foi feita a avaliação de desempenho entre versões deste código a fim de identificar melhorias nas funções de *tracking*. A avaliação indicou uma melhora nas métricas de desempenho do algoritmo. Esse resultado é detalhado na Seção 4.4.

Após ter sido obtida uma melhora do desempenho de *tracking* no algoritmo descrito em (WOJKE; BEWLEY; PAULUS, 2017) por meio do treinamento da rede neural da ferramenta DAD com o banco de imagens criado filtrado, foi iniciada a etapa de implementação dessa ferramenta no algoritmo de rastreamento de múltiplos objetos desenvolvido pela empresa. Foi implementada no algoritmo de MOT da empresa uma estratégia de reassociação de ID baseada no algoritmo SORT (BEWLEY *et al.*, 2016),

Figura 14 – Histograma de valores de desvios-padrões de todas as classes de imagens do banco de imagens criado



Fonte: Original do autor

porém, além dessa estratégia, foi usado um descritor de aparência baseado no histograma de cores das imagens dos objetos de interesse. Esse descritor de aparência tem como saída um vetor numérico. Esse vetor numérico tem a função de ser mais um parâmetro para a função de reassociação de ID no algoritmo MOT.

O objetivo desta atividade era fazer a substituição da ferramenta de descritor de aparência baseado em histograma de cores pelo descritor de aparência baseado em rede neural, DAD. Além dessa troca, foi feita também a substituição da função que calcula a distância entre vetores de características. Essa última troca de função é detalhada na Seção 3.9.

3.9 ALTERAÇÃO NA FUNÇÃO DE CÁLCULO DA DISTÂNCIA ENTRE VETORES DE CARACTERÍSTICAS NO MOT

Inicialmente no algoritmo de MOT desenvolvido pela empresa, era usada uma função que calculava a distância entre FV baseada na distância de Hellinger. Essa função apresentou bom desempenho quando o FV era extraído com o descritor de aparência baseado no histograma de cores das imagens dos objetos de interesse. Porém, com a troca do descritor de aparência pelo descritor baseado em rede neural, foi considerado também fazer a troca da função de cálculo de distância. Essa consideração veio a partir do estudo do código utilizado anteriormente baseado em

DeepSORT cuja função de cálculo da distância era baseada em distância cosseno (WOJKE, 2018). A equação (8) apresenta como é calculada a função de similaridade cosseno, S_c, e a equação (9) demonstra qual é a relação entre similaridade cosseno e distância cosseno que aqui é representada por D_c. Nas equações (8) e (9), A e B representam vetores de características multidimensionais extraídos com a ferramenta DAD e θ representa o ângulo entre eles. Em (8), A_i e B_i são valores escalares que compõem os vetores A e B, respectivamente. Ainda nessa equação, o valor de n corresponde à dimensão do FV, que na prática correspondeu ao valor 128 que é referente ao número de neurônios de saída na última camada da FVENN (WOJKE; BEWLEY, 2018).

$$S_c(A, B) = cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{8}$$

$$D_c(A, B) = 1 - S_c(A, B) \tag{9}$$

Após essa troca de funções de distância, foram feitas avaliações de desempenho e foi notada uma melhora na métrica de troca de identidades (IDWS, do inglês *identity switch*). Esse resultado indica melhora no processo de reidentificação de objetos (LUITEN *et al.*, 2020).

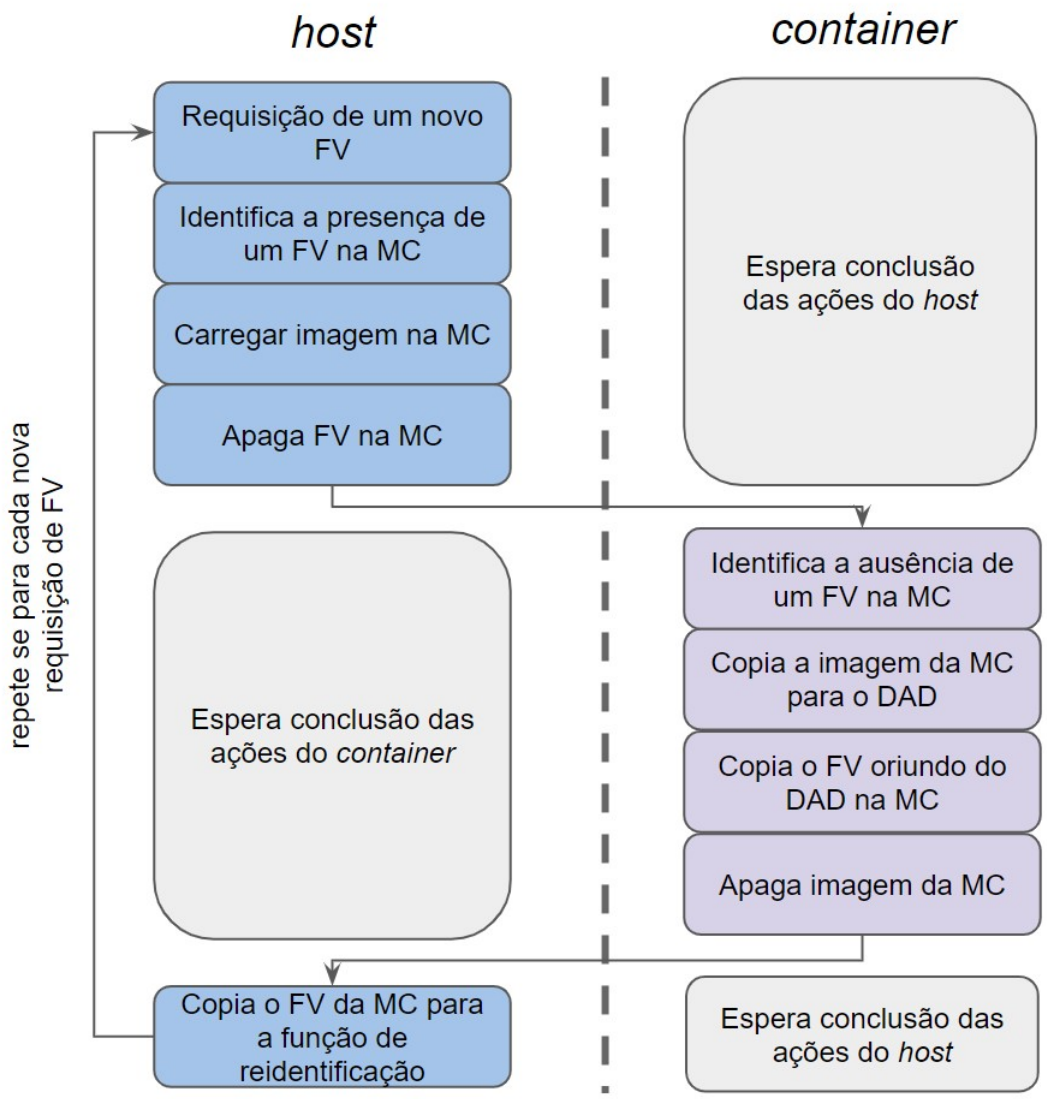
Framework utilizado para integração do DAD no MOT

Durante o processo de integração da ferramenta DAD no código de rastreamento da empresa, o aluno deparou-se com a adversidades de integração de *scripts* em diferentes linguagens de programação. Neste processo de integração, o códigos MOT foi implementado na linguagem *C++* e a ferramenta DAD foi implementada na linguagem *Python*.

Após uma reunião do aluno com colegas da área a fim de discutir qual seria a melhor estratégia de fusão dos códigos em diferentes linguagens, foi concluído que se adotaria a fusão dos códigos por meio do uso de *containers* Docker com memória compartilhada entre o *kernel* Docker e o *Host*. Dessa forma, as funções do DAD implementadas em *Python* seriam executadas por um *kernel* do *container* enquanto as demais funções do MOT seriam executadas normalmente no *Operating System* (OS) do *host* e a troca de informações entre esses dois *kernels* seria feita por memória compartilhada.

A Figura 15 ilustra qual foi a estratégia usada para garantir a troca de informações entre *kernels host* e *kernels container*.

Figura 15 – Diagrama do funcionamento da comunicação *host* e *container*



*FV - *Features Vector* *MC - Memória compartilhada *DAD - Deep Appearance Descriptor

Fonte: Original do autor

Diagrama de sequência do algoritmo de MOT da empresa

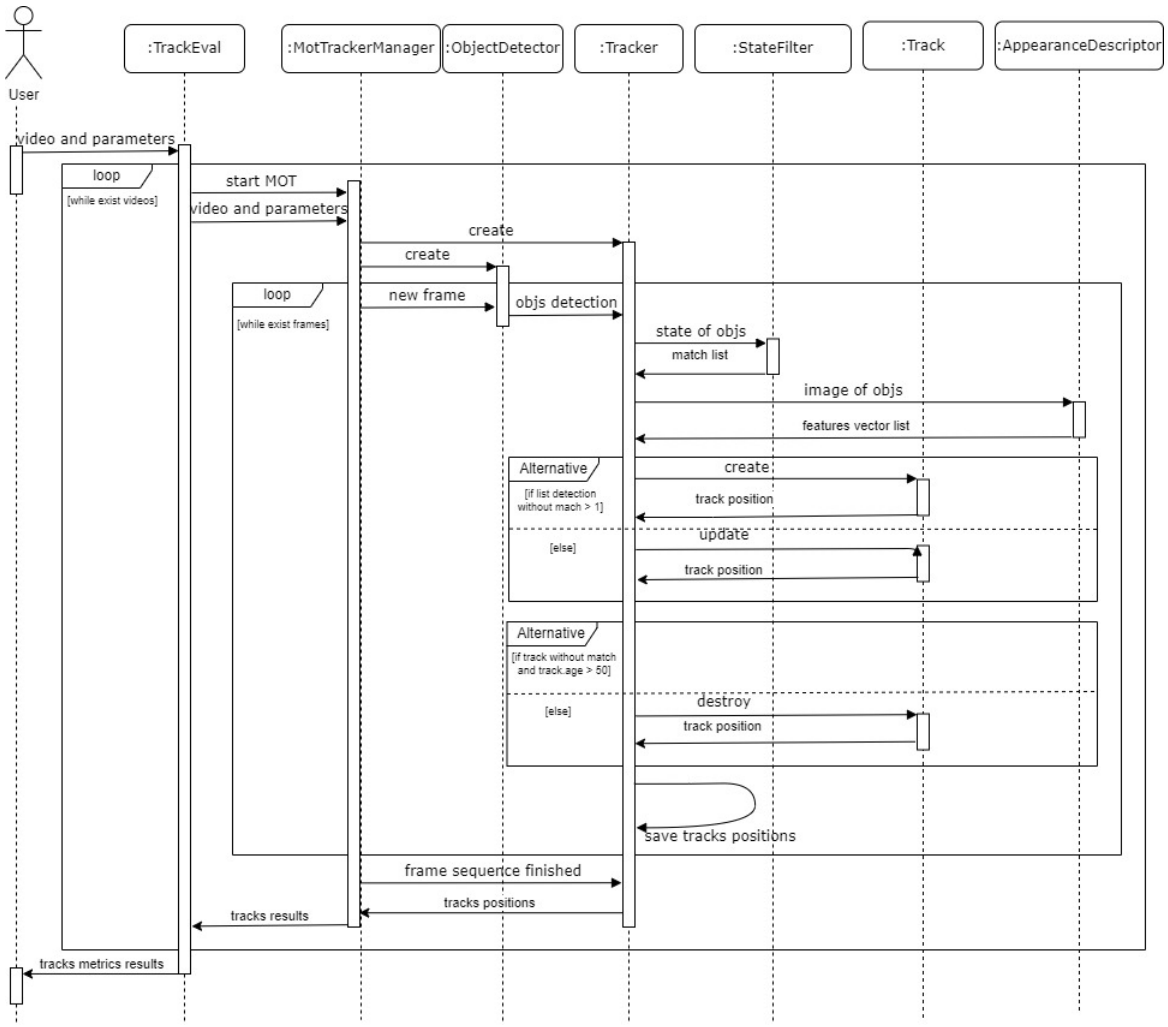
O algoritmo de *tracking* de múltiplos objetos desenvolvido pela empresa tem como entrada uma sequência de imagens e tem como saída as anotações das inferências do *tracker*. Nestas anotações estão as informações de BB e ID de cada objeto de cada frame.

A Figura 16 ilustra o diagrama de sequência de atividades do algoritmo MOT. Esta cadeia de funções se inicia com a entrada de uma sequência de imagens pelo usuário. Em seguida, é iniciado um laço de repetição que tem a duração da quantidade

de imagens da entrada. Para cada quadro, de início é feita uma inferência com uma rede neural detectora de objetos a fim de detectar objetos de interesse na imagem. Em seguida, para cada objeto detectado no quadro, é extraído um vetor de características usando a ferramenta DAD. Na sequência, o algoritmo faz a estimativa da posição de cada ID para o quadro seguinte baseada na velocidade do objeto vinculado ao ID .Subsequentemente, é iniciada a função de associação de IDs com os objetos detectados no quadro atual. Nessa função de associação, são utilizados os vetores de características das detecções e as estimativas das posições dos IDs no quadro atual. Ao término dessa função, nota-se que alguns objetos podem ser associados a IDs já existentes, outros objetos podem não ser associados e alguns dos IDs podem permanecer sem um objeto detectado. Essas três variantes são tratadas de formas diferentes:

- no caso em que o objeto detectado é associado à um ID, o vetor de característica desta detecção é adicionado às informações contidas no ID, consequentemente facilitando a reassociação desse objeto para os próximos quadros. Este caso representa o comportamento esperado no rastreamento de objetos em uma sequência de imagens;
- para o caso de objetos detectados no quadro e não associados a um ID já existente, são criados novos IDs para essas detecções. Este caso representa novos objetos de interesse sendo detectados na sequência de imagens;
- por fim, para quadros cujos IDs não são associados a detecções, é feita a contagem de quantos quadros consecutivos cada ID se encontra sem associação a detecções. Caso esse valor de ‘idade‘ deste ID atinja um valor limiar, o rastreamento é considerado inativo e para de fazer parte da função de reassociação para os próximos quadros. Este caso representa objetos de interesse que não são mais detectados no quadro pois deixaram a região coberta na captura das imagens.

Figura 16 – Diagrama de sequência do algoritmo de MOT



Fonte: Original do autor

4 RESULTADOS

Neste capítulo são apresentados os resultados das principais atividades desenvolvidas no período do projeto de fim de curso. Inicialmente, são expostos os resultados referentes ao primeiro treinamento da RNA da ferramenta de *Deep Appearance Descriptor*. Em seguida, é apresentado o resultado da primeira comparação entre os desempenhos de duas versões do algoritmo *DeepSORT*. A primeira verão é a versão padrão do código disponibilizado on-line e a segunda contém a rede da ferramenta DAD treinada com o *dataset* criado.

Ainda nesta Seção de resultados, é apresentado o resultado da primeira análise das características do *dataset* criado. Na sequência, é exposta uma segunda análise entre os desempenhos das versões do algoritmo DeepSORT. Nesta parte, são introduzidos os resultados de desempenho de uma terceira versão do código cujos pesos da rede extratora de vetores de características foram obtidos após o treinamento com o *dataset* criado e limpo. Por fim, é exibida a comparação entre os desempenhos das versões com e sem DAD do algoritmo MOT desenvolvido pela empresa.

4.1 RESULTADOS PRIMEIRO TREINAMENTO DA RNA EXTRATORA DE CARACTERÍSTICAS COM O *DATASET* CRIADO

Após mapeados os códigos e *datasets* disponíveis, levantadas as hipóteses para melhoria de desempenho das funções de reidentificação de objetos e construída a primeira versão do banco de imagens criado, foi feito o treinamento da RNA com essas imagens. O processo de treinamento assim como o código base usado *metric learning* são descritos com mais detalhes na Seção 3.3.

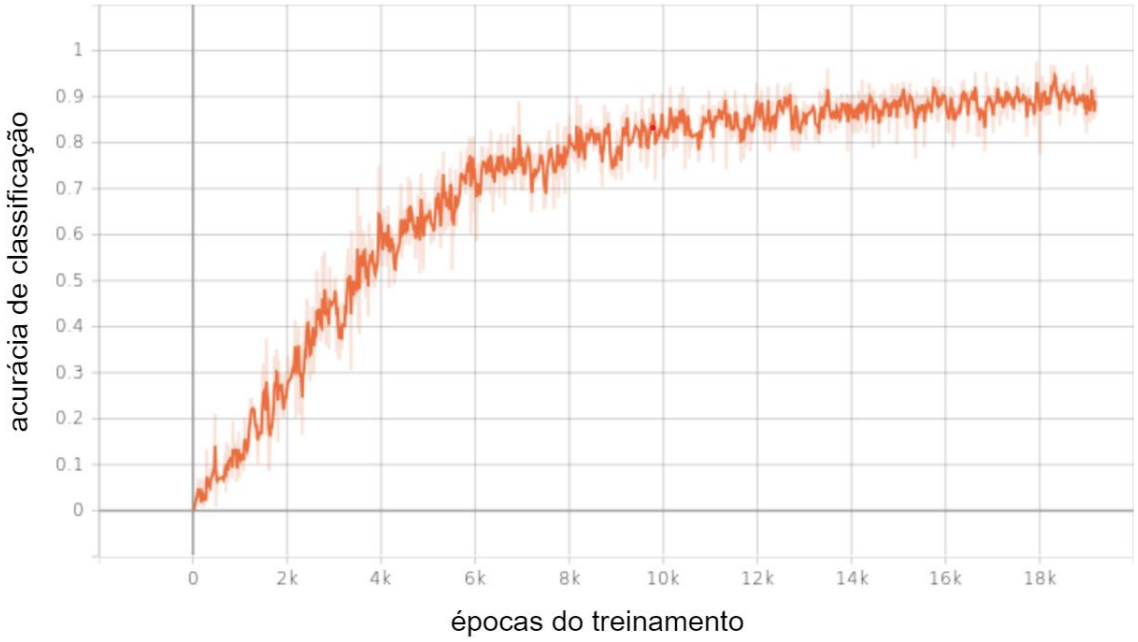
A Figura 17 apresenta a evolução do valor de acurácia durante as épocas do primeiro treinamento da RNA extratora de vetores de características. No total, foram usadas 18958 épocas para o treinamento alcançar a acurácia de $0,89$.

Após este treinamento, foi gerado o arquivo de extensão *.pb*, padrão dos modelos gerados pela biblioteca Tensorflow, com as informações do modelo e o valor dos pesos da RNA recém treinada. Na sequência, este arquivo foi usado no *pipeline* do código DeepSORT a fim de fazer o primeiro experimento com a ferramenta de DAD treinada com um *dataset* personalizado para a aplicação.

4.2 RESULTADO DA PRIMEIRA COMPARAÇÃO ENTRE O DESEMPENHO DE *TRACKING* DA VERSÃO PADRÃO DEEPSORT E DA VERSÃO TREINADA COM O BANCO DE IMAGENS CRIADO

Depois de ter sido concluído o treinamento da FVENN com a primeira versão do banco de imagens criado, os pesos resultantes desse treinamento foram adicionados

Figura 17 – Evolução do valor do parâmetro *classification accurancy* durante o treinamento da FVENN com a primeira versão do dataset criado.



Fonte: Original do autor.

à rotina do código DeepSORT. Em seguida, foi feita uma avaliação de desempenho de duas versões do DeepSORT usando duas sequências de quadros da seção de teste do *dataset* VisDrone. Para fazer essa avaliação, foi utilizado o algoritmo *TrackEval* (JONATHON LUITEN, 2020).

A Tabela 1 apresenta a primeira comparação entre a avaliação das versões padrão do *DeepSORT*, V0, e a versão com os pesos da FVENN resultantes do treinamento. Nesta primeira comparação foram usadas somente as métricas MOTA e MOTP.

Tabela 1 – Primeira análise entre versões de algoritmos DeepSORT.

Versão	**MOTA**	**MOTP**
V0	**0,336289**	**0,796588**
V1	0,334848	0,796225
V0 - DeepSORT com pesos da FVENN padrão		
V1 - DeepSORT com pesos da FVENN modificados		

Fonte: Original do autor.

Nota-se nesta comparação que os resultados das avaliações foram similares, com uma leve redução do desempenho na versão com os pesos da FVENN modificados, V1. Esse resultado foi de encontro ao que se esperava, visto que as imagens usadas para o treinamento dessa RNA compartilhavam as mesmas propriedades das imagens dos objetos de interesse encontrados nas sequências de quadros usadas

para esta avaliação. Por este fato, era esperado que os desempenhos das funções de reidentificação do código fossem melhores e, consequentemente, era esperada uma melhora na pontuação das métricas usadas nesta avaliação, o que não ocorreu.

É importante salientar que não foi encontrado um método que avaliava o desempenho da ferramenta DAD de forma isolada. Logo, a forma encontrada de avaliar possíveis melhorias de desempenho da ferramenta foi através da comparação do desempenho de *tracking* do código como um todo. nesta comparação as únicas alterações entre as versões do DeepSORT foram os pesos da FVENN.

4.3 RESULTADO DA ANÁLISE DO *DATASET* CRIADO

Após ter sido feita a criação do banco de imagens por derivação das imagens do *dataset* VisDrone, como descrito na Seção 3.4, e terem sido obtidos os resultados dos algoritmos de *tracking* com as FVENN obtidas do treinamento com este novo *dataset*, foi levantada a hipótese de existência de imagens dentro deste banco que estariam prejudicando o processo de treinamento dessa FVENN, como detalhado na Seção3.6. Depois de levantada essa hipótese, foi usada a distância de Mahalanobis para a identificação destas imagens que estariam prejudicando o treinamento da RNA. Após a identificação destas imagens concluiu-se que durante a derivação das imagens do *dataset* VisDrone, algumas delas, cujas características não convergiam com as características comuns das figuras da sua classe, foram adicionadas no *dataset*. Essas imagens tornavam a variância e o valor de moda das distâncias de Mahalanobis entre os FVs da mesma classe mais altos, ou seja, as imagens que divergiam das demais da respectiva classe aumentavam a distância entre imagens da mesma classe.

Os principais motivos da presença destas imagens errôneas no banco criado é que nas anotações das BB do VisDrone, objetos completamente ou parcialmente oclusos continuaram sendo delimitados e anotados sendo estimado sua extensão total na área do quadro. Além deste motivo também foi levado em consideração possíveis erros humanos durante a anotação dos quadros.

Concluiu-se, também, nessa análise que usar a distância de Mahalanobis apresentou melhores resultados na função de *clustering* do que os valores de distância euclidiana ou a ferramenta de incorporação de vizinhos estocásticos t-distribuídos, (TSNE, do inglês *t-distributed stochastic neighbor embedding*).

4.3.1 Discussão sobre o resultado do uso das distâncias das imagens para identificação de imagens prejudiciais ao treinamento da FVENN

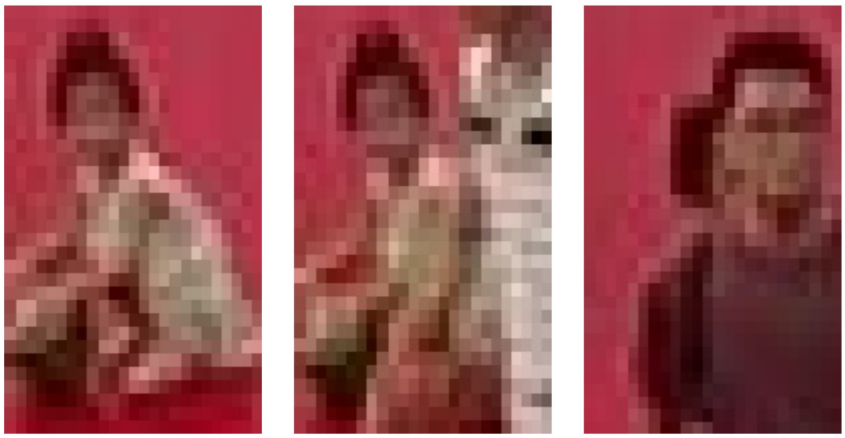
Após a identificação das imagens que continham propriedades diferentes das características gerais das outras imagens da mesma classes, com o uso da distância de Mahalanobis, notou-se que as imagens encontradas não divergiam tanto em

relações às características visuais, ou seja, visualmente elas não eram tão diferente das demais. Além disso, em algumas classes, notou-se que existiam imagens que visualmente eram consideradas mais diferentes das demais, porém essas imagens diferentes não eram as que continham o maior valor de distância observadas.

A Figura 18 apresenta três imagens da classe ‘1503‘, que foi umas das classes estudadas durante as análises das distâncias. Nota-se que a imagem a) é a considerada a mais comum, pois presenta menor valor de soma de distâncias. A imagem b) da Figura 18 foi a imagem com maior valor de distância de Mahalanobis dentro da classe e a imagem c), ainda na Figura 18, foi considerada, após uma avaliação visual, a mais dissemelhante dentro da classe.

Além deste caso da classe ‘1503‘, foram constatados outros exemplos de classes nas quais a imagem com maior valor de soma das distâncias não necessariamente apresentava maior dissimilaridade visual em relação às demais imagens da respectiva classe.

Figura 18 – Análise de imagem com maior valor de soma das distâncias de Mahalanobis da classe ‘1503‘ do banco de imagens criado.: (a) imagem com menor valor de soma das distâncias; b) imagem com maior valor da soma das distâncias; (c) imagem considerada com mais divergia em relação às características gerais da classe.



a) b) c)

Fonte: Original do autor

Concluiu-se, após esse estudo, que o método utilizado para identificar imagens com propriedades visuais diferentes das demais, em alguns casos, diverge quando comparado com critérios visuais de um observador.

4.4 RESULTADOS DA SEGUNDA ANÁLISE ENTRE AS VERSÕES DO ALGORITMO DEEPSORT

O resultado desta segunda comparação mostrou que obteve-se um ganho de desempenho na versão do DeepSORT cujos pesos da RNA da ferramenta DAD foram resultantes do treinamento com as imagens do dataset limpo. A Tabela 2 apresenta os valores de avaliação de três diferentes versões do código DeepSORT com as métricas HOTA, MOTA e MOTP. Com esse resultado pode-se concluir que a limpeza do banco de imagens criado, proporcionou um treinamento da FVENN mais adaptado para as imagens da aplicação, aperfeiçoando o processo de reidentificação do algoritmo e consequentemente melhorando o desempenho de rastreamento do código.

Tabela 2 – Segunda análise entre versões de algoritmos DeepSORT.

Versão	HOTA	MOTA	MOTP
V0	0,346355	0,141691	**0,772698**
V1	0,339879	0,139686	0,771713
V2	**0,350356**	**0,142106**	0,771610

V0 - DeepSORT com pesos da FVENN padrão
V1 - DeepSORT com pesos da FVENN modificados
V2 - DeepSORT com pesos da FVENN modificados - dataset limpo

Fonte: Original do autor.

Após o dataset ser limpo pelo processo descrito em 3.7, foi feita uma segunda comparação da avaliação de desempenho entre as versões do algoritmo de tracking. Nesta avaliação foi adicionada HOTA às métricas usadas para quantificar o desempenho de tracking das versões. Como detalhado na Seção 2.7, a métrica HOTA foi criada a fim de corrigir falhas na métrica MOTA, porém os valores resultantes das métricas MOTA e MOTP continuaram sendo relevantes para a análise de ganho de desempenho entre as versões do DeepSORT.

Após este resultado, a ferramenta de DAD foi considerada pronta para ser integrada à rotina do MOT e fazer as avaliações a fim de identificar variações de desempenho do algoritmo de rastreamento da empresa.

Ao final desse processo da segunda avaliação, foi cogitado a possibilidade investir mais esforços na limpeza deste banco de imagens, porém por questões de tempo essa tarefa é sugerida como atividade para continuação do presente trabalho.

4.5 RESULTADOS DA ANÁLISE ENTRE VERSÕES DO ALGORITMO MOT

Como comentado na Seção 3.8, foi feita uma adaptação para integrar a ferramenta DAD, desenvolvida em *Python*, para o algoritmo de MOT da empresa, desenvolvido em *C++*. Nessa mesma Seção, é comentado que, além da troca da ferramenta descritora de aparência, foi feita a troca da função que retorna o valor da dissimilari-

dade entre os FVs. O resultado da avaliação entre essas versões é apresentado na Tabela 3.

Tabela 3 – Comparações entre as versões do código MOT da empresa.

Versão	HOTA	MOTA	IDS	IDSW
V0	-	0,1878	-	-
V1	**0,429847**	0,208848	8484	5411
V2	0,360029	0,209556	**1780**	4757
V3	0,388067	**0,209783**	3190	**4506**

V0 - MOT sem um descritor de aparência
V1 - MOT com descritor de aparência baseado em histograma de cores
V2 - MOT com DAD e com threshold = 0,7
V3 - MOT com DAD e com threshold = 0,4

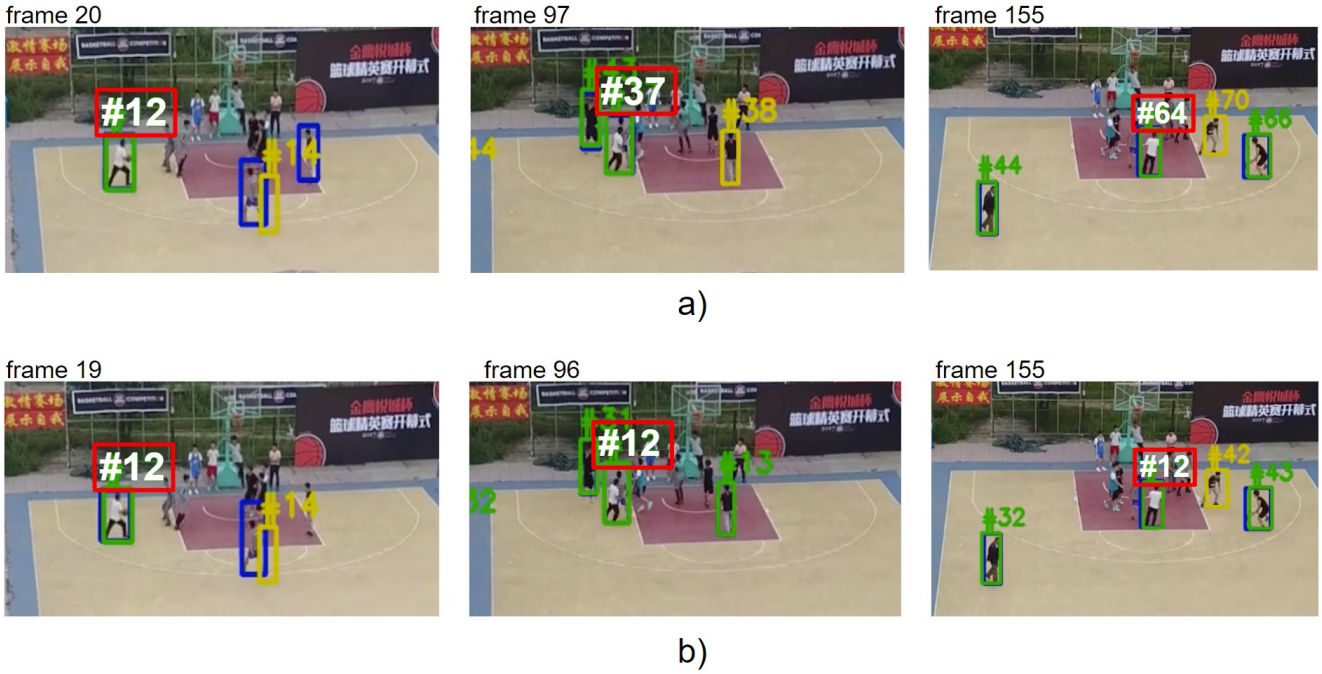
Fonte: Original do autor.

A Figura 19 ilustra o resultado do rastreamento de duas versões do MOT: a primeira, em a), se refere à versão com o descritor de aparência baseado no histograma de cores; em b), tem-se a versão com o descritor de aparência baseado em rede neural extratora de vetor de característica, DAD. Essas imagens são da sequência de teste *uav*0000079_00480_*v* do *dataset* VisDrone (ZHU *et al.*, 2021). Nessa sequência de quadros, há interrupções na detecção do objeto que inicialmente é associado ao número de identificação #12. Nota-se que o rastreamento da versão com a ferramenta DAD, em b), é mais fiel do que se espera, pois, mesmo com interrupções da detecção, a função de reidentificação reassocia o número de identificação já criado ao objeto correto. Porém, não é observado esse mesmo resultado na versão do MOT com o descritor de aparência baseado no histograma de cores, apresentado em a). Neste último resultado, para o mesmo objeto são atribuídos erroneamente os IDs #37 e #64. Vale ressaltar que para ambas as versões do MOT comentadas acima, as inferências da rede neural de detecção de objetos são as mesmas e as únicas diferenças entre as versões são o descritor de aparência e a função do cálculo de distância.

A fim de certificar se a utilização do *dataset* criado pela derivação do banco de imagens VisDrone (ZHU *et al.*, 2021), descrito com mais detalhes na Seção 3.4, trouxe uma melhora no desempenho do processo de reidentificação e criação de identidades no treinamento da rede neural da ferramenta de DAD, foi feita uma comparação entre uma versão do MOT com DAD que usava os pesos padrão da rede extratora de FVs disponibilizados com o código base DeepSORT (GUY, 2021) e a versão do MOT com DAD que utilizava os pesos oriundos do treinamento com o *dataset* de imagens criado.

A Tabela 3 apresenta a comparação entre versões do MOT com diferentes descritores de aparência. A versão V0 corresponde ao MOT sem um descritor de aparência. V1 corresponde a versão do MOT com descritor de aparência baseado no histograma de cores, já as versões V2 e V3 correspondem ao MOT com a ferramenta DAD. A diferença entre V2 e V3 é o valor do parâmetro usado como distância mínima entre vetores de características no processo de reidentificação do *tracker*. Ou seja, na

Figura 19 – Comparação de inferências de diferentes versões do MOT da empresa para a sequência de quadros *uav*0000079_00480_*v*: (a) resultado inferência com MOT usando histograma de cores como descritor de aparência; (b) resultado inferência com MOT usando DAD aperfeiçoado.

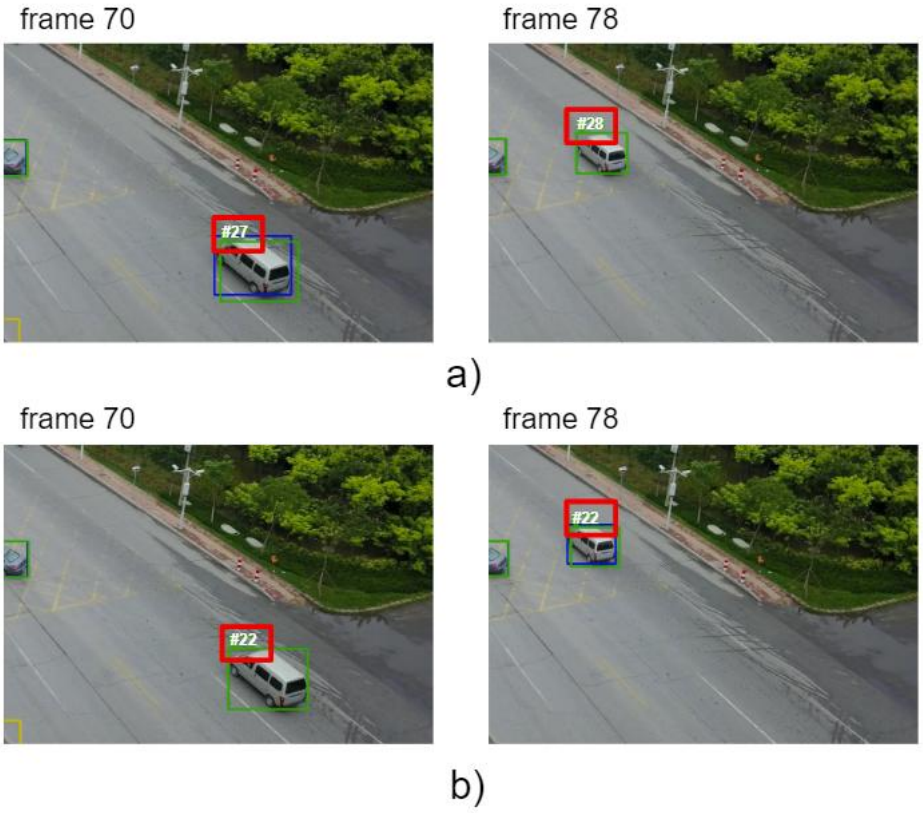


Fonte: Original do autor

versão V2 do MOT, quando comparado dois objetos em quadros diferentes durante a execução da função de reidentificação, caso a distância cosseno entre as FV destas imagens fossem maior que o limiar 0,7, não seria considerado pelo algoritmo que estes dois objetos pertencem ao mesmo ID. Com valor de distância máxima tolerada mais alto, observado em V2, o processo de reidentificação ficou mais tolerante, mais IDs foram reutilizados e consequentemente reduziu a métrica IDS, que mede a quantidade de IDs criados durante a avaliação do *tracker*. Porém, com essa tolerância maior do limiar da distância permitiu que as reidentificações errôneas de objetos acontecessem com mais frequência. Esta última observação é sustentada ao comparar-se o valor da métrica de IDSW entre as versões V2 e V3. Portanto, baseado nas observações feitas a partir da Tabela 3 e no objetivo de melhorar o desempenho da função de reidentificação, foi selecionada a versão V2 como a mais adequada.

Outra observação relevante no estudo da Tabela 3 foi a redução significativa na métrica HOTA que tende a apontar o desempenho geral do algoritmo de rastreamento (LUITEN *et al.*, 2020). Essa redução somada à melhoria na função de reidentificação, representada na redução da métrica IDSW, foram fatores que levantaram a seguinte questão: quais foram os motivos da redução do valor de avaliação pela métrica HOTA visto que o processo de reidentificação foi melhorado? Para identificar a causa dessa

Figura 20 – Comparação das inferências de diferentes versões do MOT da empresa - Sequência de quadros *uav*0000323_01173_*v*: (a) resultado inferência com MOT usando histograma de cores como descritor de aparência; (b) resultado inferência com MOT usando DAD aperfeiçoado.



Fonte: Original do autor

redução da HOTA, foi feita uma análise das submétricas que compõem a métrica HOTA. O resultado desta análise indicou uma redução na submétrica que mede o desempenho de associação na HOTA, o AssA 2.7. Porém por falta de tempo dentro deste projeto, não foi alcançada uma resposta para esta questão levantada.

4.5.1 Discussão sobre resultados da integração da ferramenta DAD e da Análise entre versões do algoritmo MOT

Após ser adicionada a ferramenta DAD ao MOT e feito testes de inferências com esse *tracker* usando as sequencias de *frames* do banco VisDrone, obteve-se um período de inferência por quadro que variou de $5,2ms$ a $24,82ms$. Essa variação depende da quantidade de objetos de interesse na imagens analisada pelo *tracker*. O hardware usado para estes testes um *Desktop* equipado com duas placas de vídeo GeForce GTX 1080, um processador Intel core i7-6850k de 3,6 GHz e uma memória RAM de 64 GB. Esse período de inferência foi considerado satisfatório visto que nesta

versão do MOT a troca de informação entre a ferramenta DAD e o restante da rotina do código MOT não estava otimizada.

Uma vez feita a análise e identificada as variações dos valores das métricas usadas após a adição do DAD no MOT e partindo da premissa que nas versões com e sem o DAD, as detecções de objetos são as mesmas, viu-se uma redução do número de IDs, o que indica menos criação de IDs novos e uma redução do IDSW. Este resultado indica que a função de reassociação teve um melhor desempenho com a adição do DAD. Porém, nota-se uma redução na submétrica de associação AssA dentro da métrica HOTA, que acarreta uma redução no valor desta última. A redução dessa submétrica indica que entre as versões sem e com DAD, o processo de associação e reassociação de IDs do MOT teve uma queda de desempenho com a adição do DAD. E essa última conclusão vai de encontro à conclusão obtida ao observar a redução das métricas IDS e IDSW.

4.5.2 Discussão sobre métricas para avaliação de desempenho

Devido ao constante desenvolvimento e aperfeiçoamento das estratégias de *tracking* de múltiplos objetos, as formas como esses algoritmos são avaliados também têm sido aperfeiçoadas. Para este trabalho, foram usadas principalmente as métricas HOTA, MOTA e IDSW, que representam a quantidade de trocas de reidentificação, e IDS, que representa a quantidade de identificações diferentes criadas.

Durante este trabalho, foram usadas métricas diferentes para avaliar as mudanças de funções no algoritmo a fim de detectar melhorias. Essa falta de padronização de métricas durante os experimentos dos algoritmos, gerou incertezas nas comparações e abriu espaço para interpretações divergentes. Deduz-se que a adoção de uma única métrica como foco de melhoria tornaria o trabalho mais eficiente, que pudesse trazer melhoras mais explícitas e que demais métricas deveriam ter sido utilizadas apenas para auxiliar no diagnóstico de comportamentos indesejados do código.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este trabalho relata as atividades desenvolvidas pelo aluno com o objetivo de aprimorar e implementar a ferramenta de *Deep Appearance Descriptor* no algoritmo de rastreamento de múltiplos objetos baseado em sequências de quadros capturados por VANTs. Esta ferramenta é aplicada na função de reidentificação na rotina desse algoritmo, e essa função é fundamental no processo de rastreamento. O objetivo geral do projeto foi melhorar o desempenho total do algoritmo de rastreamento de múltiplos objetos da empresa através do aprimoramento da função de reidentificação pelo refinamento do descritor de aparência desta função. O resultado alcançado foi uma melhora de desempenho na função de reidentificação do *tracker*.

5.1 CONCLUSÕES

Durante o estudo da literatura relacionada ao tema, identificou-se uma grande variedade de abordagens de códigos de rastreamento de objetos baseado em visão computacional. Ao analisar essas abordagens, concluiu-se que a aplicação DeepSORT atendia os requisitos levantados pela empresa. Também durante esse estudo, foi levantado um *dataset* de imagens disponíveis, porém concluiu-se que para atender os requisitos do algoritmo usado para o treinamento do RNA da ferramenta *Deep Appearance Descriptor* e os requisitos do projeto seria necessário criar um novo banco de imagens.

Mesmo com a criação de um banco de imagens que atendesse os requisitos mencionados do projeto, não foram obtidos melhores resultados nas métricas MOTA e MOTP após a substituição da FVENN padrão pela RNA treinada com esse banco de imagens criado. Levantou-se a hipótese de que esse *dataset* criado continha um número suficientemente grande de classes mal comportadas e a limpeza desta coleção de imagens traria melhores resultados nas métricas usadas para avaliar o desempenho do algoritmo. Essa hipótese se mostrou verdadeira após obter-se novos valores de desempenho da versão do algoritmo cuja ferramenta *Deep Appearance Descriptor* fora treinada com este *dataset* já limpo. Durante esse processo de limpeza, a distância de Mahalanobis se mostrou uma boa ferramenta para identificação de classes e imagens consideradas prejudiciais ao treinamento da FVENN. Com este último resultado, concluiu-se que o uso da ferramenta *Deep Appearance Descriptor* cuja FVENN é treinada com um *dataset* cujas imagens apresentam características similares àquelas encontradas nos cenários de aplicação traz melhoria nas métricas HOTA e MOTA.

Após essa conclusão, iniciou-se o processo de implementação dessa ferramenta no MOT da empresa, substituindo o descritor de aparência, que anteriormente era baseado em histograma de cores, pelo DAD. Para atender os requisitos de integração da ferramenta DAD aprimorada à rotina do MOT, definiu-se que a abordagem mais

eficiente seria através do uso de *container* Docker, com utilização de memória compartilhada. Essa abordagem se mostrou uma boa opção e possibilitou que o MOT continuasse atendendo a velocidade da taxa de quadros.

A comparação das avaliações de desempenho entre as versões do MOT apresentou uma melhora nas métricas IDSW e IDS ao adicionar a ferramenta DAD ao *pipeline* do algoritmo. Baseado nessa comparação, concluiu-se que a integração da ferramenta DAD aperfeiçoada pelo aluno trouxe uma melhora no desempenho nas funções de reidentificação do MOT. Porém, o valor de HOTA também apresentou uma redução com essa integração, o que indica uma perda de desempenho geral de rastreamento. Foi identificado que a redução da submétrica de associação, AssA, que compõe a métrica HOTA, foi a origem dessa perda de desempenho geral, porém não foi possível concluir esse estudo durante o período deste projeto.

5.2 TRABALHOS FUTUROS

Como sugestão de trabalhos futuros, podem ser citados os itens elencados abaixo.

- Criação de um banco de imagens por derivação das imagens dos *dataset* MARS e do VERI, que contêm imagens de pessoas e de veículos, respectivamente. Em seguida, fazer o treinamento da RNA *Wide Residual Network* (WRN) com a fusão desses dois *datasets*, usando o *script* Wojke (2018).
- Fazer a limpeza do banco de imagens criado usando a distância cosseno ao invés da distância de Mahalanobis entre os FVs para identificar imagens com atributos visuais divergentes das demais imagens da respectiva classe.
- Fazer a utilização de outros algoritmos de *metric learning* para o treinamento da rede neural extratora a FV. Foi mapeado o algoritmo *Open Metric Learning* (SHABANOV, 2022), que apresenta algumas vantagens em relação ao algoritmo usado neste trabalho como testes de épocas baseados no desempenho de extração de características ao invés de testes baseado em acurácia de classificação de objetos como é feito em (WOJKE, 2018).
- Identificar a causa da redução do valor da submétrica de associação AssA que compõem a métrica HOTA que por sua vez mede as performance total do algoritmo de rastreamento.

REFERÊNCIAS

BEWLEY, Alex; GE, Zongyuan; OTT, Lionel; RAMOS, Fabio; UPCROFT, Ben. Simple online and realtime tracking. *In*: 2016 IEEE International Conference on Image Processing (ICIP). Phoenix: IEEE, set. 2016. DOI: 10.1109/icip.2016.7533003.

BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark. **YOLOv4: Optimal Speed and Accuracy of Object Detection**. [*S.l.*]: arXiv, 2020. DOI: 10.48550/ARXIV.2004.10934.

GUY, The AI. **yolov4-deepsort**. [*S.l.*: *s.n.*], 2021. https://github.com/theAIGuysCode/yolov4-deepsort.

HENSOLDT. **Intelligence, Surveillance, Target Acquisition and Reconnaissance (ISR and ISTAR)**. 2022. Disponível em: https://www.hensoldt.net/what-we-do/air/intelligence-surveillance-target-acquisition-and-reconnaissance-isr-and-istar/. Acesso em: 1 jun. 2022.

JONATHON LUITEN, Arne Hoffhues. **TrackEval**. [*S.l.*: *s.n.*], 2020. https://github.com/JonathonLuiten/TrackEval.

LI, Xiying; ZHOU, Zhihao. Visual Object Tracking with Deep Neural Networks. *In*: [*S.l.*]: IntechOpen, 2019. Object Re-Identification Based on Deep Learning, p. 87–111.

LIN, Tsung-Yi *et al.* **Microsoft COCO: Common Objects in Context**. [*S.l.*]: arXiv, 2014. DOI: 10.48550/ARXIV.1405.0312.

LUITEN, Jonathon; OSEP, Aljosa; DENDORFER, Patrick; TORR, Philip; GEIGER, Andreas; LEAL-TAIXÉ, Laura; LEIBE, Bastian. HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking. **International Journal of Computer Vision**, Springer, p. 1–31, 2020.

SHABANOV, Aleksei. **Open Metric Learning**. [*S.l.*: *s.n.*], 2022. https://github.com/OML-Team/open-metric-learning.

WOJKE, Nicolai. **Cosine Metric Learning Code**. [*S.l.*: *s.n.*], 2018. https://github.com/nwojke/cosine_metric_learning.

WOJKE, Nicolai; BEWLEY, Alex. Deep Cosine Metric Learning for Person Re-identification. *In*: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). Lake Tahoe, NV, USA: IEEE, mar. 2018. DOI: 10.1109/wacv.2018.00087.

WOJKE, Nicolai; BEWLEY, Alex; PAULUS, Dietrich. **Simple Online and Realtime Tracking with a Deep Association Metric**. [*S.l.*]: arXiv, 2017. DOI: 10.48550/ARXIV.1703.07402.

SPRINGER, [*S.l.*]. **MARS: A Video Benchmark for Large-Scale Person Re-identification**. [*S.l.*: *s.n.*], 2016.

ZHU, Pengfei; WEN, Longyin; DU, Dawei; BIAN, Xiao; FAN, Heng; HU, Qinghua; LING, Haibin. Detection and Tracking Meet Drones Challenge. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, p. 1–1, 2021. DOI: 10.1109/TPAMI.2021.3119563.

APÊNDICE A – PRINCIPAIS FUNÇÕES DO ALGORITMO *PARSER*

```
def integrateTxtFiles(path_to_folder_annotations):
    ## runs the whole path from path_to_folder_annotations and generates a list with the names of all the
    # annotation files
    list_of_annotation_files_names = []
    for file in os.listdir(path_to_folder_annotations):
        if file.endswith(".txt"):
            # generates a list with the names of the annotations without the ".txt" extention
            list_of_annotation_files_names.append(file[:-4])
    print('len(list_of_annotation_files_names) = ',len(list_of_annotation_files_names))
    print('...')
    ## runs the entire list of annotated txt file names
    # create an empty pandas array
    concatenated_array = pd.DataFrame()
    debug_index = 0
    while debug_index < len(list_of_annotation_files_names):
        # read a single txt file and turn it into a pandas array
        data_raw = pd.read_csv(path_to_folder_annotations+'/'+list_of_annotation_files_names[debug_index]+".txt",
            %sep=",", header=None)
        # define the name of the Pandas Array columns
        data_raw.columns = ["frame_index", "target_id", "bbox_left", "bbox_top", "bbox_width", "bbox_height", "score",
             "object_category", "truncation", "occlusion"]
        # filter and leave only the information of the bbox that belong to the classes of people, pedestrians and %cars.
            # reduce. just use the classes people, pedestrians or car in "object class" = 1,2,4
        not_interres_index_class = [0,3,5,6,7,8,9,10,11]
        for i in not_interres_index_class:
            data_raw = data_raw[ data_raw["object_category"]!=i]

        # remove the columns ['score', 'object_category', 'truncation', 'occlusion']
        data_raw = data_raw[["frame_index", "target_id", "bbox_left", "bbox_top", "bbox_width", "bbox_height"]]

        ## get the name of corresponding images collection and integrates with target_id info column
        # for loop to run all the row of the array
        for index in data_raw.index:
            target_id = data_raw["target_id"][index]
            data_raw["target_id"][index] = list_of_annotation_files_names[debug_index]+"_"+str(target_id)

        # concatenate arrays
        concatenated_array=pd.concat([concatenated_array,data_raw])
        debug_index+=1
        progress_value = float("{:.2f}".format((debug_index/len(list_of_annotation_files_names))*100))




    # filter threshold, IDs with an img amount less than the threshold won't integrate the dataset
    list_of_ids_names = concatenated_array['target_id'].unique()

    print("debug:: start the filter threshold!")
    # verify the amount of images of each ID
    debug_index_aux = 0

    for id in list_of_ids_names:
        debug_index_aux += 1
        unique_id_array = concatenated_array[concatenated_array['target_id']==id]
        if(len(unique_id_array.index)< int(arg_thresholder)):
            concatenated_array = concatenated_array[concatenated_array['target_id']!=id]
    print("debug:: finished the filter threshold!")

    # Generates a list of identified ids
    print('At the end,',debug_index,' lists were concatenades')
    return concatenated_array

```

```
63  # Restart blank directories
64  def reset_folders_image(PATH_bbox_train,PATH_bbox_test):
65
66      if os.path.isdir(PATH_bbox_train):
67          shutil.rmtree(PATH_bbox_train)
68      if os.path.isdir(PATH_bbox_test):
69          shutil.rmtree(PATH_bbox_test)
70
71      os.mkdir(PATH_bbox_train)
72      os.mkdir(PATH_bbox_test)
73
74  # function that receives the image path, cuts out the selected part and saves it in the specified path
75  def crop_an_img_and_save_it(origin_img_path, bbox_info, goal_img_path_with_name,is_reshape):
76      global unsaved_imgs_counter
77
78      img = cv2.imread(origin_img_path)
79      '''
80          bbox_info[0]=(bbox_left)(x)
81          bbox_info[1]=(bbox_top)(y)
82          bbox_info[2]=(bbox_width)(dx)
83          bbox_info[3]=(bbox_height)(dy)
84          crop_img = img[y:y+h, x:x+w]
85      '''
86      crop_img = img[bbox_info[1]:bbox_info[3], bbox_info[0]:bbox_info[2]]
87
88      if(is_reshape):
89          #height = bbox_info[3]-bbox_info[1]
90          #width = bbox_info[2]-bbox_info[0]
91          crop_img = cv2.resize(crop_img, (width,height), interpolation= cv2.INTER_NEAREST)
92
93      try:
94          RGB_im = cv2.cvtColor(crop_img, cv2.COLOR_BGR2RGB)
95          # saves the image to the path: goal_img_path_with_name
96          cv2.imwrite(goal_img_path_with_name,crop_img)
97          # plt.imshow(RGB_im)
98          # plt.show()
99
100     except:
101         unsaved_imgs_counter += 1
102         print("unsaved_imgs_counter= {}".format(unsaved_imgs_counter))
103         print("image:{},does not save".format(goal_img_path_with_name))
104
105
106
107
```

```
107
108 def image_separator_by_id(list_all_ids_bboxs,PATH_bbox_train,PATH_bbox_test,PATH_VisDro_imgs_orig):
109     # Generates a list of identified ids
110     list_of_ids_names = list_all_ids_bboxs['target_id']
111     print("debug::list_of_ids_names = ",list_of_ids_names)
112
113
114     list_of_ids_names = list_all_ids_bboxs['target_id'].unique()
115     # for each id name create a mini array with the bbox info
116     debug_aux_index = 0
117
118     for name in list_of_ids_names:
119         # prints the progress of the code for monitoring purpose
120         progress_value = float("{:.2f}".format((debug_aux_index/len(list_of_ids_names))*100))
121         print('building dataset...',progress_value,'%' )
122         if(debug_aux_index//400==0):
123             print_time_counter()
124
125
126         one_id_name_array = list_all_ids_bboxs[list_all_ids_bboxs['target_id']== name]
127         # If the class folder name is an odd number, it goes to bbox_train. If it's even, it goes to the bbox_test folder
128         if(debug_aux_index%2==0):
129             id_folder_imgs_name = PATH_bbox_test +'/'+ str(debug_aux_index).zfill(4)
130         else:
131             id_folder_imgs_name = PATH_bbox_train +'/'+ str(debug_aux_index).zfill(4)
132
133         os.mkdir(id_folder_imgs_name)
134         # run all row of a array of just one unique id
135         for index, row in one_id_name_array.iterrows():
136
137             path_of_img = PATH_VisDro_imgs_orig + '/' + str(row['target_id'])[0:18] + '/'+ str(row['frame_index']).zfill(7) +\
138                 '.jpg'
139             bbox_info = (row['bbox_left'],row['bbox_top'],row['bbox_left']+row['bbox_width'],row['bbox_top']+\
140                 row%'bbox_height'])
141             ## create a name for img
142             # <(0001)ID do pedestrian><(C1)used camera ><(T0002)captured tracklet><(F016)number of the
143             #  frame inside the tracklet>.jpg
144             img_name = str(row['target_id'])[19:].zfill(4) + 'C1' + 'T0001' + 'F' + str(row['frame_index']).zfill(3)+ '.jpg'
145             crop_an_img_and_save_it(path_of_img,bbox_info,id_folder_imgs_name+'/'+ img_name,True)
146
147             # break
148         debug_aux_index+=1
149
150         # print('debug_aux_index = ', debug_aux_index)
151         # stop just for debug reasons
152         # if (debug_aux_index>35):
153         #     break
154
155     # 7702 class in 430 minutes(7h and 10 min)
156     print(' dataset completed ')
```

ANEXO A – ARQUITETURA DA REDE NEURAL CONVOLUCIONAL *WIDE RESIDUAL NETWORKS*, WRN.

Name	**Patch Size/Stride**	**Output Size**
Conv 1	3 X 3/1	32 X 128 X 64
Conv 2	3 X 3/1	32 X 128 X 64
Max Pool 3	3 X 3/2	32 X 64 X 32
Residual 4	3 X 3/1	32 X 64 X 32
Residual 5	3 X 3/1	32 X 64 X 32
Residual 6	3 X 3/2	64 X 32 X 16
Residual 7	3 X 3/1	64 X 32 X 16
Residual 8	3 X 3/2	128 X 16 X 8
Residual 9	3 X 3/1	128 X 16 X 8
Dense 10		128
Batch and l_2 normalization		128

Fonte: Wojke, Bewley e Paulus (2017)